Sardar Patel Institute of Technology,Mumbai

Department of Computer Science Engineering

B.E. Sem-VII- PE-IV (2024-2025)

IT 24 - AI in Healthcare

Experiment 9: Implement NLTK, spaCY (NLP libraries in Python) to extract important information like patient diagnosis,medications and lab results.

Name: Adwait Purao **UID: 2021300101** **Date: 17/11/24**

Objective: To implement NLTK library in healthcare dataset.

Theory:

Introduction to Natural Language Processing (NLP)

Natural Language Processing (NLP) is a transformative field within artificial intelligence (AI) that enables machines to interpret, understand, and respond to human language. NLP intersects linguistics, computer science, and AI, combining statistical, machine learning, and deep learning methods to process and analyze text and speech. NLP applications are widespread, powering technologies like chatbots, voice-activated assistants, translation systems, and predictive text. By bridging the gap between human communication and machine interpretation, NLP has become fundamental in creating interactive and user-friendly AI systems.

1. Core Concepts in NLP

1.1. Language Processing Pipeline

The NLP pipeline involves several stages to transform raw text into actionable insights. A standard pipeline may include:

- **Text Preprocessing**: Removing noise, converting to lowercase, and tokenizing sentences and words.
- **Tokenization**: Splitting text into individual words or phrases.
- **Normalization**: Converting words to their base forms (stemming and lemmatization).
- **Vectorization**: Representing text numerically for machine processing.

- **Analysis**: Conducting sentiment analysis, entity recognition, and syntactic parsing.
- **Generation**: Producing text, such as in chatbots or summarization systems.

1.2. Tokenization

Tokenization is one of the foundational steps, breaking down text into smaller units, typically words or sentences. This simplification is necessary because NLP algorithms often require distinct units of language to analyze patterns, sentiments, and context. Tokenization allows analysis of individual words, word frequencies, n-grams, and co-occurrence patterns.

1.3. Stop Words Removal

Stop words are common words (e.g., "the," "is," "in") that don't add significant meaning to a sentence and are often filtered out in NLP tasks. Removing stop words reduces noise in the data, focusing the analysis on more meaningful content. The NLTK library provides a list of common stop words in various languages.

1.4. Stemming and Lemmatization

- **Stemming** reduces words to their root forms, which may not always be linguistically accurate (e.g., "running" becomes "run").
- **Lemmatization** is more sophisticated and aims to map words to their base or dictionary forms, such as converting "am," "are," and "is" into "be."

NLTK provides both stemming and lemmatization methods to ensure text standardization.

---

2. NLTK: An Overview of the Natural Language Toolkit

The Natural Language Toolkit (NLTK) is a powerful library for Python that supports a wide range of NLP operations, from basic tokenization and stop word removal to more advanced techniques like parsing, entity recognition, and sentiment analysis. Here's a detailed look at some of NLTK's primary components and functionalities:

2.1. Text Processing Tools

NLTK's primary advantage lies in its collection of tools for breaking down and preparing text. Some of the key text processing functionalities include:

- **Tokenizers**: For both word and sentence tokenization, useful in breaking down raw text.
- **Corpus and Lexical Resources**: Large datasets of text (corpora) like the Brown Corpus, WordNet, and Gutenberg, enabling model training and language insights.
- **Stemmers and Lemmatizers**: NLTK includes several stemmers like the Porter and Lancaster stemmers, along with WordNet lemmatizer, to standardize words.

2.2. Text Classification and Sentiment Analysis

NLTK has built-in tools for text classification tasks, often leveraging statistical or machine learning algorithms. Using labeled data, NLTK can classify text by training classifiers like Naive Bayes, MaxEnt, or decision trees. Sentiment analysis, which classifies text based on positive, neutral, or negative sentiment, is another common application. NLTK enables creating custom sentiment analyzers or using pretrained models.

2.3. Named Entity Recognition (NER)

Named Entity Recognition (NER) identifies and categorizes entities like people, organizations, locations, and dates within a text. NLTK's NER capabilities are based on pre-trained models and can also be customized using NLTK's tagging functionalities, aiding in tasks like information extraction and text summarization.

2.4. Parsing and Syntactic Analysis

NLTK provides parsers, including recursive descent and shift-reduce parsers, to analyze the syntactic structure of sentences. Parsing helps in understanding the grammatical structure, enabling extraction of relationships between entities in sentences. This understanding is essential for deeper text comprehension, such as finding subject-verb-object relationships.

2.5. Corpus Access and Text Corpora

NLTK includes several corpora, which are invaluable resources for NLP research. Some popular corpora within NLTK include:

- **Brown Corpus**: One of the first widely used corpora, containing a variety of English text genres.
- **WordNet**: A lexical database for English, providing synonyms, antonyms, and semantic relationships.
- **Movie Reviews**: A dataset of movie reviews with labeled sentiments, often used for sentiment analysis.
- **Gutenberg Corpus**: A collection of literary texts from authors like Shakespeare, Jane Austen, and Charles Dickens.

Using these corpora, NLTK enables users to train and evaluate models without needing to source extensive external datasets.

---

3. Key NLP Techniques Using NLTK

3.1. Text Tokenization

Tokenization is essential for breaking down text into smaller components. NLTK provides several tokenizers for various applications. For instance, `word_tokenize` splits text into

words, while `sent_tokenize` splits it into sentences. The library's regex-based tokenizers are also useful for custom tokenization tasks, such as splitting by punctuation.

3.2. Stop Word Removal

By filtering out commonly used words, NLP models focus on meaningful words. NLTK includes a list of stop words for various languages, accessible using the `nltk.corpus.stopwords` module.

3.3. Stemming and Lemmatization

Stemming and lemmatization normalize words. NLTK provides the `PorterStemmer` and `WordNetLemmatizer`, each suited to different applications. Stemming is typically faster but less accurate, while lemmatization requires part-of-speech tags for greater precision.

3.4. Named Entity Recognition (NER)

NER identifies and classifies entities. Using NLTK's `ne_chunk`, named entities can be automatically recognized and categorized.

3.5. Part-of-Speech (POS) Tagging

POS tagging labels words as nouns, verbs, adjectives, etc., based on context. This step is crucial for tasks that rely on understanding sentence structure, such as syntactic parsing and entity recognition.

3.6. Text Classification

Text classification in NLTK can be achieved using supervised learning, such as Naive Bayes or decision trees. NLTK also supports feature extraction, essential for creating models that can categorize text into different classes, such as sentiment or topic.

---

4. Applications of NLP and NLTK

4.1. Sentiment Analysis

Sentiment analysis determines the emotional tone in text, making it popular for analyzing customer feedback, social media, and reviews. NLTK's text classification tools allow for creating models that classify text as positive, neutral, or negative.

4.2. Healthcare Data Analysis

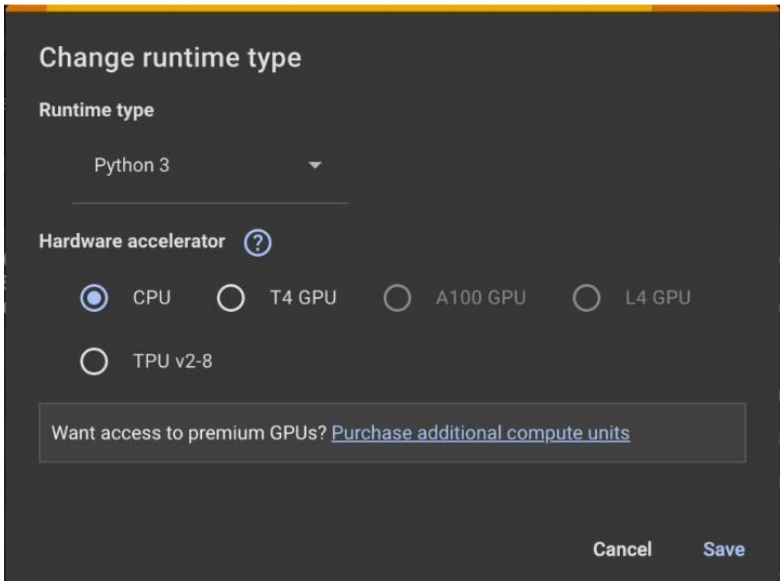
In healthcare, NLP helps analyze patient records, diagnoses, and prescriptions. NLTK can extract essential information like diseases, symptoms, and treatments, assisting in research, patient management, and diagnostic support.

4.3. Chatbots and Virtual Assistants

Chatbots use NLP to understand user input and provide relevant responses. With NLTK, chatbots can be trained to process natural language queries and respond appropriately, making them useful for customer service, virtual assistants, and interactive systems.

Steps:

1. Setting up the Environment



2.Import libraries and and download necessary NLTK data file

```
import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize, sent_tokenize

from nltk import pos_tag, ne_chunk

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('averaged_perceptron_tagger')
```

```
nltk.download('maxent_ne_chunker')

nltk.download('words')

import pandas as pd
```

3.Loading the Dataset

```
data = pd.read_csv('medquad.csv')
```

4.Data Preprocessing

```
data['answer']                                                    =
data['answer'].str.lower().str.replace(r'[^a-z\s]', '')
```

5.Tokenization

```
# Fill NaN values if any

data['answer']                                                    =
data['answer'].fillna('').astype(str).str.lower()



# Now apply sent_tokenize and word_tokenize

data['sent_tokens'] = data['answer'].apply(sent_tokenize)

data['word_tokens'] = data['answer'].apply(word_tokenize)
```

6.Removing Stop Words

```
stop_words = set(stopwords.words('english'))

data['filtered_words']   =   data['word_tokens'].apply(lambda   x:
[word for

word in x if word not in stop_words])
```

7.Named Entity Recognition (NER)

```
def get_named_entities(text):
    words = word_tokenize(text)
    pos_tags = pos_tag(words)
    chunks = ne_chunk(pos_tags)
    entities = []
    for chunk in chunks:
        if hasattr(chunk, 'label'):
            entity = " ".join(c[0] for c in chunk)
            entities.append((entity, chunk.label()))
    return entities


data['named_entities'] =
data['answer'].apply(get_named_entities)
```

8.Extracting Diagnoses and Medications

```
diagnosis_keywords = ['glaucoma', 'vision loss', 'optic nerve']
medication_keywords = ['medication', 'drug', 'laser surgery',
'eye drops']


def extract_diagnoses(text):
      return list(set([word for word in word_tokenize(text) if
word in diagnosis_keywords]))


def extract_medications(text):
      return list(set([word for word in word_tokenize(text) if
word in medication_keywords]))
```

```
data['diagnoses'] = data['answer'].apply(extract_diagnoses)

data['medications'] = data['answer'].apply(extract_medications)
```

9.Summarizing Extracted Information

```
summary = data[['question', 'diagnoses', 'medications']]

summary.head()
```

Output:

	question	diagnoses	medications
0	What is (are) Glaucoma ?	[glaucoma]	[]
1	What causes Glaucoma ?	[glaucoma]	[]
2	What are the symptoms of Glaucoma ?	[glaucoma]	[]
3	What are the treatments for Glaucoma ?	[glaucoma]	[medication]
4	What is (are) Glaucoma ?	[glaucoma]	[]

CONCLUSION:

In this experiment, I used NLTK on a healthcare dataset to process text by tokenizing, removing stop words, and applying Named Entity Recognition (NER). This allowed me to identify and summarize key entities like diagnoses and treatments, highlighting NLTK's effectiveness in transforming healthcare text data into accessible insights.