Text analytics is the process of deriving meaning out of a text and nowadays it has a significant role in many fields such as Banking, Entertainment, Insurance, etc. These vast fields use multiple text-based products such as intelligent chatbots, sentiment recommendations, text classifiers, etc. So getting deep dive into Text analytics is very important and we should be able to process texts in multiple languages too.
Since the text data available will be unstructured / semi-structured most of the time, it requires efficient tools to derive insights out of it. Thus solving an NLP problem is not a single-stage procedure and there are multiple sources and tools available to do Text analytics efficiently.

NLTK

Data cleaning consists of multiple steps and Tokenization is the First and very basic step in Text analytics. So let's begin with Tokenizationuch as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition and it helps the computer to analyze, preprocess, and understand the written text.

Let's start with Tokenization!!!

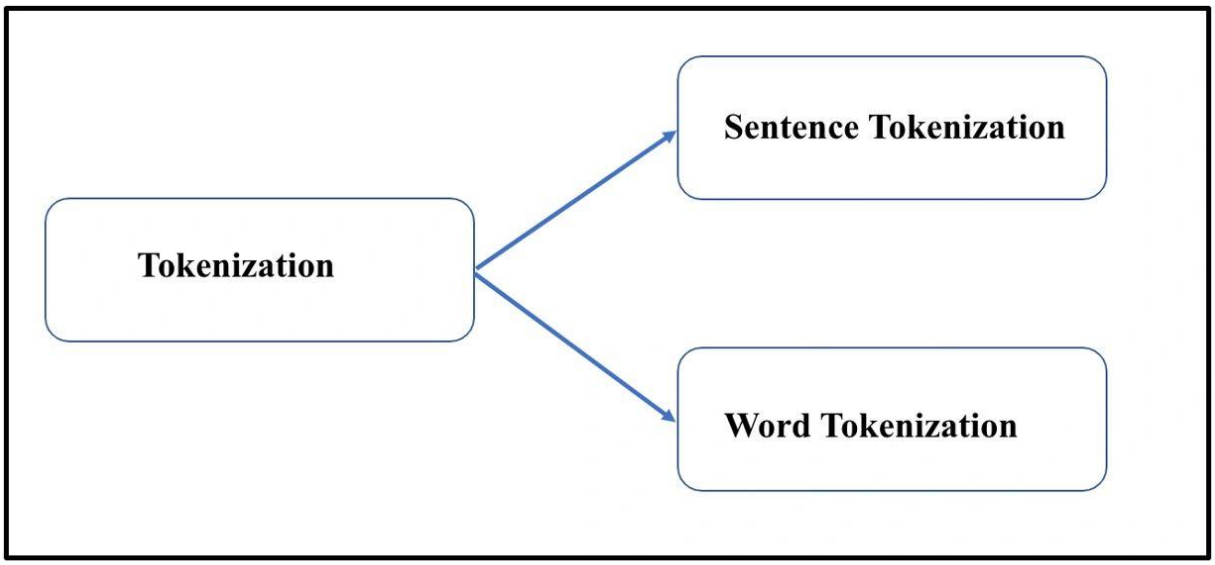
Data cleaning consists of multiple steps and Tokenization is the First and very basic step in Text analytics. So let's begin with Tokenization.

What is Tokenization ???

Tokenization is the process of breaking a text into meaningful elements such as words or phrases or symbols etc, where each individual element is known as a Token. Tokenization in NLP is the process by which a large quantity of text is divided into smaller parts called tokens.

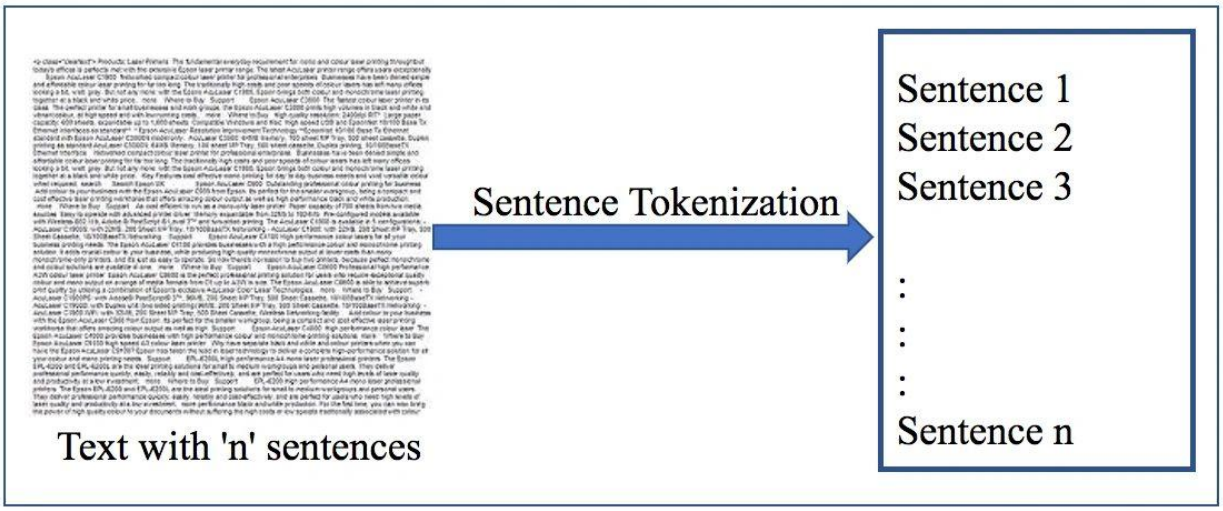
Types of Tokenization

Tokenization can be broadly classified into two based on their functionality.



Sentence Tokenization

Sentence tokenizer breaks a text paragraph into sentences. NLTK 'sent_tokenize' function works by considering the punctuation full stop ('.') followed by a single white space as the separation between 2 sentences. So it will check for this separation condition and will take a break once it meets the condition in a text.



Here, the given Text is tokenized into Sentences.

Example

We can import the sent_tokenize function from the nltk library.

```
import nltk
from nltk.tokenize import sent_tokenize
```

Here we are trying to analyze text segmentation using the 'sent_tokenize' function and have a few examples showing how this tokenizer will work on different scenarios.

```
sentence = "Hello World. It's good to see you. How are you doing?"
sent_tokenize(sentence)
```

```
['Hello World.', "It's good to see you.", 'How are you doing?']
```

```
sentence = "Hello World::: It's good to see you, How are you doing?"
sent_tokenize(sentence)
```

```
["Hello World::: It's good to see you, How are you doing?"]
```

```
para ="Hello World,It's good to see you, How are you doing?"
sent_tokenize(para)
```

```
["Hello World,It's good to see you, How are you doing?"]
```

```
para = "Hello. World: It's good to see you@How are you doing?.    fgj"
sent_tokenize(para)
```

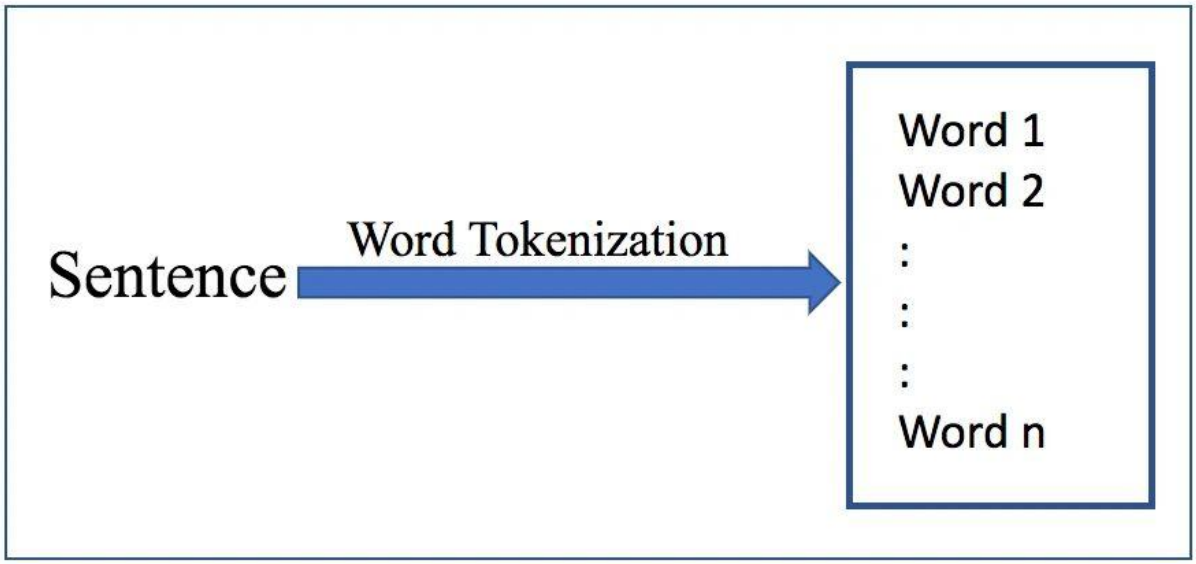
```
['Hello.', "World: It's good to see you@How are you doing?.", 'fgj']
```

```
para="Hello Mr. Sing, how are you today? The sky is pinkish-blue. the weather is good here."
sent_tokenize(para)
```

```
['Hello Mr.',
 'Sing, how are you today?',
 'The sky is pinkish-blue.',
 'the weather is good here.']
```

Word Tokenization

Word tokenizer breaks a sentence into words. These tokenized words can be used as the input for text cleaning steps such as punctuation removal, numeric character removal, etc. NLTK tokenizer splits based on a single white space and its default list of punctuations. So it will check for this separation condition and will take a break once it meets the condition in a sentence.



Here, the given Sentence is tokenized into words.

Example

We can import the word_tokenize function from the nltk library.

```
from nltk.tokenize import word_tokenize
```

Edit Image

Here we are trying to analyze sentence segmentation using the 'word_tokenize' function and have a few examples showing how this tokenizer will work on different scenarios.

```
sentence = 'Hello World.'
print(word_tokenize(sentence))
```

```
['Hello', 'World', '.']
```

```
sentence = "Hello World: It's good to see you, Thanks for buying this book."
print(word_tokenize(sentence))
```

```
['Hello', 'World', ':', 'It', "'s", 'good', 'to', 'see', 'you', ',', 'Thanks', 'for', 'buying', 'this', 'book', '.']
```

```
sentence = "Hello World// It's good to see you// thanks for buying this book."
print(word_tokenize(sentence))
```

```
['Hello', 'World//', 'It', "'s", 'good', 'to', 'see', 'you//', 'thanks', 'for', 'buying', 'this', 'book', '.']
```

WordPunctTokenizer

Another alternative word tokenizer is WordPunctTokenizer. It splits all punctuation into separate tokens.

```
from nltk.tokenize import WordPunctTokenizer
tokenizer = WordPunctTokenizer()
tokenizer.tokenize("Can't is a contraction.")
```

```
['Can', "'", 't', 'is', 'a', 'contraction', '.']
```

```
tokenizer.tokenize("http://www.packtpub.com.")
```

```
['http', '://', 'www', '.', 'packtpub', '.', 'com', '.']
```

```
tokenizer.tokenize("http: //www.packtpub.com.")
```

```
['http', ':', '//', 'www', '.', 'packtpub', '.', 'com', '.']
```

```
tokenizer.tokenize("http?://www.packtpub.com.")
```

```
['http', '?://', 'www', '.', 'packtpub', '.', 'com', '.']
```

Simple whitespace tokenizer¶

The following is a simple example of using RegexpTokenizer to tokenize on whitespace. Here punctuation still remains in the tokens.

```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer('\s+', gaps=True)
tokenizer.tokenize("Can't is a contraction.")
```

```
["Can't", 'is', 'a', 'contraction.']
```

```
tokenizer.tokenize("http://www.packtpub.com.")
```

```
['http://www.packtpub.com.']
```

```
tokenizer.tokenize("My mother-in-law's rants make me furious !")
```

```
['My', "mother-in-law's", 'rants', 'make', 'me', 'furious', '!']
```

```
tokenizer.tokenize("Hello,Mr.Hari")
```

```
['Hello,Mr.Hari']
```

```
tokenizer.tokenize("0, 2, 4, ... , 100")
```

```
['0,', '2,', '4,', '...', ',', '100']
```

The above discussion is a quick glance at the Tokenization procedure, which is the key component for any Text analysis procedure. There are different libraries such as Spacy, Gensim, etc available to perform this Tokenization process. We can choose any of these options depending on the requirement.