# EXPERIMENT -1

## Count Word Frequency

Natural Language Processing is one of the most commonly used technique which is implemented in machine learning applications — given the wide range of analysis, extraction, processing and visualising tasks that it can perform. In this article, you will learn how to implement all of these aspects and present your project. The primary goal of this project is to tokenize the textual content, remove the stop words and find the high frequency words. We shall implement this in Python 3.6.4.

```
from bs4 import BeautifulSoup
import urllib.request
import nltk
response = urllib.request.urlopen('http://
php.net/')
html = response.read()
soup = BeautifulSoup(html,"html5lib")
text = soup.get_text(strip=True)
tokens = [t for t in text.split()]
freq = nltk.FreqDist(tokens)
for key,val in freq.items():
    print (str(key) + ':' + str(val))
```

# EXPERIMENT -2

## Remove Stop Words Using NLTK

It is common practice to remove words that appear frequently in the English language such as 'the', 'of' and 'a' (known as stop-words) because they're not so interesting.
The package nltk has a list of stop-words in English which you'll now store as sw and of which you'll print the first several elements.
If you get an error here, run the command nltk.download ('stop-words') to install the stop-words on your system.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example_sent = "This is a sample sentence,
showing off the stop words filtration."

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example_sent)

filtered_sentence = [w for w in word_tokens
if not w in stop_words]

filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

print(word_tokens)
print(filtered_sentence)
```

OUTPUT:

['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']

# EXPERIMENT -3

## Tokenize Text Using NLTK

The NLTK module is a massive tool kit, aimed at helping you with the entire Natural Language Processing (NLP) methodology.
In order to install NLTK run the following commands in your terminal.

- sudo pip install nltk
- Then, enter the python shell in your terminal by simply typing python
- Type import nltk
- nltk.download('all')

The above installation will take quite some time due to the massive amount of tokenisers, chunkers, other algorithms, and all of the corpora to be downloaded.

- Some terms that will be frequently used are :
- Corpus – Body of text, singular. Corpora is the plural of this.
- Lexicon – Words and their meanings.
- Token – Each "entity" that is a part of whatever was split up based on rules. For examples, each word is a token when a sentence is "tokenized" into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph.
- So basically tokenizing involves splitting sentences and words from the body of the text.

```
# import the existing word and sentence tokenizing
# libraries
from nltk.tokenize import sent_tokenize, word_tokenize

text = "Natural language processing (NLP) is a field " + \
       "of computer science, artificial intelligence " + \
       "and computational linguistics concerned with " + \
       "the interactions between computers and human " + \
       "(natural) languages, and, in particular, " + \
       "concerned with programming computers to " + \
       "fruitfully process large natural language " + \
       "corpora. Challenges in natural language " + \
       "processing frequently involve natural " + \
       "language understanding, natural language" + \
       "generation frequently from formal, machine" + \
       "-readable logical forms), connecting language " + \
       "and machine perception, managing human-" + \
       "computer dialog systems, or some combination " + \
       "thereof."

print(sent_tokenize(text))
print(word_tokenize(text))`
```

OUTPUT:

[‘Natural language processing (NLP) is a field of computer science, artificial intelligence and computational linguistics concerned with the interactions between computers and human (natural) languages, and, in particular, concerned with programming computers to fruitfully process large natural language corpora.’, ‘Challenges in natural language processing frequently involve natural language understanding, natural language generation (frequently from formal, machine-readable logical forms), connecting language and machine perception, managing human-computer dialog systems, or some combination thereof.’]
[‘Natural’, ‘language’, ‘processing’, ‘(‘, ‘NLP’, ‘)’, ‘is’, ‘a’, ‘field’, ‘of’, ‘computer’, ‘science’, ‘,’, ‘artificial’, ‘intelligence’, ‘and’, ‘computational’, ‘linguistics’, ‘concerned’, ‘with’, ‘the’, ‘interactions’, ‘between’, ‘computers’, ‘and’, ‘human’, ‘(‘, ‘natural’, ‘)’, ‘languages’, ‘,’, ‘and’, ‘,’, ‘in’, ‘particular’, ‘,’,

‘concerned’, ‘with’, ‘programming’, ‘computers’, ‘to’, ‘fruitfully’, ‘process’, ‘large’, ‘natural’, ‘language’, ‘corpora’, ‘.’, ‘Challenges’, ‘in’, ‘natural’, ‘language’, ‘processing’, ‘frequently’, ‘involve’, ‘natural’, ‘language’, ‘understanding’, ‘,’, ‘natural’, ‘language’, ‘generation’, ‘(‘, ‘frequently’, ‘from’, ‘formal’, ‘,’, ‘machine-readable’, ‘logical’, ‘forms’, ‘)’, ‘,’, ‘connecting’, ‘language’, ‘and’, ‘machine’, ‘perception’, ‘,’, ‘managing’, ‘human-computer’, ‘dialog’, ‘systems’, ‘,’, ‘or’, ‘some’, ‘combination’, ‘thereof’, ‘.’]

# EXPERIMENT -4

## NLTK Word Stemming

In the areas of Natural Language Processing we come across situation where two or more words have a common root. For example, the three words - agreed, agreeing and agreeable have the same root word agree. A search involving any of these words should treat them as the same word which is the root word. So it becomes essential to link all the words into their root word. The NLTK library has methods to do this linking and give the output showing the root word.

```
import nltk
from nltk.stem.porter import PorterStemmer
porter_stemmer = PorterStemmer()

word_data = "It originated from the idea
that there are readers who prefer learning
new skills from the comforts of their
drawing rooms"
# First Word tokenization
nltk_tokens = nltk.word_tokenize(word_data)
#Next find the roots of the word
for w in nltk_tokens:
       print "Actual: %s  Stem: %s"  %
(w,porter_stemmer.stem(w))
```

```
OUTPUT:

Actual: It  Stem: It
Actual: originated  Stem: origin
```

```
Actual: from  Stem: from
Actual: the  Stem: the
Actual: idea  Stem: idea
Actual: that  Stem: that
Actual: there  Stem: there
Actual: are  Stem: are
Actual: readers  Stem: reader
Actual: who  Stem: who
Actual: prefer  Stem: prefer
Actual: learning  Stem: learn
Actual: new  Stem: new
Actual: skills  Stem: skill
Actual: from  Stem: from
Actual: the  Stem: the
Actual: comforts  Stem: comfort
Actual: of  Stem: of
Actual: their  Stem: their
Actual: drawing  Stem: draw
Actual: rooms  Stem: room
```

# EXPERIMENT -5

## Lemmatising Words

Lemmatisation is similar ti stemming but it brings context to the words.So it goes a steps further by linking words with similar meaning to one word. For example if a paragraph has words like cars, trains and automobile, then it will link all of them to automobile. In the below program we use the WordNet lexical database for lemmatisation.

```
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

word_data = "It originated from the idea
that there are readers who prefer learning
new skills from the comforts of their
drawing rooms"
nltk_tokens = nltk.word_tokenize(word_data)
for w in nltk_tokens:
        print "Actual: %s  Lemma: %s"  %
(w,wordnet_lemmatizer.lemmatize(w))

OUTPUT:

Actual: It  Lemma: It
Actual: originated  Lemma: originated
Actual: from  Lemma: from
Actual: the  Lemma: the
Actual: idea  Lemma: idea
Actual: that  Lemma: that
Actual: there  Lemma: there
Actual: are  Lemma: are
```

```
Actual: readers  Lemma: reader
Actual: who  Lemma: who
Actual: prefer  Lemma: prefer
Actual: learning  Lemma: learning
Actual: new  Lemma: new
Actual: skills  Lemma: skill
Actual: from  Lemma: from
Actual: the  Lemma: the
Actual: comforts  Lemma: comfort
Actual: of  Lemma: of
Actual: their  Lemma: their
Actual: drawing  Lemma: drawing
Actual: rooms  Lemma: room
```