Experiment 3

NAME	**Shreya Shetty**
UID	**2019140059**
CLASS	**TE IT**
BATCH	**B**
SUBJECT	**NLP Lab**

AIM:

1. Calculate bigrams from a given corpus and calculate probability of a sentence.
2. To apply add-one smoothing on sparse bigram table

THEORY:

A combination of words forms a sentence. However, such a formation is meaningful only when the words are arranged in some order.

Eg: Sit I car in the

Such a sentence is not grammatically acceptable. However some perfectly grammatical sentences can be nonsensical too!

Eg: Colorless green ideas sleep furiously

One easy way to handle such unacceptable sentences is by assigning probabilities to the strings of words i.e, how likely the sentence is.

Probability of a sentence can be calculated by the probability of the sequence of words occurring in it. We can use Markov assumption that the probability of a word in a sentence depends on the probability of the word occurring just before it. Such a model is called the first order Markov model or the bigram model.

$$P(W_n | W_{n-1}) = P(W_{n-1}, W_n)/P(W_{n-1})$$

Here, Wn refers to the word token corresponding to the nth word in a sequence.

Probability of a sentence:

If we consider each word occurring in its correct location as an independent event,the probability of the sentences is : P(w(1), w(2)..., w(n-1), w(n))

Using chain rule: = P(w(1)) * P(w(2) | w(1)) * P(w(3) | w(1)w(2)) ... P(w(n) | w(1)w(2) ... w(n-1))

Bigrams:

We can avoid this very long calculation by approximating that the probability of a given word depends only on the probability of its previous words. This assumption is called Markov assumption and such a model is called Markov model- bigrams. Bigrams can be generalized to the n-gram which looks at (n-1) words in the past. A bigram is a first-order Markov model.

Therefore , P(w(1), w(2)..., w(n-1), w(n)) = P(w(2)|w(1)) P(w(3)|w(2)) ... P(w(n)|w(n-1)) We use the (eos) tag to mark the beginning and end of a sentence.

A bigram table for a given corpus can be generated and used as a lookup table for calculating probability of sentences.

Eg: Corpus - (eos) You book a flight (eos) I read a book (eos) You read (eos)

Bigram Table:

	(eos)	you	book	a	flight	I	read
(eos)	0	0.33	0	0	0	0.25	0
you	0	0	0.5	0	0	0	0.5
book	0.5	0	0	0.5	0	0	0
a	0	0	0.5	0	0.5	0	0
flight	1	0	0	0	0	0	0
I	0	0	0	0	0	0	1
read	0.5	0	0	0.5	0	0	0

P((eos) you read a book (eos))
= P(you|eos) * P(read|you) * P(a|read) * P(book|a) * P(eos|book)
= 0.33 * 0.5 * 0.5 * 0.5 * 0.5
= 0.020625

N-Grams smoothing:

One major problem with standard N-gram models is that they must be trained from some corpus, and because any particular training corpus is finite, some perfectly acceptable N-grams are bound to be missing from it.

We can see that the bigram matrix for any given training corpus is sparse. There are a large number of cases with zero probability bigrams and that should really have some non-zero probability. This method tends to underestimate the probability of strings that happen not to have occurred nearby in their training corpus.

There are some techniques that can be used for assigning a non-zero probability to these 'zero probability bigrams'. This task of reevaluating some of the zero-probability and low-probability N-grams, and assigning them non-zero values, is called smoothing.

	eos	I	booked	a	flight	took
eos	0	300	0	0	0	300
I	0	0	300	0	0	0
booked	0	0	0	300	0	0
a	0	0	0	0	600	0
flight	600	0	0	0	0	0
took	0	0	0	300	0	0

Valid bigrams absent in the training corpus:
How could I eos I have a booked room eos I took a flight eos

Add-One Smoothing:

In Add-One smoothing, we add one to all the bigram counts before normalizing them into probabilities. This is called add-one smoothing.

Application on unigrams:

The unsmoothed maximum likelihood estimate of the unigram probability can be computed by dividing the count of the word by the total number of word tokens N.

$$P(wx) = c(wx)/sumi\{c(wi)\} = c(wx)/N$$

Let there be an adjusted count c.

$$ci = (c\ i+1 * N/(N+V))$$

where V is the total number of word types in the language.
Now, probabilities can be calculated by normalizing counts by N.

$$pi^* = (c\ i+1)/(N+V)$$

Application on bigrams:

Normal bigram probabilities are computed by normalizing each row of counts by the unigram count:

$$P(wn|wn-1) = C(wn-1wn)/C(wn-1)$$

For add-one smoothed bigram counts we need to augment the unigram count by the number of total word types in the vocabulary V:

$$p^*(wn|wn-1) = (\ C(wn-1wn)+1\ )/(\ C(wn-1)+V\ )$$

IDE USED: Jupyter Notebook

LIBRARIES USED:

Nltk:

The Natural Language Toolkit (NLTK) is a Python package for natural language processing. NLTK requires Python 3.7, 3.8, 3.9 or 3.10. It can be installed as :

pip install nltk

Counter:

A counter is a sub-class of the dictionary. It is used to keep the count of the elements in an iterable in the form of an unordered dictionary where the key represents the element in the iterable and value represents the count of that element in the iterable.

PROCEDURE:

1. Firstly import the required libraries
2. Perform tokenization of sample data and then remove the stop words
3. Calculate number of unigrams i.e. number of unique words in the corpus
4. Generate bigrams and Calculate the number of bigrams
5. Calculate the probability of each Bigram
6. Input a sentence and perform tokenisation, stop word removal and generate bigrams
7. Generate Bigram table and Probability Table
8. Calculate sentence probability
9. Perform add one smoothing and print the tables

CODE:

```
# Importing Libraries
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from collections import Counter

# Sample data
text_data = "The Vampire Diaries is an American supernatural horror romance television series created by Ke
# Tokenization
tokenized = word_tokenize(text_data)
print("Tokenized Text :\n",tokenized)
```

```
# Displaying stop words
stop_words = set (stopwords.words('english'))
print("Stop Words List :\n",stop_words)
```

```
# Stop word Removal
filtered_words=[]
for w in tokenized:
    if w not in stop_words and len(w) >= 2:
        filtered_words.append(w.lower())
print('Filtered Text :\n', filtered_words)
```

```
# Calculating number of unigrams i.e. number of unique words in the corpus
count_unigrams = Counter(filtered_words)
print("Frequency of Unigrams :\n",count_unigrams)

# Generation of bigrams
bigrams = [(filtered_words[i], filtered_words [i + 1]) for i in range(len(filtered_words) - 1)]
print("All Possible Bigrams List :\n",bigrams)
```

```
# Calculating the number of bigrams
count_bigrams = Counter(bigrams)
print("Frequency of Bigrams :\n",count_bigrams)
```

```
# Calculating the probab. of each Bigram
bigram_probability = {}
for bigram in bigrams:
    word1 = bigram[0]
    Word2 = bigram[1]
    bigram_probability[bigram] = count_bigrams[bigram] / count_unigrams[word1]
print("Probability of each bigram : \n",bigram_probability)
```

```
# Input Sentence
input_sent = 'Nina Dobrev portrayed Elena'
# Tokenisation of input sentence
tokenized_in = word_tokenize(input_sent)
print("Tokenised Input Sentence :\n",tokenized_in)
# Removal of stop words from input sentence
filtered_sent = [w.lower() for w in tokenized_in if w not in stop_words and len(w) >= 2]
# Printing filtered input sentence
print("\nFiltered Input Sentence :\n",filtered_sent)
```

```
# Generating Bigrams from input sentence
print("The bigrams in given sentence are :")
bigrams_input = [(filtered_sent[i], filtered_sent[i + 1]) for i in range(len(filtered_sent) - 1)]
print(bigrams_input)

# Generating Bigram table and Probability Table
bigram_table = [['']  + [i for i in filtered_sent]]
probability_table = [['']  + [i for i in filtered_sent]]
for w1 in filtered_sent:
    row1 = [w1]
    row2 = [w1]
    for w2 in filtered_sent:
        row1.append(str(count_bigrams [(w1, w2)]))
        row2.append(str(count_bigrams[(w1, w2)] / count_unigrams[w1]))
    bigram_table.append(row1)
    probability_table.append(row2)
```

```
 # Printing the Probability Table
 print("PROBABILITY TABLE :\n")
 for row in probability_table:
     for i in row:
         print(i.center(25, ' '), end='')
     print()

 # Printing the Bigram Table
 print("\nBIGRAM TABLE :\n")
 for row in bigram_table:
     for i in row:
         print(i.center(25, ' '), end='')
     print()
```

```
# Calculating the given sentence probability
sent_prob = 1
for bigram in bigrams_input:
    sent_prob = sent_prob * count_bigrams[bigram] / count_unigrams[bigram[0]]
# Printing Sentence Probability
print("Sentence Probability :",sent_prob)
```

```
# Add-one smoothing
nrows = len(bigram_table)
ncols = len(bigram_table[0])
V = len(count_unigrams)
for i in range(1, nrows):
    for j in range(1, ncols):
        bigram_table[i][j] = str(int(bigram_table[i][j]) + 1)
        probability_table[i][j] = str(int(bigram_table[i][j]) / (count_unigrams[bigram_table[i][0]] + V))

print("********************************** ADD ONE SMOOTHING **********************************\n")
# Printing the Probability Table
print("PROBABILITY TABLE :\n")
for row in probability_table:
    for i in row:
        print(i.center(25, ' '), end='')
    print()

print("\n\nBIGRAM TABLE :\n")
# Printing the Bigram Table
for row in bigram_table:
    for i in row:
        print(i.center(25,' '), end='')
    print()
```

OUTPUT:

```
Tokenized Text :
 ['The', 'Vampire', 'Diaries', 'is', 'an', 'American', 'supernatural', 'horror', 'romance', 'television', 'series', 'created', 'by', 'Kevi
n', 'Williamson', 'based', 'on', 'the', 'novels', 'of', 'the', 'same', 'name', 'by', 'author', 'L.', 'J.', 'Smith', 'centered', 'aroung',
'Elena', 'Gilbert', '.', 'It', 'was', 'officially', 'picked', 'up', 'for', 'the', '2009–10', 'season', 'on', 'May', '19', ',', '2009',
'.', 'It', 'premiered', 'on', 'September', '10', ',', '2009', ',', 'on', 'The', 'CW', 'and', 'received', 'the', 'highest', 'ratings', 'fo
r', 'a', 'series', 'premiere', 'in', 'the', 'network', "'s", 'history', 'at', 'that', 'point', ',', 'scoring', '4.91', 'million', 'live',
'viewers.The', 'series', 'focuses', 'on', 'the', 'fictional', 'town', 'of', 'Mystic', 'Falls', ',', 'Virginia', ',', 'that', 'is', 'charge
d', 'with', 'supernatural', 'history', '.', 'It', 'follows', 'Elena', 'Gilbert', 'portrayed', 'by', 'Nina', 'Dobrev', 'as', 'she', 'begin
s', 'to', 'get', 'over', 'her', 'parents', "'", 'death', ',', 'when', 'two', 'vampires', 'pull', 'her', 'into', 'a', 'world', 'she', 'di
d', "n't", 'know', 'before', ',', 'Stefan', 'and', 'Damon', 'Salvatore', ',', 'portrayed', 'by', 'Paul', 'Wesley', 'and', 'Ian', 'Somerhal
der', ',', 'respectively', '.', 'Kayla', 'Ewell', 'portrayed', 'Vicki', 'Donovan', 'for', 'the', 'first', 'seven', 'episodes', 'until', 'h
er', 'character', 'was', 'killed', 'off', '.', 'Matt', 'Davis', 'was', 'later', 'cast', 'as', 'a', 'history', 'teacher', 'in', 'a', 'recur
ring', 'role', 'to', 'fill', 'the', 'void', '.', 'He', 'was', 'later', 'upgraded', 'to', 'series', 'regular', 'status', '.', 'Nina', 'Dobr
ev', 'portrayed', 'the', 'Elena', 'Gilbert', 'as', 'well', 'as', 'Katherine', 'Pierce']

Stop Words List :
 {'which', 'than', 'then', 'its', 'hasn', 'she', 'were', 'been', 'no', 'just', 'such', 'd', 'too', 'it', 'their', 'couldn', 'you', 'that',
'doing', 'them', 'did', 'my', 'in', 'there', 'he', 'until', 'am', 'yourselves', 'over', 'ourselves', 'weren', 'should', 'between', "shoul
d've", 'now', 'how', "don't", "you're", "couldn't", 'those', 've', "doesn't", 'but', 'o', 're', 'more', 'me', 'the', 'what', "mightn't",
'be', 'where', 'wouldn', 'mightn', 'him', 'his', "it's", 'themselves', 'mustn', 'they', 'aren', 'will', 'are', 'through', 'having', 'furth
er', 'before', "wasn't", 'and', 'some', 't', 'a', 'yours', "you'll", 'why', 'against', 'own', 'isn', 'very', "you've", 'doesn', 'from', 'b
oth', 'itself', 'ma', "wouldn't", "shouldn't", 'most', 'is', 'her', "won't", 's', 'other', 'off', "aren't", 'for', 'being', 'or', 'this',
'has', 'hers', 'not', 'as', 'll', 'does', 'won', 'i', "isn't", 'few', 'with', 'had', 'on', "hasn't", "you'd", 'above', 'm', 'your', "tha
t'll", 'of', 'up', 'during', 'theirs', 'same', 'to', 'nor', 'under', "mustn't", 'was', 'at', "weren't", 'do', 'any', 'myself', 'so', 'thes
e', "didn't", "hadn't", 'when', 'didn', 'only', 'after', 'have', 'ours', 'down', "haven't", 'whom', 'our', 'all', 'ain', 'haven', 'y', 'ag
ain', 'while', 'out', 'because', 'by', 'each', 'shouldn', 'can', 'we', "needn't", 'hadn', 'wasn', 'yourself', 'below', 'needn', 'don', "sh
e's", 'into', 'an', 'if', 'shan', 'herself', 'about', "shan't", 'himself', 'once', 'who', 'here'}

Filtered Text :
 ['the', 'vampire', 'diaries', 'american', 'supernatural', 'horror', 'romance', 'television', 'series', 'created', 'kevin', 'williamson',
'based', 'novels', 'name', 'author', 'l.', 'j.', 'smith', 'centered', 'aroung', 'elena', 'gilbert', 'it', 'officially', 'picked', '2009–1
0', 'season', 'may', '19', '2009', 'it', 'premiered', 'september', '10', '2009', 'the', 'cw', 'received', 'highest', 'ratings', 'series',
'premiere', 'network', "'s", 'history', 'point', 'scoring', '4.91', 'million', 'live', 'viewers.the', 'series', 'focuses', 'fictional', 't
own', 'mystic', 'falls', 'virginia', 'charged', 'supernatural', 'history', 'it', 'follows', 'elena', 'gilbert', 'portrayed', 'nina', 'dobr
ev', 'begins', 'get', 'parents', 'death', 'two', 'vampires', 'pull', 'world', "n't", 'know', 'stefan', 'damon', 'salvatore', 'portrayed',
'paul', 'wesley', 'ian', 'somerhalder', 'respectively', 'kayla', 'ewell', 'portrayed', 'vicki', 'donovan', 'first', 'seven', 'episodes',
'character', 'killed', 'matt', 'davis', 'later', 'cast', 'history', 'teacher', 'recurring', 'role', 'fill', 'void', 'he', 'later', 'upgrad
ed', 'series', 'regular', 'status', 'nina', 'dobrev', 'portrayed', 'elena', 'gilbert', 'well', 'katherine', 'pierce']


Frequency of Unigrams :
 Counter({'series': 4, 'portrayed': 4, 'elena': 3, 'gilbert': 3, 'it': 3, 'history': 3, 'the': 2, 'supernatural': 2, '2009': 2, 'nina': 2,
'dobrev': 2, 'later': 2, 'vampire': 1, 'diaries': 1, 'american': 1, 'horror': 1, 'romance': 1, 'television': 1, 'created': 1, 'kevin': 1,
'williamson': 1, 'based': 1, 'novels': 1, 'name': 1, 'author': 1, 'l.': 1, 'j.': 1, 'smith': 1, 'centered': 1, 'aroung': 1, 'officially':
1, 'picked': 1, '2009–10': 1, 'season': 1, 'may': 1, '19': 1, 'premiered': 1, 'september': 1, '10': 1, 'cw': 1, 'received': 1, 'highest':
1, 'ratings': 1, 'premiere': 1, 'network': 1, "'s": 1, 'point': 1, 'scoring': 1, '4.91': 1, 'million': 1, 'live': 1, 'viewers.the': 1, 'fo
cuses': 1, 'fictional': 1, 'town': 1, 'mystic': 1, 'falls': 1, 'virginia': 1, 'charged': 1, 'follows': 1, 'begins': 1, 'get': 1, 'parent
s': 1, 'death': 1, 'two': 1, 'vampires': 1, 'pull': 1, 'world': 1, "n't": 1, 'know': 1, 'stefan': 1, 'damon': 1, 'salvatore': 1, 'paul':
1, 'wesley': 1, 'ian': 1, 'somerhalder': 1, 'respectively': 1, 'kayla': 1, 'ewell': 1, 'vicki': 1, 'donovan': 1, 'first': 1, 'seven': 1,
'episodes': 1, 'character': 1, 'killed': 1, 'matt': 1, 'davis': 1, 'cast': 1, 'teacher': 1, 'recurring': 1, 'role': 1, 'fill': 1, 'void':
1, 'he': 1, 'upgraded': 1, 'regular': 1, 'status': 1, 'well': 1, 'katherine': 1, 'pierce': 1})

All Possible Bigrams List :
 [('the', 'vampire'), ('vampire', 'diaries'), ('diaries', 'american'), ('american', 'supernatural'), ('supernatural', 'horror'), ('horro
r', 'romance'), ('romance', 'television'), ('television', 'series'), ('series', 'created'), ('created', 'kevin'), ('kevin', 'williamson'),
('williamson', 'based'), ('based', 'novels'), ('novels', 'name'), ('name', 'author'), ('author', 'l.'), ('l.', 'j.'), ('j.', 'smith'), ('s
mith', 'centered'), ('centered', 'aroung'), ('aroung', 'elena'), ('elena', 'gilbert'), ('gilbert', 'it'), ('it', 'officially'), ('official
ly', 'picked'), ('picked', '2009–10'), ('2009–10', 'season'), ('season', 'may'), ('may', '19'), ('19', '2009'), ('2009', 'it'), ('it', 'pr
emiered'), ('premiered', 'september'), ('september', '10'), ('10', '2009'), ('2009', 'the'), ('the', 'cw'), ('cw', 'received'), ('receive
d', 'highest'), ('highest', 'ratings'), ('ratings', 'series'), ('series', 'premiere'), ('premiere', 'network'), ('network', "'s"), ("'s",
'history'), ('history', 'point'), ('point', 'scoring'), ('scoring', '4.91'), ('4.91', 'million'), ('million', 'live'), ('live', 'viewers.t
he'), ('viewers.the', 'series'), ('series', 'focuses'), ('focuses', 'fictional'), ('fictional', 'town'), ('town', 'mystic'), ('mystic', 'f
alls'), ('falls', 'virginia'), ('virginia', 'charged'), ('charged', 'supernatural'), ('supernatural', 'history'), ('history', 'it'), ('i
t', 'follows'), ('follows', 'elena'), ('elena', 'gilbert'), ('gilbert', 'portrayed'), ('portrayed', 'nina'), ('nina', 'dobrev'), ('dobre
v', 'begins'), ('begins', 'get'), ('get', 'parents'), ('parents', 'death'), ('death', 'two'), ('two', 'vampires'), ('vampires', 'pull'),
('pull', 'world'), ('world', "n't"), ("n't", 'know'), ('know', 'stefan'), ('stefan', 'damon'), ('damon', 'salvatore'), ('salvatore', 'port
rayed'), ('portrayed', 'paul'), ('paul', 'wesley'), ('wesley', 'ian'), ('ian', 'somerhalder'), ('somerhalder', 'respectively'), ('respecti
vely', 'kayla'), ('kayla', 'ewell'), ('ewell', 'portrayed'), ('portrayed', 'vicki'), ('vicki', 'donovan'), ('donovan', 'first'), ('first',
'seven'), ('seven', 'episodes'), ('episodes', 'character'), ('character', 'killed'), ('killed', 'matt'), ('matt', 'davis'), ('davis', 'lat
er'), ('later', 'cast'), ('cast', 'history'), ('history', 'teacher'), ('teacher', 'recurring'), ('recurring', 'role'), ('role', 'fill'),
('fill', 'void'), ('void', 'he'), ('he', 'later'), ('later', 'upgraded'), ('upgraded', 'series'), ('series', 'regular'), ('regular', 'stat
us'), ('status', 'nina'), ('nina', 'dobrev'), ('dobrev', 'portrayed'), ('portrayed', 'elena'), ('elena', 'gilbert'), ('gilbert', 'well'),
('well', 'katherine'), ('katherine', 'pierce')]
```

```
Frequency of Bigrams :
 Counter({('elena', 'gilbert'): 3, ('nina', 'dobrev'): 2, ('the', 'vampire'): 1, ('vampire', 'diaries'): 1, ('diaries', 'american'): 1,
('american', 'supernatural'): 1, ('supernatural', 'horror'): 1, ('horror', 'romance'): 1, ('romance', 'television'): 1, ('television', 'se
ries'): 1, ('series', 'created'): 1, ('created', 'kevin'): 1, ('kevin', 'williamson'): 1, ('williamson', 'based'): 1, ('based', 'novels'):
1, ('novels', 'name'): 1, ('name', 'author'): 1, ('author', 'l.'): 1, ('l.', 'j.'): 1, ('j.', 'smith'): 1, ('smith', 'centered'): 1, ('cen
tered', 'aroung'): 1, ('aroung', 'elena'): 1, ('gilbert', 'it'): 1, ('it', 'officially'): 1, ('officially', 'picked'): 1, ('picked', '2009
–10'): 1, ('2009–10', 'season'): 1, ('season', 'may'): 1, ('may', '19'): 1, ('19', '2009'): 1, ('2009', 'it'): 1, ('it', 'premiered'): 1,
('premiered', 'september'): 1, ('september', '10'): 1, ('10', '2009'): 1, ('2009', 'the'): 1, ('the', 'cw'): 1, ('cw', 'received'): 1, ('r
eceived', 'highest'): 1, ('highest', 'ratings'): 1, ('ratings', 'series'): 1, ('series', 'premiere'): 1, ('premiere', 'network'): 1, ('net
work', "'s"): 1, ("'s", 'history'): 1, ('history', 'point'): 1, ('point', 'scoring'): 1, ('scoring', '4.91'): 1, ('4.91', 'million'): 1,
('million', 'live'): 1, ('live', 'viewers.the'): 1, ('viewers.the', 'series'): 1, ('series', 'focuses'): 1, ('focuses', 'fictional'): 1,
('fictional', 'town'): 1, ('town', 'mystic'): 1, ('mystic', 'falls'): 1, ('falls', 'virginia'): 1, ('virginia', 'charged'): 1, ('charged',
'supernatural'): 1, ('supernatural', 'history'): 1, ('history', 'it'): 1, ('it', 'follows'): 1, ('follows', 'elena'): 1, ('gilbert', 'port
rayed'): 1, ('portrayed', 'nina'): 1, ('dobrev', 'begins'): 1, ('begins', 'get'): 1, ('get', 'parents'): 1, ('parents', 'death'): 1, ('dea
th', 'two'): 1, ('two', 'vampires'): 1, ('vampires', 'pull'): 1, ('pull', 'world'): 1, ('world', "n't"): 1, ("n't", 'know'): 1, ('know',
'stefan'): 1, ('stefan', 'damon'): 1, ('damon', 'salvatore'): 1, ('salvatore', 'portrayed'): 1, ('portrayed', 'paul'): 1, ('paul', 'wesle
y'): 1, ('wesley', 'ian'): 1, ('ian', 'somerhalder'): 1, ('somerhalder', 'respectively'): 1, ('respectively', 'kayla'): 1, ('kayla', 'ewel
l'): 1, ('ewell', 'portrayed'): 1, ('portrayed', 'vicki'): 1, ('vicki', 'donovan'): 1, ('donovan', 'first'): 1, ('first', 'seven'): 1, ('s
even', 'episodes'): 1, ('episodes', 'character'): 1, ('character', 'killed'): 1, ('killed', 'matt'): 1, ('matt', 'davis'): 1, ('davis', 'l
ater'): 1, ('later', 'cast'): 1, ('cast', 'history'): 1, ('history', 'teacher'): 1, ('teacher', 'recurring'): 1, ('recurring', 'role'): 1,
('role', 'fill'): 1, ('fill', 'void'): 1, ('void', 'he'): 1, ('he', 'later'): 1, ('later', 'upgraded'): 1, ('upgraded', 'series'): 1, ('se
ries', 'regular'): 1, ('regular', 'status'): 1, ('status', 'nina'): 1, ('dobrev', 'portrayed'): 1, ('portrayed', 'elena'): 1, ('gilbert',
'well'): 1, ('well', 'katherine'): 1, ('katherine', 'pierce'): 1})

Probability of each bigram :
 {('the', 'vampire'): 0.5, ('vampire', 'diaries'): 1.0, ('diaries', 'american'): 1.0, ('american', 'supernatural'): 1.0, ('supernatural',
'horror'): 0.5, ('horror', 'romance'): 1.0, ('romance', 'television'): 1.0, ('television', 'series'): 1.0, ('series', 'created'): 0.25,
('created', 'kevin'): 1.0, ('kevin', 'williamson'): 1.0, ('williamson', 'based'): 1.0, ('based', 'novels'): 1.0, ('novels', 'name'): 1.0,
('name', 'author'): 1.0, ('author', 'l.'): 1.0, ('l.', 'j.'): 1.0, ('j.', 'smith'): 1.0, ('smith', 'centered'): 1.0, ('centered', 'aroun
g'): 1.0, ('aroung', 'elena'): 1.0, ('elena', 'gilbert'): 1.0, ('gilbert', 'it'): 0.3333333333333333, ('it', 'officially'): 0.333333333333
3333, ('officially', 'picked'): 1.0, ('picked', '2009–10'): 1.0, ('2009–10', 'season'): 1.0, ('season', 'may'): 1.0, ('may', '19'): 1.0,
('19', '2009'): 1.0, ('2009', 'it'): 0.5, ('it', 'premiered'): 0.3333333333333333, ('premiered', 'september'): 1.0, ('september', '10'):
1.0, ('10', '2009'): 1.0, ('2009', 'the'): 0.5, ('the', 'cw'): 0.5, ('cw', 'received'): 1.0, ('received', 'highest'): 1.0, ('highest', 'ra
tings'): 1.0, ('ratings', 'series'): 1.0, ('series', 'premiere'): 0.25, ('premiere', 'network'): 1.0, ('network', "'s"): 1.0, ("'s", 'hist
ory'): 1.0, ('history', 'point'): 0.3333333333333333, ('point', 'scoring'): 1.0, ('scoring', '4.91'): 1.0, ('4.91', 'million'): 1.0, ('mil
lion', 'live'): 1.0, ('live', 'viewers.the'): 1.0, ('viewers.the', 'series'): 1.0, ('series', 'focuses'): 0.25, ('focuses', 'fictional'):
1.0, ('fictional', 'town'): 1.0, ('town', 'mystic'): 1.0, ('mystic', 'falls'): 1.0, ('falls', 'virginia'): 1.0, ('virginia', 'charged'):
1.0, ('charged', 'supernatural'): 1.0, ('supernatural', 'history'): 0.5, ('history', 'it'): 0.3333333333333333, ('it', 'follows'): 0.33333
33333333333, ('follows', 'elena'): 1.0, ('gilbert', 'portrayed'): 0.3333333333333333, ('portrayed', 'nina'): 0.25, ('nina', 'dobrev'): 1.
0, ('dobrev', 'begins'): 0.5, ('begins', 'get'): 1.0, ('get', 'parents'): 1.0, ('parents', 'death'): 1.0, ('death', 'two'): 1.0, ('two',
'vampires'): 1.0, ('vampires', 'pull'): 1.0, ('pull', 'world'): 1.0, ('world', "n't"): 1.0, ("n't", 'know'): 1.0, ('know', 'stefan'): 1.0,
('stefan', 'damon'): 1.0, ('damon', 'salvatore'): 1.0, ('salvatore', 'portrayed'): 1.0, ('portrayed', 'paul'): 0.25, ('paul', 'wesley'):
1.0, ('wesley', 'ian'): 1.0, ('ian', 'somerhalder'): 1.0, ('somerhalder', 'respectively'): 1.0, ('respectively', 'kayla'): 1.0, ('kayla',
'ewell'): 1.0, ('ewell', 'portrayed'): 1.0, ('portrayed', 'vicki'): 0.25, ('vicki', 'donovan'): 1.0, ('donovan', 'first'): 1.0, ('first',
'seven'): 1.0, ('seven', 'episodes'): 1.0, ('episodes', 'character'): 1.0, ('character', 'killed'): 1.0, ('killed', 'matt'): 1.0, ('matt',
'davis'): 1.0, ('davis', 'later'): 1.0, ('later', 'cast'): 0.5, ('cast', 'history'): 1.0, ('history', 'teacher'): 0.3333333333333333, ('te
acher', 'recurring'): 1.0, ('recurring', 'role'): 1.0, ('role', 'fill'): 1.0, ('fill', 'void'): 1.0, ('void', 'he'): 1.0, ('he', 'later'):
1.0, ('later', 'upgraded'): 0.5, ('upgraded', 'series'): 1.0, ('series', 'regular'): 0.25, ('regular', 'status'): 1.0, ('status', 'nina'):
1.0, ('dobrev', 'portrayed'): 0.5, ('portrayed', 'elena'): 0.25, ('gilbert', 'well'): 0.3333333333333333, ('well', 'katherine'): 1.0, ('ka
therine', 'pierce'): 1.0}

 Tokenised Input Sentence :
  ['Nina', 'Dobrev', 'portrayed', 'Elena']

 Filtered Input Sentence :
  ['nina', 'dobrev', 'portrayed', 'elena']


The bigrams in given sentence are :
[('nina', 'dobrev'), ('dobrev', 'portrayed'), ('portrayed', 'elena')]

 PROBABILITY TABLE :

                             nina                dobrev              portrayed                elena
          nina                0.0                 1.0                   0.0                  0.0
        dobrev                0.0                 0.0                   0.5                  0.0
      portrayed               0.25                0.0                   0.0                  0.25
        elena                 0.0                 0.0                   0.0                  0.0

 BIGRAM TABLE :

                             nina                dobrev              portrayed                elena
          nina                 0                   2                     0                    0
        dobrev                 0                   0                     1                    0
      portrayed                1                   0                     0                    1
        elena                  0                   0                     0                    0
```

```
Sentence Probability : 0.125

*********************************** ADD ONE SMOOTHING ***********************************

PROBABILITY TABLE :
```

	nina	dobrev	portrayed	elena
nina	0.009615384615384616	0.028846153846153848	0.009615384615384616	0.009615384615384616
dobrev	0.009615384615384616	0.009615384615384616	0.019230769230769232	0.009615384615384616
portrayed	0.018867924528301886	0.009433962264150943	0.009433962264150943	0.018867924528301886
elena	0.009523809523809525	0.009523809523809525	0.009523809523809525	0.009523809523809525

```
BIGRAM TABLE :
```

	nina	dobrev	portrayed	elena
nina	1	3	1	1
dobrev	1	1	2	1
portrayed	2	1	1	2
elena	1	1	1	1

REFERENCES:

1. https://www.geeksforgeeks.org/n-gram-language-modelling-with-nltk/
2. https://medium.com/swlh/language-modelling-with-nltk-20eac7e70853
3. https://web.stanford.edu/~jurafsky/slp3/3.pdf
4. https://www.educative.io/edpresso/what-is-a-bigram-language-model
5. https://www.youtube.com/watch?v=dZvCHz6lcGU