# AI LAB EXPERIMENT NO: 11

**Implementation of NLP – Cleaning Text**

# WORKING PRINCIPLE:-

In natural language processing, human language is separated into fragments so that the grammatical structure of sentences and the meaning of words can be analyzed and understood in context.

Let’s see the various different steps that are followed while preprocessing the data also used for dimensionality reduction.

1. **Tokenization**
2. **Lower casing**
3. **Stop words removal**
4. **Stemming**
5. **Lemmatization**

Each term is the axis in the vector space model. In muti-dimensional space, the text or document are constituted as vectors. The number of different words represents the number of dimensions.

The python library that is used to do the preprocessing tasks in nlp is [nltk](https://www.analyticssteps.com/blogs/what-natural-language-toolkitnltk-nlp). You can install the nltk package using ***“pip install nltk”***.

### **1. Tokenization:**

### 

It is a method in which sentences are converted into words.

import nltk

from nltk.tokenize import word\_tokenize

token = word\_tokenize("My Email address is: taneshbalodi8@gmail.com")

token

**tokenization in natural language processing**

Tokenization

(Read also: [Sentiment Analysis of YouTube Comments](https://www.analyticssteps.com/blogs/sentiment-analysis-youtube-comments))

### **2. Lowercasing:**

### 

the tokenized words into lower case format. (NLU -> nlu). Words having the same meaning like nlp and NLP if they are not converted into lowercase then these both will constitute as non-identical words in the vector space model.

Lowercase = []

for lowercase in token:

Lowercase.append(lowercase.lower())

Lowercase

**lowercasing with the help of lower() function in natural language processing**

Lowercasing

### **3. Stop words removal:**

### 

These are the most often used that do not have any significance while determining the two different documents like (a, an, the, etc.) so they are to be removed. Check the below image wherefrom the sentence “**Introduction to Natural Language Processing**” the “**to**” word is removed.

from nltk.corpus import stopwords

stop\_words = stopwords.words('english')

from string import punctuation

punct = list(punctuation)

print(dataset[1]['quote'])

tokens = word\_tokenize(dataset[1]['quote'])

len(tokens)

**output of the dataset without stopwords removal**

Without removing Stopwords

We got to see 50 tokens without removing stopwords, Now we shall remove stopwords.

cleaned\_tokens = [token for token in tokens if token not in stop\_words

and token not in punctuation]

len(cleaned\_tokens)

By cleaning the stopwords we got the length of the dataset as 24.

(Referred blog: [What is SqueezeBERT in NLP?](https://www.analyticssteps.com/blogs/what-squeezbert-nlp))

### **4. Stemming:**

 It is the process in which the words are converted to its base from. Check the below code implementation where the words of the sentence are converted to the base form.

from nltk.stem import PorterStemmer

ps = PorterStemmer()

print(ps.stem('jumping'))

print(ps.stem('lately'))

print(ps.stem('assess'))

print(ps.stem('ran'))

****

Stemming

## 

### **5. Lemmatization:**

### 

Different from [stemming, lemmatization](https://www.analyticssteps.com/blogs/what-stemming-and-lemmatization-nlp) lowers the words to word in the present language for example check the below image where word has and is are changed to ha and be respectively.

from nltk import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize('ran', 'v'))

print(lemmatizer.lemmatize('better', 'a'))

****

Lemmatization

# CODE:-

raw\_docs = ["Here are some very simple basic sentences.",

"They won't be very interesting, I'm afraid.",

"The point of these examples is to \_learn how basic text cleaning works\_ on \*very simple\* data."]

# Tokenizing text into bags of words

from nltk.tokenize import word\_tokenize

tokenized\_docs = [word\_tokenize(doc) for doc in raw\_docs]

print(tokenized\_docs)

# Removing punctuation

import re

import string

regex = re.compile('[%s]' % re.escape(string.punctuation)) #see documentation here: http://docs.python.org/2/library/string.html

tokenized\_docs\_no\_punctuation = []

for review in tokenized\_docs:

new\_review = []

for token in review:

new\_token = regex.sub(u'', token)

if not new\_token == u'':

new\_review.append(new\_token)

tokenized\_docs\_no\_punctuation.append(new\_review)

print(tokenized\_docs\_no\_punctuation)

nltk.download('stopwords')

from nltk.corpus import stopwords

tokenized\_docs\_no\_stopwords = []

for doc in tokenized\_docs\_no\_punctuation:

new\_term\_vector = []

for word in doc:

if not word in stopwords.words('english'):

new\_term\_vector.append(word)

tokenized\_docs\_no\_stopwords.append(new\_term\_vector)

print(tokenized\_docs\_no\_stopwords)

from nltk.stem.porter import PorterStemmer

from nltk.stem.snowball import SnowballStemmer

from nltk.stem.wordnet import WordNetLemmatizer

porter = PorterStemmer()

snowball = SnowballStemmer('english')

wordnet = WordNetLemmatizer()

preprocessed\_docs = []

for doc in tokenized\_docs\_no\_stopwords:

final\_doc = []

for word in doc:

final\_doc.append(porter.stem(word))

#final\_doc.append(snowball.stem(word))

#final\_doc.append(wordnet.lemmatize(word))

preprocessed\_docs.append(final\_doc)

print(preprocessed\_docs)

# OUTPUT:-

# 

# 

# 

**RESULT:-**

Hence, the Implementation of NLP for cleaning text is done successfully.