**Natural Language Processing**

**NLP:** - NLP is a method which is communicating with an intelligent system using an NLP or we can also say that the ability of artificial system to process a human spoken language is called natural language processing.

**Why we use NLP?**

Let us take example of amazon website, so people are usually buying their product and give their reviews and we human can read that reviews whether there is positive or negative statement. But machine can’t understand this, so that why we use the concept of NLP. It used to handle these kinds of scenario. Some e.g. Alexa, Siri, sentimental analysis etc.

In NLP we use NLTK (Natural Language Took kit) libraries to perform different-different function

The first thing you need to do in any NLP project is Text Preprocessing. Preprocessing input text simply means putting the data into a predictable and analyzable form. It’s a crucial step for building an amazing NLP application.

1. **Stopwards**

The words which are generally filtered out before processing a naturallanguage are called stop words. These are actually the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc.) and does not add much information to the text. Examples of a few stop words in English are “the”, “a”, “an”, “so”, “what”.

Example of removing stopwords

from nltk.corpus import stopwards

sw\_nltk = stopwards.words(‘english’)

text = "When I first met her she was very quiet. She remained quiet during the entire two-hour long journey from Stony Brook to New York."

words=[word for word in text.split() if word.lower() not in sw\_nltk]

new\_text=” ”. join (words)

print(new\_text)

print (“Old length: “, len(text))

print (“New length:”, Len(new\_text))

output:

first met quiet. remained quiet entire two-hour long journey Stony Brook New York.  
Old length: 129  
New length: 82

We can clearly see that the removal of stop words reduced the length of the sentence from 129 to 82.

1. **Tokenization**

It is used to splitting a phase, sentence, paragraph or an entire text document into smaller units such as individual words or terms. Each of these smaller units are called tokens When we split the text into sentences, we call it sentence tokenization. For words, we call it word tokenization.

**Example of sentence tokenization**

from nltk.tokenize import sent\_tokenize

text = “Hello everyone. Life is matter of Choice. Every choice you make makes you.”

sent\_tokenize(text)

output:

[‘hello everyone.’,

‘Life is matter of Choice.’

‘Every choice you make makes you.’]

**Example of word tokenization**

from nltk.tokenize import word\_tokenize

text = “Hello everyone. Life is matter of Choice.”

sent\_tokenize(text)

output:

[‘hello’, ‘everyone.’, ’Life’, ‘is’, ‘matter’. ‘of’, ‘Choice.’]

1. **Stemming and Lemmatization**

Stemming is a technique used to extract the base form of the words by removing affixes from them or we can also say that it reduces a word to its stem the result is less readable by human but makes the test more comparable across observations.

Example: “Historical” and “history” have the same stem: “Histori”

Example of using stemming

from nltk.stem import PorterStemmer

porter = PorterStemmer ()

words = [‘Connects’,’Connecting’,’Connections’]

for word in words:

print (word,” ---> “, porter.stem(word))

output:

Connects ---> connect

Connecting ---> connect

Connections ---> connect

**Lemmatization**: - It consider the context and converts the word to its meaningful base form. It takes more time in processing as compared to stemming

Example: “History” and “Historical” have the word: “History”

from nltk. stem import WordNetLemmatizer ()

lemm = WordNetLemmatizer ()

words = [‘historical’,’Connecting’,’Connect]

for word in words:

print (word,” ---> “lemm. Stem(word))

output:

Historical ---> History

Connecting ---> connect

Connect ---> connect

**Bag of Word**: A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things: A vocabulary of known words. A measure of the presence of known words

Let’s understand this with an example

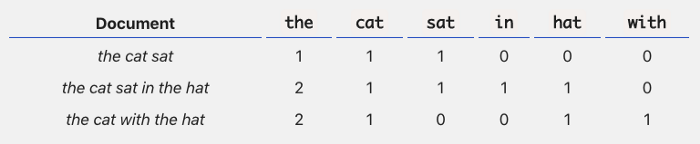
* the cat sat
* the cat sat in the hat
* the cat with the hat

**Step 1: Determine the Vocabulary**

We first define our vocabulary, which is the set of all words found in our document set. The only words that are found in the 3 documents above are: the, cat, sat, in, the, hat, and with.

# Step 2: Count

To vectorize our documents, all we have to do is **count how many times each word appears**:



Now we have length-6 vectors for each document!

* *the cat sat*: [1, 1, 1, 0, 0, 0]
* *the cat sat in the hat*: [2, 1, 1, 1, 1, 0]
* *the cat with the hat*: [2, 1, 0, 0, 1, 1]

when we use BOW. It’s like a literal **bag**-of-words: it only tells you what words occur in the document, not where they occurred.

**TF-IDF (Term Frequency -Inverse Document Frequency)**

TF-IDF stands for “**Term Frequency — Inverse Document Frequency**”. This is a technique to quantify words in a set of documents. We generally compute a score for each word to signify its importance in the document and corpus. This method is a widely used technique in Information Retrieval and Text Mining

All of this process is done using the vectorized form of query and documents.

Here its formula

TF-IDF = Term Frequency (TF) \* Inverse Document Frequency (IDF)

## **Term Frequency:**

## This measures the frequency of a word in a document. This highly depends on the length of the document and the generality of the word. TF is individual to each document and word; hence we can formulate TF as follows:

TF(t,d) = number of rep. of words in sentence/ number of words in sentence

## **Inverse Document Frequency**

IDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because they are present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage.

IDF(t) = N/df

Now there are few other problems with the IDF, when we have a large corpus size say N=10000, the IDF value explodes. So, to dampen the effect we take the log of IDF.

At query time, when the word is not present in is not in the vocab, it will simply be ignored. But in few cases, we use a fixed vocab and few words of the vocab might be absent in the document, in such cases, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator.

IDF(t) = log (N/ (df + 1))