



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

DIPTI SJARNAGAT CSEDS 55

Experiment No.1
Study various applications of NLP and Formulate the Problem Statement for Mini Project based on chosen real world NLP applications
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: Study various applications of NLP and Formulate the Problem Statement for Mini Project based on chosen real world NLP applications.

Objective: Understand the different applications of NLP and their techniques by reading and critiquing IEEE/ACM/Springer papers.

Theory:

1. Machine Translation

Machine translation is a process of converting the text from one language to the other automatically without or minimal human intervention.

2. Text Summarization

Condensing a lengthy text into a manageable length while maintaining the essential informational components and the meaning of the content is known as summarization. Since manually summarising material requires a lot of time and is generally difficult, automating the process is becoming more and more popular, which is a major driving force behind academic research.

Text summarization has significant uses in a variety of NLP-related activities, including text classification, question answering, summarising legal texts, summarising news, and creating headlines. Additionally, these systems can incorporate the creation of summaries as a middle step, which aids in shortening the text.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The quantity of text data from many sources has multiplied in the big data era. This substantial body of writing is a priceless repository of data and expertise that must be skillfully condensed in order to be of any use. A thorough investigation of NLP for automatic text summarization has been necessitated by the increase in the availability of documents. Automatic text summarising is the process of creating a succinct, fluid summary without the assistance of a human while maintaining the original text's meaning.

3. Sentiment Analysis

Sentiment analysis, often known as opinion mining, is a technique used in natural language processing (NLP) to determine the emotional undertone of a document. This is a common method used by organisations to identify and group ideas regarding a certain good, service, or concept. Text is mined for sentiment and subjective information using data mining, machine learning, and artificial intelligence (AI).

Opinion mining can extract the subject, opinion holder, and polarity (or the degree of positivity and negative) from text in addition to identifying sentiment. Additionally, other scopes, including document, paragraph, sentence, and sub-sentence levels, can be used for sentiment analysis.

Businesses must comprehend people's emotions since consumers can now communicate their views and feelings more freely than ever before. Brands are able to listen carefully to their customers and customise their products and services to match their demands by automatically evaluating customer input, from survey replies to social media chats.

4. Information Retrieval

A software programme that deals with the organisation, storage, retrieval, and evaluation of information from document repositories, particularly textual information, is known as information 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 pulled up by the ideal IR system.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

5. Question Answering System (QAS)

Building systems that automatically respond to questions presented by humans in natural language is the focus of the computer science topic of question answering (QA), which falls under the umbrella of information retrieval and natural language processing (NLP).

OUTPUT:-

TITLE NAME :- "AUTO RECOMMENDATION TEXT ENTRY"

Conclusion:

The Auto Recommendation Text Entry system is a promising solution to address the challenges of efficient and accurate text input in NLP and other text-based applications. With its real-time word suggestions and user-friendly GUI, it aims to improve productivity and reduce typing errors. This report has outlined the problem statement, objectives, scope, and introduced the proposed system, setting the stage for its potential applications in a variety of task.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

DIPTI SHARNAGAT CSE DS 55

Experiment No.2
Apply Tokenization on given English and Indian Language Text
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: Apply Tokenization on given English and Indian Language Text

Objective: Able to perform sentence and word tokenization for the given input text for English and Indian Language.

Theory:

Tokenization is one of the first step in any NLP pipeline. Tokenization is nothing but splitting the raw text into small chunks of words or sentences, called tokens. If the text is split into words, then its called as 'Word Tokenization' and if it's split into sentences then its called as 'Sentence Tokenization'. Generally 'space' is used to perform the word tokenization and characters like 'periods, exclamation point and newline char are used for Sentence Tokenization. We have to choose the appropriate method as per the task in hand. While performing the tokenization few characters like spaces, punctuations are ignored and will not be the part of final list of tokens.

Why Tokenization is Required?

Every sentence gets its meaning by the words present in it. So by analyzing the words present in the text we can easily interpret the meaning of the text. Once we have a list of words we can also use statistical tools and methods to get more insights into the text. For example, we can use word count and word frequency to find out important of word in that sentence or document.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Input Text

Tokenization is one of the first step in any NLP pipeline. Tokenization is nothing but splitting the raw text into small chunks of words or sentences, called tokens.

Word Tokenization

Tokenization	is	one	of
the	first	step	in
any	NLP	pipeline	Tokenization
is	nothing	but	splitting
the	raw	text	into
small	chunks	of	words
or	sentences	called	tokens

Sentence Tokenization

Tokenization is one of the first step in any NLP pipeline

Tokenization is nothing but splitting the raw text into small chunks of words or sentences, called tokens

```
NLP exp 02.pdf at main · d...
github.com/diptonamajet/NLP/blob/main/NLP20exp%2002.pdf
Gmail YouTube Maps Learn Programming

Files
main
Go to file
NLP EXP 01.pdf
NLP exp 02.pdf
NLDPXP03.ipynb
NLDPXP04.ipynb
NLDPXP05.ipynb
NLP_EXP_06.ipynb
NLP_EXP_07.ipynb
NLP_EXP_8.ipynb
README.md
Documentation · Share feedback

593 KB Code 55% faster with GitHub Copilot
8/18/23, 2:20 AM Copy of NLP exp 02.ipynb - Collaboratory

!pip install nltk
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.3)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.6)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex<2023.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)

import nltk
nltk.download()

D NLTK Downloader
-----
d) Download  l) List  u) Update  c) Config  h) Help  q) Quit
Downloader> q
True

Sentence Tokenization

from nltk.tokenize import sent_tokenize

text = '''Martha is at the grocery store, getting ready for a house party. She has a list of what she needs with her as she goes along. The firs
t section she comes has produce. Martha sees apples, bananas, cherries, grapes, and strawberries.'

sentences = sent_tokenize(text)
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows a GitHub repository for 'NLP / NLP exp 02.pdf'. The left sidebar lists files including 'NLP EXP 01.pdf', 'NLP exp 02.pdf', and several Jupyter Notebook files. The main content area displays a Python script for tokenization using the 'nltk.tokenize' and 'wordpunct_tokenize' functions. The script includes a list of words to be tokenized and a function to filter text by converting it into lower case. The output of the script is shown as a list of tokens: 'MARTHA IS AT THE GROCERY STORE, GETTING READY FOR A HOUSE PARTY. SHE HAS A LIST OF WHAT SHE NEEDS WITH HER AS SHE GOES ALONG. THE FIRST SECTION SHE COMES HAS PRODUCE MARTHA SEES APPLEES BANANAS PEACHES GRAPES AND STRAWBERRIES'.

Conclusion:

okenization is the process of breaking down the given text in natural language processing into the smallest unit in a sentence called a token. Punctuation marks, words, and numbers can be considered tokens.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

DIPTI SHARNAGAT CSEDS 55

Experiment No.3
Apply Stop Word Removal on given English and Indian Language Text
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: Apply Stop Word Removal on given English and Indian Language Text.

Objective: To write program for Stop word removal from a sentence given in English and any Indian Language.

Theory:

The process of converting data to something a computer can understand is referred to as pre-processing. One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words.

Stopwords are the most common words in any natural language. For the purpose of analyzing text data and building NLP models, these stopwords might not add much value to the meaning of the document.

Stop Words: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We need to perform tokenization before removing any stopwords.

Why do we need to Remove Stopwords?

Removing stopwords is not a hard and fast rule in NLP. It depends upon the task that we are working on. For tasks like text classification, where the text is to be classified into different categories, stopwords are removed or excluded from the given text so that more focus can be given to those words which define the meaning of the text.

Here are a few key benefits of removing stopwords:

- On removing stopwords, dataset size decreases and the time to train the model also decreases
- Removing stopwords can potentially help improve the performance as there are fewer and only meaningful tokens left. Thus, it could increase classification accuracy



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

- Even search engines like Google remove stopwords for fast and relevant retrieval of data from the database

We can remove stopwords while performing the following tasks:

- Text Classification
 - Spam Filtering
 - Language Classification
 - Genre Classification
- Caption Generation
- Auto-Tag Generation

Avoid Stopword Removal

- Machine Translation
- Language Modeling
- Text Summarization
- Question-Answering problems

Different Methods to Remove Stopwords

1. Stopword Removal using NLTK

NLTK, or the Natural Language Toolkit, is a treasure trove of a library for text preprocessing. It's one of my favorite Python libraries. NLTK has a list of stopwords stored in 16 different languages.

You can use the below code to see the list of stopwords in NLTK:

```
import nltk  
from nltk.corpus import stopwords  
set(stopwords.words('english'))
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

2. Stopword Removal using spaCy:

spaCy is one of the most versatile and widely used libraries in NLP. We can quickly and efficiently remove stopwords from the given text using SpaCy.

It has a list of its own stopwords that can be imported as **STOP_WORDS** from the `spacy.lang.en.stop_words` class.

3. Stopword Removal using Gensim

Gensim is a pretty handy library to work with on NLP tasks. While pre-processing, gensim provides methods to remove stopwords as well. We can easily import the `remove_stopwords` method from the class `gensim.parsing.preprocessing`.

```
In [1]: !pip install nltk

Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)

In [2]: text = "ParagraphParagraphs are the group of sentences combined together, about a certain topic. It is a very important for"
In [3]: text

Out[3]: 'ParagraphParagraphs are the group of sentences combined together, about a certain topic. It is a very important form of writing as we write almost everything in paragraphs, be it an answer, essay, story, email, etc. We can say that a well-structured paragraph is the essence of good writing. The purposes of the paragraph are to give information, to explain something, to tell a story, and to convince someone that our idea is right.'

In [8]: import nltk

In [9]: nltk.download('stopwords')

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
In [47]: snow_stemmed = [snow.stem(x) for x in words]
          print(snow_stemmed)

['play', 'play', 'play', 'play', 'player']

Lancaster Stemmer

In [26]: lancaster_stemmed = list()
          for w in words:
              stemmed_words = lancaster.stem(w)
              lancaster_stemmed.append(stemmed_words)

In [27]: lancaster_stemmed

Out[27]: ['play', 'play', 'play', 'play', 'play']

Lancaster Stemmer List Comprehension

In [28]: lancaster_stemmed = [lancaster.stem(x) for x in words]
          print(lancaster_stemmed)

['play', 'play', 'play', 'play', 'play']

Lemmatization : This has a more expansive vocabulary than Stemming

In [29]: from nltk.stem import WordNetLemmatizer
          wordnet = WordNetLemmatizer()

In [33]: nltk.download('wordnet')
```

Conclusion:

- We provide a brief description about experimental setup to see the effect of stopwords in the Indian languages Marathi, Bengali, Gujarati and Sanskrit. Stopwords are the most frequently used words like articles, pronouns, conjunctions, prepositions, prefixes, adverbs and adjectives.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

DIPTI SHARNAGAT CSEDS 55

Experiment No.4
Apply Stemming on the given Text input
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: Apply Stemming on the given Text input.

Objective: Understand the working of stemming algorithms and apply stemming on the given input text.

Theory:

Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "conect".

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words "chocolates", "chocolatey", "choco" to the root word, "chocolate" and "retrieval", "retrieved", "retrieves" and reduces to the stem "retrieve". Stemming is an important part of the pipelining process in Natural language processing. The input to the stemmer is tokenized words.

Applications of stemming :

1. Stemming is used in information retrieval systems like search engines.
2. It is used to determine domain vocabularies in domain analysis.

Porter's Stemmer Algorithm:

It is one of the most popular stemming methods proposed in 1980. It is based on the idea that the suffixes in the English language are made up of a combination of smaller and simpler suffixes. This stemmer is known for its speed and simplicity. The main applications of Porter Stemmer include data mining and Information retrieval. However, its applications are only limited to English words. Also, the group of stems is mapped on to the same stem and the output stem is not necessarily a meaningful word. The algorithms are fairly lengthy in nature and are known to be the oldest stemmer.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Example: EED -> EE means “if the word has at least one vowel and consonant plus EED ending, change the ending to EE” as ‘agreed’ becomes ‘agree’.

Advantage: It produces the best output as compared to other stemmers and it has less error rate.

Limitation: Morphological variants produced are not always real words.

The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer lists several files, with NLPEXP04.ipynb selected. The code editor shows Python code for calculating n-grams and their probabilities. The output displays the most common unigrams, bigrams, and trigrams with their respective probabilities.

```
print("\nMost common unigrams: ", str(ngrams_prob[1]))
print("\nMost common bigrams: ", str(ngrams_prob[2]))
print("\nMost common trigrams: ", str(ngrams_prob[3]))
print("\nMost common fourgrams: ", str(ngrams_prob[4]))
```

Most common n-grams without stopword removal and with a

Most common unigrams: [[('the',), 0.05598462224968249],
[('a',), 0.02155631071293722], [('she',), 0.01846703051
48797], [('said',), 0.015892630350461675], [('i',), 0.0

Most common bigrams: [[('said', 'the'), 0.005339571308
e'), 0.0029494774848076483], [('in', 'a'), 0.0024917995
0.0020086958732741743], [('it', 'was'), 0.0019069897531
0.0016781509827353861], [('as', 'she'), 0.0015764448625

Most common trigrams: [[('the', 'mock', 'turtle'), 0.0
5], [('said', 'the', 'king'), 0.0006599062933063505], [

- **Conclusion:** Stemming is a natural language processing technique that is used to reduce words to their base form, also known as the root form. The process of stemming is used to normalize text and make it easier to process. It is an important step in text pre-processing, and it is commonly used in information retrieval and text mining applications.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

DIPTI SHARNAGAT CSEDS 55

Experiment No.5
Implement Bi-Gram model for the given Text input
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: Implement Bi-Gram model for the given Text input

Objective: To study and implement N-gram Language Model.

Theory:

A language model supports predicting the completion of a sentence.

Eg:

- Please turn off your cell _____
- Your program does not _____

Predictive text input systems can guess what you are typing and give choices on how to complete it.

N-gram Models:

Estimate probability of each word given prior context.

$P(\text{phone} \mid \text{Please turn off your cell})$

- Number of parameters required grows exponentially with the number of words of prior context.
- An N-gram model uses only $N-1$ words of prior context.
 - Unigram: $P(\text{phone})$
 - Bigram: $P(\text{phone} \mid \text{cell})$
 - Trigram: $P(\text{phone} \mid \text{your cell})$

- The Markov assumption is the presumption that the future behavior of a dynamical system only depends on its recent history. In particular, in a k th-order Markov model,



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

the next state only depends on the k most recent states, therefore an N-gram model is a (N1)-order Markov model.

N-grams: a contiguous sequence of n tokens from a given piece of text

Mary was scared because of the terrifying noise. ...

The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer lists several files: NLP EXP 01.pdf, NLP exp 02.pdf, NLPEXP03.ipynb, NLPEXP04.ipynb, NLPEXP05.ipynb (selected), NLP_EXP_06.ipynb, NLP_EXP_07.ipynb, NLP_EXP_08.ipynb, and README.md. The code editor displays the following code:

```
In [1]: text = "ParagraphParagraphs are the group of sentences combined together, about a certain  
Importing necessary dependencies  
In [2]: import nltk  
from nltk.tokenize import word_tokenize  
Word Tokenization  
In [4]: nltk.download('punkt')  
[nltk_data] Downloading package punkt to /root/nltk_data...  
[nltk_data] Unzipping tokenizers/punkt.zip.
```

Fig. Example of Trigrams in a sentence

Conclusion:

the model implemented here is a "Statistical Language Model". I have used "BIGRAMS" so this is known as Bigram Language Model. In Bigram language model we find bigrams which means two words coming together in the corpus (the entire collection of words/sentences). In the sentence "DEV is awesome and user friendly" the bigrams are :



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

DIPTI SHARNAGAT CSEDS 55

Experiment No.6
Perform POS tagging on the given English and Indian Language Text
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology
Department of Artificial Intelligence & Data Science



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: Perform POS tagging on the given English and Indian Language Text

Objective: To study POS Tagging and tag the part of speech for given input in english and an Indian Language.

Theory:

The primary target of Part-of-Speech (POS) tagging is to identify the grammatical group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. based on the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.

POS Tagging (Parts of Speech Tagging) is a process to mark up the words in text format for a particular part of a speech based on its definition and context. It is responsible for text reading in a language and assigning some specific token (Parts of Speech) to each word. It is also called grammatical tagging.

Steps Involved in the POS tagging example:

- Tokenize text (word_tokenize)
- apply pos_tag to above step that is nltk.pos_tag(tokenize_text)

```
In [1]: import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] /root/nltk_data...
[nltk_data] Unzipping taggers/averaged_perceptron_tagger.zip.

Out[1]: True

In [2]: from nltk.chunk import RegexParser
from nltk.tokenize import word_tokenize

In [3]: sentence = "Paragraph are the group of sentences combined together, about a certain topic"

Tokenization

In [4]: tokens = word_tokenize(sentence)

In [5]: tokens

Out[5]: ['ParagraphParagraphs',
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows a Jupyter Notebook titled 'NLP / NLP_EXP_06.ipynb' on a GitHub page. The notebook contains the following code and output:

```
Out[9]: '\n NP: {<DT>?<JJ>*<NN>} # Chunk noun phrases\n VP: {<VB.*><NP|PP>} # Chunk verb phrases\n'
```

Create a chunk parser

```
In [10]: chunk_parser = RegexpParser(chunk_patterns)
```

```
In [12]: chunk_parser
```

```
Out[12]: <chunk.RegexpParser with 2 stages>
```

Perform chunking

```
In [13]: result = chunk_parser.parse(pos_tags)
          print(result)
```

```
{5
ParagraphParagraphs/MIS
(VP are/VBP (NP the/DT group/NN))
off/IN
sentences/MIS
combined/VBI
together/RB
./
about/IN
(NP a/DT certain/JJ topic/NN))
```

Conclusion:

POS Tagging (Parts of Speech Tagging) is a process to mark up the words in text format for a particular part of a speech based on its definition and context.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

DIPTI SHARNAGAT CSEDS 55

Experiment No.7
Implement Named Entity Recognizer for the given Text input
Date of Performance:
Date of Submission:

Aim: Implement Named Entity Recognizer for the given Text input

Objective: Understand the importance of NER in NLP and Implement NER.

Theory:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The named entity recognition (NER) is one of the most data preprocessing task. It involves the identification of key information in the text and classification into a set of predefined categories. An entity is basically the thing that is consistently talked about or refer to in the text.

NER is the form of NLP.

At its core, NLP is just a two-step process, below are the two steps that are involved:

- Detecting the entities from the text
- Classifying them into different categories

Some of the categories that are the most important architecture in NER such that:

- Person
- Organization
- Place/ location

Other common tasks include classifying of the following:

- date/time.
- expression
- Numeral measurement (money, percent, weight, etc)
- E-mail address

Ambiguity in NE

For a person, the category definition is intuitively quite clear, but for computers, there is some ambiguity in classification. Let's look at some ambiguous example:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

England (Organisation) won the 2019 world cup vs The 2019 world cup happened in England(Location).

Washington(Location) is the capital of the US vs The first president of the US was Washington(Person).

```
# Initialize variables to store named entities
named_entities = []

# Define a function to extract named entities
def extract_named_entities(doc):
    entities = []
    current_entity = None

    for token in doc:
        if token.ent_type_:
            if current_entity and token.ent_type_ == current_entity[1]:
                current_entity = (current_entity[0] + " " + token.text, token.ent_type_)
            else:
                if current_entity:
                    entities.append(current_entity)
                current_entity = (token.text, token.ent_type_)
            else:
                if current_entity:
                    entities.append(current_entity)
                current_entity = None

            if current_entity:
                entities.append(current_entity)

    return entities

# Extract named entities
named_entities = extract_named_entities(doc)

# Print the named entities
for entity, label in named_entities:
    print(f"Entity: {entity}, Label: {label}")
```

```
import spacy

# Load the spaCy language model
nlp = spacy.load("en_core_web_sm")

# Sample text input
text = "A paragraph is a self-contained unit of discourse in writing dealing with a particular point or idea."

# Process the text using spaCy
doc = nlp(text)

# Initialize variables to store named entities
named_entities = []

# Define a function to extract named entities
def extract_named_entities(doc):
    entities = []
    current_entity = None

    for token in doc:
        if token.ent_type_:
            if current_entity and token.ent_type_ == current_entity[1]:
                current_entity = (current_entity[0] + " " + token.text, token.ent_type_)
            else:
                if current_entity:
                    entities.append(current_entity)
                current_entity = (token.text, token.ent_type_)
            else:
                if current_entity:
                    entities.append(current_entity)
                current_entity = None

            if current_entity:
                entities.append(current_entity)

    return entities

# Extract named entities
named_entities = extract_named_entities(doc)

# Print the named entities
for entity, label in named_entities:
    print(f"Entity: {entity}, Label: {label}")
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Conclusion:

A named entity is basically a real-life object which has proper identification and can be denoted with a proper name. Named Entities can be a place, person, organization, time, object, or geographic entity. For example, named entities would be Roger Federer, Honda city, Samsung Galaxy S10. Named entities are usually instances of entity instance

Experiment No. 8
Implement word sense disambiguation using LSTM/GRU



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Date of Performance:

DIPTI

Date of Submission:

SHARNAGAT CSEDs 55



Vidyavardhini's College of Engineering and Technology

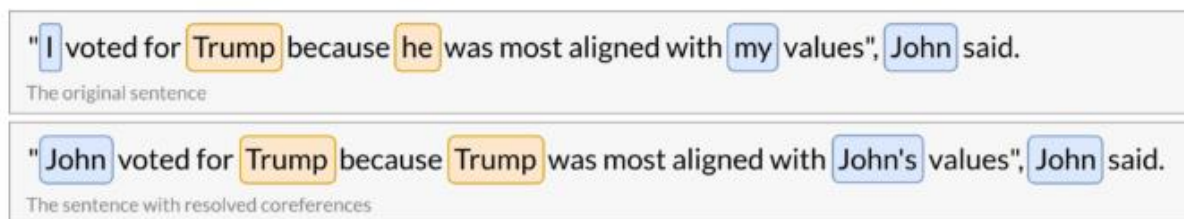
Department of Artificial Intelligence & Data Science

Aim: Apply Reference Resolution Technique on the given Text input.

Objective: Understand the importance of resolving references and implementing reference resolution for the given text input.

Theory:

Coreference resolution (CR) is the task of finding all linguistic expressions (called mentions) in a given text that refer to the same real-world entity. After finding and grouping these mentions we can resolve them by replacing, as stated above, pronouns with noun phrases.



Coreference resolution is an exceptionally versatile tool and can be applied to a variety of NLP tasks such as text understanding, information extraction, machine translation, sentiment analysis, or document summarization. It is a great way to obtain unambiguous sentences which can be much more easily understood by computers.

```
In [1]: import torch
import torch.nn as nn
import torch.nn.functional as F

Sample data (context and senses)

In [2]: data = [
    ({
        "text": "Trump", "type": "Noun", "sense": "1", "index": 1, "context": "Trump"},
        {"text": "he", "type": "Pronoun", "sense": "1", "index": 2, "context": "Trump"},
        {"text": "my", "type": "Pronoun", "sense": "1", "index": 3, "context": "Trump"},
        {"text": "Trump", "type": "Noun", "sense": "1", "index": 4, "context": "Trump"},
        {"text": "John", "type": "Noun", "sense": "1", "index": 5, "context": "John"},
        {"text": "John", "type": "Noun", "sense": "1", "index": 6, "context": "John"}
    ])

Create a vocabulary

In [3]: vocab = Vocabulary.from_instances(data, min_count={'tokens': 1})
vocab.get_vocab_size('tokens')
vocab.get_vocab_size('senses')

Map sense labels to integers

In [4]: sense_labels = {}
for i, (label, sense) in enumerate(sense_labels.items()):
    sense_labels[label] = i
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows a GitHub repository for 'NLP_EXP_8.ipynb'. The left sidebar lists files including NLP_EXP_01.pdf, NLP_EXP_02.pdf, NLP_EXP_03.ipynb, NLP_EXP_04.ipynb, NLP_EXP_05.ipynb, NLP_EXP_06.ipynb, NLP_EXP_07.ipynb, NLP_EXP_8.ipynb (selected), and README.md. The main area displays the code for NLP_EXP_8.ipynb, which includes a preview of the code, tabs for Code, Blame, and a status bar indicating 338 lines (338 loc) and 8.8 KB. The code is organized into sections: a function definition for prediction, Hyperparameters (vocab_size, embedding_dim, hidden_dim, sense_count, learning_rate, epochs), Initialize the model (model = WSDModel), Define the loss function and optimizer (criterion = nn.CrossEntropyLoss, optimizer = optim.Adam), and a Training loop (def train). The code is written in Python and uses libraries like torch.nn and torch.optim.

Conclusion:

Word Sense Disambiguation is considered one of the challenging problems in natural language processing (NLP). LSTM-based Word Sense Disambiguation techniques have been shown effective through experiments. Models have been proposed before that employed LSTM to achieve state-of-the-art results.