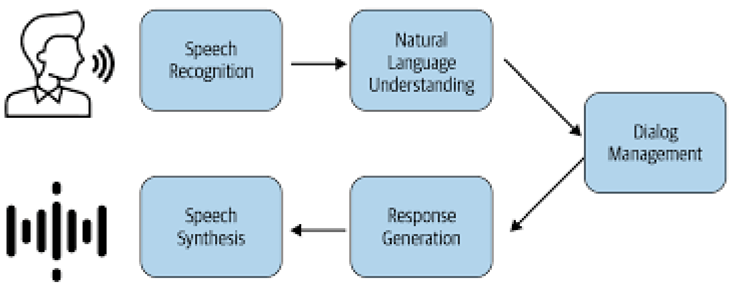
R V College of Engineering
Department of Computer Science and Engineering
CIE 1: Scheme

Subject : (Code)	**Natural Language Processing (Group B) (IS355TBC)**	**Semester:** **5th BE**
Date :	**Duration: 120** minutes	**Staff: Dr.Sindhu D V**
Name :	**USN** :	**Section :** **A**

512

Q. No	Part-A	Marks	Level	CO
1	True	1	2	2
2	Transformers model the textual context but not in a sequential manner. Given a word in the input, it prefers to look at all the words around it and represent each word with respect to its context.	1	1	1
3	NLTK	2	3	2
4	Because of aspects like ambiguity, the need for contextual information, and idioms	1	1	1
5	The number of times a word appears in a document	2	3	1
6	[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}	2	2	1
7	**Tokenization**: • Tokenization is the process of **splitting text** into smaller units called **tokens**. These tokens could be words, characters, or subwords, depending on the level of tokenization. The main goal is to break down a sentence or a piece of text into manageable chunks for further analysis. • **Example**: o Input text: "I love NLP." o Tokens: ['I', 'love', 'NLP', '.'] **Lemmatization**: • Lemmatization is the process of reducing words to their **base or root form**, known as a **lemma**. Unlike stemming (which simply removes suffixes), lemmatization considers the context of the word and applies proper linguistic rules to transform words to their base form. • **Example**: o Input word: "running" o Lemmatized form: "run" o Input word: "better" o Lemmatized form: "good"	2	1	1

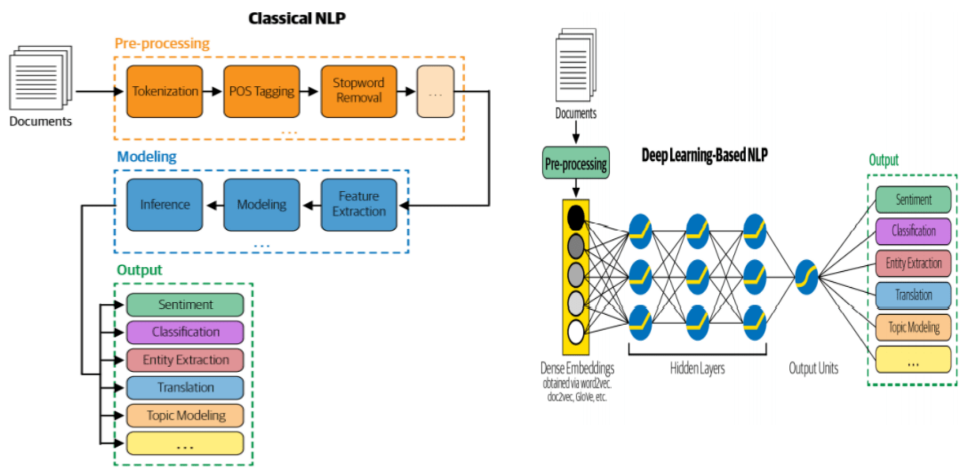
Q. No	**Answer all questions**	**Marks**	**Level**	**CO**
1a	Despite such tremendous success, DL is still not the silver bullet for all NLP tasks when it comes to the industrial applications. Some of the key reasons for this are as follows: Overfitting on small datasets Few-shot learning and synthetic data generation Domain adaption Interpretable models Common sense and world knowledge.	5	2	3
1b	 1. Speech recognition 2. Natural Language Understanding 3. Dialog management 4. Response generation 5. Speech synthesis	5	2	1
2	Data is a heart of any ML system. In most of the industrial projects, it is often the data becomes the bottleneck. ▪ Use public dataset ▪ Scrape data ▪ Product intervention ▪ Data augmentation 1. Synonym replacement 2. Back translation 3. Bigram flipping 4. Replacing entities 5. Adding noise to data	10	4	2
3	1. Machine Translation	2m×	2	1

	2. Sentimental analysis	5=1 0		

4a	import nltk
	from nltk.corpus import stopwords
	from nltk.tokenize import word_tokenize
	import string
	
	# Ensure necessary resources are downloaded
	nltk.download('punkt')
	nltk.download('stopwords')
	
	# Define the function
	def load_and_clean_corpus(file_path):
	    # Load the corpus from a file
	    with open(file_path, 'r') as file:
	        text = file.read()
	
	    # Tokenize the text into words
	    tokens = word_tokenize(text)
	
	    # Convert to lowercase
	    tokens = [word.lower() for word in tokens]
	
	    # Remove punctuation
	    tokens = [word for word in tokens if word.isalnum()]
	
	    # Remove stopwords
	    stop_words = set(stopwords.words('english'))
	    cleaned_tokens = [word for word in tokens if word not in stop_words]
	
	    return cleaned_tokens
	
	# Test the function
	file_path = "sample_corpus.txt" # Replace with your file's path
	cleaned_corpus = load_and_clean_corpus(file_path)
	
	print("Cleaned Tokens:", cleaned_corpus)
		6	1,2	1

4b	There are two approaches 1. A classical NLP 2. DL pipeline	4	3	3

<table>
<tr>
<td></td>
<td>



</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5a.</td>
<td>

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

# Preprocessing function
def preprocess_review(review):
    # Remove non-alphanumeric characters
    review = re.sub(r'\W', ' ', review)

    # Convert to lowercase
    review = review.lower()

    # Tokenization
    tokens = nltk.word_tokenize(review)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    return ' '.join(tokens)

# Preprocess all reviews
processed_reviews = [preprocess_review(review) for review in reviews]
```

</td>
<td>10</td>
<td>3</td>
<td>2</td>
</tr>
</table>

	from sklearn.feature_extraction.text import CountVectorizer
	
	# Initialize CountVectorizer
	vectorizer = CountVectorizer()
	
	# Transform reviews into a Bag of Words representation
	X = vectorizer.fit_transform(processed_reviews)
	from sklearn.model_selection import train_test_split
	from sklearn.naive_bayes import MultinomialNB
	from sklearn.metrics import classification_report
	
	# Sample labels (1 for positive, 0 for negative)
	# In a real-world scenario, these would be obtained through manual labeling or pre-labeled data.
	labels = [1 if 'good' in review or 'excellent' in review else 0 for review in processed_reviews]
	
	# Split the data into train and test sets
	X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2, random_state=42)
	
	# Train Naive Bayes classifier
	classifier = MultinomialNB()
	classifier.fit(X_train, y_train)
	
	# Test the classifier
	y_pred = classifier.predict(X_test)
	
	# Evaluate the model
	print(classification_report(y_test, y_pred))			

Marks Distribution	Particulars		CO1	CO2	CO3	CO4	L1	L2	L3	L4	L5	L6
	Test	Max Marks	28	19	13		6	26	18	10		

BT-Blooms Taxonomy, CO-Course Outcomes, M-Marks