



NLP: Tokenization, Stemming, Lemmatization and Part of Speech Tagging



Kerem Kargin · [Follow](#)

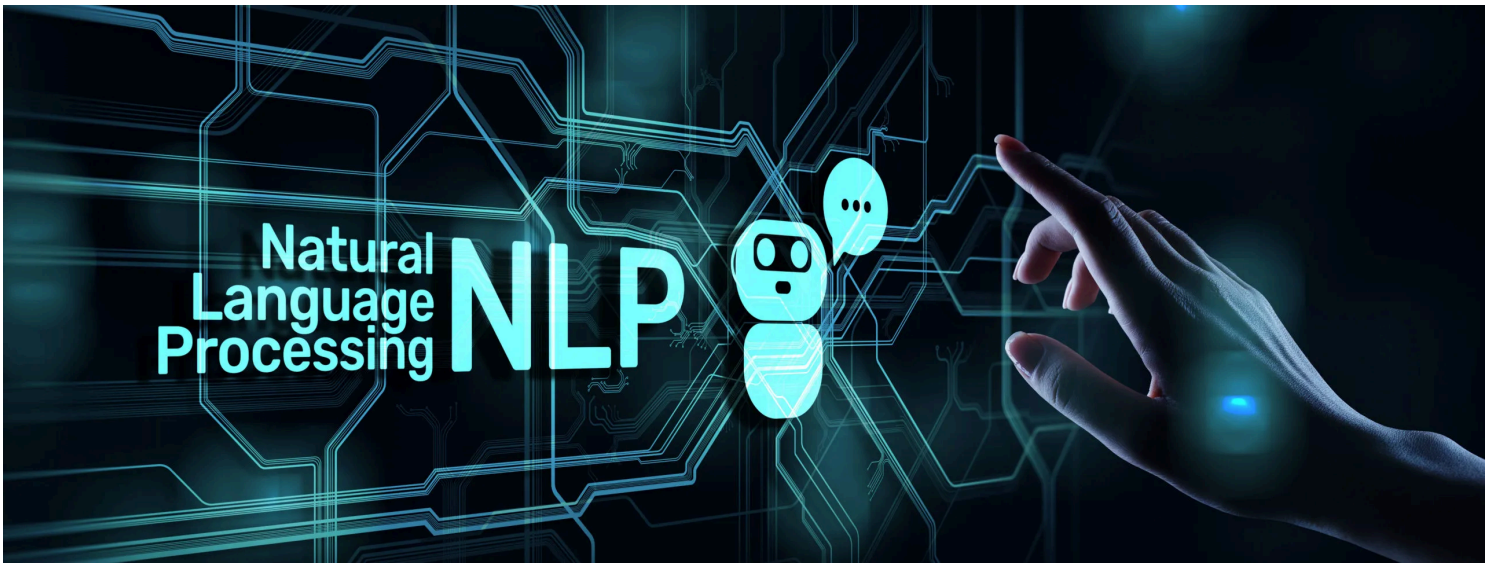
6 min read · Feb 27, 2021



305



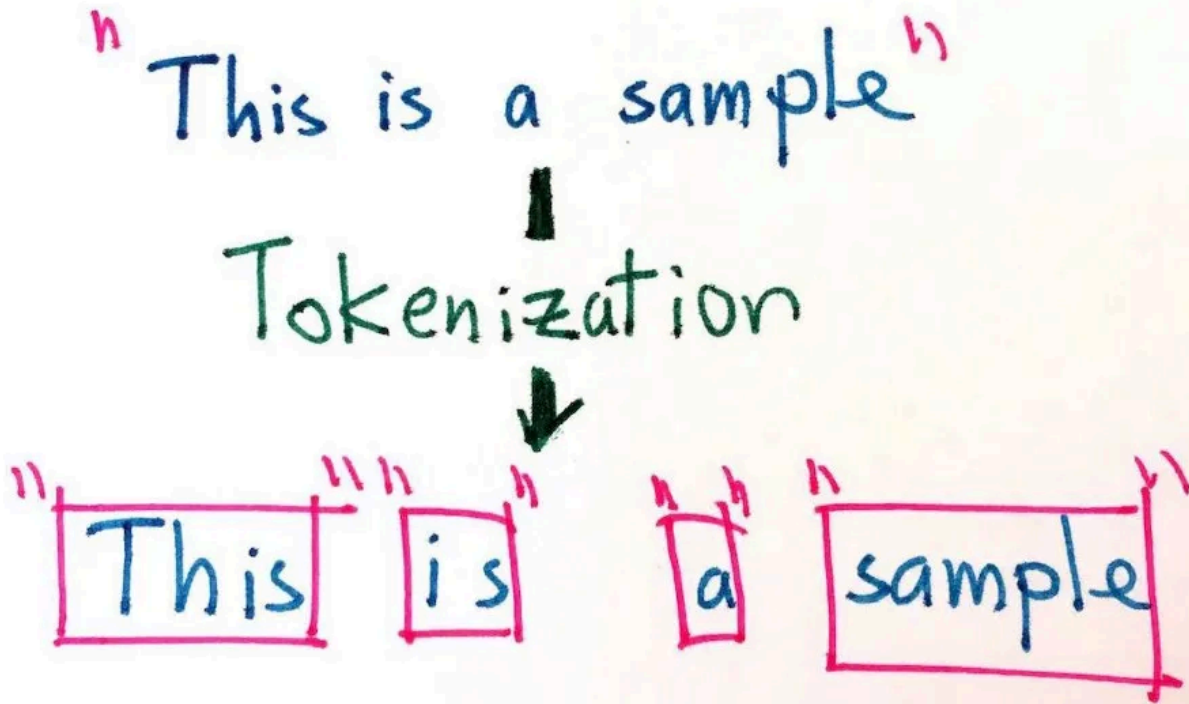
In this blog post, I'll talk about Tokenization, Stemming, Lemmatization, and Part of Speech Tagging, which are frequently used in Natural Language Processing processes. We'll have information about how to use them by reinforcing them with applications. Enjoyable readings.



Resource: <https://www.asksid.ai/resources/what-is-natural-language-processing/>

Tokenization

Tokenization is the process of breaking down the given text in natural language processing into the smallest unit in a sentence called a token. Punctuation marks, words, and numbers can be considered tokens. So why do we need Tokenization? We may want to find the frequencies of the words in the entire text by dividing the given text into tokens. Then, models can be made on these frequencies. Or we may want to tag tokens by word type. I'll mention this while explaining Part of Speech Tagging.



Resource: <https://medium.com/data-science-in-your-pocket/tokenization-algorithms-in-natural-language-processing-nlp-1fceb8454af>

Let's make an application for Tokenization. We will examine the tokenization process by applying it with 2 different libraries. First, let's make an example with the `TextBlob` library. First, we need to download the library.

```
!pip install textblob
```

Hosted on [Jovian](#)

[View File](#)

```
import textblob
from textblob import TextBlob
```

Hosted on [Jovian](#)

[View File](#)

After downloading the library and importing it, let's define a text.

```
text = "Hello everyone! Welcome to my blog post on Medium. We are studyin
```

In order to do tokenization, we can access tokens by calling words from the TextBlob object. As a result, you will see that the text we have is allocated to tokens as below.

```
TextBlob(text).words
```

```
WordList(['Hello', 'everyone', 'Welcome', 'to', 'my', 'blog', 'post',  
'on', 'Medium', 'We', 'are', 'studying', 'Natural', 'Language',  
'Processing'])
```

As you can see, we were able to split it into tokens quite simply. Let's do this with the NLTK (Natural Language Toolkit) library.

```
import nltk  
from nltk import sent_tokenize  
from nltk import word_tokenize
```

Hosted on [Jovian](#)

[View File](#)

As you can see, we have called word_tokenize and sent_tokenize objects from the NLTK library. With sent_tokenize we'll be able to split the text into sentences. We'll split it into the same text words with word_tokenize.

```
tokens_sents = nltk.sent_tokenize(text)
print(tokens)
```

```
['Hello everyone!', 'Welcome to my blog post on Medium.', 'We are studying  
Natural Language Processing.']
```

Hosted on [Jovian](#)

[View File](#)

The `sent_tokenize` function uses an instance of `PunktSentenceTokenizer` from the `nltk.tokenize.punkt` module, which is already been trained and thus very well knows to mark the end and beginning of sentence at what characters and punctuation.

```
tokens_words = nltk.word_tokenize(text)
print(tokens_words)
```

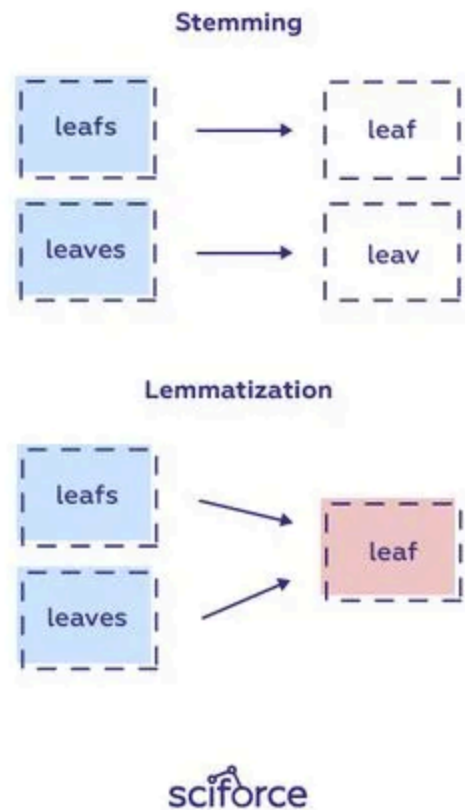
```
['Hello', 'everyone', '!', 'Welcome', 'to', 'my', 'blog', 'post', 'on',  
'Medium', '.', 'We', 'are', 'studying', 'Natural', 'Language',  
'Processing', '.']
```

Hosted on [Jovian](#)

[View File](#)

`word_tokenize()` function is a wrapper function that calls `tokenize()` on an instance of the `TreebankWordTokenizer` class.

Thus, we can do the split into tokens in a very practical way with two different libraries.



Resource: <https://tr.pinterest.com/pin/706854104005417976/>

Stemming

Stemming is the process of finding the root of words. Let's examine a definition made about this.

Stemming is definitely the simpler of the two approaches. With stemming, words are reduced to their word stems. A word stem need not be the same root as a dictionary-based morphological root, it just is an equal to or smaller form of the word.

When you are breaking down words with stemming, you can sometimes see that finding roots is erroneous and absurd. Because Stemming works rule-based, it cuts the suffixes in words according to a certain rule. This reveals inconsistencies regarding stemming. Overstemming and understemming.

Overstemming occurs when words are over-truncated. In such cases, the meaning of the word may be distorted or have no meaning.

Understemming occurs when two words are stemmed from the same root that is not of different stems.

We'll examine the stemming example with two different algorithms.

- Porter Stemmer (Algorithm details are in this [link](#).)
- Snowball Stemmer (Algorithm details are in this [link](#).)

First, let's import the `PorterStemmer`.

```
from nltk.stem import PorterStemmer
```

Hosted on [Jovian](#)

[View File](#)

Then, let's define a `ps` an object that will implement `PorterStemmer`. After defining the word to be stemming, all that remains is to run the code.

```
ps = PorterStemmer()  
word = ("civilization")  
ps.stem(word)
```

```
'civil'
```

Hosted on [Jovian](#)

[View File](#)

Let's do a similar process with `SnowballStemmer`. For this, we do import the `SnowballStemmer`.

```
from nltk.stem.snowball import SnowballStemmer
```

Hosted on [Jovian](#)

[View File](#)

Then, we define the stemmer object. Here, in addition to PorterStemmer, we can also choose in which language we will stemming in SnowballStemmer.

```
stemmer = SnowballStemmer(language = "english")  
word = "civilization"  
stemmer.stem(word)
```

```
'civil'
```

Hosted on [Jovian](#)

[View File](#)

Lemmatization

Lemmatization is the process of finding the form of the related word in the dictionary. It is different from Stemming. It involves longer processes to calculate than Stemming. Let's examine a definition made about this.

The aim of lemmatization, like stemming, is to reduce inflectional forms to a common base form. As opposed to stemming, lemmatization does not simply chop off inflections. Instead, it uses lexical knowledge bases to get the correct base forms of words.

NLTK provides WordNetLemmatizer class which is a thin wrapper around the wordnet corpus. This class uses morphy() function to the WordNet CorpusReader class to find a lemma .

First, let's do import NLTK and WordNetLemmatizer.


```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

Hosted on [Jovian](#)

[View File](#)

Let's see how the lemmatizer works in a single word.

```
# Lemmatize single word
```

```
print(lemmatizer.lemmatize("workers"))
print(lemmatizer.lemmatize("beeches"))
```

worker

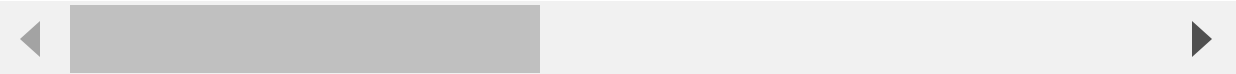
beech

Hosted on [Jovian](#)

[View File](#)

Then we have a text. Let's break this text down to tokens first. Then let's apply the lemmatizer one by one on these tokens.

```
text = "Let's lemmatize a simple sentence. We first tokenize the sentence"
word_list = nltk.word_tokenize(text)
print(word_list)
```



```
['Let', "'", 's', 'lemmatize', 'a', 'simple', 'sentence', '.', 'We',  
'first', 'tokenize', 'the', 'sentence', 'into', 'words', 'using',  
'nltk.word_tokenize', 'and', 'then', 'we', 'will', 'call',  
'lemmatizer.lemmatize', '(', ')', 'on', 'each', 'word', '.']
```

```
lemmatized_output = ' '.join([lemmatizer.lemmatize(w) for w in word_list])
print(lemmatized_output)
```

Let 's lemmatize a simple sentence . We first tokenize the sentence into word using `nltk.word_tokenize` and then we will call `lemmatizer.lemmatize ()` on each word .

Hosted on [Jovian](#)

[View File](#)

In the first example of Lemmatizer, we used WordNet Lemmatizer from the NLTK library. Let's do similar operations with TextBlob. As a result, we will reach similar results.

```
# pip install textblob

from textblob import TextBlob, Word
```

Hosted on [Jovian](#)

[View File](#)

```
word = 'stripes'
w = Word(word)
w.lemmatize()
```

'stripe'

Hosted on [Jovian](#)

[View File](#)

When we apply the '*lemmatize*' process to the word '*stripes*', it deletes the 's' suffix and reaches the word '*stripe*', which is the dictionary form of the word. Now let's do the same on a sentence.

```
text = "The striped bats are hanging on their feet for best"  
sent = TextBlob(text)  
" ".join([w.lemmatize() for w in sent.words])
```

'The striped bat are hanging on their foot for best'

Hosted on [Jovian](#)

[View File](#)

Thus, we examined how the ‘lemmatization’ process is implemented on both sentences and a single word with two different libraries.

Part of Speech Tagging

Part of Speech Tagging (POS-Tag) is the labeling of the words in a text according to their word types (noun, adjective, adverb, verb, etc.). Let’s look at how it is explained in a definition.



Resource: <https://blog.aaronccwong.com/2019/building-a-bigram-hidden-markov-model-for-part-of-speech-tagging/>

It is a process of converting a sentence to forms — list of words, list of tuples (where each tuple is having a form (word, tag)). The tag in case of is a part-of-speech tag, and signifies whether the word is a noun, adjective, verb, and so on.

Let’s examine the most used tags with examples.

- Noun (N)- Daniel, London, table, dog, teacher, pen, city, happiness, hope
- Verb (V)- go, speak, run, eat, play, live, walk, have, like, are, is
- Adjective(ADJ)- big, happy, green, young, fun, crazy, three
- Adverb(ADV)- slowly, quietly, very, always, never, too, well, tomorrow
- Preposition (P)- at, on, in, from, with, near, between, about, under
- Conjunction (CON)- and, or, but, because, so, yet, unless, since, if
- Pronoun(PRO)- I, you, we, they, he, she, it, me, us, them, him, her, this
- Interjection (INT)- Ouch! Wow! Great! Help! Oh! Hey! Hi!

So, how does POS Tagging works?

POS tagging is a supervised learning solution that uses features like the previous word, next word, is first letter capitalized etc. NLTK has a function to get pos tags and it works after tokenization process.

Let's understand Part of Speech Tagging with an application. Let's import the NLTK library and word_tokenize object. When applying this, we first need to split a sentence into tokens. Tagging works after splitting to tokens.

```
import nltk
from nltk import word_tokenize
```

Hosted on [Jovian](#)

[View File](#)

```
text = "The striped bats are hanging on their feet for best"
tokens = nltk.word_tokenize(text)
print("Parts of Speech: ",nltk.pos_tag(tokens))
```

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

Hosted on [Jovian](#)

[View File](#)

After separating the words in a sentence into tokens, we applied the POS-Tag process. For example, the word ‘The’ has gotten the tag ‘DT’. The word ‘feet’ has been labeled ‘NNS’. You can review this [link](#) to investigate in detail what these tags are.

Thank you for reading my blog post. Your suggestions and feedback regarding the content are very important to me. You can indicate your thoughts by commenting. Happy days everyone!

Mlearning.ai Submission Suggestions

How to become a writer on Mlearning.ai

[medium.com](#)

Resources

1. <https://www.geeksforgeeks.org/nlp-part-of-speech-default-tagging/>
2. <https://pythonexamples.org/nltk-tokenization/>
3. <https://towardsdatascience.com/part-of-speech-tagging-for-beginners-3a0754b2ebba>

4. <https://www.machinelearningplus.com/nlp/lemmatization-examples-python/>
5. <https://www.geeksforgeeks.org/introduction-to-stemming/>
6. https://www.geeksforgeeks.org/python-nltk-nltk-tokenizer-word_tokenize/
7. <https://medium.com/@gianpaul.r/tokenization-and-parts-of-speech-pos-tagging-in-pythons-nltk-library-2d30f70af13b>
8. <https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/>
9. <https://www.geeksforgeeks.org/introduction-to-stemming/>
10. <https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8>
11. <https://medium.com/@datamonsters/text-preprocessing-in-python-steps-tools-and-examples-bf025f872908>
12. https://www.tutorialspoint.com/natural_language_toolkit/natural_language_toolkit_stemming_lemmatization.htm
13. <https://www.geeksforgeeks.org/nlp-part-of-speech-default-tagging/>
14. <https://medium.com/greyatom/learning-pos-tagging-chunking-in-nlp-85f7f811a8cb>

Data Science

NLP

Tokenization

Stemming

Pos Tagging