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

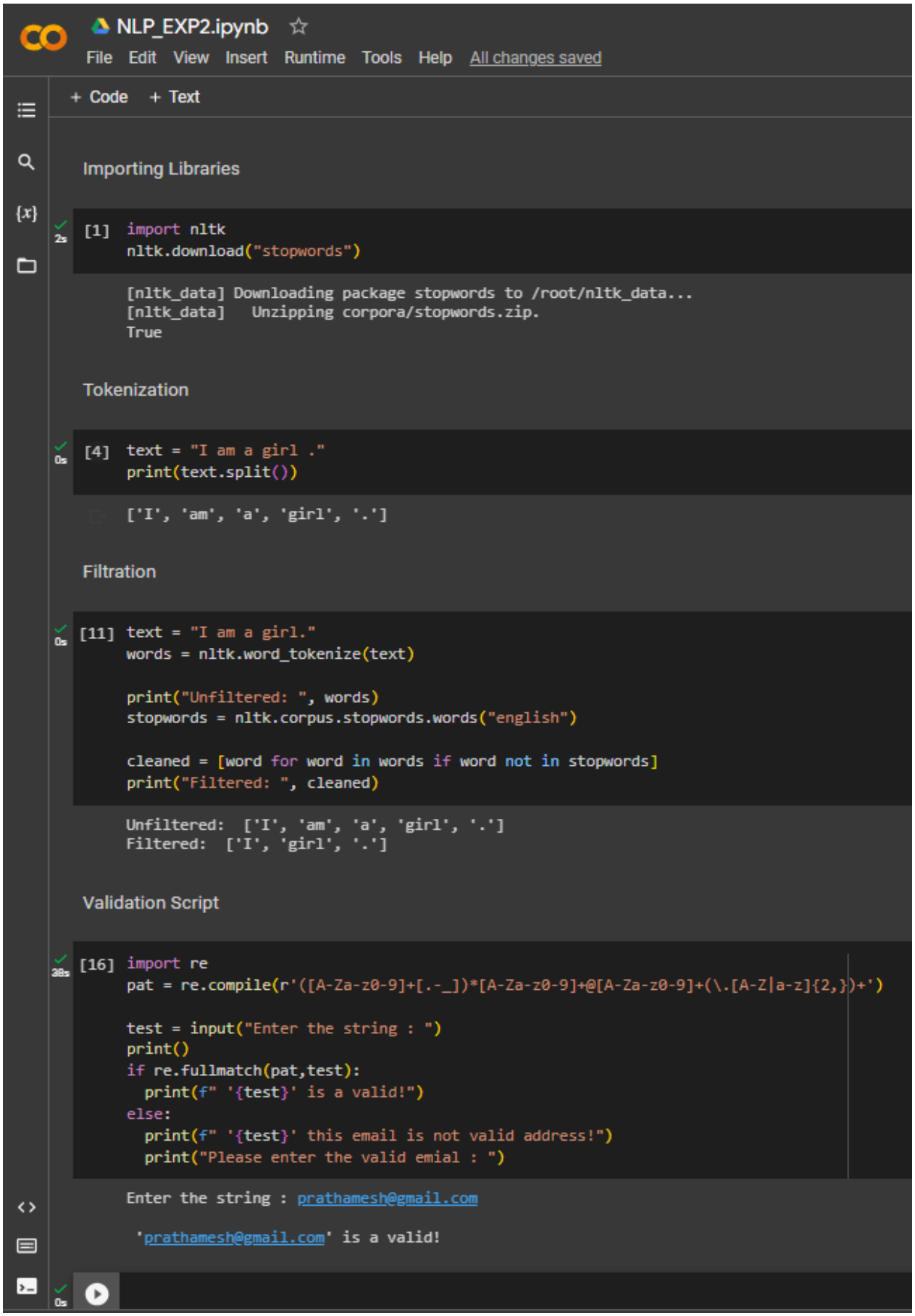
NAME	**PRATHAMESH S. CHIKANKAR**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML11**
SUBJECT	**NATURAL LANGUAGE PROCESSING LAB**
COURSE CODE	**CSDOL7011**
PRACTICAL NO.	
DOP	
DOS	

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	**PRATHAMESH S. CHIKANKAR**
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
[16] 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 : prathamesh@gmail.com
'prathamesh@gmail.com' is a valid!

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	**PRATHAMESH S. CHIKANKAR**
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	**PRATHAMESH S. CHIKANKAR**
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	**PRATHAMESH S. CHIKANKAR**
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	**PRATHAMESH S. CHIKANKAR**
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)

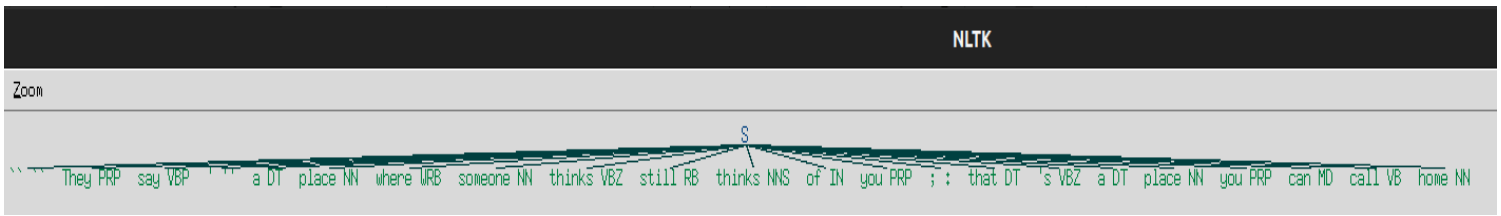
B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**PRATHAMESH S. CHIKANKAR**
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	**PRATHAMESH S. CHIKANKAR**
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	**PRATHAMESH S. CHIKANKAR**
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 (‘AI-Powered Autocompleter’).

Abstract:

Most of us have experienced the wonderful world of autocomplete, where you type the first few letters of what you’re looking for into a search engine and click on one of the search suggestions from the drop-down, but there’s a lot that goes on in the background to make this possible.The ability to select from a list of suggestions without having to type the complete term in the search.

Google set the standard, being the first organization to implement an autocomplete feature. Very quickly, users grew to expect predictive search. Amazon, Shopify, Spotify, and Walmart are some examples of sites and applications that have implemented predictive search capability to show popular queries, offer contextual suggestions, avoid typos, and filter suggestions based on a user’s location or preferences.

Introduction:

Today, organizations, big and small, offer some form of autocomplete.Predictive search technology attempts to anticipate search terms based on the behavior, previous searches, geolocation, and other attributes of the end user, as well as trending searches across all user sessions, and displays the terms as suggestions in or under the search field.

Autocomplete, autosuggest, and predictive search are often used interchangeably in the tech industry, especially in the UI/UX department.A search query is the term that a user types in the search bar of a page. Predictive search analyzes each letter of the search query as they are being typed and provides a set of suggested queries in a drop-down menu before the user has even finished typing.
Multiple machine learning and natural language processing algorithms and models are used to generate matches, starting with a simple string in order to identify, match, and predict the outcome of the unfinished search query. These suggestions help users to get results faster with less typing, which helps prevent typos and create queries that will return the desired results.

The algorithms behind these suggestions begin with simple string matching from static lists or existing data, and get increasingly complex by using natural language processing to manage typos and synonyms to help users get the desired results.

AI-Powered Autocompleter, often referred to as "autocomplete" or "predictive text," is a technology or feature that assists users in generating text by predicting what they are likely to type next and suggesting possible completions. It is commonly used in a variety of applications, including word processors, search engines, messaging apps, and more, to save time and improve typing efficiency.

- **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.

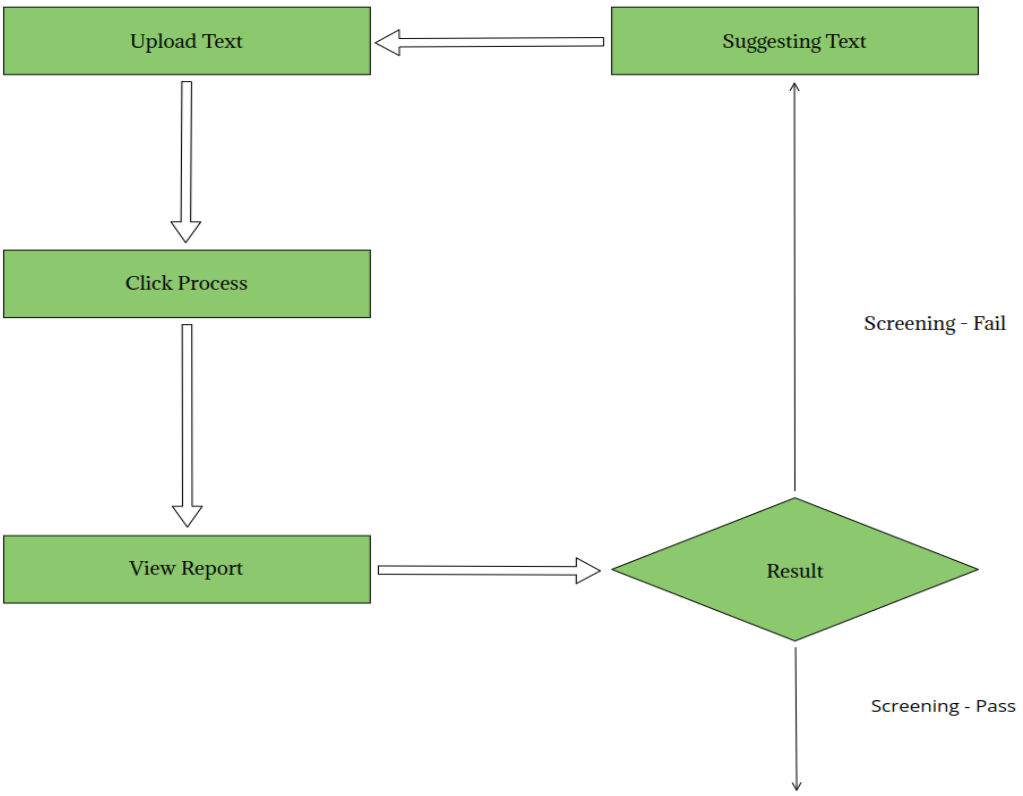
6) **Autocorrect:** Autocompletion systems often include autocorrect functionality, which can automatically correct misspelled words as users type. This helps improve the quality of the text and reduces errors.

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.

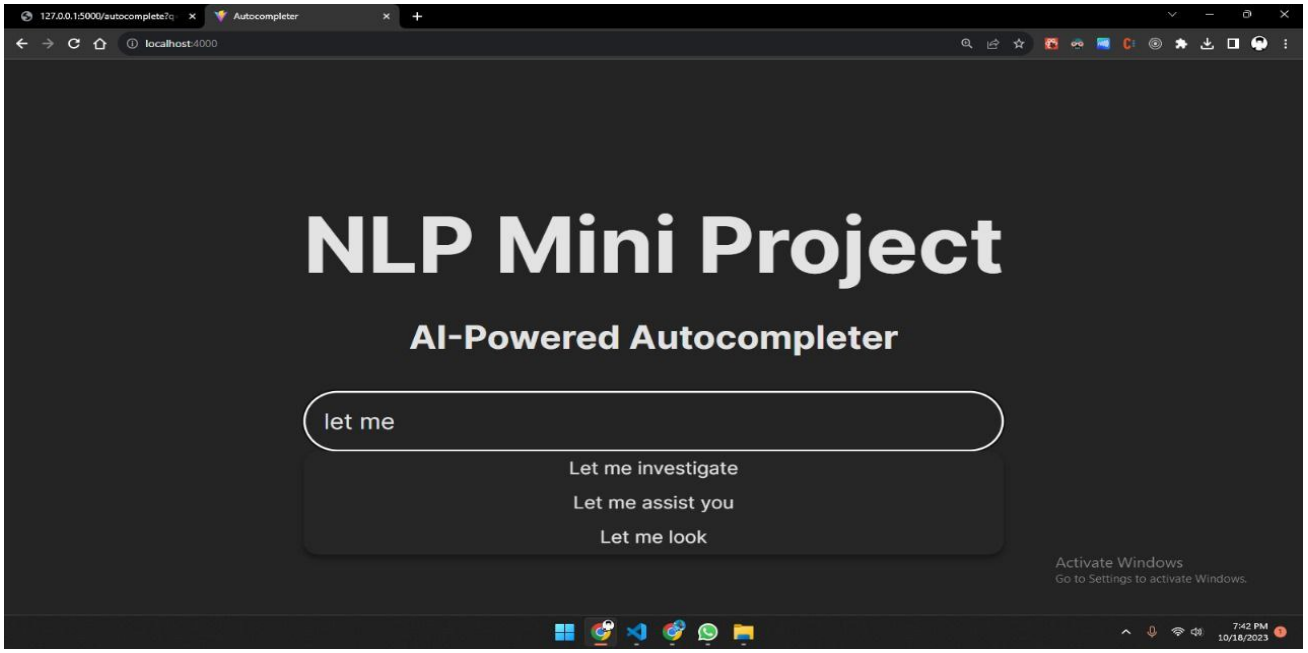
9) **Application Integration:** Autocomplete is integrated into various software applications, including web browsers, email clients, text editors, and messaging platforms. It enhances the user experience in these contexts.

Data Flow Diagram:

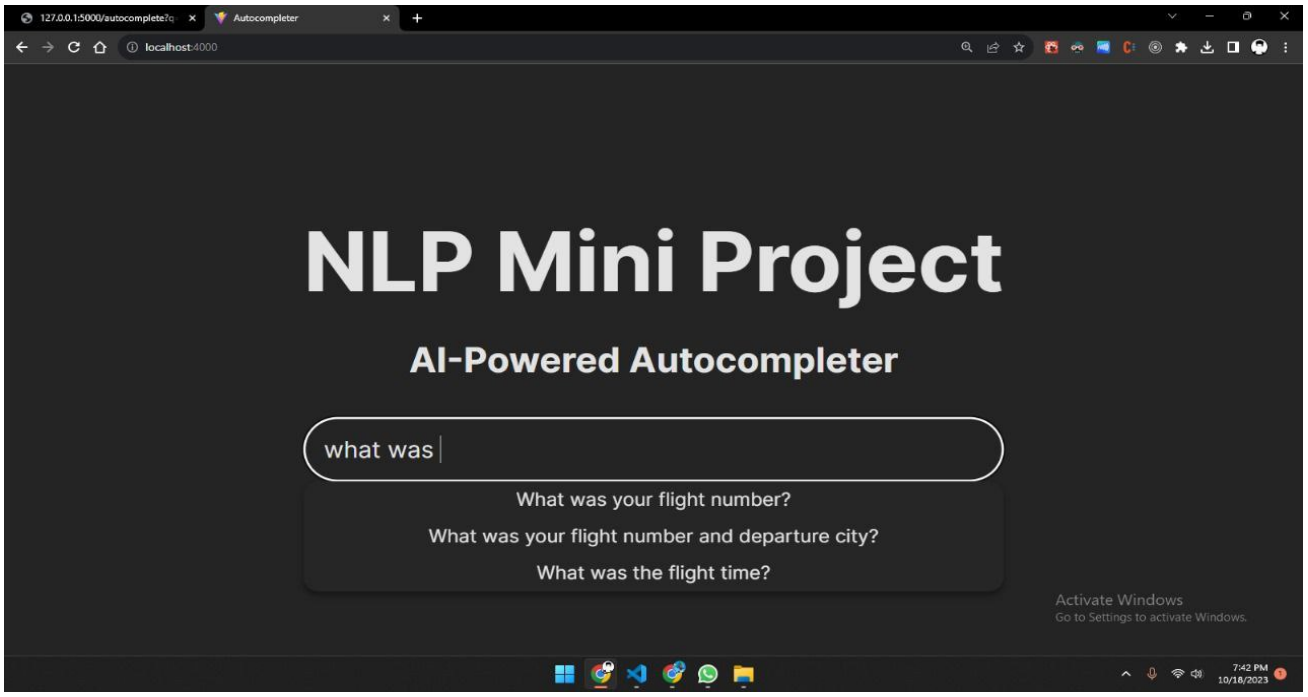


Output:

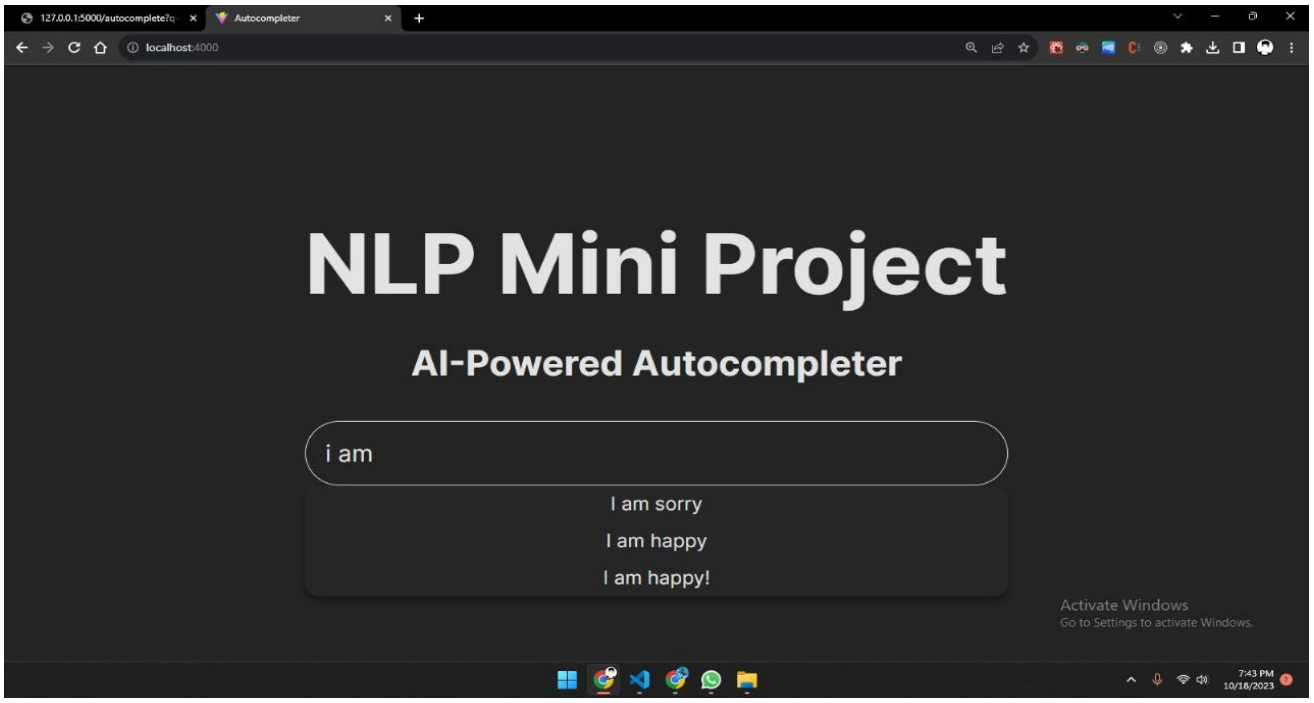
1) Example: let me



2) Example: what was



3) Example: i am



Conclusion: We have successfully implemented real world NLP Application in AI Powered Autocompleter.

Name :- Prathamesh S. Chikankar

Roll No. :- AIML11 Branch: CSE- (AI & ML)

Subject :- Natural Language processing (NLP)

Topic :- Assignment No. 01

Date :- August '23 Sign :- Prathu

Q.1. What is NLP?

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.

Explain different phases involved in the NLP process with suitable examples.
The different phases involved in the NLP process are as followed:

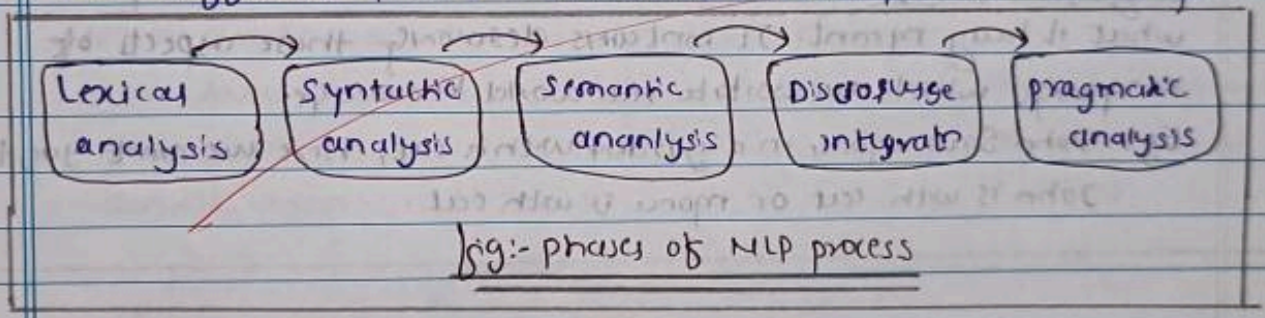


Fig:- phases of NLP process

> 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." is divided into
["sudham", "is", "Genious", "."]

> Syntactic analysis: It involves analysis of words in sentence for grammar and ordering words in way that show relationship among words.
Eg:- The sentence such as The school goes to boy is rejected by English syntactic analyzer.

PTO

> 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 syntactic structures and objects in the task domain.
Eg :- The semantic analyzer neglects sentence such as "hot ice-cream".

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

> pragmatic analysis : During this, what was said is re-interpreted on what it truly meant. It contains designing those aspects of language which necessitate real world knowledge.
Eg :- John saw mona in a garden with a cat, here we can't say that John is with cat or mona is with cat.

Q.2. What is stemming ?
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 is algorithm is a language linguistic normalization process in which the variant forms of a words are reduced to standard form.
Eg :- "eating", "eats", "eaten" => eat

Explain porters stemming algorithm in detail.
It is common algorithm for stemming English.

PTO

The algorithm consist of 5 sets of rules appord in order
- phases appord sequencially
- each phase consist of SET of command.

Sample Convention : of the rules in a compound command, select the one that appies to the longest suffix.

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

With the definition all words are of the form (C)(VC)^m(V)
C = String of one or more consonant
V = String of one or more vowels
m = measure of words or word part when represented in the form of VC
VC = combination of vowel and consonant

eg:- How calculate value of 'm'
The combination TREE, BY, TR doesn't contain ∴ m=0 } There is combination of consonant & vowel
VC form
TROUBLE, OATS, TREES, IVY ∴ m=1
TROUBLES, PRIVATE, OATEN ∴ m=2

* Rules for removing suffix
To use porters stemmer, there are certain rules to follow
The rules are of the form (condn) S1 → S2 ∴ S1 & S2 are suffixes.
eg:- word = REPLACEMENT → if rule is (m>1) EMENT →
in this S1 is ement & S2 is null ∴ m>1
String part will be replace by null REPLAC m=4.

The condn may also contain following :

PTO

m	the measure of the 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 (not w, x or y)

the condn 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'.

★ Step 1:

SSES ⇒ SS eg:- caresses → caress ; IES → I
ponies → poni ; ties → ti
SS ⇒ SS eg:- pass → pass & S ⇒ ε (null) eg:- cats → cat

★ step 2: stemming with condn

① (m>0) EED → EE eg:- agreed → agree ...if condn verified
stem contains vowel feed → feed ...if not verified
② (*v*) ED → ε (null) eg:- plastered → plaster ...if verified
bled → bled ...if not verified
③ (*v*) ING → ε eg:- motoring → motor ...if verified
sing → sing ...if not verified

★ step 3: Clean up

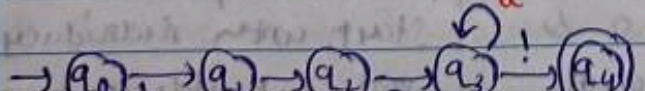
these rules are run if second & third rule in 1b apply)
① AT → ATE eg:- conflat (ed) → conflate
② BL → BLE eg:- Troubling) → Trouble
③ [*d & !, (*L or *s or *z)] → single letter
eg:- hopp (ing) → hop ... condn verified
fall (ing) → fall .. condn not verified

PTO

④ (m=1 & *0) → E
eg:- fil(ing) → file ... (word verified
fwl → fwl ... (word 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: $(Q, \Sigma, \delta, q_0, F)$
Q → set of finite state ; Σ = set of alphabets
δ → transition ; q_0 = initial state
F → set of final state { where any string get finished! generally it is one but may be some time occur more than one.
$F \subseteq Q$

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



I/p / States	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 parsing?

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, and machine translation.

Compare Top-Down and Bottom-up approach with examples.

Aspect	Top-down parsing	Bottom-up parsing
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 all individual words constituents.
Step 2:	apply rules 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.
Example outcome:	produces a parse tree that starts with S & breaks down into constituents.	produces a parse tree that starts with individual words & builds up to S.

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

PTO

Q. 6. Describe open class words and 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 you which tend to be very short, occur frequently & often have structural uses in grammar
eg:- of closed class
- determiners: a, an, the pronouns: he, she, I
- preposition: on, under, over, by, at, from, to, with
- pronouns, conjuction, 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-
- Nouns: 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 (i.e black pen) are the content that carry primary meaning of sentence, while (blue pen) are closed class that provide structure, grammar & relation b/w the content words.

PT

Q.7. what is a language model?

language model estimate the relative likelihood of different phases & are useful in many different NLP applicat^n.

eg:- They have been used twitty, Bots for 'robot' accounts to form their own sentences.

→ The goal of the probabilistic language modelling is to calculate the probability of a sentence of sequence of words.

$$P(W) = P(w_1, w_2, w_3, \cdots w_n)$$

write a short note on the N-Gram model

N-gram model (language) is nothing but a sequence of 'n' words.

"I am the KING"

* 1-gram (unigram) ⇒ (I) (am) (the) (KING)
i.e. single words is a sentence.

* 2-gram ⇒ It is two words sequence (bigram)
i.e. (I am) (am the) (the KING)

* 3-gram (trigram) ⇒ It is a three words sequence
(I am the) (am the KING)

* Unigram probability ⇒ $P(w) = count(w) / N$
where N = words in entire corpus
w = can be any word

* Bigram / Trigram ⇒ $P(A|B) = P(A \cap B) / P(B)$

Bigram ⇒ $P(w_i / w_{i-1}) = count(w_{i-1}, w_i) / count(w_{i-1})$

Trigram ⇒ $P(w_i / w_{i-2}, w_{i-1}) = count(w_{i-2}, w_{i-1}, w_i) / count(w_{i-2}, w_{i-1})$

PT

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

part-of-speech (POS) Tagging is a process of converting a sentence to form a list of words, list of tuples (where each tuple is having a form (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 hand-written rules to identify the correct tag.
- For eg, if the parameters preceding word of a word is article or adjective, then the word must be noun.
- Rule-based architecture of pos tagging can be visualized by its two-stage architecture
  1) First stage: Here dictionary is used to assign word a list of potential part-of-speech.
  2) Second stage: Here, the method use large files list of hand-written
- disambiguate rules to sort down the list to single pos for each word.

* properties of rule-based POS tagging
  1) These taggers are knowledge-driven taggers.
  2) There are around 1000 no. of rules.

B) Stochastic POS tagging

- Stochastic model is the model that include frequency or probability.
- Different approaches to the problem of a model that includes probability to the problem of pos tagging refer to stochastic tagger.

1) word-frequency approach:
   - Here the stochastic taggers disambiguate the words, i.e. the

PTO

words based on the probability that a word occurs with particular tag.
- The tag is encountered most frequently with word in training set.

2) Tag sequence probability:
- This is a different approach of stochastic tagging.
- Here the tagger calculates the probability of given sequence of tags occurring.

* properties of stochastic POS tagging
1) POS tagging is based on probability of tag occurring
2) 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 rule-based algorithm for automatic tagging of POS to given text

* Working of TBL
- To understand concept governing transformation-based taggers, we have to understand working of TBL.
- Steps of working of TBL:
1) Begin with the solution
2) Choosing most beneficial transformation.
3) Applying to the problem.

* Advantages of TBL
1) We have to learn small set of simple rules & these rules are enough for tagging.
2) Transforming TBL is much faster than markov-model tagger.

Q.9 Explain Derivational and Inflectional morphology in details with examples.

PTO

Inflectional morphology:

It is one of the ways to combine morphemes with stems.

1) Inflectional morphology conveys grammatical information, such as number, tense, agreement or case.
2) One can say that the word (root) stem is inflected to form other words of same meaning & category.
3) Inflection creates different forms of same word.
4) Examples of inflectional morphemes.
   - Inflectional morphemes are suffixes that get added to a word, thus, adding a grammatical value to it.
   - plural: Bikes, cars, Trucks, Lions, etc..
   - possessive: Boys, Girl's, Man's, Mark's, etc..
   - Tense: cooked, played, market, waited, etc..
   - comparisons: fast, slower, quicker, etc..
   - Superlative: fastest, slowest, quickest, etc..

* Types of morphology

1) -s → 3rd person singular present → he waits
2) -en → past participle → he has eaten
3) -s → plural → Three tables
4) 's → possessive → Hollys cat
5) -er → comparative → you are taller

Derivational morphology:

Derivation is the process of creating new words from a sentence of stem/base form of a word.

1) one of the most common ways to derive new word is to combine derivational affixes with root words (stems)

The new words formed through derivational morphology may be a stem for another affix.

PTO

2) New words are derived from the words in this type of morphology.
3) Example:
Black + bird combine to form black bird, dis + connect combine to form disconnect.
More eg of English derivational patterns & their suffixes.
a) adjective-to-noun, -ness (slow → slowness)
b) adjective-to-verb, -en (weak - weaken)
c) adjective-to-adjective, -ly (personal - personally)
d) noun-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 hidden variable model which can give an observation of another hidden state using Markov assumption.
- A hidden markov model consists of five important components:
a) initial 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 probability.
d) A sequence of observation.
e) Emission probabilities: A sequence of observation likelihood, also called as emission probability. Each observation expresses the probability of an observation, generated from a state.

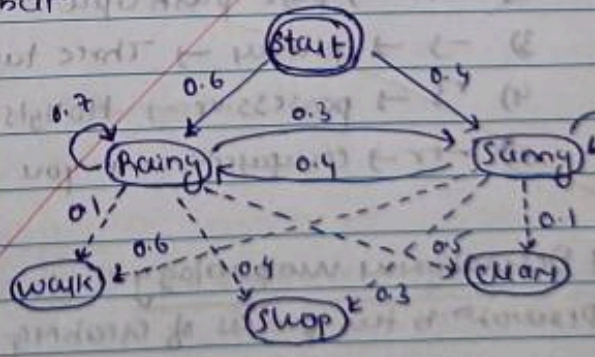


fig: state transitive diagram

PTO

from fig. a)

i) There are two hidden states such as rainy and sunny. They are hidden states because the process output is whether the person is shopping, walking or cleaning.

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

iii) 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 probability of observing the output.

Q.3. Consider the following corpus

<s> I tell you to sleep and 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 probability and predict the next word for the word "to".

$P(I \mid <s>) = C(<s> I) / C(<s>) = 2/3 = 0.667$

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

$P(I \mid <s>) = C(<s> I) / C(<s>) = 2/3 = 0.667$

$P(tell \mid I) = C(I\ tell) / C(I) = 1/2 = 0.5$

$P(you \mid tell) = C(tell\ you) / C(tell) = 1/1 = 1$

$P(to \mid you) = C(you\ to) / C(you) = 1/1 = 1$

PTO

P(sleep/to) = C(to sleep) / C(to) = 2/3 = 0.667
P(and/sleep) = C(sleep and) / C(sleep) = 1/3 = 0.33
P(rest/and) = C(and rest) / C(and) = 1/1 = 1
P(</s>/rest) = C(rest </s>) / C(rest) = 1/1 = 1
P(would/I) = C(I would) / C(I) = 1/2 = 0.5
P(like/would) = C(would like) / C(would) = 1/1 = 1
P(to/like) = C(like to) / C(like) = 1/1 = 1
P(for/sleep) = C(sleep for) / C(sleep) = 1/3 = 0.33
P(an/for) = C(for an) / C(for) = 1/1 = 1
P(hour/an) = C(an hour) / C(an) = 1/1 = 1
P(</s>/hour) = C(hour </s>) / C(hour) = 1/1 = 1
P(<s>/sleep) = C(<s> sleep) / C(<s>) = 1/3 = 0.33
P(helps/sleep) = C(sleep helps) / C(sleep) = 1/3 = 0.33
P(our/helps) = C(helps our) / C(helps) = 1/1 = 1
P(to/our) = C(our to) / C(our) = 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.

Name :- Prathamesh S. Chikankar

Roll No. :- AIML11 Branch :- (SE- (AI (ML)

Year :- BE Subject :- Natural Language Processing (NLP)

Topic :- Assignment No. 02

Date :- October'23 Sign :- Prathu

Q.1. Explain with suitable Examples relat^n slip blw word meaning.

a) Homonymy: Homonymy refers to the relat^nslip blw words that have the same form (spelling and pronunciat^n) but different meanings. These words are unrelated in terms of their meanings.

for example:

① "bank" (meaning a financial institut^n) and "bank" (meaning the side of river).

② "bat" (an animal) & "bat" (a sports equipment)

b) polysemy: polysemy refers to the relat^nslip blw words that have multiple related meanings, all derived from a common origin. These meanings are often related in some way.

for example:

①"run" can mean moving quickly or operating a machine. polysemy means many signs.

c) Antonymy: Antonymy refers to the relationslip blw words that have opposite meanings. Antonomys are words that are contrasting or opposing in meaning.

for example:

① "hot" and "cold" are antonyms.

② "happy" and "sad" are antonyms.

d) Hyponymy: Hyponymy is a relationslip in which one word (the hyponymy) is a specific instance or type of another word (the hypernym). Hyponyms are a subset of the hypernymy meaning.

for example:

① "rose" is a hyponym of the hypernym "flower".

② "poodle" is a hyponym of the hypernym "dog".

PTO

e) Synonymy: Synonymy refers to the relationship b/w words that have similar or identical meanings. Synonyms can be used interchangeably in many contexts.
For example:
① "happy" and "joyful" are synonyms.
② "big" and "large" are synonyms.

Q.2. Define

> Discourse analysis: Discourse analysis is the practise of analyzing texts or languages that entails understanding social interaction and text interpretation. Dealing with anaphoresis, n-grams, tenses, linguistic features page layouts and other elements can be a part of discourse analysis. One unit of discourse is a series of sentences.

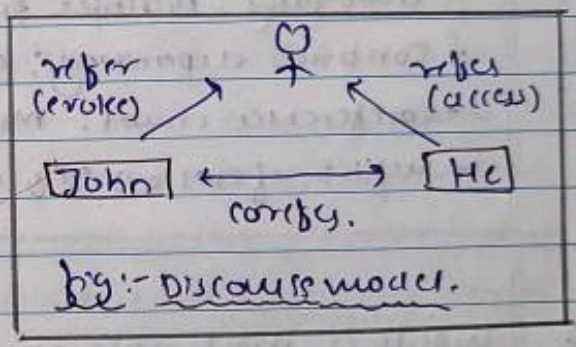
> pragmatic analysis: pragmatics is the study of how people use language in context to convey meaning. pragmatic analysis involves examining how language users employ implicatures, presupposition, speech acts and context to understand & interpret utterances.

> Discuss ~~dictionary based~~ Reference resolution problem in detail.
This problem involves determining the referent (the entity or object) that a word or phrase refers to within a given context. It can be challenging because words like pronouns (eg. he, she, it) or definite descriptions (eg. book) often require context to identify their reference referents.
> The discourse level is rich with occurrences in language. Think about the conversation in the example,

PTO

John went to Bills car dealership to check out an Acura Integra. He looked at it for about an hour.

- what do pronouns like "he" and and "it" mean? The reader might not have issue deducing that he refers to John and not Bill, and that it refers to the Integra and not Bills auto business, but for a computers?
- The mechanism through which speakers ability turns like 'John' & 'he' in passage (5.1) to signify a person called 'John' is the subject of this section's consideration of the reference problem. Reference Resolution is a process by which when speakers use expressions like 'John' and 'he' in passage (5.1) to denote a person named 'John' this relation is identified.



Types :-

① Indefinite noun phrases :
- unfamiliar entities
- the determiner 'a' (or 'an') is used to indicate indefinite reference most frequently.
eg:- I saw 'an' Acura Integra today.

② definite noun phrases :
- entity that identifiable to the hearer.
- refer represented as discourse models.
eg:- The fasty car in Indianapolis 500 was an Integra.

③ pronouns :
- another type of definite reference
eg:- I saw an Acura Integra today. It was white and needed to be washed.

④ Demonstratives: simple demonstrative pronouns like 'it' function differently than demonstrative pronouns like 'this' and 'that'. They may show up both by themselves & as determiners, such that 'this Aroma' and 'that Aroma'.
eg:- Bob (pointing): I like 'this' better than 'that'.

⑤ Names: prevalent type of referring expression & include names of persons, org. and places. Name may refer both literal & new entities.
eg:- 'Miss Woodhouse' certainly has not done long justice.

> challenges in Reference resolution.
- ambiguity: multiple possible interpretations.
- context dependency: depends on broader context of discourse.
- coreference chain: multiple words refers to same entity.
- implicit reference: rely on shared knowledge by authors.

Q. 3 Write a short note on

1. Machine Translation:

machine translation is the process of converting the text from one language to the other automatically w/o or minimal human intervention.

example:
- English - Sudham likes ice-cream.
- Sanskrit - सुधाम आइसक्रीम रोचते।

machine translator aims to bridge language barriers and enable communication b/w speakers of different languages.

Examples of machine Translator systems include-
Google translate and DeepL.

PTO

2. WordNet:

WordNet is a lexical Database and resource for natural language processing (NLP) and linguistics. It groups words into sets of synonyms (synsets) and provides structured relationships b/w words, including hypernyms (superordinate categories) and hyponyms (subordinate categories). WordNet is commonly used in tasks like word sense disambiguation and text classification.

example: eg synsets for "car".

- synset 1: car, auto, automobile, machine, motor
- synset 2: car, railcar, railway car, railway car, railroad car.

Q.4. What do you mean by word sense disambiguation (WSD)?

The task of word sense disambiguation is to examine word tokens in context and specify which sense of each word is being used. WSD, a challenging in NLP, is the process of figuring out which "sense" (meaning) of a word is activated by the use of a word in a certain context.

eg:- WordNet lists all possible senses example for a 'pen'.

- pen: writing implement from which ink flows.
- pen: enclosure for confining animals
- playpen: portable enclosure in which baby left to play.
- pen: female swan.

Discuss dictionary based approach for WSD: (knowledge-based)

These avoid the use of corpus-based evidence in favour of dictionaries, thesaurus and lexical knowledge bases.

> The first dictionary-based approach is the Lesk method. (1986).

It is a predicted on the idea that words employed in tending

PTO

in a text have a relationship to one another, a relationship that may be seen in the definitions of the terms and their senses.
- the usage of general-word-sense relatedness and computing the semantic similarity of each pair of word senses based on a specific lexical knowledge-base, such as WordNet, are alternative to using the definition.
- selection preferences (also known as selectional constraints) might be helpful. For instance one can identify that the phrase I am cooking bass (i.e. it is not a musical instrument) doesn't apply to the activity of cooking.

Q.5. Explain Hobbs algorithm for pronoun resolution

One method for pronoun resolving is the Hobbs's algorithm. The syntactic parse tree of the sentences form forms the foundation of the algorithm.

only a few sentences can be referenced with pronouns, and entities that are closest to the referring word are more significant than those are farther away.

- Example : Let is look at two sentences:

- sentence 1 (S1) : Jack is an engineer.
- sentence 2 (S2) : Jill likes him

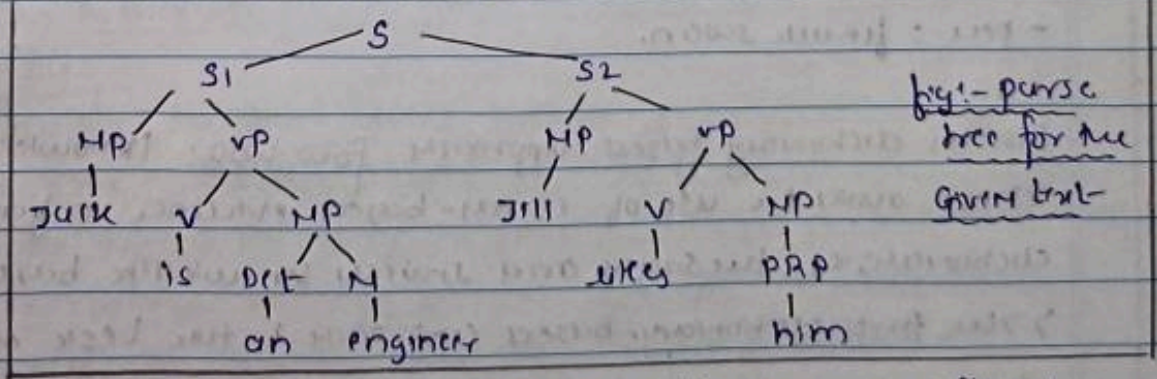


fig:- parse tree for the given text-

the probing identified is to resolve the probing pronoun 'him'.

PTO

Based on the algorithm: (explanation with example)

(1) Go back to 'PRP' to 'NP' from 'him' and thus 'NP' will be called as 'X' and the path 'him' to 'PRP', to 'NP' to 'P'.

(2) As there is no path to the left of 'P', we again go back from 'NP' to 'VP' to 'S2', which is our new 'X'.

(3) Due to the syntactic restriction imposed by Binding theory, it does not, however, shrug that branch. According to binding theory, a non-reflexive cannot refer to the subject of the most immediate clause in which it appears. However, reflexive can. Reflexive words include such like himself, herself themselves, etc.. So we again go backwards to 'S', becomes my new 'X'.

(4) From 'S' we move to 'S1', which is our new 'X'.

(5) Now we have encountered 'NP' when search the path. This 'NP' does not violate any constraints hence, this is our solution or answer. That is, the 'him' in the given sentences refers to 'Jack'.

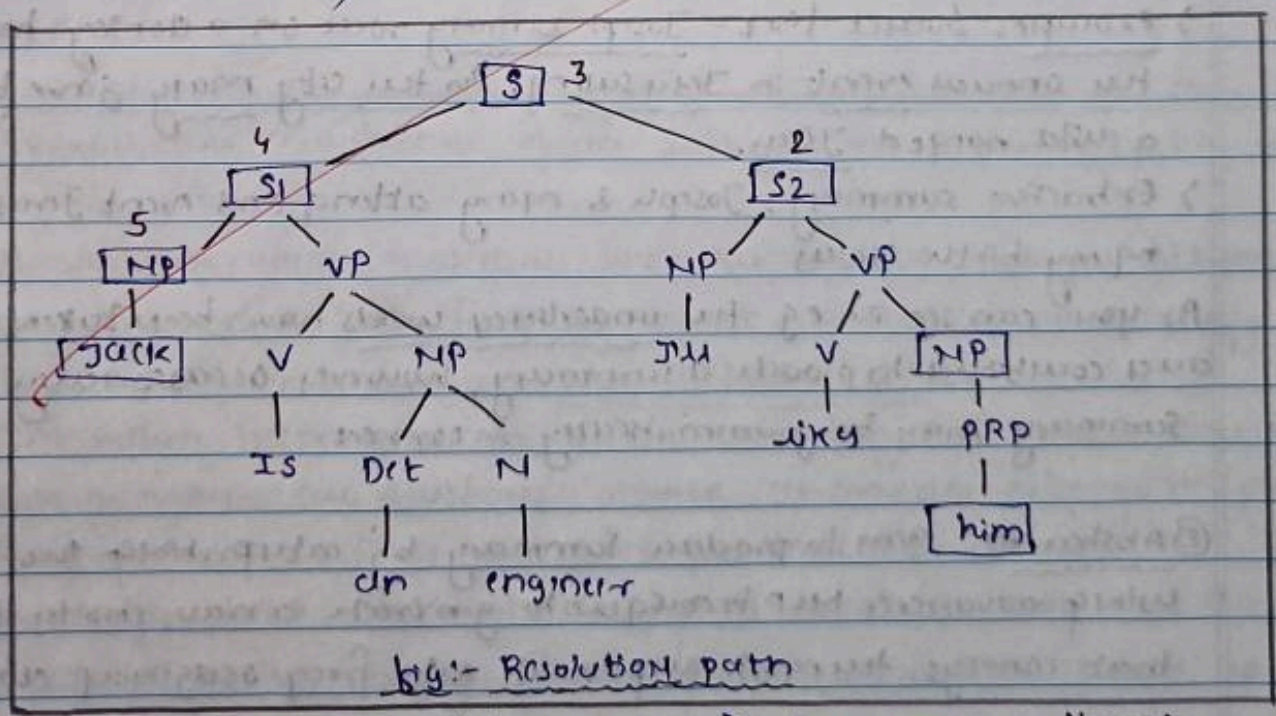


fig:- Resolution path

shows the pronoun resolution path for the pronoun 'him'.

Q.6. Explain Text summarization in detail.

Condensing a lengthy text into a manageable length while maintaining the essential information components and the meaning of the content is known as summarization

> Text summarization has significant uses in a variety of NLP-based activities including text classification, question answering, summarizing legal texts, summarizing news and creating headlines.

> Automatic text summarizing is the process of creating a succinct, fluent summary w/o the assistance of a human while maintaining the original text's meaning.

> Text summarization types :- Extractive and Abstractive

① Extraction : Using a scoring algorithm, extractive summarizers select phrases from the source text to create a meaningful summary. Extractive summaries include the key phrases from the I/p, which may be a single document or collection of documents.

> Example : Source text - Joseph & Mary rode on a donkey to attend the annual event in Jerusalem. In the city Mary gave birth to a child named Jesus.

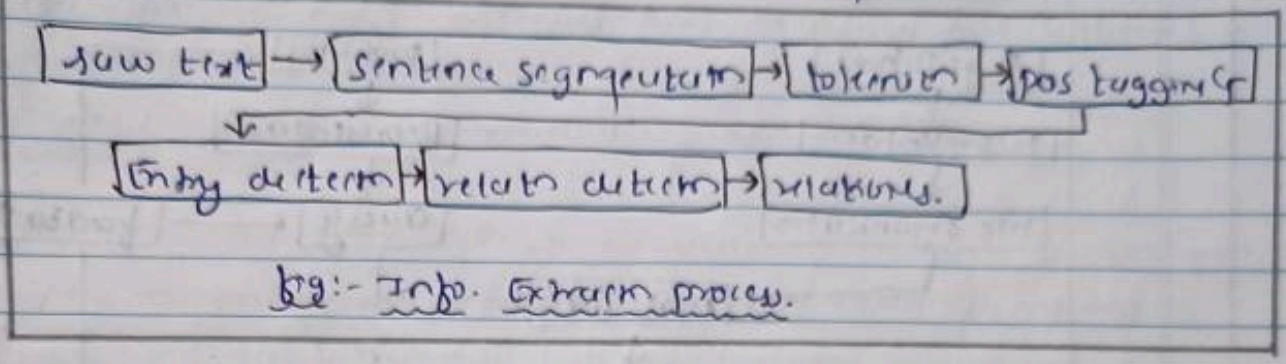
> Extractive summary : Joseph & Mary attend end event Jerusalem Mary birth Jesus

As you can see in e.g. the underlined words have been taken out and combined to provide a summary, however occasionally the summary may be grammatically incorrect.

② Abstraction : Aims to produce summary by interpreting the text using advanced NLP techniques to generate a new, shorter text that conveys the most important info. from original document They are not trained, them, to merely picking and choosing from the original text

PTO

In order to extract the relevant information from the text the following info. extraction process should be followed.



Eg:- Info. Extraction process.

Q.7. Explain Yarowsky bootstrapping approach of semi-supervised learning
pg.no.12

Q.8. Analyse the Naive Bayes classifier approach to word sense Disambiguation in NLP.
pg.no.13

Q.9. Explain the architecture of an Information Retrieval system with a neat diagram.

A sw programme that deals with the organisation, storage, retrieval, and evaluation of info. from document repositories, particularly textual info. is known as Info. Retrieval (IR).

The system helps users locate the data they need, but it does not clearly return the questions' answers. It provides information about the presence and placement of papers that may contain the necessary data. Relevant documents are those that meet the needs of the user. Only relevant documents will be passed up by the ideal IR system.

PTO

Working of a Typical IR System.

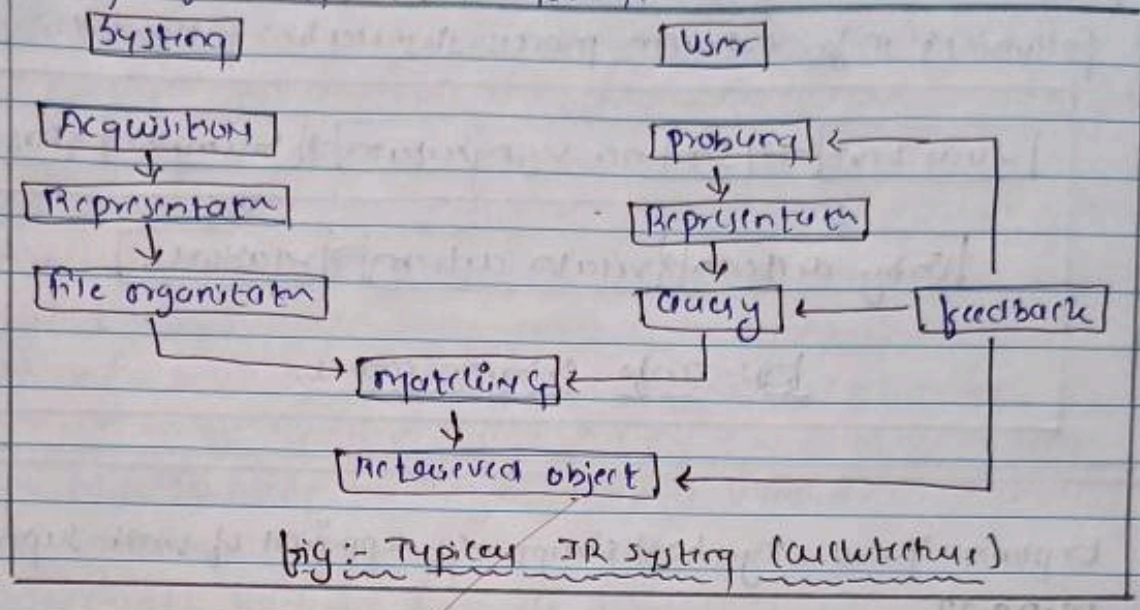


fig:- Typical IR system (architecture)

1) Acquisition: In this step, documents and other objective objects are chosen from a variety of web resources that are made up of text-based content. Web crawlers gather the necessary info. and store it in the database.

2) Representation: it includes indexing that uses both human & automatic procedures, as well as regularised & regulated vocabulary and free-text phrases.

3) File organization: There are two diff. sorts of file organization techniques, that is, sequential: By document data, contents docs.

4) Inverted: It includes a list of records broken down by term. A blending of the two.

5) Query: When a user submits a query to the system, an IR process begins, for instance, search terms in web search engines are formal declaration of info. needs. A query used for info. retrieval doesn't specially identify every thing in the collection. Instead, a number of things may match the query, maybe to varying degrees.

Q.10. Describe Text coherence.

Text coherence refers to the logical and organized flow of ideas and information within a text, ensuring that sentences & paragraphs are connected and form a coherent whole. Text coherence is essential for readability and comprehension.

Discuss the significance of Text cohesence in Discourse Segmentation.

(1) Reader comprehension: Coherent texts are easier for readers to understand and follow.

(2) Segmented Boundary identification: Coherence helps identify natural boundaries b/w text segments. Text coherence aids the accurate identification of these boundaries.

(3) Content organization: Coherent texts are organized in a way that presents ideas and information logically and cohesively.

(4) Engagement: Coherent texts are more engaging for readers.

(5) Transition phrases: Coherent texts often use transitional phrases and devices (eg:- "however", "on the other hand," "in conclusion") to signal shifts in topics or ideas.

(6) Reducing Cognitive load: On readers when readers don't have to work hard to connect disjointed or unrelated segments.

(7) Natural flow: Readers can move more seamlessly from one segment to another, making the reading experience more efficient.

(8) Information Retrieval: Search engines and information retrieval systems can use coherent segments to provide more relevant search results, as they can better understand the relevance of each segment.

(9) Summarization: Summarization algorithms can effectively extract key information from well-structured, coherent segments to create concise and informative summaries.

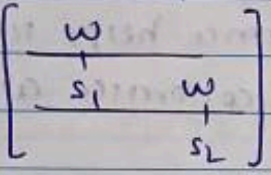
Q. 17 Semi supervised method (YAROWSKY)

In computational linguistics the yarowsky algorithm is a semi-supervised or unsupervised learning algorithm, for word sense disambiguation.

From observation, words tend to exhibit only one sense in most given discourse and in a given collocation.

P(s_j) P(f_i|s_i) ; how can find out the value.

So for this we required labelled dataset.



this is for every word & this is really expensive...

By using semi-supervised some sort of labelled & create many labels.

- Bootstrapping or co-training
  - start with (small) seed) learn decision list
  - use decision list to label rest of corpus.
  - Retain confident labels, treat as annoted data to learn new list (decision)
  - repeat.

- derived from observation
  - one sense per discourse
  - one sense per collocation.

example:
Cricket comedy word using ball.

Range in the discourse the word occur multiple time having single sense.

One sense per discourse :-
A word tends to preserve its meaning across all its occurance in a given discourse..

example: Disambiguate "plant"

PTO

- Think of seed feature for each sense
  ① industrial sense - (manufactured)
  ② living thing sense - (live)

from whole corpus having

[O] servering sentence occurred from with word manufacture.. } treated as labelled data.

[O] serves 2 ← send } word occur with live

& apply decision list algorithm to find good collocation. & having 3,4 collocation with the word it can be diff. thing like growth, cat, etc..
now use this collocation to test new list classifies you have new data keep on repeating..

example:
divide list into plant & animal kingdom

collocation

collocation → plant & animal
animal rather than plant
it is it can be...

Q. 18| Supervised (Naive Bayes)
Naive Bayes classifies Algorithm is a family of probability algorithm
It is based on applying Bayes theorem with the assumption of conditional independence b/w every pair of feature.
Naive bayes theorem calculates probability P(c/x) ..
mostly used in NLP problems..

PTO

Naive bayes classifier chooses the most likely sense for a word given the features of the context.

Naive bayes

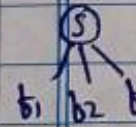
* 'f' is feature vector consisting of
  - pos of 'w'
  - semantic & syntactic features of 'w'
  - collocation vector (set of words around it)
    - nearest word (+1, +2, -1, -2 & their pos's)
  - co-occurance vector..

set parameters of Naive bayes using maximum likelihood estimation (MLE) from training data.

- $P(s_i) = \frac{count(s_i, w_j)}{count(w_j)}$ ... no. of times the sense is used divided by no. of times this word is used.
- $P(f_i|s_i) = \frac{count(f_i, s_i)}{count\ s_i}$ ... no. of times the feature is observed with the sense divided by no. of time sense is used..

finds out the sense of word that gives maximum probability..

$$\hat{s} = \underset{s}{argmax}\ P(s|f)$$



$$= \underset{s}{argmax}\ P(s)\ P(f|s)$$

P(s) → prior sense probability
P(f|s) → probability of diff. features generated by sense..

Naive Bayes predict the tag of a text.
They calculate the probability of each tag for a given text and then o/p the tag with the highest one..