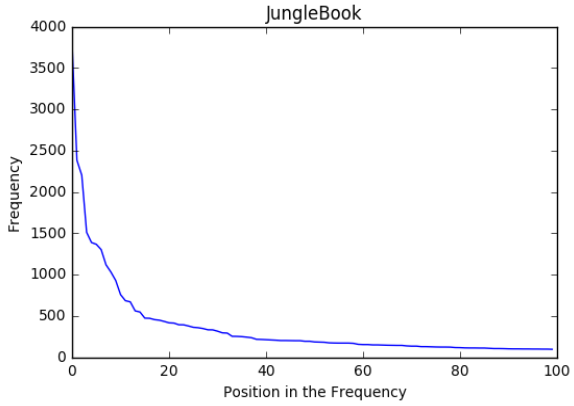
ASSIGNMENT 1

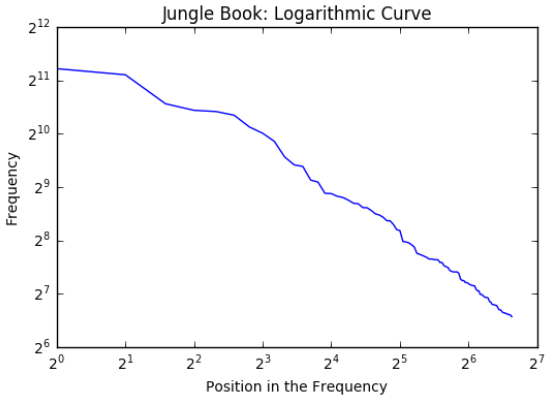
WRITE UP BY HARSHITA JHAVAR (MSc IN Computer Science (Sem 1))
Matriculation number- 2566267

Writeup on Solution for Question 1

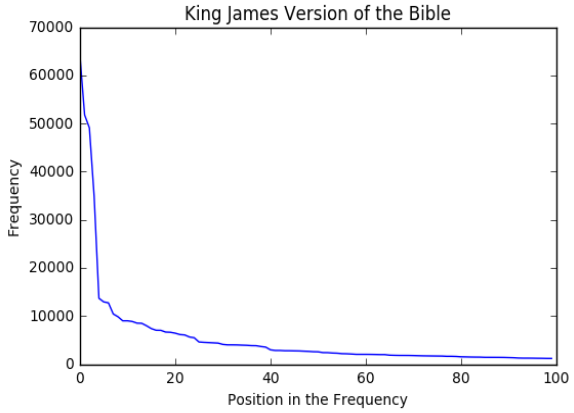
Below are the plots obtained for the three different corpus given in the question. I have plotted only the top 100 results from the Frequency Position Descending Sorted List so that the results look clear. :

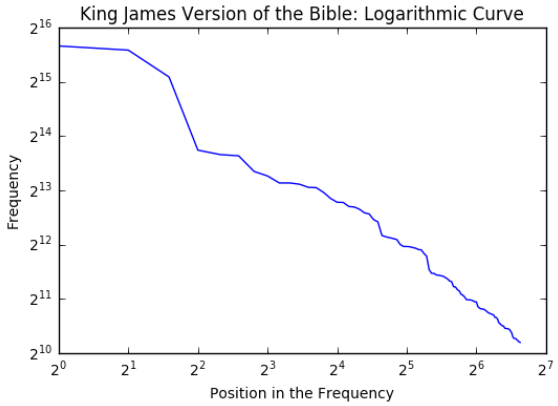
Corpus1: The Jungle Book (Linear Curve and Logarithmic Curve)



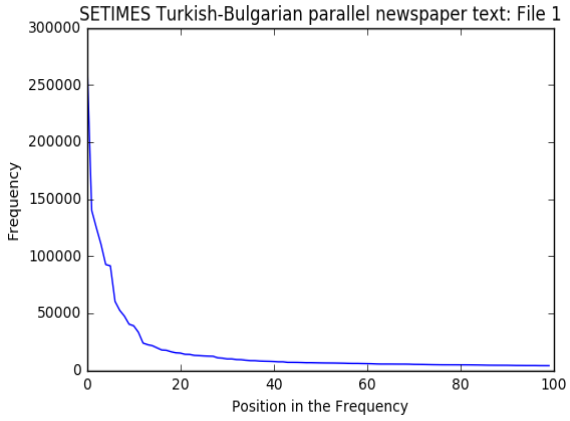


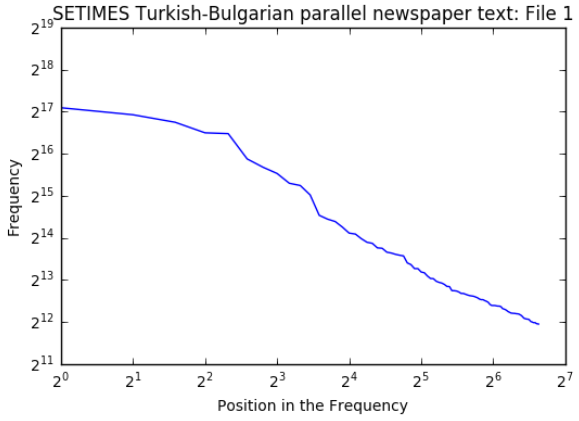
Corpus 2: The King James Version of The Bible (Linear Curve and Logarithmic Curve)

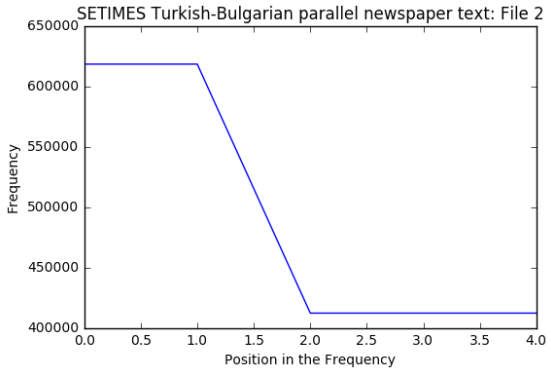


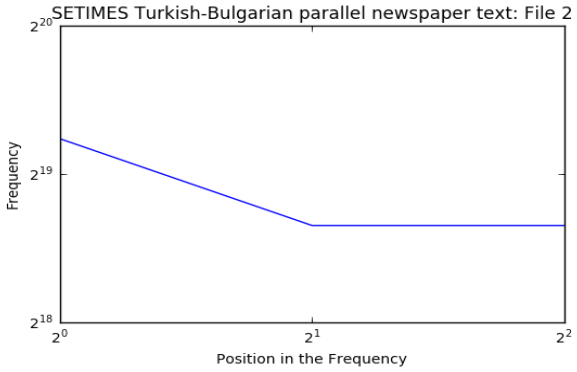


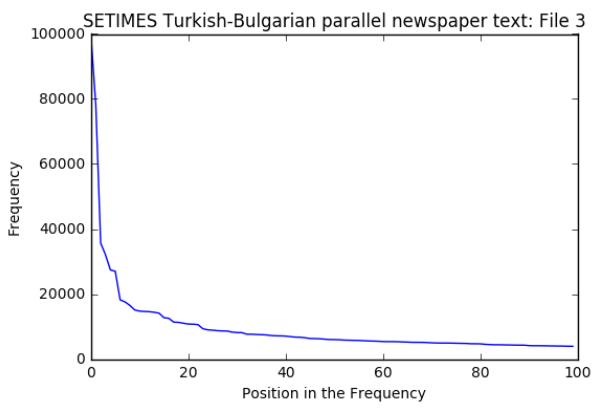
Corpus3: SETIMES Turkish-Bulgarian Parallel News Corpus:
This corpus was a zip file which consisted of three text files. Below are the linear (On the left) and the logarithmic curve (On the right) of the three text files in the corpus.

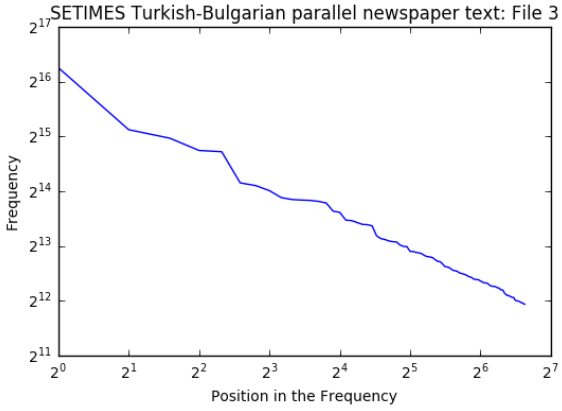












Discussion: It is pretty clear from the above curves that the Zipf's Law is true for all the above curves as the earlier the rank of the word in the Frequency Position Descendingly sorted list, the higher is the frequency.

The source code and the notebook file can be found in the folder Question1.

Writeup on Solution for Question 2

The corpus taken here in the code is the James Bible corpus whose link is mentioned in Question 1. The code below shows the different varieties of texts generated:

For a BiGram Model (n=2)

First Word	
'a'	a serpent p p p p be fulfilled p and the wicked perish p and he needeth p p as for moses according to pass over israel were on thy god and all that scattereth the cities and two ends of all his righteousness p p thy hand of thine own foot from millo p p paul the wheels and confirmation of his kingdom of the hittite and he hath in the god p p and forsook all the pursuers be clean shall set on his garner but he shall come also it p p now the same thing that
'the'	the first and the children for me p so that he hath broken his left in the son p and wept over all the sons specially for him out of jeroboam and smote him the name of god gave him that hath trodden down at that the crown of merari p p then shall fall down many words of the children of silver p p p p p and discretion shall be very many great stand and the lord unworthily eateth any cuttings in order unto whom i said unto anger and i would give you p p p is
'grass'	grass and they thy sight that doeth not your land of the rest to do to bethjeshimoth and discovered cyprus we given and did so aaron came and the lord and the eighteenth year by captivity saying what was with those days the lord he had been ambassadors by little child i sent you life p p he should not crow thou wilt thou shalt set me to their partners which slept with part as for my lord your god p see all the besom of heaven clothed with me and tarried still sacrificed for he said behold the shapes
'blue'	blue and though ye heavens and he shall pharaoh king for righteousness and the lord sent for the almighty for it came to preach saying go down their doings to draw her whelps in the sixth angel but ye have wars of the son had thus spake and all and they had ordained in that she down from blood which thou shalt thou shalt thou shalt thou art come to him is a burnt offering an angry with an angel and all that i have everlasting life for thy kingdom of egypt for he may without p p p and
'bird'	bird and shobab and if now i pray thee thou me but the rain they not that i will render unto his house of the holy ghost and king and ye idolaters as a lie down to the place of this before you shortly the spirit is mine iniquity and to the dead men shall walk in the wagons and mightier than that are the flock and caleb the plains of one generation and he should find good things it true and are wind bloweth where all yea one ordinance both ye have no rain or ah so that are

For a TriGram Model (n=3)

First Two Words	
god remember	god remember me i will cause him to me they pursue thee sith thou hast said i am weary with forbearing and i did bear p p now the lord had shewed him the blood of christ p p and it is better to thee thy beautiful flock what wilt thou canst not bear false witness against thee rebuke him shall glory p p and joshua said why is the feast of unleavened bread seven days and whosoever offereth a burnt offering unto one lamb thou shalt remember all the congregation for all the beasts that perish foolishness but unto cain

egypt and	egypt and in wrath and behold all things p p and it shall accept of the lord said unto him if thou be as god sitteth upon the altar shall be at naioth in ramah even in this mount abarim and pitched in judah and in fury even a god p p having therefore these promises dearly beloved let us come near by their generations after their families were thirteen cities with their eyes were set before you for i have given you the wall above the hills shall flow rivers of waters therefore he lodged there p p furthermore he
another stick	another stick and write it before the inhabitants of the sanctuary according to his father and my statutes which i lifted up his spear and with whom if his father amaziah had done giving him food and one loaf p p if any man draw back unto her fear not have done abominable works there is death p p and if any man shall pass through her belly so shall i more and more also to him as he hath broken thee in the house of togarmah traded in thy word p p and the lord god shewed unto him if
this freedom	this freedom and paul dwelt two whole years in jerusalem p p and he shall direct his steps p p and as soon as the latter days because ye are partakers of the basket of unleavened bread and your mighty sins they shall eat unleavened bread as i was weary p p then joshua called for all the abominations that are left and they laid the pleasant places of the congregation when he had prepared into the tabernacle of the sodomites out of the lord spake unto moses and the captains over thousands of manasseh of valiant men of this project
for horsemen	for horsemen and bowmen they shall say the burden but they which have not known man let me die with the officers shall speak shall come up unto you saying the land of canaan with the angel of the way of the children of israel were forty and two months that i shall see him not but saith unto him in he shall rise and measure the length shall be after him and desired of him that the lord your redeemer the mighty with thy might p p they have read understand agree to indemnify and hold him guiltless that taketh
law judge	law judge thee in righteousness and your earth as brass and in jerusalem that are appointed to sound with cymbals psalteries and harps for the lord god p p and it was restored whole as the lord i pray you bring him before i have nothing to pay tribute until this very thing that ye have not accepted ye might know that i may have right to eat bread p p but their laying await was known unto you at the door of the ephod all of brass his legs and put no fire under p p and he said unto\n"

For a FourGram Model (n=4)

First Three words	
in the end	in the end of his talk is mischievous madness p p a good man some would even dare to die p p hell and destruction are in their ways and the lord appeared unto all the ends of the mercy seat and the vail of the covering and covered the ark of the god of your fathers and i said unto her father let this thing be and he said unto them is he well and they said unto him micaiah shall we go thou hast the males shall be the first month in the ninth year in the month zif which
i will provoke	i will provoke you to jealousy by them that are dead to sin live any longer therein know ye not brethren for i speak to them that are numbered half a shekel after the shekel of the sanctuary both of them full of fine flour mingled with half an hin of oil p p thus saith the lord p p and ahab came into his heart to seek the lord p p turn away thine eyes from

	me for now if i do this great wickedness and sin against god and lettest such words go out of the dungeon and cast a
is my meditation	is my meditation all the day p p because ye were more in number than any people for ye were strangers in the land upon the mountains neither hath lifted up his eyes and looked and behold the rod of his mouth cometh knowledge and understanding p p he hath destroyed me on every side p p he that hath an issue and of him and through him and to say unto the children of judah round about according to all thy lovers have forgotten thee they seek thee not for i will not pity nor spare nor have mercy but destroy
him that soul	him that soul shall be cut off in the sight of the lord your god but if ye refuse and rebel ye shall be gathered together ye shall blow with the trumpets over your burnt offerings and peace offerings p p and david said unto him go thy way shew thyself to the priest to the war against hazael king of syria escaped out of jerusalem that the king enquired of them diligently what time the chest was brought unto the priest with the wave breast and heave shoulder and the wave breast and the heave offering of the sin offering of
the waters fail	the waters fail from the house of our god shall deliver them but they have not known i will make a league with thee but amaziah would not hear saith the lord god of israel hath spoken it p p jesus said take ye away the stone from the well of harod so that the days of my people that dwellest in the clefts of the rock for their thirst and promisedst them that they should not hear unto this day p p honour widows that are widows indeed p p i also will ask you one thing and answer me p

Discussion:

It looks like when n=4, the text generated is much better in quality than text generated for n=2. This means, as the value of n increases, the quality of the generated text will increase. This has to be true because the probability distribution for n=4 will be less variated as compared to that for n=2, so the text will come-up as more quality text directly extracted in lengths of three from the corpus to frame a paragraph of length 100 for n=4.

Source Code: The source code can be found in the folder named Question 2.

```
# coding: utf-8

# In[229]:

import nltk


# In[230]:

from urllib import request


# In[231]:

url1 = "http://www.gutenberg.org/files/10/10-h/10-h.htm" ##Importing the text from the html link
provided in the question
url2 = "http://www.gutenberg.org/files/35997/35997-h/35997-h.htm"
f3_1 = open('/home/harshita/Assignment_1/bg-tr.txt/SETIMES2.bg-tr.bg')
f3_2 = open('/home/harshita/Assignment_1/bg-tr.txt/SETIMES2.bg-tr.ids')
f3_3 = open('/home/harshita/Assignment_1/bg-tr.txt/SETIMES2.bg-tr.tr')


# In[232]:

response1 = request.urlopen(url1)
response2 = request.urlopen(url2)


# In[233]:

raw1 = response1.read().decode('utf8')
raw2 = response2.read().decode('latin-1')

raw3_1 = f3_1.read()
raw3_2 = f3_2.read()
raw3_3 = f3_3.read()


# In[234]:

tokens1 = nltk.wordpunct_tokenize(raw1)
tokens2 = nltk.wordpunct_tokenize(raw2)
tokens3_1 = nltk.wordpunct_tokenize(raw3_1)
tokens3_2 = nltk.wordpunct_tokenize(raw3_2)
tokens3_3 = nltk.wordpunct_tokenize(raw3_3)


# In[235]:

text1 = nltk.Text(tokens1)
text2 = nltk.Text(tokens2)
text3_1 = nltk.Text(tokens3_1)
text3_2 = nltk.Text(tokens3_2)
text3_3 = nltk.Text(tokens3_3)


# In[236]:

words1 = [w.lower() for w in text1 if w.isalpha()]
words2 = [w.lower() for w in text2 if w.isalpha()]
words3_1 = [w.lower() for w in text3_1 if w.isalpha()]
words3_2 = [w.lower() for w in text3_2 if w.isalpha()]
words3_3 = [w.lower() for w in text3_3 if w.isalpha()]


# In[237]:

fdist1 = nltk.FreqDist(words1)
fdist2 = nltk.FreqDist(words2)
fdist3_1 = nltk.FreqDist(words3_1)
fdist3_2 = nltk.FreqDist(words3_2)
```

```
fdist3_3 = nltk.FreqDist(words3_3)


# In[238]:

high_freqs1 = sorted(fdist1.values(), reverse=True) ## Considering only the top 100 greatest fequ
ency of words
high_freqs2 = sorted(fdist2.values(), reverse=True)
high_freqs3_1 = sorted(fdist3_1.values(), reverse=True)
high_freqs3_2 = sorted(fdist3_2.values(), reverse=True)
high_freqs3_3 = sorted(fdist3_3.values(), reverse=True)


# In[239]:

get_ipython().magic('matplotlib inline')
import matplotlib
import matplotlib.pyplot as plt
plt.plot(high_freqs1)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('King James Version of the Bible')


# In[240]:

import matplotlib.pyplot as plt
plt.loglog(high_freqs1, basex=2, basey=2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('King James Version of the Bible: Logarithmic Curve')
plt.show()


# In[241]:

get_ipython().magic('matplotlib inline')
import matplotlib
import matplotlib.pyplot as plt
plt.plot(high_freqs2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('JungleBook')


# In[242]:

import matplotlib.pyplot as plt
plt.loglog(high_freqs2, basex=2, basey=2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('Jungle Book: Logarithmic Curve')
plt.show()


# In[243]:

get_ipython().magic('matplotlib inline')
import matplotlib
import matplotlib.pyplot as plt
plt.plot(high_freqs3_1)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 1')


# In[244]:

import matplotlib.pyplot as plt
plt.loglog(high_freqs3_1, basex=2, basey=2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
```

```
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 1')
plt.show()


# In[245]:

get_ipython().magic('matplotlib inline')
import matplotlib
import matplotlib.pyplot as plt
plt.plot(high_freqs3_2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 2')


# In[246]:

import matplotlib.pyplot as plt
plt.loglog(high_freqs3_2, basex=2, basey=2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 2')
plt.show()


# In[247]:

get_ipython().magic('matplotlib inline')
import matplotlib
import matplotlib.pyplot as plt
plt.plot(high_freqs3_3)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 3')


# In[248]:

import matplotlib.pyplot as plt
plt.loglog(high_freqs3_3, basex=2, basey=2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 3')
plt.show()
```

```
# coding: utf-8

# In[207]:

import nltk


# In[208]:

from urllib import request

## Importing the text from the html link provided in the question
# In[209]:

url1 = "http://www.gutenberg.org/files/10/10-h/10-h.htm"
url2 = "http://www.gutenberg.org/files/35997/35997-h/35997-h.htm"

## Reading the files after unzipping the third corpus from my hard disk
f3_1 = open('/home/harshita/Assignment_1/bg-tr.txt/SETIMES2.bg-tr.bg')
f3_2 = open('/home/harshita/Assignment_1/bg-tr.txt/SETIMES2.bg-tr.ids')
f3_3 = open('/home/harshita/Assignment_1/bg-tr.txt/SETIMES2.bg-tr.tr')


# In[210]:

response1 = request.urlopen(url1)
response2 = request.urlopen(url2)

##Reading the text
# In[211]:

raw1 = response1.read().decode('utf8')
raw2 = response2.read().decode('latin-1')

raw3_1 = f3_1.read()
raw3_2 = f3_2.read()
raw3_3 = f3_3.read()


# In[212]:
##Removing the punctuations
tokens1 = nltk.wordpunct_tokenize(raw1)
tokens2 = nltk.wordpunct_tokenize(raw2)
tokens3_1 = nltk.wordpunct_tokenize(raw3_1)
tokens3_2 = nltk.wordpunct_tokenize(raw3_2)
tokens3_3 = nltk.wordpunct_tokenize(raw3_3)


# In[213]:
##Tokenizationation of the words
text1 = nltk.Text(tokens1)
text2 = nltk.Text(tokens2)
text3_1 = nltk.Text(tokens3_1)
text3_2 = nltk.Text(tokens3_2)
text3_3 = nltk.Text(tokens3_3)


# In[214]:

words1 = [w.lower() for w in text1 if w.isalpha()]
words2 = [w.lower() for w in text2 if w.isalpha()]
words3_1 = [w.lower() for w in text3_1 if w.isalpha()]
words3_2 = [w.lower() for w in text3_2 if w.isalpha()]
words3_3 = [w.lower() for w in text3_3 if w.isalpha()]

#3 Calculating the frequency for different words and sorting them.
## Considering only the top 100 greatest fequency of words to get a better plot
# In[215]:

fdist1 = nltk.FreqDist(words1)
fdist2 = nltk.FreqDist(words2)
```

```
fdist3_1 = nltk.FreqDist(words3_1)
fdist3_2 = nltk.FreqDist(words3_2)
fdist3_3 = nltk.FreqDist(words3_3)
```

```
# In[216]:

high_freqs1 = sorted(fdist1.values(), reverse=True)[:100]
high_freqs2 = sorted(fdist2.values(), reverse=True)[:100]
high_freqs3_1 = sorted(fdist3_1.values(), reverse=True)[:100]
high_freqs3_2 = sorted(fdist3_2.values(), reverse=True)[:100]
high_freqs3_3 = sorted(fdist3_3.values(), reverse=True)[:100]

Plotting the linear and the logarithmic curves
# In[217]:

get_ipython().magic('matplotlib inline')
import matplotlib
import matplotlib.pyplot as plt
plt.plot(high_freqs1)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('King James Version of the Bible')


# In[218]:

import matplotlib.pyplot as plt
plt.loglog(high_freqs1, basex=2, basey=2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('King James Version of the Bible: Logarithmic Curve')
plt.show()


# In[219]:

get_ipython().magic('matplotlib inline')
import matplotlib
import matplotlib.pyplot as plt
plt.plot(high_freqs2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('JungleBook')


# In[220]:

import matplotlib.pyplot as plt
plt.loglog(high_freqs2, basex=2, basey=2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('Jungle Book: Logarithmic Curve')
plt.show()


# In[221]:

get_ipython().magic('matplotlib inline')
import matplotlib
import matplotlib.pyplot as plt
plt.plot(high_freqs3_1)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 1')


# In[222]:

import matplotlib.pyplot as plt
plt.loglog(high_freqs3_1, basex=2, basey=2)
plt.xlabel('Position in the Frequency')
```

```
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 1')
plt.show()


# In[223]:

get_ipython().magic('matplotlib inline')
import matplotlib
import matplotlib.pyplot as plt
plt.plot(high_freqs3_2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 2')


# In[224]:

import matplotlib.pyplot as plt
plt.loglog(high_freqs3_2, basex=2, basey=2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 2')
plt.show()


# In[225]:

get_ipython().magic('matplotlib inline')
import matplotlib
import matplotlib.pyplot as plt
plt.plot(high_freqs3_3)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 3')


# In[226]:

import matplotlib.pyplot as plt
plt.loglog(high_freqs3_3, basex=2, basey=2)
plt.xlabel('Position in the Frequency')
plt.ylabel('Frequency')
plt.title('SETIMES Turkish-Bulgarian parallel newspaper text: File 3')
plt.show()

## Discussion about this question can be read from the pdf file available in the solution folder
```

```
# coding: utf-8

# In[107]:

from nltk.probability import (FreqDist, ConditionalFreqDist, ConditionalProbDist, MLEProbDist, Si
mpleGoodTuringProbDist)
from nltk.util import ngrams


def ml_estimator(freqdist):
    return MLEProbDist(freqdist)

def goodturing_estimator(freqdist):
    return SimpleGoodTuringProbDist(freqdist)

class BasicNgram(ConditionalProbDist):
    """
    Define and train an Ngram Model over the corpus represented by the list words.
    Given an BasicNgram instance ngram and a (n-1)-gram context (i.e., a tuple of n-1 strings),
    a call to ngram[context] returns a nltk.probability.ProbDistI object representing the Probabi
lity distribution P(.|context) over possible values for the next word.
    Be aware that context has to be a tuple, even if context is a unigram (see example below)
     >>> corpus=['a','b','b','a']
    >>> bigram=BasicNgram(2,corpus)
    >>> bigram.contexts()
    [('<$>',), ('a',), ('b',)]
    >>> p_b=bigram[('b',)] #not bigram['b']!!!
    >>> p_b.prob('a')
    0.5
    >>> p_b.prob('b')
    0.5


    :param n: the dimension of the n-grams (i.e. the size of the context+1).
    :type n: int
    :param corpus:
    :type corpus: list(Str)

    other parameters are optional and may be omitted. They define whether to add artificial symbo
ls before or after the word list,
    and whether to use another estimation methods than maximum likelihood.
    """
    def __init__(self, n, words, start_symbol="<$>", end_symbol="</$>", pad_left=True, pad_right=
False, estimator=ml_estimator):
        assert (n > 0)
        self._n=n
        self._words=words
        self._counter=ConditionalFreqDist()
        self._start_symbol=start_symbol
        self._end_symbol=end_symbol
        self._pad_left=pad_left
        self._pad_right=pad_right
        self._train()
        super().__init__(self._counter, estimator)


    def _train(self):
        _ngrams=self.generate_ngrams()
        for ngram in _ngrams:
            context=ngram[0:-1]
            outcome=ngram[-1]
            self._counter[context][outcome]+=1

    """
    returns an iterable over the ngrams of the word corpus
    """
    def generate_ngrams(self):
        return ngrams(self._words, self._n, self._pad_left, self._pad_right,
                      left_pad_symbol=self._start_symbol,
                      right_pad_symbol=self._end_symbol)
```

```
    """


    Return the list of contexts


    """
    def contexts(self):
        return list(self.conditions())


## My solution begins here. Above, we had just imported the n-gram class given by Antoine.
# In[108]:

import nltk
from urllib import request

## Reading the text from the given URL to the html file
# In[109]:

url1 = "http://www.gutenberg.org/files/10/10-h/10-h.htm"
response1 = request.urlopen(url1)
raw1 = response1.read().decode('utf8')

##Removing the punctuations and tokenizing only the words
tokens1 = nltk.wordpunct_tokenize(raw1)
text1 = nltk.Text(tokens1)
words1 = [w.lower() for w in text1 if w.isalpha()]
words1


# In[110]:

## Creating instance of ngram class with n =2
corpus = words1
bigram=BasicNgram(2,corpus)
bigram.contexts()

## A loop for creating a text length of size = 100. This I have ran 5 times with different starti
ng words for different values of n whose information is given in the pdf file found in the soluti
on folder
# In[111]:

previous_word = 'a'
sample = previous_word
for i in range(1, 100):
    probs = bigram[(previous_word,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word = next_word
print(sample)


# In[112]:

previous_word = 'the'
sample = previous_word
for i in range(1, 100):
    probs = bigram[(previous_word,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word = next_word
print(sample)


# In[113]:

previous_word = 'grass'
```

```
sample = previous_word
for i in range(1, 100):
    probs = bigram[(previous_word,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word = next_word
print(sample)


# In[114]:

previous_word = 'blue'
sample = previous_word
for i in range(1, 100):
    probs = bigram[(previous_word,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word = next_word
print(sample)


# In[115]:

previous_word = 'bird'
sample = previous_word
for i in range(1, 100):
    probs = bigram[(previous_word,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word = next_word
print(sample)


# In[116]:

## Creating instance of ngram class with n = 3
corpus = words1
trigram=BasicNgram(3,corpus)
trigram.contexts()


# In[117]:

previous_word1 = 'god'
previous_word2 = 'remember'
sample = previous_word1 + ' '+previous_word2
for i in range(1, 100):
    probs = trigram[(previous_word1,previous_word2,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
    previous_word2 = next_word
print(sample)


# In[118]:

previous_word1 = 'egypt'
previous_word2 = 'and'
sample = previous_word1 + ' '+previous_word2
for i in range(1, 100):
    probs = trigram[(previous_word1,previous_word2,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
    previous_word2 = next_word
print(sample)


# In[119]:

previous_word1 = 'another'
```

```
previous_word2 = 'stick'
sample = previous_word1 + ' '+previous_word2
for i in range(1, 100):
    probs = trigram[(previous_word1,previous_word2,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
    previous_word2 = next_word
print(sample)


# In[120]:

previous_word1 = 'this'
previous_word2 = 'freedom'
sample = previous_word1 + ' '+previous_word2
for i in range(1, 100):
    probs = trigram[(previous_word1,previous_word2,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
    previous_word2 = next_word
print(sample)


# In[121]:

previous_word1 = 'for'
previous_word2 = 'horsemen'
sample = previous_word1 + ' '+previous_word2
for i in range(1, 100):
    probs = trigram[(previous_word1,previous_word2,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
    previous_word2 = next_word
print(sample)


# In[122]:

previous_word1 = 'law'
previous_word2 = 'judge'
sample = previous_word1 + ' '+previous_word2
for i in range(1, 100):
    probs = trigram[(previous_word1,previous_word2,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
    previous_word2 = next_word
print(sample)


# In[123]:

## Creating instance of ngram class with n =4
corpus = words1
fourgram=BasicNgram(4,corpus)
fourgram.contexts()


# In[124]:

previous_word1 = 'in'
previous_word2 = 'the'
previous_word3 = 'end'
sample = previous_word1 + ' '+previous_word2+' '+previous_word3
for i in range(1, 100):
    probs = fourgram[(previous_word1,previous_word2,previous_word3,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
```

```
    previous_word2 = previous_word3
    previous_word3 = next_word
print(sample)


# In[125]:

previous_word1 = 'i'
previous_word2 = 'will'
previous_word3 = 'provoke'
sample = previous_word1 + ' '+previous_word2+' '+previous_word3
for i in range(1, 100):
    probs = fourgram[(previous_word1,previous_word2,previous_word3,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
    previous_word2 = previous_word3
    previous_word3 = next_word
print(sample)


# In[126]:

previous_word1 = 'is'
previous_word2 = 'my'
previous_word3 = 'meditation'
sample = previous_word1 + ' '+previous_word2+' '+previous_word3
for i in range(1, 100):
    probs = fourgram[(previous_word1,previous_word2,previous_word3,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
    previous_word2 = previous_word3
    previous_word3 = next_word
print(sample)


# In[127]:

previous_word1 = 'him'
previous_word2 = 'that'
previous_word3 = 'soul'
sample = previous_word1 + ' '+previous_word2+' '+previous_word3
for i in range(1, 100):
    probs = fourgram[(previous_word1,previous_word2,previous_word3,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
    previous_word2 = previous_word3
    previous_word3 = next_word
print(sample)


# In[128]:

previous_word1 = 'the'
previous_word2 = 'waters'
previous_word3 = 'fail'
sample = previous_word1 + ' '+previous_word2+' '+previous_word3
for i in range(1, 100):
    probs = fourgram[(previous_word1,previous_word2,previous_word3,)]
    next_word = probs.generate()
    sample = sample + ' ' + next_word
    previous_word1 = previous_word2
    previous_word2 = previous_word3
    previous_word3 = next_word
print(sample)

## Discussion on this can be read in the pdf file found in the solution folder
```