

```
pip install SpeechRecognition
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: SpeechRecognition in /usr/local/lib/python3.7/dist-packages
```

```
import os
import speech_recognition as sr

r = sr.Recognizer()
audio = sr.AudioFile('food.wav')
```

▼ Data Preprocessing

```
import wave
obj = wave.open('food.wav','r')
print("Number of channels",obj.getnchannels())
print("Sample width",obj.getsampwidth())
print("Frame rate.",obj.getframerate())
print("Number of frames",obj.getnframes())
print("parameters:",obj.getparams())
obj.close()
```

```
Number of channels 2
Sample width 2
Frame rate. 44100
Number of frames 175584384
parameters: _wave_params(nchannels=2, sampwidth=2, framerate=44100, nframes=175584384, (
```

```
#importing the required libraries
import os
import speech_recognition as sr
import wave
import numpy as np
import matplotlib.pyplot as plt
import IPython.display as ipd
import librosa
import librosa.display
```

```
#opening the file using wave
raw = wave.open("food.wav")
```

```

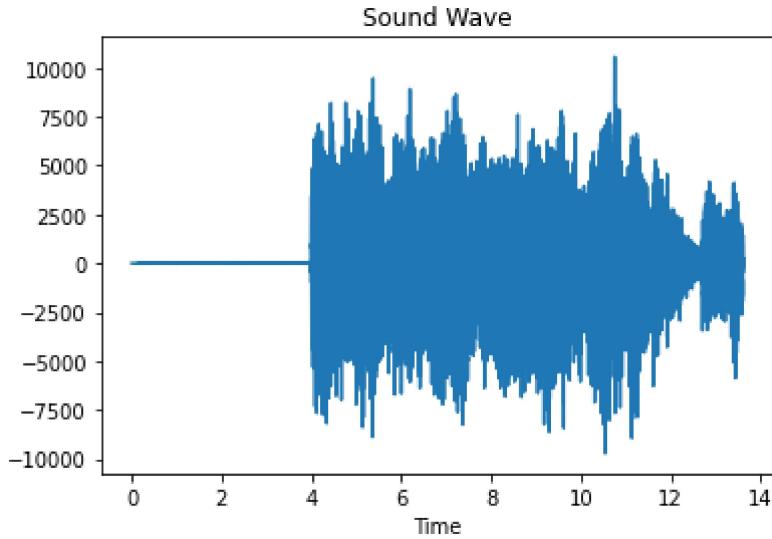
#reading the signal of 300000 frames
signal = raw.readframes(300000)
signal = np.frombuffer(signal, dtype ="int16")

# gets the frame rate
f_rate = raw.getframerate()

#for plotting the time, we divide the len of the signal with the frame rate.
time = np.linspace(
    0, # start
    len(signal) / f_rate,
    num = len(signal)
)

#using matplotlib for the plot
plt.figure(1)
plt.title("Sound Wave")
plt.xlabel("Time")
plt.plot(time, signal)
plt.show()

```

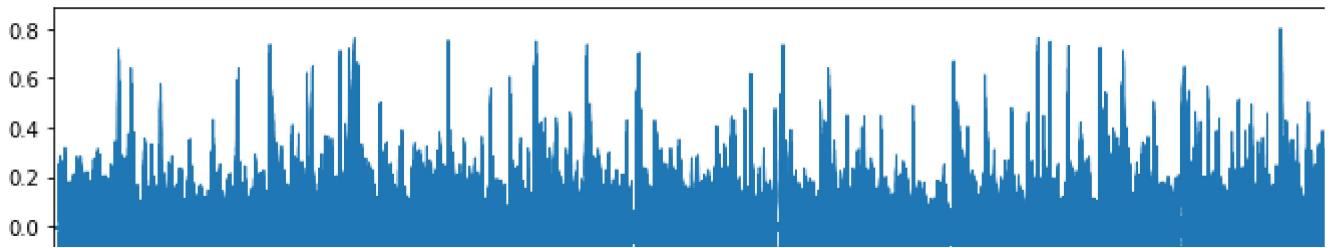


```

#using the librosa library and plot the wave file for time and sample rate for the provided w
plt.figure(figsize=(15,4))
data1,sample_rate1 = librosa.load('food.wav', sr=44100, mono=True, offset=0.0, duration=900,
librosa.display.waveplot(data1,sr=sample_rate1, max_points=50000.0,x_axis='time', offset=0.0,

```

```
<matplotlib.collections.PolyCollection at 0x7f5aa4d41710>
```



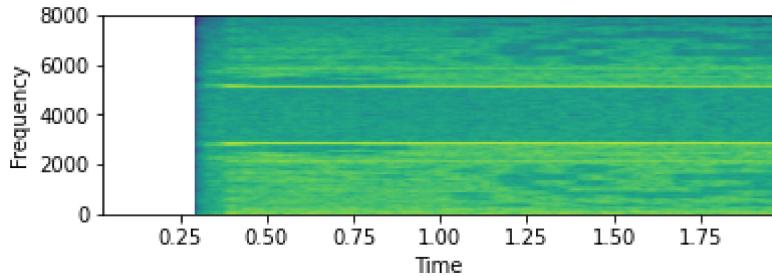
```
#using the wav file to plot the energry vs sample rate and time vs frequency graphs
```

```
signal_wave = wave.open("food.wav", 'r')
sig = np.frombuffer(signal_wave.readframes(16000), dtype=np.int16)
plt.figure(1)
```

```
plot_b = plt.subplot(212)
plot_b.specgram(sig, NFFT=1024, Fs=16000, noverlap=900)
plot_b.set_xlabel('Time')
plot_b.set_ylabel('Frequency')
```

```
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/axes/_axes.py:7592: RuntimeWarning: di
    Z = 10. * np.log10(spec)
```



▼ Speech to Text

```
#intializing text to empty first
finalText=""
#the intial offset is 0
offset=0
#for i in the range of for the duration offset, we repeat the conversion
for i in range(10):
    #using the recogniser to find the text from the audio
    r = sr.Recognizer()
    audio = sr.AudioFile('food.wav')
    with audio as source:
        audio = r.record(source, duration=300, offset=offset)
    # set offset as 300
    offset=offset+300
```

```
text=r.recognize_google(audio)
finalText=finalText+text ## recognizing the speech and converting into text
print(finalText)
```

hello my name is food toxicology would like to welcome back once again today the subject

▼ Adding Punctuations

```
pip install deepmultilingualpunctuation
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public
Requirement already satisfied: deepmultilingualpunctuation in /usr/local/lib/python3.7/dist-packages (1.0.0)
Requirement already satisfied: torch>=1.8.1 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: transformers in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: huggingface-hub<1.0,>=0.10.0 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
Requirement already satisfied: urllib3!=1.25.0,!>=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from deepmultilingualpunctuation)
```

```
pip install transformers[sentencepiece]
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public
Requirement already satisfied: transformers[sentencepiece] in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: huggingface-hub<1.0,>=0.10.0 in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: protobuf<=3.20.2 in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
Requirement already satisfied: sentencepiece!=0.1.92,>=0.1.91 in /usr/local/lib/python3.7/dist-packages (from transformers[sentencepiece])
```

```
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-packages (from ipykernel==5.1.4) (3.7.4.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from ipykernel==5.1.4) (2.2.0)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.7/dist-packages (from ipykernel==5.1.4) (1.12.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from ipykernel==5.1.4) (0.5.2)
Requirement already satisfied: urllib3!=1.25.0,!>=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from ipykernel==5.1.4) (1.25.7)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from ipykernel==5.1.4) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from ipykernel==5.1.4) (2.8.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from ipykernel==5.1.4) (2020.6.14)
```

```
from deepmultilingualpunctuation import PunctuationModel  
!pip install transformers
```

```
model = PunctuationModel()
text = text1
clean_text=model.preprocess(text)
labeled_words=model.predict(clean_text)
result = model.restore_punctuation(text)
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: transformers in /usr/local/lib/python3.7/dist-packages (<=
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (<
Requirement already satisfied: huggingface-hub<1.0,>=0.10.0 in /usr/local/lib/python3.7/
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/pyth
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/di
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: urllib3!=1.25.0,!>=1.25.1,<1.26,>=1.21.1 in /usr/local/lit
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (1
/usr/local/lib/python3.7/dist-packages/transformers/pipelines/token_classification.py:1:
``grouped_entities` is deprecated and will be removed in version 5.0.0, defaulted to'
```

```
#predicted accuracy  
print(labeled_words)
```

```
print(result)

    hello, my name is food toxicology would like to welcome back once again. today. the sub-
```



▼ Capitalizing Text

```
def capitalizng_sentence(data):
    sent_tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
    sentences_from_summary = sent_tokenizer.tokenize(data)
    corrected_sentences = [sentence.capitalize() for sentence in sentences_from_summary]
    print(' '.join(corrected_sentences))

capitalizedres=capitalizng_sentence(result)
print(capitalizedres)
```

```
Hello, my name is food toxicology would like to welcome back once again. Today. The sub-
None
```



▼ Text Summarization(Abstractive)

```
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation
from heapq import nlargest

def summarize(text, per):
    nlp = spacy.load('en_core_web_sm')
    doc= nlp(text)
    tokens=[token.text for token in doc]
    word_frequencies={}
    for word in doc:
        if word.text.lower() not in list(STOP_WORDS):
            if word.text.lower() not in punctuation:
                if word.text not in word_frequencies.keys():
                    word_frequencies[word.text] = 1
                else:
                    word_frequencies[word.text] += 1
    max_frequency=max(word_frequencies.values())
    for word in word_frequencies.keys():
        word_frequencies[word]=word_frequencies[word]/max_frequency
    sentence_tokens= [sent for sent in doc.sents]
```

```
sentence_scores = {}
for sent in sentence_tokens:
    for word in sent:
        if word.text.lower() in word_frequencies.keys():
            if sent not in sentence_scores.keys():
                sentence_scores[sent]=word_frequencies[word.text.lower()]
            else:
                sentence_scores[sent]+=word_frequencies[word.text.lower()]
select_length=int(len(sentence_tokens)*per)
summary=nlargest(select_length, sentence_scores,key=sentence_scores.get)
final_summary=[word.text for word in summary]
summary=''.join(final_summary)
return summary
```

```
text2=summarize(text1, 0.07)
print(text2)
```

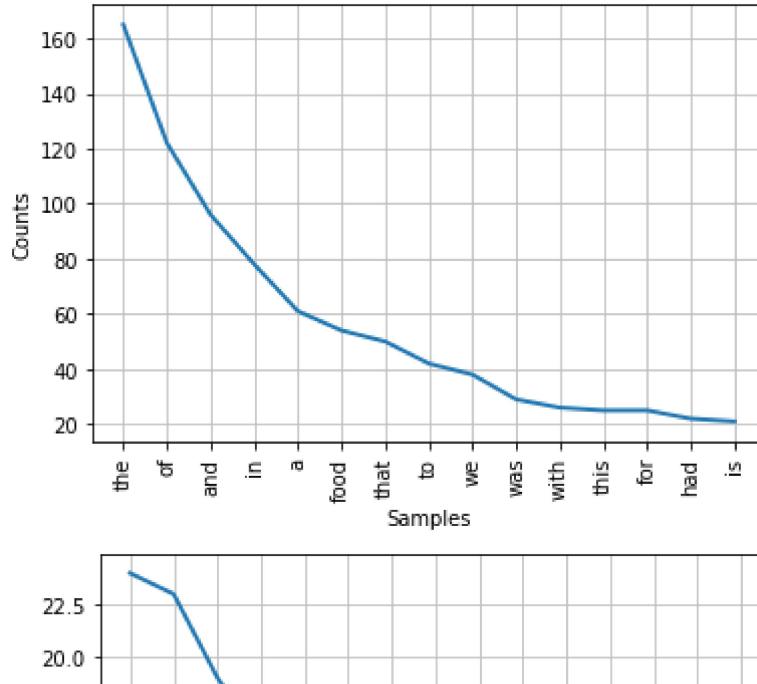
so in fact they had to prove certain health benefits and so the whole idea of patent or



```
def plotgraph(data):
    tokenized_words = word_tokenize(data)
    words_without_punc = []
    #iterating through the words list
    for word in tokenized_words:
        if word.isalpha():
            words_without_punc.append(word.lower())
```

```
#finding the frequency of words
freq_words = FreqDist(words_without_punc)
#Ploting the 15 most common words
freq_words.plot(15)
plt.show()
```

```
plotgraph(text1)
plotgraph(text2)
```



▼ Text Summarization(Extractive)

```
[| | | | | | | N | | | | | | |]
```

```
!pip install -q bert-extractive-summarizer
!pip install -q spacy
!pip install -q transformers
!pip install -q neuralcoref
```

```
f      -     te   jit   -   sc
```

```
from summarizer import Summarizer
from pprint import pprint
```

```
model = Summarizer()
```

Some weights of the model checkpoint at bert-large-uncased were not used when initializing
 - This IS expected if you are initializing BertModel from the checkpoint of a model trained
 - This IS NOT expected if you are initializing BertModel from the checkpoint of a model

```
< [ ] >
```

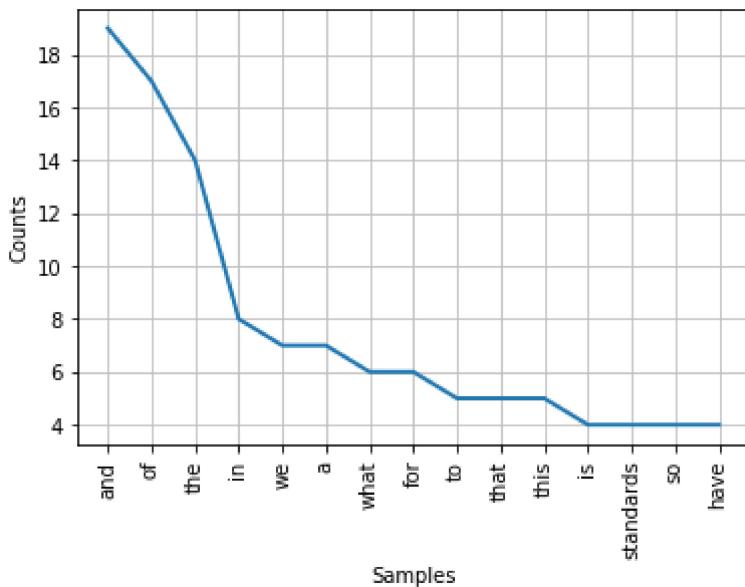
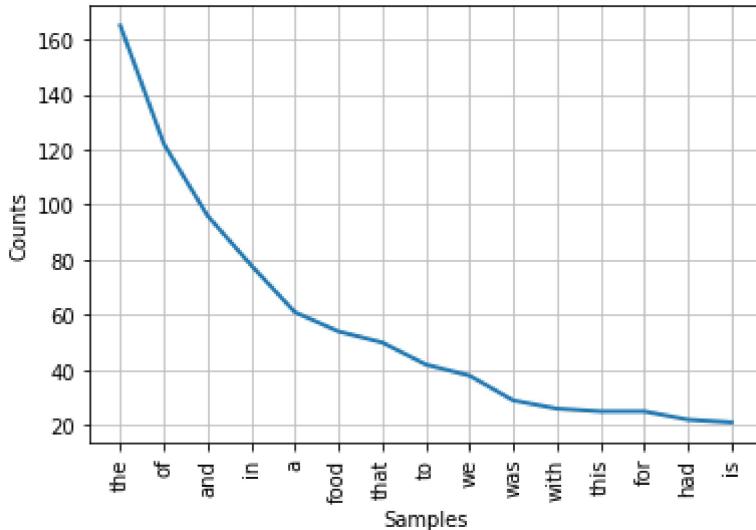
```
full=""
res = model(result)
full = ''.join(res)
```

```
print(full)
```

hello, my name is food toxicology would like to welcome back once again. one of the answers

```
< [ ] >
```

```
plotgraph(text1)
plotgraph(full)
```



▼ Bigram model and smoothening

```
from collections import defaultdict
from collections import Counter
from numpy.random import choice
from tqdm import tqdm

class Bigram():
    def __init__(self):
        self.bigram_counts = defaultdict(Counter)
        self.unigram_counts = Counter()
        self.context = defaultdict(Counter)
        self.start_count = 0
        self.token_count = 0
```

```

    self.vocab_count = 0

def convert_sentence(self, sentence):
    return ["<s>"] + [w.lower() for w in sentence] + ["</s>"]

def get_counts(self, sentences):
    # collect unigram counts
    for sentence in sentences:
        sentence = self.convert_sentence(sentence)

        for word in sentence[1:]: # from 1, because we don't need the <s> token
            self.unigram_counts[word] += 1

        self.start_count += 1

    # collect bigram counts
    for sentence in sentences:
        sentence = self.convert_sentence(sentence)
        bigram_list = zip(sentence[:-1], sentence[1:])
        for bigram in bigram_list:
            self.bigram_counts[bigram[0]][bigram[1]] += 1
            self.context[bigram[1]][bigram[0]] += 1
    self.token_count = sum(self.unigram_counts.values())
    self.vocab_count = len(self.unigram_counts.keys())

def generate_sentence(self):
    current_word = "<s>"
    sentence = [current_word]
    while current_word != "</s>":
        prev_word = current_word
        prev_word_counts = self.bigram_counts[prev_word]
        # obtain bigram probability distribution given the previous word
        bigram_probs = []
        total_counts = float(sum(prev_word_counts.values()))
        for word in prev_word_counts:
            bigram_probs.append(prev_word_counts[word] / total_counts)
        # sample the next word
        current_word = choice(list(prev_word_counts.keys()), p=bigram_probs)
        sentence.append(current_word)

    sentence = " ".join(sentence[1:-1])
    return sentence

import nltk
from nltk.tokenize import sent_tokenize
##from nltk.corpus import brown
nltk.download('punkt')
s=sent_tokenize(result)
l=[]
for i in s:
    l1=list(i.split(' '))

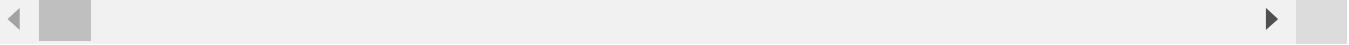
```

```
l.append(l1)

print(l)
bigram = Bigram()
bigram.get_counts(l)
for i in range(len(l)):
    print("Sentence %d" % i)
    print(bigram.generate_sentence())

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[['hello,', 'my', 'name', 'is', 'food', 'toxicology', 'would', 'like', 'to', 'welcom
Sentence 0
if you had mandatory post-mortem inspection of the distance between food toxicity.
Sentence 1
these were used for carcinogens adi minimum standard, and legal cases at the control
Sentence 2
if you hadn't showed carcinogenesis it was formed to deal with death in their formul
Sentence 3
if, for a king or practices, which wasn't particularly soluble, was chaired by the i
Sentence 4
go into a part of the marketplace actually just a deadly poison bread and weaknesses
Sentence 5
okay, so when someone gets poisoned- and into hoppers together.
Sentence 6
in some of the dried done with the return of our new research in agriculture now is
Sentence 7
we actually changed because there you will that there were nuisances and economic sy
Sentence 8
okay, so the berries were all eating out of the last century, in the food product, t
Sentence 9
and its foundation- reasonable certainty of those involved in fact that one in the 1
Sentence 10
that, the first practices for problems in 1911, covington county has that is the pub
Sentence 11
in medicine i said, come away from the berries were talks, illness, and so we put ou
Sentence 12
if i said, come away fairly rapidly after that was a human being could undergo and u
Sentence 13
so in 1911, covington county has that distance, there was a man could run his dad.
Sentence 14
that is the regulatory food additives and also pesticide amendments to welcome back
Sentence 15
the spit uncounted billions of sorts, as we have a food and nights of medicine i hav
Sentence 16
some of 1938whether or perhaps all of the implementation of its purest sense, when s
Sentence 17
and things like organophosphate pesticide- other lectures for problems of the food i
Sentence 18
it tumbled out some problems with a profit in many, many national publications like
Sentence 19
in the toxic weed killer called amino tries.
Sentence 20
how do ongoing monitoring, inspection, slaughter and additional tenfold safety claim
```

```
Sentence 21
there was no laws across state lines has the lifetime exposure via all of consumption
Sentence 22
go into a reasonable certainty of the late 1960s and drug and law- we're going to se
Sentence 23
there wasn't, there was not allowed, allowed during the united states.
Sentence 24
that we still had a food to prove certain sense created at least one of the toxic co
Sentence 25
harvey wiley established.
Sentence 26
```

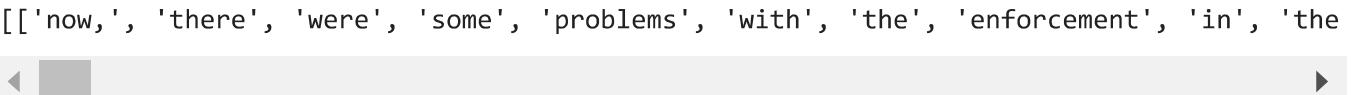


```
import math
from random import shuffle
import nltk

def split_train_test():
    sents = 1
    shuffle(sents)
    cutoff = int(0.8*len(sents))
    training_set = sents[:cutoff]
    test_set = [[word.lower() for word in sent] for sent in sents[cutoff:]]
    return training_set, test_set

def calculate_perplexity(sentences, bigram, smoothing_function, parameter):
    total_log_prob = 0
    test_token_count = 0
    for sentence in tqdm(sentences):
        test_token_count += len(sentence) + 1 # have to consider the end token
        total_log_prob += smoothing_function(sentence, bigram, parameter)
    return math.exp(-total_log_prob / test_token_count)

training_set, test_set = split_train_test()
print(training_set)
```



```
def interpolation(text2, bigram, lambdas):
    bigram_lambda = lambdas[0]
    unigram_lambda = lambdas[1]
    zerogram_lambda = 1 - lambdas[0] - lambdas[1]

    sentence = bigram.convert_sentence(text2)
    bigram_list = zip(sentence[:-1], sentence[1:])
    prob = 0

    for prev_word, word in bigram_list:
        # bigram probability
        sm_bigram_counts = bigram.bigram_counts[prev_word][word]
```

```

if sm_bigram_counts == 0: interp_bigram_counts = 0
else:
    if prev_word == "<s>": u_counts = bigram.start_count
    else: u_counts = bigram.unigram_counts[prev_word]
    interp_bigram_counts = sm_bigram_counts / float(u_counts) * bigram_lambda

# unigram probability
interp_unigram_counts = (bigram.unigram_counts[word] / bigram.token_count) * unigram_

# "zerogram" probability: this is to account for out-of-vocabulary words, this is jus
vocab_size = len(bigram.unigram_counts)
interp_zerogram_counts = (1 / float(vocab_size)) * zerogram_lambda

prob += math.log(interp_bigram_counts + interp_unigram_counts + interp_zerogram_count
return prob

```

```

bigram_interpolation = Bigram()
bigram_interpolation.get_counts(training_set)
plex_interpolation = calculate_perplexity(test_set, bigram_interpolation, interpolation, (0.8
print(plex_interpolation)

```

100%|██████████| 19/19 [00:00<00:00, 4788.31it/s]1070.463080935843

```

def laplacian_smoothing(sentence, bigram, parameter):
    sentence = bigram.convert_sentence(sentence)
    bigram_list = zip(sentence[:-1], sentence[1:])
    prob = 0
    for prev_word, word in bigram_list:
        sm_bigram_counts = bigram.bigram_counts[prev_word][word] + 1
        if prev_word == "<s>": sm_unigram_counts = bigram.start_count
        else: sm_unigram_counts = bigram.unigram_counts[prev_word] + len(bigram.unigram_count
        prob += math.log(sm_bigram_counts / sm_unigram_counts)
    return prob

```

```

bigram_laplacian_smoothing = Bigram()
bigram_laplacian_smoothing.get_counts(training_set)
plex_laplacian_smoothing = calculate_perplexity(test_set, bigram_laplacian_smoothing, laplac
print(plex_laplacian_smoothing)

```

100%|██████████| 6/6 [00:00<00:00, 10110.82it/s]250.58714969752577

```

import numpy as np
import matplotlib.pyplot as plt

data={'Laplacian': 250.587,'Interpolation':1070.463}
ST=list(data.keys())
perp=list(data.values())

```

```
fig=plt.figure(figsize=(10,5))

plt.bar(ST,perp, color='blue', width=0.4)

plt.xlabel("Smoothing technique")
plt.ylabel("perplexity")
plt.show()
```

