HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

M.Sc. (CS) Part1 SEM-2

Subject: Computational Linguistics

Sr. No.	Title	Date	Signature
1	Perform Tokenization of words and sentences using different methods.	xx-03-24	
2	Implement following concepts in NLP: stopwords, synonym, antonym, lemmas, hyponyms, hypernyms, and entailments.	xx-03-24	
3	Elaborate Part of Speech tagging and Named Entity Recognition concepts.	xx-03-24	
4	Design Context-free grammar and Parse trees.	xx-03-24	
5.	Translating text to Indian Languages and languages of different countries.	xx-03-24	
6	Develop an End-to-End Chatbot.	xx-03-24	
7	Implement Text Summarization using NLTK.	xx-03-24	
8	Build the Next Word Prediction Model.	xx-03-24	
9	Develop interactive application using Text to Speech and Speech to Text conversion.	xx-03-24	
10	Perform Sentiment Analysis on Amazon Product Reviews.	xx-03-24	

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 1	
Aim: Perform Tokenization of words and sentences using different methods.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

Tokenization is a crucial preprocessing step in natural language processing (NLP) tasks.
It involves breaking down text into smaller units, such as words or sentences, for further analysis.

The following Python code demonstrates four tokenization methods using the split() function, regular expressions (re module), NLTK's word_tokenize(), and sent_tokenize().

Tokenization Method	Advantages	Drawbacks
split() Method:	• Simple and easy to use. • Splits text based on a specified delimiter. • Fast performance.	• Limited flexibility in handling complex tokenization requirements. • Less effective with irregular text patterns.
Regular Expressions (re.findall()):	• Offers flexibility in defining tokenization patterns. • Suitable for complex tokenization tasks. • Handles special characters effectively.	• Requires understanding of regular expressions. • Performance may degrade with large texts due to regex engine overhead.
NLTK's word_tokenize():	• Specifically designed for word tokenization in NLP tasks. • Handles common tokenization challenges effectively. • Preserves contractions and hyphenated words.	• May not perform optimally with domain-specific or non-standard text. • Requires NLTK library installation and data downloads.
NLTK's sent_tokenize():	• Designed for accurate sentence tokenization. • Handles abbreviations and emoticons effectively. • Preserves sentence boundaries accurately.	• May struggle with text containing unconventional sentence structures. • Requires NLTK library installation and data downloads.

In conclusion each tokenization method has its own strengths and weaknesses, making them suitable for different scenarios. While basic methods like split() offer simplicity, more advanced techniques such as regular expressions and NLTK's tokenizers provide greater flexibility and accuracy. Choosing the appropriate tokenization method depends on the specific requirements of the NLP task and the characteristics of the text being processed.

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 1	
Aim: Perform Tokenization of words and sentences using different methods.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

CODE:

```
import re
from nltk.tokenize import word_tokenize , sent_tokenize
print("Saail Chavan - KFPMSCCS016 CL_P1")
text = "Hi this is Saail Chavan . This is an example of tokenization methods ."
tokens=text.split()
print("\noriginal text:", text)
print("\ntokenized using split()\n", tokens)
tokens = re.findall("[\w']+", text)
print("\ntokenized using re.findall()\n", tokens)
tokens=word_tokenize(text)
print("\ntokenized using word_tokenize()\n",tokens)
print("\n\nsentence tokenization")
sents=text.split('.')
print("\noriginal text\n", text)
print("\ntokenized using split('.')\n", sents)
sents = re.compile('[.!?]').split(text)
print("\ntokenized using re.compile()\n",sents)
sents=sent_tokenize(text)
print("\ntokenized using sent_tokenize()\n",sents)
```

OUTPUT:

```
Saail Chavan - KFPMSCCS016 CL_P1

original text: Hi this is Saail Chavan . This is an example of tokenization methods .

tokenized using split()
 ['Hi', 'this', 'is', 'Saail', 'Chavan', '.', 'This', 'is', 'an', 'example', 'of', 'tokenization', 'methods', '.']

tokenized using re.findall()
 ['Hi', 'this', 'is', 'Saail', 'Chavan', 'This', 'is', 'an', 'example', 'of', 'tokenization', 'methods']

tokenized using word_tokenize()
 ['Hi', 'this', 'is', 'Saail', 'Chavan', '.', 'This', 'is', 'an', 'example', 'of', 'tokenization', 'methods', '.']


sentence tokenization

original text
Hi this is Saail Chavan . This is an example of tokenization methods .

tokenized using split('.')
['Hi this is Saail Chavan ', ' This is an example of tokenization methods ', '']

tokenized using re.compile()
['Hi this is Saail Chavan ', ' This is an example of tokenization methods ', '']

tokenized using sent_tokenize()
['Hi this is Saail Chavan .', 'This is an example of tokenization methods .']
```

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 2	
Aim: Implement following concepts in NLP: stopwords, synonym, antonym, lemmas, hyponyms, hypernyms, and entailments.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

Stopwords Removal:

- Stopwords are common words in a language that carry little to no semantic meaning but appear frequently in texts. Examples include "is", "the", "and", etc.
- Removing stopwords is an essential preprocessing step in NLP tasks like text classification, sentiment analysis, and information retrieval.
- By eliminating stopwords, we can focus on the more informative words in the text, which helps improve the accuracy of NLP models and reduces computational complexity.

Synonyms and Antonyms:

- Synonyms are words with similar meanings, while antonyms are words with opposite meanings.
- Understanding synonyms and antonyms is crucial for tasks such as text understanding, sentiment analysis, and paraphrase detection.
- In NLP applications, identifying synonyms and antonyms can aid in improving the richness and accuracy of language models, as well as in enhancing semantic understanding and context analysis.

Lemmas:

- Lemmas are the canonical or base forms of words obtained by removing inflections and derivational affixes.
- In NLP, lemmatization is often used for text normalization, where different inflected forms of words are mapped to their common base form.
- Lemmatization helps in reducing word ambiguity and simplifying the vocabulary, which is beneficial for tasks like information retrieval, text mining, and machine translation.

Hyponyms and Hypernyms:

- Hyponyms are words that are more specific in meaning than a given word, while hypernyms are words that are more general.
- Understanding hyponyms and hypernyms is essential for building semantic hierarchies and taxonomies.
- In NLP, exploring hyponyms and hypernyms can aid in tasks such as word sense disambiguation, semantic similarity calculation, and knowledge representation.

Entailments:

- Entailments are logical relationships between verbs, where the truth of one statement guarantees the truth of another.
- Recognizing entailments is crucial for tasks like natural language inference, reasoning, and logic-based text understanding.
- In NLP applications, identifying entailments helps in capturing the implicit semantic relationships between sentences, which is valuable for tasks such as question answering, summarization, and inference.
- By incorporating these concepts into NLP workflows, practitioners can enhance the robustness, accuracy, and semantic understanding of their language processing systems. Each concept plays a vital role in various NLP tasks, contributing to the advancement and effectiveness of natural language understanding and generation applications.

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 2	
Aim: Implement following concepts in NLP: stopwords, synonym, antonym, lemmas, hyponyms, hypernyms, and entailments.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

CODE:

```
from nltk.tokenize import word_tokenize
from nltk.corpus import wordnet, stopwords

print("Saail Chavan - KFPMSCCS016 CL_P2\n")
text = "Natural language processing, especially when Saail discusses it, is fascinating. It involves many tasks such as text classification, sentiment analysis, and more."

# Stop words removal
stop_words = set(stopwords.words('english'))
words = word_tokenize(text)
filtered_words = [word for word in words if word.casefold() not in stop_words]
print("Stopwords:\n", "original text:",text,"\nfiltered words:",filtered_words)

synonyms = []
antonyms = []
lemmas = []

# Synonyms and antonyms
for syn in wordnet.synsets("active"):   #synsets-syntax sets
    for l in syn.lemmas():
        synonyms.append(l.name())
        if l.antonyms():
            antonyms.append(l.antonyms()[0].name())
        lemmas.append(l.name())

print("\nSynonyms of active:\n", set(synonyms))
print("\nAntonyms of active:\n", set(antonyms))

# Printing lemmas
print("\nLemmas of active:\n", set(lemmas))

# Hyponyms
hypo = wordnet.synset('car.n.01').hyponyms()
print("\nHyponym of car:\n", hypo)

# Hypernyms
hyper = wordnet.synset('car.n.01').hypernyms()
print("\nHypernym of car:\n", hyper)

# Entailments
ent = wordnet.synset('snore.v.01').entailments()
print("\nEntailment of snore:\n", ent)
```

OUTPUT:

```
Saail Chavan - KFPMSCCS016 CL_P2

Stopwords:
 original text: Natural language processing, especially when Saail discusses it, is fascinating. It involves many tasks such as
 text classification, sentiment analysis, and more.
filtered words: ['Natural', 'language', 'processing', ',', 'especially', 'Saail', 'discusses', ',', 'fascinating', '.', 'involv
es', 'many', 'tasks', 'text', 'classification', ',', 'sentiment', 'analysis', ',', '.']

Synonyms of active:
 {'fighting', 'combat-ready', 'active', 'dynamic', 'alive', 'active_agent', 'active_voice', 'participating'}

Antonyms of active:
 {'passive', 'stative', 'dormant', 'quiet', 'inactive', 'passive_voice', 'extinct'}

Lemmas of active:
 {'fighting', 'combat-ready', 'active', 'dynamic', 'alive', 'active_agent', 'active_voice', 'participating'}
```

```
Hyponym of car:
 [Synset('ambulance.n.01'), Synset('beach_wagon.n.01'), Synset('bus.n.04'), Synset('cab.n.03'), Synset('compact.n.03'), Synset(
'convertible.n.01'), Synset('coupe.n.01'), Synset('cruiser.n.01'), Synset('electric.n.01'), Synset('gas_guzzler.n.01'), Synset(
'hardtop.n.01'), Synset('hatchback.n.01'), Synset('horseless_carriage.n.01'), Synset('hot_rod.n.01'), Synset('jeep.n.01'), Syns
et('limousine.n.01'), Synset('loaner.n.02'), Synset('minicar.n.01'), Synset('minivan.n.01'), Synset('model_t.n.01'), Synset('pa
ce_car.n.01'), Synset('racer.n.02'), Synset('roadster.n.01'), Synset('sedan.n.01'), Synset('sport_utility.n.01'), Synset('sport
s_car.n.01'), Synset('stanley_steamer.n.01'), Synset('stock_car.n.01'), Synset('subcompact.n.01'), Synset('touring_car.n.01'),
Synset('used-car.n.01')]

Hypernym of car:
 [Synset('motor_vehicle.n.01')]

Entailment of snore:
 [Synset('sleep.v.01')]
```

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 3	
Aim: Elaborate Part of Speech tagging and Named Entity Recognition concepts.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 3	
Aim: Elaborate Part of Speech tagging and Named Entity Recognition concepts.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

CODE:

OUTPUT:

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 4	
Aim: Design Context-free grammar and Parse trees.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

In computational linguistics and natural language processing (NLP), Context-Free Grammar (CFG) and Parse Trees play vital roles. They are fundamental tools for understanding and generating natural language sentences according to specific rules and structures.

Context-Free Grammar (CFG):

CFG is a formal grammar where each non-terminal symbol can be replaced by a specific sequence of terminal symbols, regardless of the context in which it appears. In simpler terms, CFG consists of a set of production rules that dictate how strings of symbols can be generated in a language. It is commonly used to describe the syntax of programming languages and natural languages.

In the following code, a CFG is defined using the nltk.CFG module. The grammar consists of non-terminal symbols (S, VP, NP, PP, V, Det, N, P) and terminal symbols (words). Production rules define how these symbols can be combined to form valid sentences. For instance, S -> NP VP means that a sentence (S) can be composed of a noun phrase (NP) followed by a verb phrase (VP).

Parse Trees:

A parse tree is a graphical representation of the syntactic structure of a sentence according to a given grammar. It illustrates how the sentence can be parsed into its constituent parts, following the rules defined in the grammar. Each node in the parse tree corresponds to a symbol in the grammar, and the edges represent the relationships between these symbols.

Parse trees are invaluable for understanding the syntactic structure of sentences and for parsing input sentences to determine their grammaticality. They provide a clear visualization of how different parts of speech combine to form sentences, aiding in language analysis and processing tasks.

Usage and Implementation:

In the Following Python code, CFG and Parse Trees are used to generate example sentences and parse input sentences, respectively. Here is a breakdown of the code's functionality:

Defining the CFG: The code defines a CFG using the nltk.CFG.fromstring() method. Production rules specify how different parts of speech combine to form sentences.

Generating Example Sentences: The generate() function is used to produce example sentences based on the defined grammar. These sentences adhere to the rules specified in the CFG.

Parsing Input Sentences: The RecursiveDescentParser class is employed to parse input sentences according to the CFG. The parser attempts to match the input sentence with the grammar rules and generates parse trees if the sentence is syntactically valid.

Printing Parse Trees: If the input sentence is valid, parse trees are printed to illustrate the syntactic structure of the sentence according to the CFG.

In conclusion, Context-Free Grammar (CFG) and Parse Trees are indispensable tools in natural language processing for describing, generating, and parsing sentences. They enable us to model the syntactic structure of languages and facilitate various language processing tasks, including parsing, machine translation, and syntactic analysis.

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 4	
Aim: Design Context-free grammar and Parse trees.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

CODE:

```
from nltk import CFG
from nltk.parse.generate import generate
from nltk.parse import RecursiveDescentParser
print("Saail Chavan - KFPMSCCS016 CL_P4\n")

# Define a context-free grammar
grammar = CFG.fromstring("""
    S -> NP VP
    VP -> V NP | V NP PP
    PP -> P NP
    V -> "admires" | "loves" | "hates"
    NP -> "Alice" | "Bob" | "Charlie" | Det N | Det N PP
    Det -> "the" | "a"
    N -> "cat" | "dog" | "rabbit"
    P -> "on" | "under" | "beside"
""")

# Generate and print example sentences based on the defined grammar
print("Generated sentences:")
for sentence in generate(grammar, n=10):
    print(' '.join(sentence))

rd_parser = RecursiveDescentParser(grammar)
input_sentence = "Alice loves the cat".split()
print("\nParse tree for the input sentence:")
for tree in rd_parser.parse(input_sentence):
    print(tree)
```

OUTPUT:

```
Saail Chavan - KFPMSCCS016 CL_P4

Generated sentences:
Alice admires Alice
Alice admires Bob
Alice admires Charlie
Alice admires the cat
Alice admires the dog
Alice admires the rabbit
Alice admires a cat
Alice admires a dog
Alice admires a rabbit
Alice admires the cat on Alice

Parse tree for the input sentence:
(S (NP Alice) (VP (V loves) (NP (Det the) (N cat))))
```

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 5	
Aim: Translating text to Indian Languages and languages of different countries.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

Translating text into various languages serves several important purposes in today's interconnected world. It facilitates communication between people who speak different languages, fosters cultural exchange, enables access to information across language barriers, and supports businesses in reaching global markets. In the context of India, a linguistically diverse country with hundreds of languages spoken across its regions, translation plays a crucial role in promoting unity, understanding, and accessibility.

The following code demonstrates two methods of translating text into Indian languages as well as languages from different countries. Let's discuss each method and its implementation:

Method 1: Using the translate module

This method utilizes the translate module to perform translation. It requires installation of the module (pip install translate). The translate_to_language function takes the text to be translated and the target language code as inputs. It then initializes a translator object with the target language and translates the text using the translate method.

Method 2: Using the googletrans library

This method utilizes the googletrans library for translation. Similar to the previous method, it requires installation of the library (pip install googletrans==4.0.0-rc1). The translate_to_language function here takes the text and destination language code as inputs. It initializes a Translator object and translates the text using the translate method, specifying the destination language.

Key Differences:

Library Used: Method 1 uses the translate module, while Method 2 uses the googletrans library. These libraries provide different interfaces for accessing translation services.

Translation Service: Method 1 uses the translation service provided by the translate module, while Method 2 relies on Google Translate service. Each service may have its own strengths, limitations, and performance characteristics.

Language Codes: Both methods use language codes to specify the target languages. These codes may vary slightly between libraries or translation services.

Functionality: While both methods achieve the same goal of translating text into different languages, they may offer different features or levels of customization. For example, Google Translate might offer more advanced features like language detection, transliteration, and automatic language switching.

In conclusion, translating text to Indian languages and languages of different countries is essential for promoting communication, understanding, and accessibility on a global scale. The provided code demonstrates two methods of implementing translation, each utilizing different libraries and services. The choice between these methods may depend on factors such as preferred features, performance, and compatibility with specific use cases.

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 5	
Aim: Translating text to Indian Languages and languages of different countries.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

CODE:

Method 1:

```
from translate import Translator
print("Saail Chavan - KFPMSCCS016 CL_P5a")

def translate_to_language(text, to_lang):
    translator = Translator(to_lang=to_lang)
    return translator.translate(text)
translations = {}
text = "Good Morning!"

# Translate to Marathi
translations["Marathi"] = translate_to_language(text, "marathi")
# Translate to Bengali
translations["Bengali"] = translate_to_language(text, "bengali")
# Translate to Kannada
translations["Kannada"] = translate_to_language(text, "kannada")
# Translate to Spanish
translations["Spanish"] = translate_to_language(text, "spanish")
# Translate to German
translations["German"] = translate_to_language(text, "german")
# Translate to Japanese
translations["Japanese"] = translate_to_language(text, "japanese")

for lang, translation in translations.items():
    print(f"{lang}: {translation}")
```

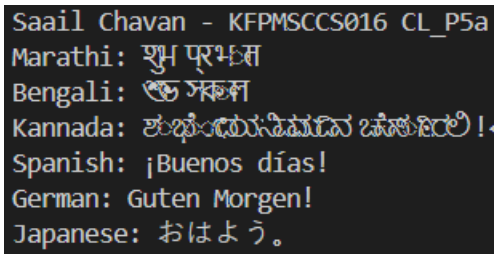
Method 2:

```
from googletrans import Translator
print("Saail Chavan - KFPMSCCS016 CL_P5b")

def translate_to_language(text, dest):
    translator = Translator()
    translation = translator.translate(text, dest=dest)
    return translation.text

to_translate = input("Enter a sentence to be translated: ")
translations = {}

# Translate to Marathi
translations["Marathi"] = translate_to_language(to_translate, "mr")
# Translate to Bengali
translations["Bengali"] = translate_to_language(to_translate, "bn")
# Translate to Kannada
translations["Kannada"] = translate_to_language(to_translate, "kn")
# Translate to Spanish
translations["Spanish"] = translate_to_language(to_translate, "es")
```

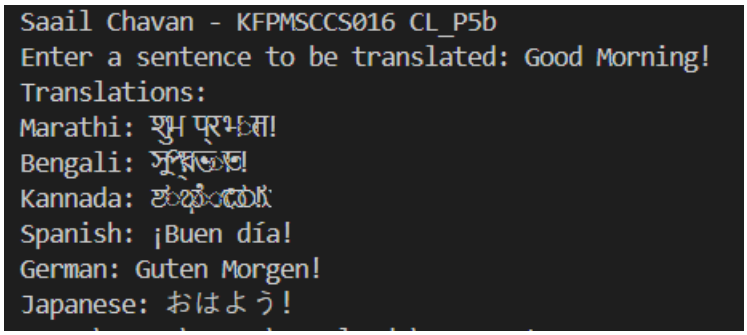
```
# Translate to German
translations["German"] = translate_to_language(to_translate, "de")
# Translate to Japanese
translations["Japanese"] = translate_to_language(to_translate, "ja")
print("Translations:")

for lang, translation in translations.items():
    print(f"{lang}: {translation}")
```

OUTPUT:

Method1:



Method2:



HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 6	
Aim: Develop an End-to-End Chatbot.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

Chatbots have become increasingly prevalent across various industries, serving as efficient tools for customer service, information retrieval, and even companionship. Developing an end-to-end chatbot involves creating a system capable of understanding and responding to user input in a conversational manner.

Chatbots offer several advantages, including:

- **24/7 Availability**: Chatbots can provide round-the-clock assistance, improving customer satisfaction and service accessibility.
- **Scalability:** They can handle multiple conversations simultaneously, making them suitable for large-scale deployments.
- **Efficiency:** Chatbots can quickly provide answers to common queries, reducing the workload on human operators.
- **Consistency:** They ensure consistent responses, maintaining brand identity and messaging standards.

Implementation Overview:

The following code utilizes Python and the NLTK library to create a simple rule-based chatbot capable of engaging in basic conversations. Let's break down its implementation:

Importing Libraries: The code starts by importing the necessary libraries, including NLTK for natural language processing capabilities.

Defining Conversation Patterns: The pairs variable holds a list of conversation patterns paired with corresponding responses. Each pattern-response pair represents a rule that the chatbot follows when processing user input.

Creating the Chatbot: The Chat object from NLTK's nltk.chat.util module is instantiated with the defined conversation patterns and a set of reflection rules. Reflections allow the chatbot to recognize pronouns and rephrase responses accordingly.

Conversing with Users: The converse() method is called on the Chat object, initiating the chatbot's interaction with users. It continuously prompts users for input and responds based on the defined patterns until the user decides to quit the conversation.

NLTK is a popular choice for developing chatbots due to its ease of use and extensive collection of tools for natural language processing and text analysis. It provides functionalities for tokenization, stemming, part-of-speech tagging, and more, making it well-suited for building rule-based chatbots like the one demonstrated in the code.

In Conclusion Building an end-to-end chatbot involves a combination of natural language processing techniques, rule-based systems, and user interface design. While the following code offers a simple starting point, the potential for customization and expansion is vast. By leveraging tools like NLTK and incorporating user feedback, developers can create chatbots that effectively address diverse user needs and contribute to enhanced user experiences across various domains.

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 6	
Aim: Develop an End-to-End Chatbot.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

CODE:

```
import nltk
from nltk.chat.util import Chat, reflections

print("Saail Chavan - KFPMSCCS016 CL_P6")

pairs = [
    [r"my name is (.*)",
     ["Hello %1, How are you today?"]
    ],
    [r"hi|hey|hello",
     ["Hello", "Hey there"]
    ],
    [r"what is your name ?",
     ["I'm Sheldon, your friendly chatbot."]
    ],
    [r"how are you ?",
     ["I'm fine"]
    ],
    [r"sorry (.*)",
     ["Its alright", "Its OK, never mind"]
    ],
    [r"i am fine",
     ["Great to hear that", "Awesome!"]
    ],
    [r"what can you do ?",
     ["I can have a conversation with you. You can ask me about anything or just chat for fun!"]
    ],
    [r"tell me a joke",
     ["Why don't scientists trust atoms? Because they make up everything!"]
    ],
    [r"how old are you ?",
     ["I don't have an age. I'm just a computer program."]
    ],
    [r"what is the weather today ?",
     ["I'm sorry, I don't have access to real-time data. You can check a weather website or app for
that!"]
    ],
    [r"thanks|thank you",
     ["You're welcome!", "No problem!"]
    ],
    [r"bye|goodbye",
```

```
    ["Goodbye! Take care.", "Bye bye, have a great day!"]
   ],
   [r"quit",
    ["Bye bye, take care. See you soon :) "]
   ],
]

Chatbot = Chat(pairs, reflections)
Chatbot.converse()
```

OUTPUT:

```
Saail Chavan - KFPMSCCS016 CL_P6
>hello
Hey there
>my name is Saail
Hello saail, How are you today?
>what is your name ?
I'm Sheldon, your friendly chatbot.
>how are you ?
I'm fine
>what can you do ?
I can have a conversation with you. You can ask me about anything or just chat for fun!
>how old are you ?
I don't have an age. I'm just a computer program.
>what is the weather today ?
I'm sorry, I don't have access to real-time data. You can check a weather website or app for that!
>tell me a joke
Why don't scientists trust atoms? Because they make up everything!
>Thank you
You're welcome!
>goodbye
Goodbye! Take care.
>quit
Bye bye, take care. See you soon :)
```

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 7	
Aim: Implement Text Summarization using NLTK.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

Text summarization is a vital task in Natural Language Processing (NLP) that condenses lengthy documents into shorter versions while retaining the key information. NLTK (Natural Language Toolkit) is a powerful library in Python widely used for NLP tasks, including text summarization. This write-up will delve into the importance of text summarization, the essence of the provided code, and why NLTK is instrumental in this process.

Importance of Text Summarization:

- **Information Overload Management:** In today's digital age, we are inundated with vast amounts of textual data daily. Text summarization helps in managing this overload by extracting the most crucial information from documents.
- **Time Efficiency:** Summarized content provides a quick overview, saving time for readers who may not have the bandwidth to go through lengthy documents.
- **Decision Making:** Summaries offer concise insights, aiding decision-making processes in various fields such as research, business, and journalism.
- **Information Retrieval:** Summarized documents serve as effective reference points, enabling easy retrieval of pertinent information.

What is Text Summarization?

Text summarization is the process of distilling the main points from a piece of text while preserving its essence. There are two primary approaches to text summarization: extractive and abstractive.

Extractive Summarization: In this approach, key sentences or phrases are selected directly from the original text to form the summary. It involves identifying significant sentences based on various criteria such as word frequency, relevance, or importance.

Abstractive Summarization: Unlike extractive summarization, abstractive summarization generates summaries by interpreting and paraphrasing the content. This method involves understanding the meaning of the text and generating new sentences to convey the essential information.

In conclusion text summarization is a crucial component of NLP, offering numerous benefits such as managing information overload and facilitating efficient decision-making. NLTK provides powerful tools and algorithms for implementing text summarization techniques, as demonstrated in the foloowing code. By leveraging NLTK's functionalities, developers can build robust summarization systems catering to various use cases across different domains.

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 7	
Aim: Implement Text Summarization using NLTK.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

CODE:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

print("Saail Chavan - KFPMSCCS016 CL_P7")

text = """Humans communicate with each other using words and text.
The way that humans convey information to each other is called Natural Language.
Every day humans share a large quality of information with each other in various languages
as speech or text.
However, computers cannot interpret this data, which is in natural language,
as they communicate in 1s and 0s. The data produced is precious and can offer valuable insights.
Hence, you need computers to be able to understand, emulate and respond intelligently to human
speech.
Natural Language Processing or NLP refers to the branch of Artificial Intelligence
that gives the machines the ability to read, understand and derive meaning from human languages.
Natural language processing (NLP) is a machine learning technology that gives computers
the ability to interpret, manipulate, and comprehend human language.
Organizations today have large volumes of voice and text data from various
communication channels like emails, text messages, social media newsfeeds, video, audio, and more.
They use NLP software to automatically process this data, analyze the intent or
sentiment in the message, and respond in real time to human communication."""

stopWords = set(stopwords.words("english"))
words = word_tokenize(text)
freqTable = dict()

for word in words:
    word = word.lower()
    if word in stopWords:
        continue
    if word in freqTable:
        freqTable[word] += 1
    else:
        freqTable[word] = 1

sentences = sent_tokenize(text)
sentenceValue = dict()
for sentence in sentences:
    for word, freq in freqTable.items():
        if word in sentence.lower():
```

```
            if sentence in sentenceValue:
                sentenceValue[sentence] += freq
            else:
                sentenceValue[sentence] = freq

sumValues = 0
for sentence in sentenceValue:
    sumValues += sentenceValue[sentence]
average = int(sumValues / len(sentenceValue))

summary = ''
for sentence in sentences:
    if (sentence in sentenceValue) and (sentenceValue[sentence] > (average)):
        summary += " " + sentence
print(summary)
```

OUTPUT:

```
Saail Chavan - KFPMSCCS016 CL_P7
 However, computers cannot interpret this data, which is in natural language,
as they communicate in 1s and 0s. Natural Language Processing or NLP refers to th
e branch of Artificial Intelligence
that gives the machines the ability to read, understand and derive meaning from h
uman languages. Natural language processing (NLP) is a machine learning technolog
y that gives computers
the ability to interpret, manipulate, and comprehend human language. Organization
s today have large volumes of voice and text data from various
communication channels like emails, text messages, social media newsfeeds, video,
 audio, and more. They use NLP software to automatically process this data, analy
ze the intent or
sentiment in the message, and respond in real time to human communication.
```

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 8	
Aim: Build the Next Word Prediction Model.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 8	
Aim: Build the Next Word Prediction Model.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

CODE:
OUTPUT:

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 9	
Aim: Develop interactive application using Text to Speech and Speech to Text conversion.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

Text-to-Speech (TTS) and Speech-to-Text (STT) Conversion:

Text-to-Speech (TTS) and Speech-to-Text (STT) conversion are two essential components of natural language processing (NLP) systems. TTS allows computers to convert written text into spoken words, enabling human-computer interaction through auditory channels. On the other hand, STT performs the opposite operation, translating spoken language into written text, facilitating communication with machines through speech input.

Why TTS and STT are Important:

- **Accessibility:** TTS enables accessibility features for individuals with visual impairments, allowing them to interact with digital content through speech.
- **Hands-Free Operation:** STT enables hands-free operation of devices, making it convenient for users to perform tasks while driving, cooking, or engaging in other activities.
- **Natural Interaction:** Both TTS and STT contribute to more natural human-computer interaction, mimicking real-life communication patterns.
- **Increased Productivity:** STT can improve productivity by allowing users to dictate text rather than type it, especially for tasks like note-taking or composing emails.

Libraries and Modules Used:

pyttsx3: This library provides a cross-platform interface to Text-to-Speech engines in Python. It supports multiple TTS engines and allows developers to control various aspects of speech synthesis, such as voice selection, rate, and volume.

speech_recognition (sr): This library enables easy integration of Speech-to-Text functionality into Python applications. It supports multiple speech recognition engines and provides convenient APIs for capturing audio from different sources, such as microphones or audio files, and converting it into text.

The following code demonstrates the basic usage of TTS and STT functionalities using the pyttsx3 and speech_recognition libraries, respectively.

Text-to-Speech (TTS): The code initializes a TTS engine using pyttsx3.init(), prompts the user to input text, converts the text to speech using engine.say(), and finally plays the generated speech using engine.runAndWait().

Speech-to-Text (STT): The code opens an audio file, extracts the audio data using sr.WavFile, and then uses the recognize() method of the Recognizer class from speech_recognition library to perform STT conversion on the audio data, printing the recognized text.

In Conclusion Text-to-Speech (TTS) and Speech-to-Text (STT) conversion are vital technologies that enable natural and intuitive human-computer interaction. By leveraging libraries like pyttsx3 and speech_recognition, developers can easily integrate these capabilities into their applications, opening up new possibilities for accessibility, hands-free operation, and enhanced user experience.

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 9	
Aim: Develop interactive application using Text to Speech and Speech to Text conversion.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

CODE:

```
import pyttsx3
import speech_recognition as sr

#Text-To-Speech
engine = pyttsx3.init()
print("Saail Chavan - KFPMSCCS016 CL_P9")
print("TTS:")
text = input("Enter text: ")
engine.say(text)
engine.runAndWait()

#Speech-to-Text
print("\nSTT:")
filename = "16-122828-0002.wav"
# initialize the recognizer
r = sr.Recognizer()
# open the file
with sr.WavFile(filename) as source:
    # listen for the data (load audio to memory)
    audio_data = r.record(source)
    # recognize (convert from speech to text)
    text = r.recognize(audio_data)
    print(text,"\n")
```

OUTPUT:

```
Saail Chavan - KFPMSCCS016 CL_P9
TTS:
Enter text: Hello

STT:
I believe you are just talking nonsense
```

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 10	
Aim: Perform Sentiment Analysis on Amazon Product Reviews.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

In the era of e-commerce, understanding customer sentiments towards products is crucial for businesses to make informed decisions and enhance customer satisfaction. Amazon, being one of the largest online marketplaces, accumulates vast amounts of product reviews, making sentiment analysis a valuable tool for extracting insights from this data. This writeup delves into the significance of sentiment analysis on Amazon product reviews, its implementation, and its implications for businesses.

Why Sentiment Analysis on Amazon Product Reviews?

- **Consumer Insights:** Amazon reviews reflect customers' opinions, preferences, and experiences with products. Analyzing sentiments from these reviews provides valuable insights into consumer perceptions, which can be utilized for product development, marketing strategies, and brand management.
- **Competitive Analysis:** Understanding the sentiment surrounding competitor products enables businesses to identify strengths and weaknesses in their offerings. This competitive intelligence aids in refining product features and enhancing market positioning.
- **Quality Assurance:** Sentiment analysis helps in monitoring product quality and identifying issues or areas for improvement highlighted by customers. This proactive approach allows businesses to address concerns promptly and maintain customer satisfaction.
- **Brand Reputation Management:** Positive sentiments contribute to building a strong brand reputation, whereas negative sentiments can harm brand image. Analyzing sentiments allows businesses to manage their reputation by addressing negative feedback and amplifying positive experiences.

Implementation:

The following code demonstrates a basic implementation of sentiment analysis on Amazon product reviews using Python's Natural Language Toolkit (NLTK) library. Below are the key steps:

- **Data Loading:** The Amazon review dataset is loaded into a Pandas DataFrame.
- **Preprocessing:** Text preprocessing is performed to enhance the quality of text data. This includes tokenization, removing stop words, and lemmatization to normalize the text.
- **Sentiment Analysis:** NLTK's Vader Sentiment Intensity Analyzer is utilized to analyze the sentiment of preprocessed text. The analyzer provides polarity scores for positive, neutral, and negative sentiments.
- **Scoring and Classification:** Based on the polarity scores, a sentiment label is assigned to each review. If the positive score exceeds the negative score, the sentiment is classified as positive (1); otherwise, it is classified as negative (0).
- **Result Analysis:** The sentiment analysis results are analyzed to understand the distribution of positive and negative sentiments among the reviews.

In Conclusion Sentiment analysis on Amazon product reviews offers valuable insights that drive business decisions and enhance customer satisfaction. By leveraging text analytics techniques, businesses can extract meaningful information from large volumes of unstructured data, enabling them to stay competitive in the dynamic e-commerce landscape. Continuous monitoring and analysis of sentiments empower businesses to adapt, innovate, and deliver products that resonate with customer preferences and expectations.

HSNC UNIVERSITY, MUMBAI
KISHINCHAND CHELLARAM COLLEGE
M.Sc. Computer Science Semester 2 (FY. 2023-24)

Practical 10	
Aim: Perform Sentiment Analysis on Amazon Product Reviews.	
Name: Saail Chavan	Roll No: KFPMSCCS016
Performance date: xx – 03 – 2024	Sign:

CODE:

```
import pandas as pd
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

print("Saail Chavan - KFPMSCCS016 CL_P9")

# Load the amazon review dataset
df = pd.read_csv('amazondata.csv')

# create preprocess_text function
def preprocess_text(text):
    # Tokenize the text
    tokens = word_tokenize(text.lower())
    # Remove stop words
    filtered_tokens = [token for token in tokens if token not in stopwords.words('english')]
    # Lemmatize the tokens
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in filtered_tokens]
    # Join the tokens back into a string
    processed_text = ' '.join(lemmatized_tokens)
    return processed_text

# apply the function df
df['reviewText'] = df['reviewText'].apply(preprocess_text)

# initialize NLTK sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

# create get_sentiment function
def get_sentiment(text):
    scores = analyzer.polarity_scores(text)
    sentiment = 1 if scores['pos'] > 0 else 0
    return sentiment

# apply get_sentiment function
df['sentiment'] = df['reviewText'].apply(get_sentiment)
print(df)
print(df['sentiment'].value_counts())
```

OUTPUT:

```
Saail Chavan - KFPMSCCS016 CL_P9
                                                reviewText  Positive  sentiment
0   one best apps acording bunch people agree bomb...           1          1
1   pretty good version game free . lot different ...           1          1
2   really cool game . bunch level find golden egg...           1          1
3   silly game frustrating , lot fun definitely re...           1          1
4   terrific game pad . hr fun . grandkids love . ...           1          1
5   entertaining game ! n't smart play . guess 's ...           1          1
6   awesome n't need wi ti play trust . really fun...           1          1
7   awesome bet one even read review know game goo...           1          1
8   basicly free version ad . 's actually awesome ...           1          1
9   far best free app available anywhere . helped ...           1          1
10  definitely great game . get 6-year-old grand-n...           1          1
11  . fun time consuming . work great kindle fire ...           1          1
12  good like physic game , free game , bird game ...           1          1
13           n't like .i think kid . really many game           0          0
14  got end pre-installed version . confused , che...           0          0
15  hate say everything l got rid shut kindle stup...           0          0
16  hat never liked sucked boring like rio one tho...           0          1
17  hear app everywhere . everyone smartphone tabl...           0          1
18  'll admit , 'm late game getting wonderful gam...           1          1
19  love angry bird , issue ad supported game . re...           0          1
20  love game level never time enjoy far get anoth...           0          1
21  see nothing wrong game busy n't think 's fun ....           0          1
22  n't great get hard really frustrating . would ...           0          1
1    20
0     3
Name: sentiment, dtype: int64
```