DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

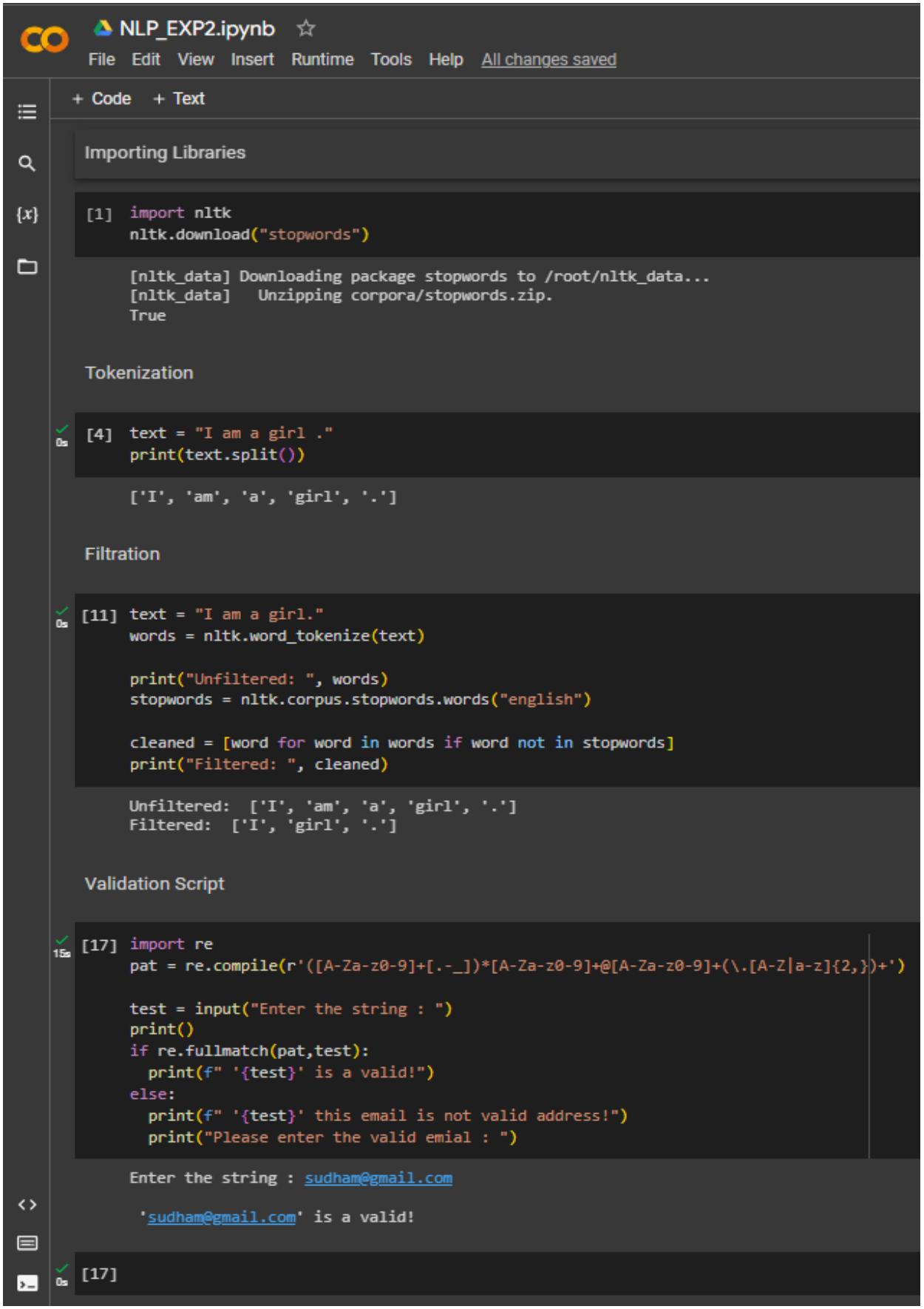
NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**NATURAL LANGUAGE PROCESSING LAB**
COURSE CODE	**CSDOL7011**
PRACTICAL NO.	
DOP	
DOS	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	
SUBJECT	**NATURAL LANGUAGE PROCESSING LAB**
COURSE CODE	**CSDOL7011**
PRACTICAL NO.	
DOP	
DOS	

Output :


NLP_EXP2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Importing Libraries
[1] import nltk
nltk.download("stopwords")
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
True
Tokenization
[4] text = "I am a girl ."
print(text.split())
['I', 'am', 'a', 'girl', '.']
Filtration
[11] text = "I am a girl."
words = nltk.word_tokenize(text)
print("Unfiltered: ", words)
stopwords = nltk.corpus.stopwords.words("english")
cleaned = [word for word in words if word not in stopwords]
print("Filtered: ", cleaned)
Unfiltered: ['I', 'am', 'a', 'girl', '.']
Filtered: ['I', 'girl', '.']
Validation Script
[17] import re
pat = re.compile(r'([A-Za-z0-9]+[.-_])*[A-Za-z0-9]+@[A-Za-z0-9]+(\.[A-Z|a-z]{2,})+')
test = input("Enter the string : ")
print()
if re.fullmatch(pat,test):
print(f" '{test}' is a valid!")
else:
print(f" '{test}' this email is not valid address!")
print("Please enter the valid emial : ")
Enter the string : sudham@gmail.com
'sudham@gmail.com' is a valid!
[17]

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	
SUBJECT	**NATURAL LANGUAGE PROCESSING LAB**
COURSE CODE	**CSDOL7011**
PRACTICAL NO.	
DOP	
DOS	

Output :

Stop word removal

Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence.

```
[4] import nltk
    nltk.download('punkt')
    nltk.download('stopwords')
    from nltk.corpus import stopwords
    from nltk.tokenize import word_tokenize

    # Add text
    text = "How to remove stop words with NLTK library in Python"
    print("Text:", text)

    # Convert text to lowercase and split to a list of words
    tokens = word_tokenize(text.lower())
    print("Tokens:", tokens)

    # Remove stop words
    english_stopwords = stopwords.words('english')
    tokens_wo_stopwords = [t for t in tokens if t not in english_stopwords]
    print("Text without stop words:", " ".join(tokens_wo_stopwords))
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
Text: How to remove stop words with NLTK library in Python
Tokens: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'library', 'in', 'python']
Text without stop words: remove stop words nltk library python
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Stemming

Stemming is a technique used to extract the base form of the words by removing affixes from them. It is just like cutting down the branches of a tree to its stems. For example, the stem of the words eating, eats, eaten is eat.

```
[5] import nltk
    from nltk.stem import PorterStemmer
    word_stemmer = PorterStemmer()
    word_stemmer.stem('writing')
```

```
'write'
```

Lemmatization

Lemmatization technique is like stemming. The output we will get after lemmatization is called 'lemma', which is a root word rather than root stem, the output of stemming.

```
[6] import nltk
    nltk.download('wordnet')
    from nltk.stem import WordNetLemmatizer
    lemmatizer = WordNetLemmatizer()
    lemmatizer.lemmatize('eating')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
'eating'
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	
SUBJECT	**NATURAL LANGUAGE PROCESSING LAB**
COURSE CODE	**CSDOL7011**
PRACTICAL NO.	
DOP	
DOS	

Output:

NLP_EXP4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Morphological Analysis

```
[4] from polyglot.text import Text, Word

    words = ["preprocessing", "processor", "invaluable", "thankful", "crossed"]
    for w in words:
     w = Word(w, language="en")
     print("{:<20}{}".format(w, w.morphemes))
```

```
preprocessing       ['pre', 'process', 'ing']
processor           ['process', 'or']
invaluable          ['in', 'valuable']
thankful            ['thank', 'ful']
crossed             ['cross', 'ed']
```

Text Generation

```
[5] import random

    # List of possible words
    words = ["Hello", "world", "Python", "is", "fun", "text", "generation"]

    # Generate a random sentence
    sentence = ' '.join(random.choice(words) for _ in range(5))
    print(sentence)
```

```
world Python generation generation fun
```

```
import random

corpus = "This is a sample corpus of words for word generation."
words = corpus.split()
chain = {words[i]: [words[i + 1]] for i in range(len(words) - 1)}

def generate_word_markov(chain, starting_word):
 generated_words = [starting_word]
 current_word = starting_word
 while current_word in chain:
  next_word = random.choice(chain[current_word])
  generated_words.append(next_word)
  current_word = next_word
 return " ".join(generated_words)

starting_word = "sample"
generated_words = generate_word_markov(chain, starting_word)
print(generated_words)
```

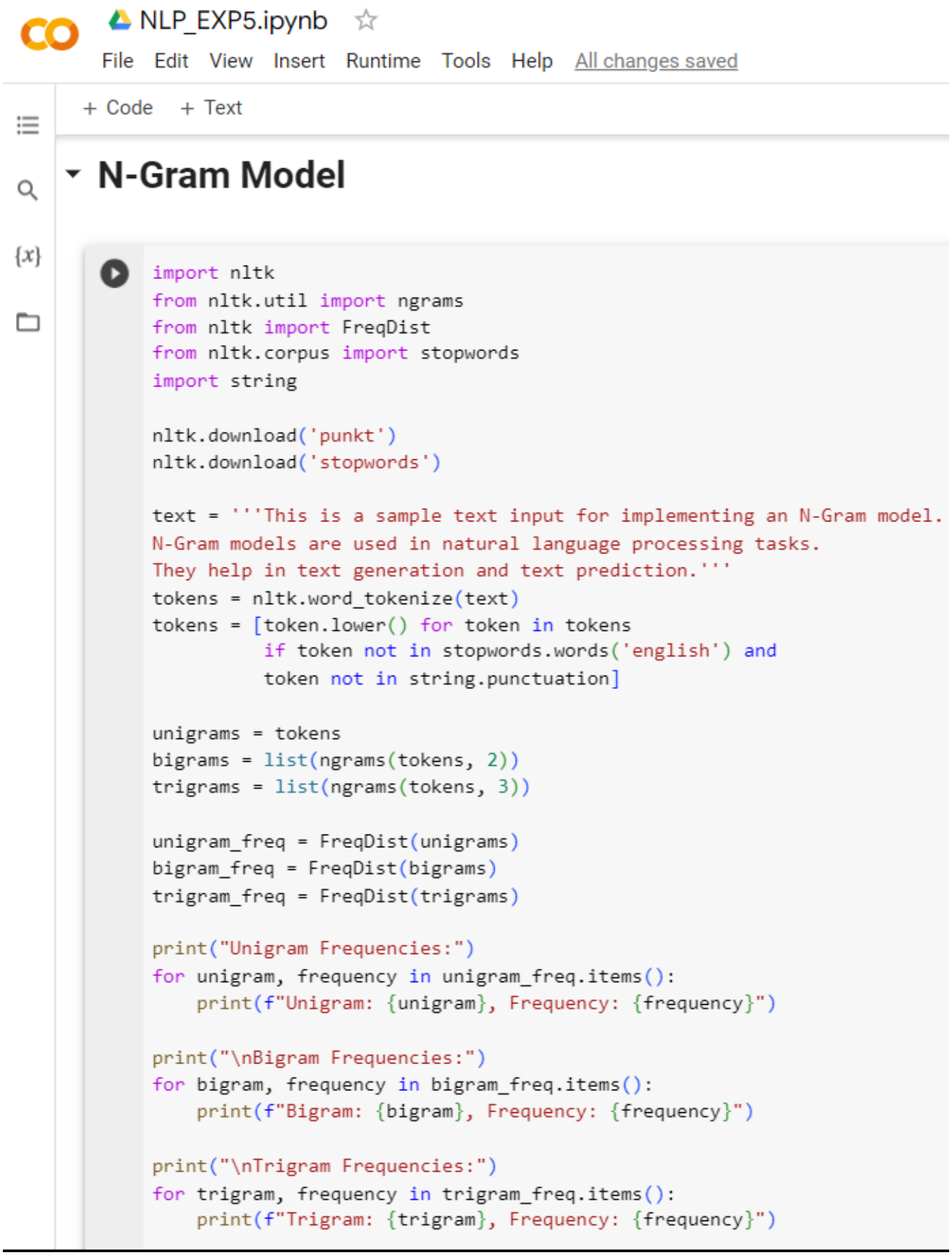
```
sample corpus of words for word generation.
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	
SUBJECT	**NATURAL LANGUAGE PROCESSING LAB**
COURSE CODE	**CSDOL7011**
PRACTICAL NO.	
DOP	
DOS	

OUTPUT :-


NLP_EXP5.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
N-Gram Model
import nltk
from nltk.util import ngrams
from nltk import FreqDist
from nltk.corpus import stopwords
import string
nltk.download('punkt')
nltk.download('stopwords')
text = '''This is a sample text input for implementing an N-Gram model.
N-Gram models are used in natural language processing tasks.
They help in text generation and text prediction.'''
tokens = nltk.word_tokenize(text)
tokens = [token.lower() for token in tokens
if token not in stopwords.words('english') and
token not in string.punctuation]
unigrams = tokens
bigrams = list(ngrams(tokens, 2))
trigrams = list(ngrams(tokens, 3))
unigram_freq = FreqDist(unigrams)
bigram_freq = FreqDist(bigrams)
trigram_freq = FreqDist(trigrams)
print("Unigram Frequencies:")
for unigram, frequency in unigram_freq.items():
print(f"Unigram: {unigram}, Frequency: {frequency}")
print("\nBigram Frequencies:")
for bigram, frequency in bigram_freq.items():
print(f"Bigram: {bigram}, Frequency: {frequency}")
print("\nTrigram Frequencies:")
for trigram, frequency in trigram_freq.items():
print(f"Trigram: {trigram}, Frequency: {frequency}")

```
Unigram Frequencies:
Unigram: this, Frequency: 1
Unigram: sample, Frequency: 1
Unigram: text, Frequency: 3
Unigram: input, Frequency: 1
Unigram: implementing, Frequency: 1
Unigram: n-gram, Frequency: 2
Unigram: model, Frequency: 1
Unigram: models, Frequency: 1
Unigram: used, Frequency: 1
Unigram: natural, Frequency: 1
Unigram: language, Frequency: 1
Unigram: processing, Frequency: 1
Unigram: tasks, Frequency: 1
Unigram: they, Frequency: 1
Unigram: help, Frequency: 1
Unigram: generation, Frequency: 1
Unigram: prediction, Frequency: 1

Bigram Frequencies:
Bigram: ('this', 'sample'), Frequency: 1
Bigram: ('sample', 'text'), Frequency: 1
Bigram: ('text', 'input'), Frequency: 1
Bigram: ('input', 'implementing'), Frequency: 1
Bigram: ('implementing', 'n-gram'), Frequency: 1
Bigram: ('n-gram', 'model'), Frequency: 1
Bigram: ('model', 'n-gram'), Frequency: 1
Bigram: ('n-gram', 'models'), Frequency: 1
Bigram: ('models', 'used'), Frequency: 1
Bigram: ('used', 'natural'), Frequency: 1
Bigram: ('natural', 'language'), Frequency: 1
Bigram: ('language', 'processing'), Frequency: 1
Bigram: ('processing', 'tasks'), Frequency: 1
Bigram: ('tasks', 'they'), Frequency: 1
Bigram: ('they', 'help'), Frequency: 1
Bigram: ('help', 'text'), Frequency: 1
Bigram: ('text', 'generation'), Frequency: 1
Bigram: ('generation', 'text'), Frequency: 1
Bigram: ('text', 'prediction'), Frequency: 1

Trigram Frequencies:
Trigram: ('this', 'sample', 'text'), Frequency: 1
Trigram: ('sample', 'text', 'input'), Frequency: 1
Trigram: ('text', 'input', 'implementing'), Frequency: 1
Trigram: ('input', 'implementing', 'n-gram'), Frequency: 1
Trigram: ('implementing', 'n-gram', 'model'), Frequency: 1
Trigram: ('n-gram', 'model', 'n-gram'), Frequency: 1
Trigram: ('model', 'n-gram', 'models'), Frequency: 1
Trigram: ('n-gram', 'models', 'used'), Frequency: 1
Trigram: ('models', 'used', 'natural'), Frequency: 1
Trigram: ('used', 'natural', 'language'), Frequency: 1
Trigram: ('natural', 'language', 'processing'), Frequency: 1
Trigram: ('language', 'processing', 'tasks'), Frequency: 1
Trigram: ('processing', 'tasks', 'they'), Frequency: 1
Trigram: ('tasks', 'they', 'help'), Frequency: 1
Trigram: ('they', 'help', 'text'), Frequency: 1
Trigram: ('help', 'text', 'generation'), Frequency: 1
Trigram: ('text', 'generation', 'text'), Frequency: 1
Trigram: ('generation', 'text', 'prediction'), Frequency: 1
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	
SUBJECT	**NATURAL LANGUAGE PROCESSING LAB**
COURSE CODE	**CSDOL7011**
PRACTICAL NO.	
DOP	
DOS	

OUTPUT :-

```
import nltk
nltk.download('state_union')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
from nltk.corpus import state_union
from nltk.tokenize import PunktSentenceTokenizer

train_text = state_union.raw("2005-GWBush.txt")
sample_text = state_union.raw("2006-GWBush.txt")

custom_sent_tokenizer = PunktSentenceTokenizer(train_text)
tokenized = custom_sent_tokenizer.tokenize(sample_text)

def process_content():
    try:
        for i in tokenized[:5]:
            words = nltk.word_tokenize(i)
            tagged = nltk.pos_tag(words)
            print(tagged)  # Print the entire tagged sentence

    except Exception as e:
        print(str(e))

process_content()
```

```
[('PRESIDENT', 'NNP'), ('GEORGE', 'NNP'), ('W.', 'NNP'), ('BUSH', 'NNP'), ("'S", 'POS'), ('ADDRESS', 'NNP'), ('BEFORE', 'IN'),
('A', 'NNP'), ('JOINT', 'NNP'), ('SESSION', 'NNP'), ('OF', 'IN'), ('THE', 'NNP'), ('CONGRESS', 'NNP'), ('ON', 'NNP'), ('THE',
'NNP'), ('STATE', 'NNP'), ('OF', 'IN'), ('THE', 'NNP'), ('UNION', 'NNP'), ('January', 'NNP'), ('31', 'CD'), (',', ','), ('200
6', 'CD'), ('THE', 'NNP'), ('PRESIDENT', 'NNP'), (':', ':'), ('Thank', 'NNP'), ('you', 'PRP'), ('all', 'DT'), ('.', '.')]
[('Mr.', 'NNP'), ('Speaker', 'NNP'), (',', ','), ('Vice', 'NNP'), ('President', 'NNP'), ('Cheney', 'NNP'), (',', ','), ('member
s', 'NNS'), ('of', 'IN'), ('Congress', 'NNP'), (',', ','), ('members', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('Supreme', 'NNP'),
('Court', 'NNP'), ('and', 'CC'), ('diplomatic', 'JJ'), ('corps', 'NN'), (',', ','), ('distinguished', 'JJ'), ('guests', 'NNS'),
(',', ','), ('and', 'CC'), ('fellow', 'JJ'), ('citizens', 'NNS'), (':', ':'), ('Today', 'VB'), ('our', 'PRP$'), ('nation', 'N
N'), ('lost', 'VBD'), ('a', 'DT'), ('beloved', 'VBN'), (',', ','), ('graceful', 'JJ'), (',', ','), ('courageous', 'JJ'), ('woma
n', 'NN'), ('who', 'WP'), ('called', 'VBD'), ('America', 'NNP'), ('to', 'TO'), ('its', 'PRP$'), ('founding', 'NN'), ('ideals',
'NNS'), ('and', 'CC'), ('carried', 'VBD'), ('on', 'IN'), ('a', 'DT'), ('noble', 'JJ'), ('dream', 'NN'), ('.', '.')]
[('Tonight', 'NN'), ('we', 'PRP'), ('are', 'VBP'), ('comforted', 'VBN'), ('by', 'IN'), ('the', 'DT'), ('hope', 'NN'), ('of', 'I
N'), ('a', 'DT'), ('glad', 'JJ'), ('reunion', 'NN'), ('with', 'IN'), ('the', 'DT'), ('husband', 'NN'), ('who', 'WP'), ('was',
'VBD'), ('taken', 'VBN'), ('so', 'RB'), ('long', 'RB'), ('ago', 'RB'), (',', ','), ('and', 'CC'), ('we', 'PRP'), ('are', 'VB
P'), ('grateful', 'JJ'), ('for', 'IN'), ('the', 'DT'), ('good', 'JJ'), ('life', 'NN'), ('of', 'IN'), ('Coretta', 'NNP'), ('Scot
t', 'NNP'), ('King', 'NNP'), ('.', '.')]
[('(', '('), ('Applause', 'NNP'), ('.', '.'), (')', ')')]
[('President', 'NNP'), ('George', 'NNP'), ('W.', 'NNP'), ('Bush', 'NNP'), ('reacts', 'VBZ'), ('to', 'TO'), ('applause', 'VB'),
('during', 'IN'), ('his', 'PRP$'), ('State', 'NNP'), ('of', 'IN'), ('the', 'DT'), ('Union', 'NNP'), ('Address', 'NNP'), ('at',
'IN'), ('the', 'DT'), ('Capitol', 'NNP'), (',', ','), ('Tuesday', 'NNP'), (',', ','), ('Jan', 'NNP'), ('.', '.')]
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

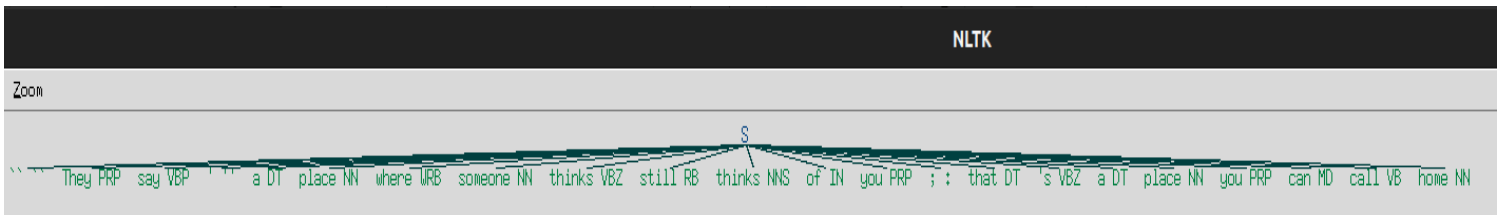
B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	
SUBJECT	**NATURAL LANGUAGE PROCESSING LAB**
COURSE CODE	**CSDOL7011**
PRACTICAL NO.	
DOP	
DOS	

OUTPUT :-

In [*]:
```
import nltk
nltk.download('averaged_perceptron_tagger')
sample_text="""
They say 'a place where someone thinks still thinks of you;
that's a place you can call home
"""
tokenized=nltk.sent_tokenize(sample_text)
for i in tokenized:
  words=nltk.word_tokenize(i)
  # print(words)
  tagged_words=nltk.pos_tag(words)
  # print(tagged_words)
  chunkGram=r"""VB: {}"""
  chunkParser=nltk.RegexpParser(chunkGram)
  chunked=chunkParser.parse(tagged_words)
  chunked.draw()
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/computer/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	
SUBJECT	**NATURAL LANGUAGE PROCESSING LAB**
COURSE CODE	**CSDOL7011**
PRACTICAL NO.	
DOP	
DOS	

OUTPUT :-

NLP_EXP8.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[8]  import spacy
     from spacy import displacy

     NER = spacy.load("en_core_web_sm")
```

```
[9]  raw_text="""
     There is only one thing that matters if you are a shinobi,
     and it isn't the number of jutsu you possess.
     All you need is the guts to never give up.
     """
```

```
[10] text1= NER(raw_text)
```

```
[11] for word in text1.ents:
         print(word.text,word.label_)
```

```
only one CARDINAL
```

```
[12] spacy.explain("ORG")
```

```
'Companies, agencies, institutions, etc.'
```

```
[13] spacy.explain("GPE")
```

```
'Countries, cities, states'
```

```
[14] displacy.render(text1,style="ent",jupyter=True)
```

There is only one CARDINAL thing that matters if you are a shinobi,

and it isn't the number of jutsu you possess.

All you need is the guts to never give up.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**SUDHAM D. SINGH**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	
SUBJECT	**NATURAL LANGUAGE PROCESSING LAB**
COURSE CODE	**CSDOL7011**
PRACTICAL NO.	
DOP	
DOS	

Experiment No. 09

Aim: Mini-project For Chosen Real World Application ('E-Commerce Customer Review Analysis').

Abstract:

Analyzing customer reviews, ecommerce businesses can gain insights into customer needs, preferences, and pain points. They can use this information to improve their products or services, identify areas for improvement, and enhance customer satisfaction.
E-commerce customer review analysis, a pivotal aspect of modern online business, involves the systematic evaluation and interpretation of customer feedback and reviews regarding products and services available on digital marketplaces and e-commerce platforms. Employing natural language processing (NLP) and data analytics techniques, this process aims to gain deeper insights into customer sentiments, opinions, and preferences. Key components of e-commerce customer review analysis include data collection, sentiment analysis, aspect-based analysis, rating analysis, keyword extraction, topic modeling, competitor analysis, quality control, and customer feedback loop.

Introduction:

Customer Review Analysis is the process of analyzing and interpreting customer reviews and feedback about products or services offered by a business. It involves using techniques such as sentiment analysis and natural language processing to understand customer feedback and identify patterns and trends in customer behavior.

By analyzing customer reviews, ecommerce businesses can gain insights into customer needs, preferences, and pain points. They can use this information to improve their products or services, identify areas for improvement, and enhance customer satisfaction.

Customer reviews help build social proof and trust among potential customers. When customers leave positive reviews, it shows that others have had a good experience with the product or service, making it more likely for new customers to make a purchase. On the other hand, negative reviews can provide valuable feedback for the business to improve its offerings and customer service.

E-commerce customer review analysis, also known as online review analysis or sentiment analysis of e-commerce reviews, is the process of systematically evaluating and extracting insights from customer reviews and feedback about products or services that are sold online. This analysis is typically performed using natural language processing (NLP) techniques and data analytics to gain a better understanding of customers' opinions and sentiments.

- **Key Features**

1) **Predictive Suggestions:** Autocompletion systems predict the next word or phrase a user intends to type based on what they've already entered. This prediction is typically based on statistical analysis of large text corpora, language models, or user-specific data.

2) **Real-Time Suggestions:** Autocomplete systems often provide real-time suggestions as users type, with a list of potential completions appearing in a dropdown or pop-up menu. Users can then select the desired suggestion, reducing the need for manual typing.
3) **Contextual Understanding:** Advanced auto completion systems consider the context of the text to provide more accurate suggestions. They take into account the preceding words or phrases, adjusting suggestions accordingly.

4) **Multilingual Support:** Autocomplete systems can offer suggestions in multiple languages, making them versatile for users with diverse language needs.

5) **Personalization:** Some autocompletion systems can be personalized to a user's writing style, preferences, and frequently used words and phrases. This enhances the accuracy and relevance of suggestions.

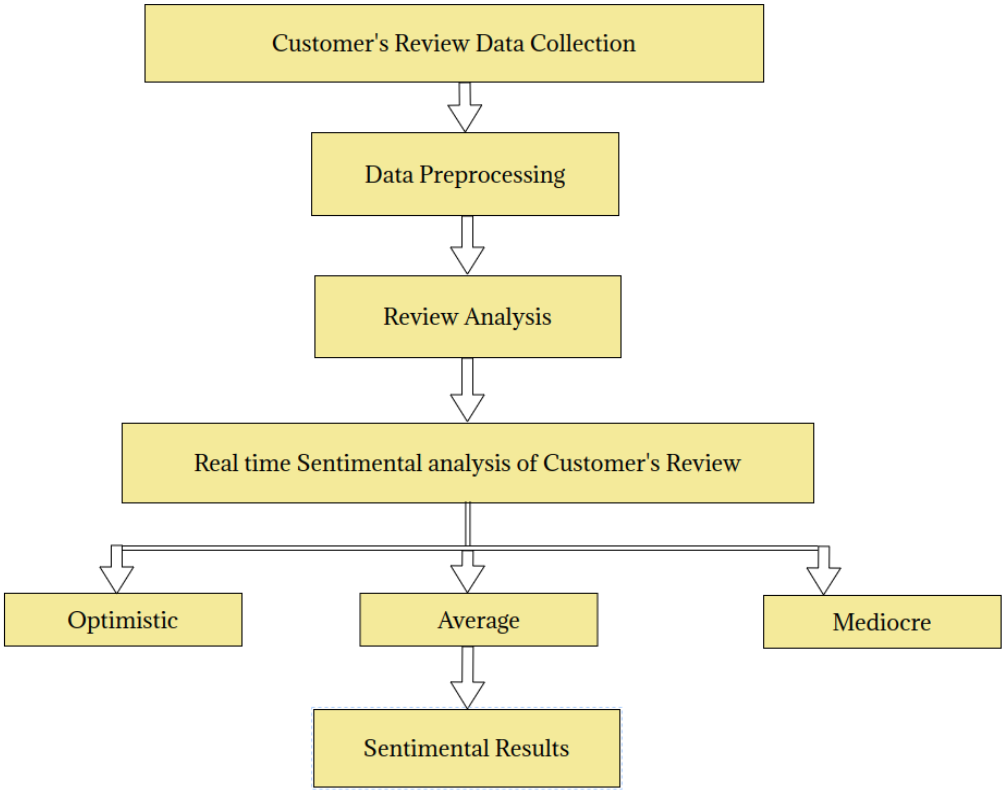
6) **Autocorrect:** Autocompletion systems often include autocorrect functionality, which can automatically correct misspelled words as users type. This helps improve the quality of the text and reduces errors.

7) **Efficiency and Time-Saving:** Autocompletion greatly improves typing speed and accuracy, reducing the effort required to input text, especially on mobile devices with small keyboards.

8) **Learning and Adaptation:** Some autocompletion systems learn from user behavior over time, becoming more accurate and tailored to the individual's writing style and needs.

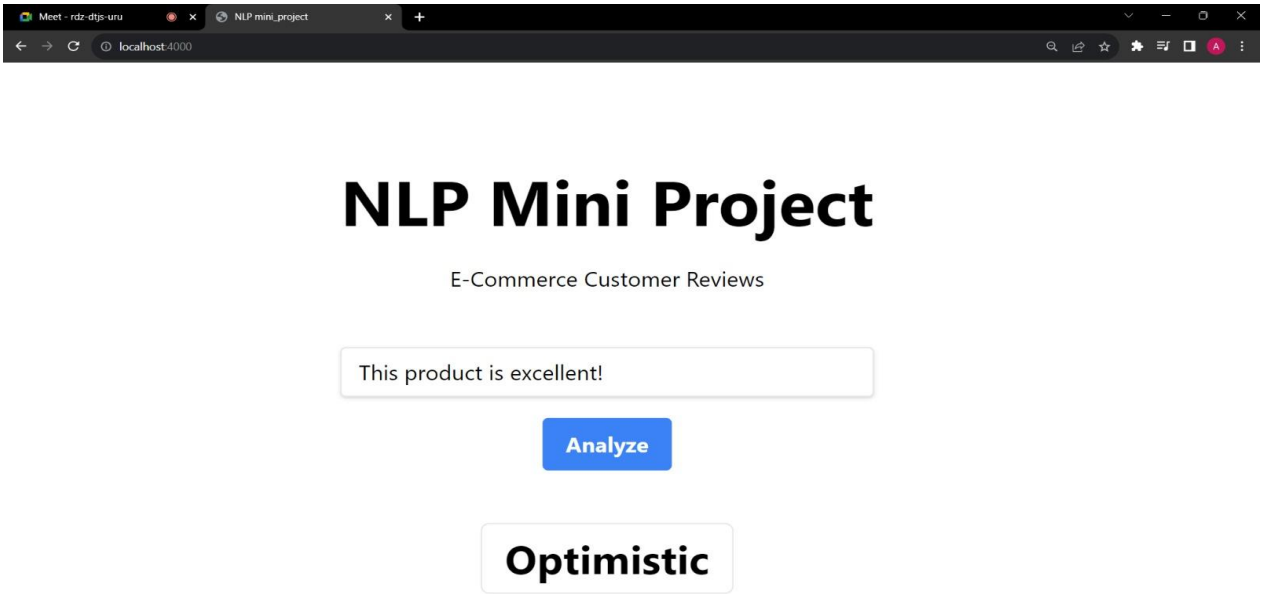
9) **Application Integration:** Autocomplete is integrated into various software applications, including web browsers, email clients, text editors, and messaging platforms. It enhances the user experience in these contexts.

Data Flow Diagram:

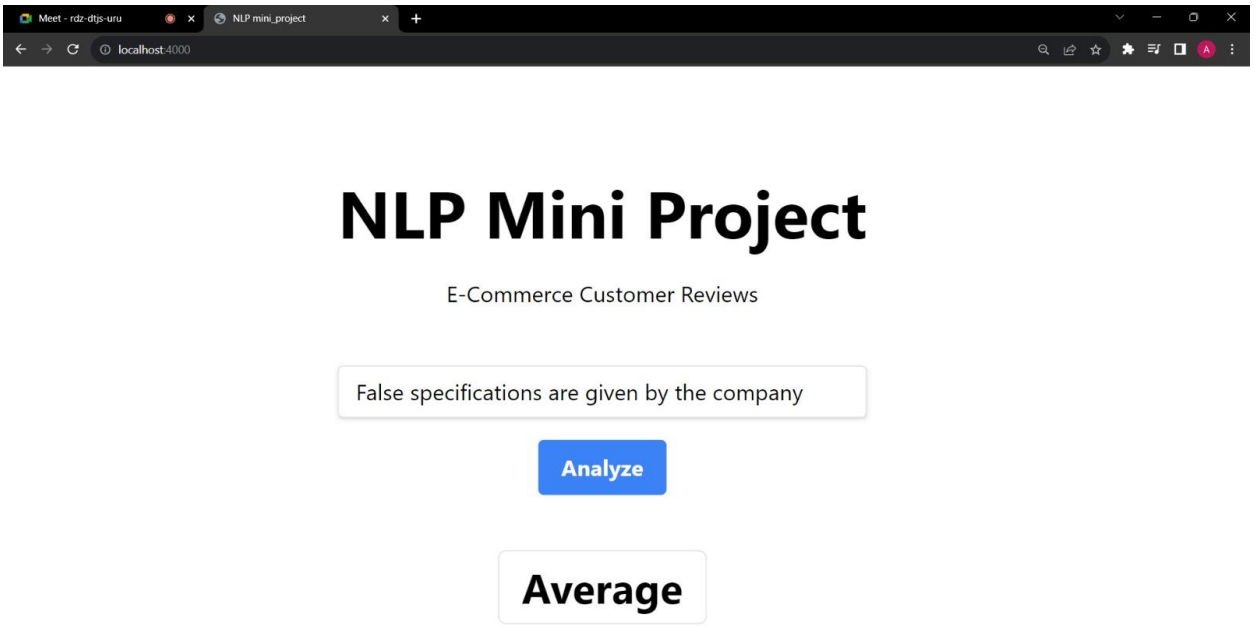


Output:

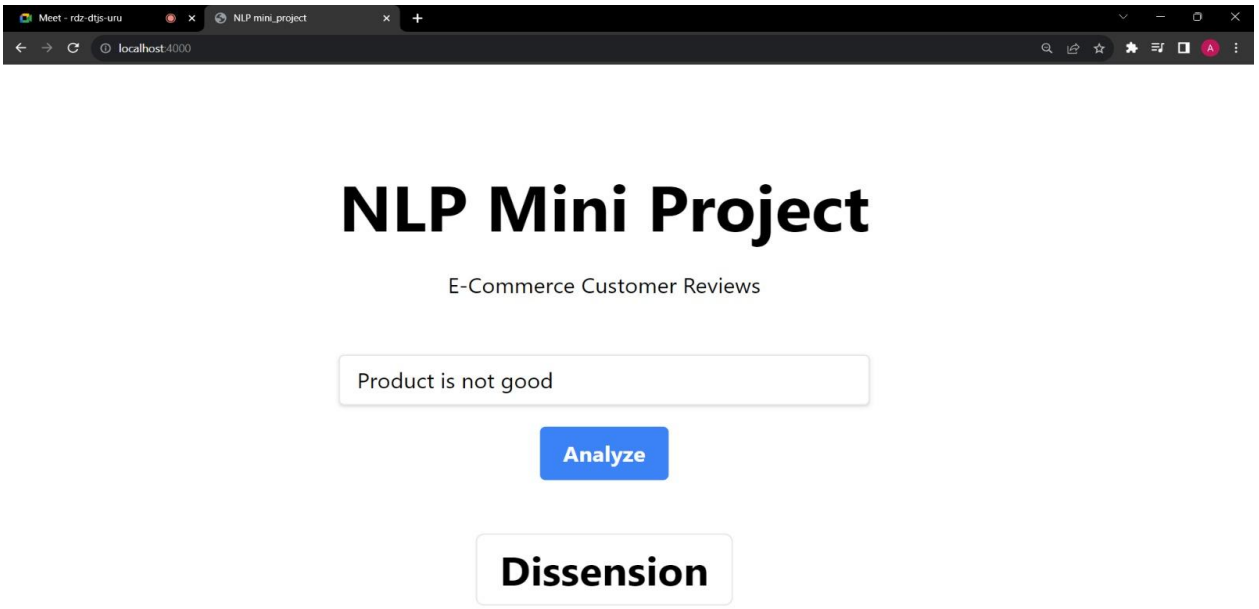
1) Optimistic review



2) Average Review



3) Dissension review



Conclusion: We have successfully implemented real world NLP Application in E-Commerce Customer Review Analysis.

Name:- Singh Sudham Dharmendra

Branch:- CSE - (AIML)

Roll no:- AIML57

Subject:- Natural Language Processing

Topic:- Assignment No:- (1)

Date of Submission:- 22/08/2023

Q.1] What is NLP ? Explain different phases involved in the NLP process with suitable examples.

→ Natural language processing is the process of computer analysis of input provided in a human language, and conversion of this input into a useful form of representation NLP is concerned with the development of computational models of aspects of human Language processing.

Two main reasons of such development are :-
- develop automatic tool for NLP
- Gain better understanding of human communication

The different phases involved in the NLP process are as follows :-

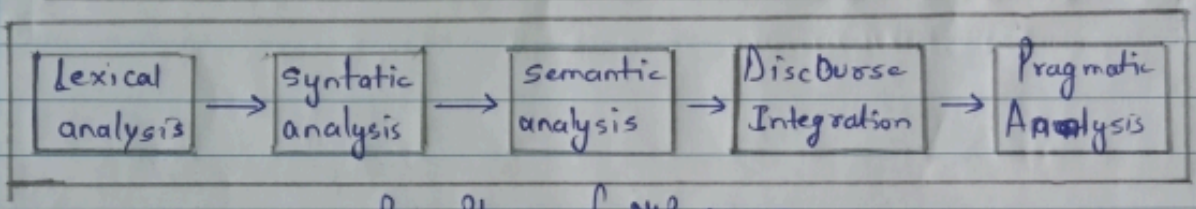


fig. Phases of NLP process

(i) Lexical analysis :- Lexical analysis is the first stage of NLP. It is also known as morphological analysis. At this stage the structure of words is identified & analyzed.
Eg :- The sentence : "sudham is Genious."
→ : ["sudham", "is", "Genious", "."]

(ii) Syntatic analysis :- It involves analysis of words in sentence for grammar & ordering words in way that show relationship among words.
Eg :- The sentence : "The school goes to boy" is rejected by English Syntatic analysis ~~because it does not make any sense~~ Because it does not make any sense.

(iii) Semantic Analysis :- Semantic analysis draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntatic structures and objects in the task domain.
Eg :- The semantic analyzer neglects sentence such as "hot ice-cream".

(iv) Discourse integration :- The meaning of any sentence depends upon the meaning of sentence just before it.
It also bring about meaning of immediately following sentences.
Eg :- "Sudham is a boy, he goes to school"
Here "he" is a dependency pointing to sudham.

(v) Pragmatic Analysis :- During this, what was said is re-interpreted on what it truly meant. It contains deriving those aspects of language which necessitate real world knowledge.
Eg :- "John saw Monu in a garden with a cat"
Here we can't say that John is with cat or Monu is with cat.

Q.2] What is Stemming ? Explain Porter's stemming algorithm in detail.

→ Stemming in NLP [Natural Language Processing] refers to the process of reducing a word to its word stem that affixes to suffixes & prefixes or the root words. It's algorithm is a language linguistic normalization process in which the varient forms of a words are reduced to standard form.
Eg :- "eating", "eats", "eaten" => eat

Porter's Stemming :- It is common algorithm for stemming English.
The algorithm consist of 5 sets of rules applied in order.
- phases applied sequentially.
- each phase consist of SET of command.
Sample Convention : of the rules is a compand command, select the one that applies to the longest suffix.

- The porter stemmer defines two terms:
  - consonant - a letter other than A, E, I, O, U & Y preceed by consonant
  - vowel - any other letter. (A, E, I, O, U) & Y

With the definition all words are of the term $(C)(VC)^m(V)$

C = string of one or more consonants
V = string of one or more vowels
m = measure of words or word part when represented in term of VC
VC = combination of vowels & consonants

Eg:- Now calculate value of 'm'

① The combination that doesn't contain VC term → TREE, BY, TR
C C V V | C C | C C

② The combinat that contain VC term → TROUBLE
C C V V C C V → m = 1

TREES
C C V V C → m = 1

PRIVATE
C C V C V C V → m = 2

*Rules for Removing suffix:-
To use porter stemmer, there are certain rules to follow:-
The rules are of the form (Condition) $S_1 \rightarrow S_2$ [S_1 & S_2 are suffixes]

Eg:- REPLACEMENT → if rule is (m>1) EMENT
C V C C V C V C V C C

in this S_1 is EMENT & S_2 is Null ∴ m > 1
Stem part will be replace by null → REPLAC [∵ m = 4]

The condition may also contain following:-
m = the measure of stem
*S = the stem ends with 'S'
V = the stem contains vowels.

*d : the stem ends with double consonant [TT, SS]
*o : the stem ends with cvc [second c not w,x or y]

The condition part may also contain expression with 'and' 'or' & 'not'
eg:- (m>1 & (*s or *t)) : tests for stem with m>1 ending in 's' or 't'

→ Step①:-

SSES ⇒ SS eg:- Caresses → Caress ; IES → I
Ponies → poni ; ties → ti

SS ⇒ SS eg:- Pass → Pass

S ⇒ E (null) eg:- Cats → cat

→ step②:- Stemming with condition

① (m>0) EED → EE eg:- agreed → agree ... if condition verified
feed → feed ... if not verified

② (*v*) ED → E [null] eg:- plastered → plaster ... if verified
bled → bled ... if not verified
(stem contain vowels)

③ (*v*) ING → E eg:- motoring → motor ... if verified
SING → SING ... If not verified

→ step③:- Clean up
these rules are run if second & third rule in ② apply.

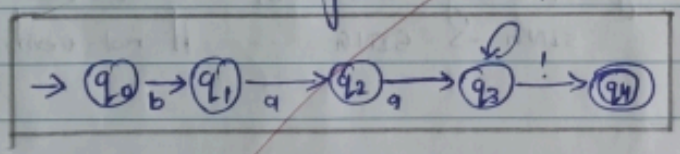
① AT → ATE eg:- conflat(ed) → conflate.
② BL → BLE eg:- Troubl(ing) → Trouble
③ [*d & ! (*L or *S or *Z)] → Single
eg:- hopp(ing) → hop ... condition verified
fall(ing) → fall ... condition not verified
④ (m=1 & *o) → E
eg:- fil(ing) → file ... condition verified
fail → fail ... Condition not verified.

Q.4] Explain the role of FSA in morphological analysis?

→ -Morphology is the study of the structure & formation of words
An automata having a finite number of states is named a finite state automata. -FSA are used to identify patterns.
- It takes the string of symbol as input & changes its state accordingly
- It is finite state machine having '5' elements or tuples.
- FSA is of two types: ① DFA [deterministic finite automata]
② NFA [Non-deterministic finite automata]

'5' elements are [Q, Σ, δ, q_0, F]
Q → set of finite state ; Σ = set of alphabets
δ → Transition. ; q_0 = initial state.
F → set of final state { where all string get finished! generally it is one but may be some time occur more than one.
F ⊆ Q

Example:- Design a finite state automata for a string baa!
baa! baaa! baaaa! etc....
but it will reject ba!x, b!x



States \ Input	a	b	!
q_0	ϕ	q_1	ϕ
q_1	q_2	ϕ	ϕ
q_2	q_3	ϕ	ϕ
q_3	q_3	ϕ	q_4
q_4	ϕ	ϕ	ϕ

Q.5] What is Phrasing? Compare Top-Down & Bottom-up approach with examples.

→ It is a method of analyzing a sentence to determine its structure according to the grammar to get meaning out from sentence. It is often a key step in various NLP tasks, such as language understanding, information extraction & machine translation.

Aspect	Top- down Phrasing	Bottom Up Phrasing.
Starting point	Begins at highest level structure [eg. sentence]	starts with individual words.
Order of Processing	Process is from top (sentence) to bottom [words]	Process is from bottom [words] to top [sentence]
Example	Sentence :- "The quick brown fox jumps over the lazy dog."	
Step ①	starts with sentence-level rule, such as S → NP, VP	Starts with intial individual words constituents.
Step ②	Apply rule to break down the sentences. eg:- NP → Det, Adj, Noun	Combine word based on grammar rules, eg:- Adj N → "quick brown fox"
Step ③	Continue breaking down until individual words match grammar rules	keep combining constituents until a sentence-level structure is found
outcome :-	Produces a phrase tree that starts with S & break down into constituents.	Produces a phrase tree that starts with individual words and builds upto the S.

The choice of phrasing technique depends on specific requirements of the phrasing task, grammar being used & efficiently of phrasing algorithm.

① 6] Describe Open class words & closed class words in English with examples.

→ * Closed class words

closed class words are those with a relatively fixed / number of words & we rarely add new words to these POS

such as prepositions, closed class words are generally functional words like of, it, & or, ya which find to be very short, occur frequently and often have structuray uses in Grammar.

eg:- of closed class:
- determinants :- a, an, the pronouns :- he, she, I
- preposition :- on, under, over, by, at, from, to, with.
- pronouns ;conjunction, articles, Auxiliary verbs, particles

* Open Class Words :-
Open class words are also known as content words. They are called "open" because new words can be added to this category over time. Open class words can be modified, combined, and expanded to create richer and more complex expressions.
eg:- Open class words are :-
- Noun :- dog, cat, love
- Verbs :- run, eat, sing
- Adjective, adverbs, interjections.

=> Sentence that illustrated both the classes :-
"The quick brown fox jumps over the lazy dog."

in above sentence, the open class words [in ~~black~~ underlined] are the content that carry primary meaning of sentence, while [non-underlined] are closed class that provide structures, grammar & relation b/w the ~~the~~ content words.

Q.7] What is a Language model ? Write a short note on the N-Gram model

→ Language model estimate the relative likelihood of different phrases and are useful in many different NLP applications.
Eg:- They have been used twitter, Bots for 'robot' accounts to form their own sentences.
→ The goal of the probabilistic language ~~programs~~ modelling is to calculate the probability of sentence of sentence of words.

$$P(w) = P(w_1, w_2, w_3, \ldots w_n)$$

N-Gram Model is nothing but a sequence of 'n' words

"I am the king"

* 1-Gram [unigram] => (I) (am) (the) (king)
  ie, Single word is a Sentence.
* 2-Gram [Bigram] => It is two words sequence [bigram]
  ie, (I am) (am the) (the king)
* 3-Gram [trigram] => It is a three words sequence
  ie; [I am the] [am the king]
* Unigram Probability => $P(w) = \text{Count}(w) / N$
  ∴ where N = words in entire corpus
  w = can be any word.

* Bigram / trigram = $P(A/B) = P(A \cap B) / P(B)$

Bigram => $P(w_i / w_{i-1}) = \text{count}(w_{i-1} w_i) / \text{count}(w_{i-1})$

trigram => $P(w_i / w_{i-2}, w_{i-1}) = \text{Count}(w_{i-2}, w_{i-1}, w_i) / \text{Count}(w_{i-2}, w_{i-1})$

Q.8] Discuss various approaches to perform POS tagging.

→ • Part-of-Speech [POS] Tagging is a process of converting a sentence to form - list of words, list of tuples [word, tag].

Approaches to perform POS tagging are:-

A] Rule-based POS tagging:-

- Rule-based taggers use dictionary or lexicon for obtaining possible tags for tagging each word.
- If the word has more than one possible tag, then rule-based taggers use hard-written rules to identify the correct tag.

Eg:- If the preceding word of a word is article or adjective, then the word must be noun.

Rule-based POS tag

Rule-based POS tagging can be visualized by its two-stage architecture:-

(i) First stage :- Here dictionary is used to assign each word of a list of potential parts-of-speech.

(ii) Second stage :- Here, the method use large list of hand-written disambiguation rules to sort down the list of to a single part-of-speech for each word

* Properties for Rule-based Pos tagging :-

(i) These taggers are knowledge-driven taggers.

(ii) The rules in Rule-based POS tagging are done manually.

(iii) There are around 1000 number of rules.

B] Stochastic POS Tagging :-

- Stochastic model is the model that includes frequency or probability
- Different approaches to the problem of a model that includes probability to the problem of part-of-speech tagging is referred to as stochastic tagger.
- The simplest stochastic taggers uses the following approaches for POS tagging :-

(i) Word-frequency approach :-

- Here, the stochastic taggers disambiguate the words. ie, the words based on the probability that a word occur with a particular tag.
- The tag that is encountered most frequently with the word in the training set is assigned to an ambiguous instance of word.
- The main problem with this approach is that it may yield inadmissible sequence of tags.

(ii) Tag sequence Probabilities :-

- This is a different approach of stochastic tagging. Here the tagger calculates the probability of a given sequence of tags occuring

* Properties of stochastic POS tagging :-
(i) The POS tagging is based on the probability of tag occuring.
(ii) Training corpus is required here.

c] Transformation - based Tagging [TBL]
- Transformation based tagging is also called Brill tagging.
- It is the instance of the transformation - based learning. It is a rule-base algorithm for automatic tagging of POS to the given text.

Working :-
- To understand the concept governing transformation - based taggers, we have to understand the working of transformation - based learning.
- We mention below the steps of the working of TBL :-

(i) Begin with the solution
(ii) Choosing most beneficial transformation.
(iii) Applying to the problem.

Advantages of TBL :-
(i) We have to learn small set of simple rules and these rules are enough for tagging.
(ii) Transformation - based tagger is much faster than Markov - model tagger.

Q.9] Explain Derivational & Inflectional morphology in detail with examples :-
→ * Inflectional Morphology :-
- It is one of the ways to combine morphemes with stems.
- Inflectional morphology conveys grammatical information, such as number, tense, agreement to case.
- One can say that the root word [stem] is inflected to form other words of same meaning and category.
- Inflection creates different forms of the same words.
- Eg :- Inflectional morphemes are suffixes that get added to a word, thus, adding a grammatical value to it.

- Plural : Bikes , cars
- Possession :- Boy's , Girl's
- Tense :- cooked
- Compasison :- Faster ; slower ; quicker.
- Superlative :- Fastest ; slowest , Quickest

* Types of Morphology :-

(i) -s	3rd person singular Present	he waits
(ii) -en	past participle	he has eaten
(iii) -s	Plural	Three tables.
(iv) 's	Possessive	Holly's cat.
(v) -er	Comparative	You are taller.

* Derivational Morphology :-

- Derivation is the process of creating new words from a steam/ base form of a word.
- One of the most common ways to derive new word is to combine derivational affixes with root words [stem].
- The word formed through derivational morphology may be a stem for another affix.
- New words are derived from the root word in this type of morphology.
- Eg :- Black + Bird combine to form black birds.
  dis connect combine to form disconnect.

Eg of English derivational patterns and their suffixes

(a) adjective - to - noun , - ness [slow → slowness]
(b) adjective - to - verb , -en [weak - weaken]
(c) adjective - to - adjective - ly [personal - personally]
(d) nown - to - adjective - al [recreation - recreational]

Q.10] Explain Hidden Markov Model in detail.

→ • The hidden Markov Model [HMM] is another type of Markov model where there are a few states hidden

• It is a hidden variable model which can give an observation of another hidden state using Markov assumption.

• A hidden Markov model consist of five important components :-

(a) Intial probability distribution,

(b) One or more hidden states,

(c) Transition probability distribution :- The transition matrix is used to show the hidden state to hidden state transition probabilities.

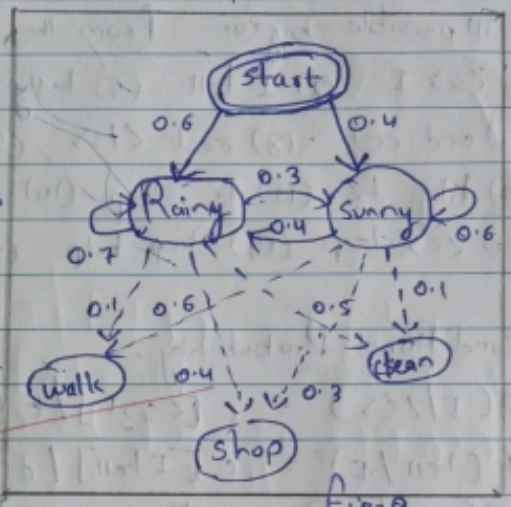


fig.a

(d) A sequence of observation

(e) emission probabilities :- A sequence of observation likelihood, also called as emission probabilities ; Each observation expresses the probability of an observation, generated from a state.

From fig a,

(i) There are two hidden states such as rainy & Sunny. These are hidden states because the process output is whether the person is shopping, walking or cleaning.

(ii) The sequence of operation is shop, walk & clean.

(iii) An initial probability distribution is start probability.

(iv) Transition probability is transition of one state [rainy or Sunny] to another state given the current state.

(v) Emission probability is the probability of observing the output.

Q.3] Consider the following corpus :- <s> I tell you to sleep & rest </s>
<s> I would like to sleep for an hour </s>
<s> Sleep helps one to relax </s>

List all possible bigrams. Compute conditional probabilities & predict the next word for the word "to"

→ $P(I|<s>) = C(<s> I) / C(<s>) = 2/3 = 0.667$

All possible bigrams from the given corpus:

(1) <s> I (2) I tell (3) tell you (4) you to (5) to sleep (6) sleep and (7) and rest (8) rest </s> (9) <s> I (10) I would (11) would like (12) like to (13) to sleep (14) sleep for (15) for an (16) an hour (17) hour </s> (18) <s> sleep (19) sleep helps (20) helps one (21) one to (22) to relax (23) relax </s>

Conditional Probability :-

$P(I|<s>) = C[<s> I] / C[<s>] = 2/3 = 0.667$

$P(\text{tell}|I) = C[I \text{ tell}] / C[I] = 1/2 = 0.5$

$P(\text{you}|\text{tell}) = C[\text{tell you}] / C[\text{tell}] = 1/1 = 1$

$P[\text{to}|\text{you}) = C[\text{you to}] / C[\text{you}] = 1/1 = 1$

$P[\text{sleep}|\text{to}) = C[\text{to sleep}] / C[\text{to}] = 2/3 = 0.667$

$P(\text{and}|\text{sleep}) = C[\text{sleep and}] / C[\text{sleep}] = 1/3 = 0.33$

$P[\text{rest}|\text{and}) = C[\text{and rest}] / C[\text{and}] = 1/1 = 1$

$P(</s>|\text{rest}) = C[\text{rest } </s>] / C[\text{rest}] = 1/1 = 1$

$P(\text{would}|I) = C[I \text{ would}] / C[I] = 1/2 = 0.5$

$P(\text{like}|\text{would}) = C[\text{would like}] / C[\text{would}] = 1/1 = 1$

$P(\text{to}|\text{like}) = C[\text{like to}] / C[\text{like}] = 1/1 = 1$

$P(\text{for}|\text{sleep}) = C[\text{sleep for}] / C[\text{sleep}] = 1/3 = 0.33$

$P(\text{an}|\text{for}) = C[\text{for an}] / C[\text{for}] = 1/1 = 1$

$P[\text{hour}|\text{an}) = C[\text{an hour}] / C[\text{an}] = 1/1 = 1$

$P(</s>|\text{hour}) = C[\text{hour } </s>] / C[\text{hour}] = 1/1 = 1$

$P(\text{sleep}|<s>) = C[<s> \text{sleep}] / C[<s>] = 1/3 = 0.33$

$P(\text{helps}|\text{sleep}) = C[\text{sleep helps}] / C[\text{sleep}] = 1/3 = 0.33$

$P(\text{one}|\text{helps}) = C[\text{helps one}] / C[\text{helps}] = 1/1 = 1$

P (to | one) = c [one to] / c [one] = 1/1 = 1
P (relax | to) = c [to relax] / c [to] = 1/3 = 0.33
P (</s> | relax) = c [relax </s>] / c [relax] = 1/1 = 1

From above probability calculation.
P (sleep | to) = c [to sleep] / c [to] = 2/3 = 0.667
P [relax | to] = c [to relax] / c [to] = 1/3 = 0.33

The next word after to sleep.