**Natural language toolkit (NLTK)** is the most popular library for natural language processing (NLP) which was written in Python and has a big community behind it. NLTK also is very easy to learn, actually, it’ s the easiest natural language processing (NLP) library that we are going to use. It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

**Gensim** is billed as a Natural Language Processing package that does ‘Topic Modeling for Humans’. But it is practically much more than that. It is a leading and a state-of-the-art package for processing texts, working with word vector models (such as Word2Vec, FastText etc)

Topic models and word embedding are available in other packages like scikit, R etc. But the width and scope of facilities to build and evaluate topic models are unparalleled in gensim, plus many more convenient facilities for text processing. Another important benefit with gensim is that it allows you to manage big text files without loading the whole file into memory.

First, let's install nltk and gensim by following commands:

pip install nltk

pip install gensim

**Tokenization of words (NLTK)**

We use the method word\_tokenize() to split a sentence into words. Take a look example below

from nltk.tokenize import word\_tokenize

data = "Mars is approximately half the diameter of Earth."

print(word\_tokenize(data))

Output:

['Mars', 'is', 'approximately', 'half', 'the', 'diameter', 'of', 'Earth']

**Tokenization of sentences (NLTK)**

An obvious question in your mind would be why sentence tokenization is needed when we have the option of word tokenization. We need to count average words per sentence, so for accomplishing such a task, we use sentence tokenization as well as words to calculate the ratio.

from nltk.tokenize import sent\_tokenize

data = "Mars is a cold desert world. It is half the size of Earth. "

print(sent\_tokenize(data))

Output:

['Mars is a cold desert world', 'It is half the size of Earth ']

Now, you know how these methods is useful when handling text classification. Let's implement it in our similarity algorithm.

**Open file and tokenize sentences**

Create a .txt file and write 4-5 sentences in it. Include the file with the same directory of your Python program. Now, we are going to open this file with Python and split sentences.

import nltk

from nltk.tokenize import word\_tokenize, sent\_tokenize

file\_docs = []

with open ('demofile.txt') as f:

tokens = sent\_tokenize(f.read())

for line in tokens:

file\_docs.append(line)

print("Number of documents:",len(file\_docs))

Program will open file and read it's content. Then it will add tokenized sentences into the array for word tokenization.

**Tokenize words and create dictionary**

Once we added tokenized sentences in array, it is time to tokenize words for each sentence.

gen\_docs = [[w.lower() for w in word\_tokenize(text)]

for text in file\_docs]

Output:

[['mars', 'is', 'a', 'cold', 'desert', 'world', '.'],

['it', 'is', 'half', 'the', 'size', 'of', 'earth', '.']]

In order to work on text documents, Gensim requires the words (aka tokens) be converted to unique ids. So, Gensim lets you create a Dictionary object that maps each word to a unique id. Let's convert our sentences to a [list of words] and pass it to the corpora.Dictionary() object.

dictionary = gensim.corpora.Dictionary(gen\_docs)

print(dictionary.token2id)

Output:

{'.': 0, 'a': 1, 'cold': 2, 'desert': 3, 'is': 4, 'mars': 5,

'world': 6, 'earth': 7, 'half': 8, 'it': 9, 'of': 10, 'size': 11, 'the': 12}

A dictionary maps every word to a number. Gensim lets you read the text and update the dictionary, one line at a time, without loading the entire text file into system memory

-----------------------------------------------------------------------------------------------------------------------------