# Data Cleaning, Splitting, Normalizing, and Stemming - NLP Course 01

By Sunil Ghimire- "Your legacy will never be erased"

Welcome to **SUNIL's** new tutorial series on Natural Language Processing. We have already completed tutorials on "**Machine Learning and Deep Learning**". If you have not gone through those tutorials you can check them now. In today's session, I will be covering a basic course on Natural Language Processing.

## TABLE OF CONTENTS



## INTRODUCTION TO NATURAL LANGUAGE PROCESSING

Language is the most important tool of communication invented by human civilization. It is either spoken or written, consisting of the use of words in a structured and conventional way. Language helps us share our thoughts, and understand others.

Natural Language Processing is, a form of artificial intelligence, all about trying to analyze and understand either written or spoken language and the context that it's being used in. The ultimate objective of NLP is to read, decipher, understand, and make sense of human languages in a manner that is valuable.

Wikipedia defines NLP as " ***a subfield of AI concerned with the interactions between computers and human (natural) languages, in particular, how to program computers to process and analyze large amounts of natural language data.*** "

## DATA PREPROCESSING - NLP

In this session, I will show you the ways of cleaning the text for the preparation of the dataset in NLP. I will be using the built-in python function and also introduce the NLTK library in the next session. Taking about data preprocessing in NLP, I encounter the steps like splitting the documents into sentences, words and there are various ways to split the texts. Here I will go through some of the ways:-

1. **SPLIT BY WHITE SPACES**

Splitting by white spaces refers to the splitting of documents or texts by word. Applying **split()** with no input

parameters calls the function to split text by looking at white spaces only. It doesn't take account of any apostrophe.

Example: Look how who's is split.

```
text = 'Albert Einstein is  one of the most brilliant scientists who's
ever lived.'
# split into words by white space
words = text.split()
print(words[:100])

OUTPUT:
  ['Albert', 'Einstein', 'is', 'one', 'of', 'the', 'most', 'brilliant',
'scientists', 'who's', 'ever', 'lived.']
```

## 2. SPLIT BY WORDS

It is already clear by title about its function. Do you know the difference between split by words and split by white space? Notice the difference in "who's".

```
import re

# split based on words only
words = re.split(r'\W+', text)
print(words[:100])

Output:
  ['Albert', 'Einstein', 'is', 'widely', 'celebrated', 'as', 'one', 'of',
'the', 'most', 'brilliant', 'scientists', 'who', 's', 'ever', 'lived', '']
```

## 3. NORMALIZATION

In NLP, we convert all uppercase characters to lowercase. We don't recommend using this step in every dataset preprocessing. Normalizing the words can change the entire meaning.

Example: Orange is a French telecom company whereas orange s fruit.

```
# split based on words only
words = re.split(r'\W+', text)

# convert to lower case
words = [word.lower() for word in words]
print(words[:100])


Output:
  ['albert', 'einstein', 'is', 'widely', 'celebrated', 'as', 'one', 'of',
```

```
'the', 'most', 'brilliant', 'scientists', 'who', 's', 'ever', 'lived', '']
```

## NATURAL LANGUAGE TOOLKIT (NLTK)

NLTK, Natural Language Toolkit, is an open-source Python platform to work on Natural Language Processing. This library requires Python 3.5, 3.6, 3.7, or 3.8.

### 1. SPLIT BY SENTENCE

The tokenizer divides a text into a list of sentences by using an unsupervised algorithm to build a model for abbreviation words, collocations, and words that start sentences. We should train on a large collection of plain text in the target language before using them.

The NLTK data package includes a pre-trained Punkt tokenizer for the English language.

```
import nltk
from nltk import sent_tokenize
nltk.download('punkt')

# split into sentences
sentences = sent_tokenize(text)
for sentence in sentences:
  print(sentence)
```

### 2. SPLIT BY WORDS

Make sure you check out the output and spot the differences in “who’s”

```
from nltk.tokenize import word_tokenize

# split into words
tokens = word_tokenize(text)
print(tokens[:100])

Output:
  ['Albert', 'Einstein', 'is', 'widely', 'celebrated', 'as', 'one', 'of',
'the', 'most', 'brilliant', 'scientists', 'who', '’', 's', 'ever',
'lived', '.']
```

### 3. FILTERING

Python includes the built-in function **isalpha()** that can be used in order to determine whether or not the scanned

word is alphabetical or else (numerical, punctuation, special characters, etc.). Male sure you check out the output and spot the difference.

```
# split into words
tokens = word_tokenize(text)

# remove all tokens that are not alphabetic
words = [word for word in tokens if word.isalpha()]
print(words[:100])

Output:
  ['Albert', 'Einstein', 'is', 'widely', 'celebrated', 'as', 'one', 'of',
'the', 'most', 'brilliant', 'scientists', 'who', 's', 'ever', 'lived']
```

## 4. REMOVE STOPWORDS

Stopwords are words that do not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. The most common are short function words such as the, is, at, which, and on, etc.

In this case, removing stopwords can cause problems when searching for phrases that include them, particularly in names such as “The Who” or “Take That”.

Including the word “not” as a stopword also changes the entire meaning if removed (try “this code is not good”)

```
# let's list all the stopwords for NLTK
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')

stop_words = stopwords.words('english')
print(stop_words)
```

As you can see, the stopwords are all lower case and don’t have punctuation. If we’re to compare them with our tokens, we need to make sure that our text is prepared the same way.

This cell recaps all that we have previously learned in this collab: tokenizing, lower casing, and checking for alphabetic words.

```
# clean our text

# split into words
tokens = word_tokenize(text)

# convert to lower case
tokens = [w.lower() for w in tokens]

# remove all tokens that are not alphabetic
words = [word for word in tokens if word.isalpha()]

# filter out stop words
stop_words = set(stopwords.words('english'))
words = [w for w in words if not w in stop_words]
print(words[:100])

Output:
  ['albert', 'einstein', 'widely', 'celebrated', 'one', 'brilliant',
'scientists', 'ever', 'lived']
```

## STEMMING IN NATURAL LANGUAGE PROCESSING

Stemming refers to the process of reducing each word to its root or base. There are two types of stemmers for suffix stripping: Porter and Lancaster and each has its own algorithm and sometimes displays different outputs.

a. **Porter**

```
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer

# stemming of words
porter = PorterStemmer()
stemmed = [porter.stem(word) for word in words]
print(stemmed[:100])

Output:
  ['albert', 'einstein', 'wide', 'celebr', 'one', 'brilliant',
'scientist', 'ever', 'live']
```

b. **Lancaster**

```
from nltk.tokenize import word_tokenize
from nltk.stem.lancaster import LancasterStemmer

# stemming of words
lancaster = LancasterStemmer()
stemmed = [lancaster.stem(word) for word in words]
print(stemmed[:100])

Output:
  ['albert', 'einstein', 'wid', 'celebr', 'on', 'bril', 'sci', 'ev',
'liv']
```

This is all for today's tutorial. We will cover more details in the next tutorials. Stay Safe & Happy Coding.

# ☺ Thanks for your time ☺

What do you think of this "**Data Cleaning, Splitting, Normalizing, and Stemming - NLP Course**"? (Appreciation, Suggestions, and Questions are highly appreciated).