

## Class 13: Introduction to NLP

Natural Language Processing (NLP) enables machines to understand, interpret, and generate human language. This class covers the fundamental NLP tasks, text preprocessing techniques, and regular expressions (regex) for pattern matching.

---

### 1. NLP Tasks

NLP tasks involve various operations on text data. Here are some common tasks:

1. **Text Classification:** Categorizing text into predefined labels (e.g., spam detection, news classification).
2. **Sentiment Analysis:** Determining whether a text expresses a positive, negative, or neutral sentiment.
3. **Named Entity Recognition (NER):** Identifying names of people, places, organizations, etc.
4. **Part-of-Speech (POS) Tagging:** Labeling words with their grammatical roles (noun, verb, adjective, etc.).
5. **Machine Translation:** Translating text from one language to another.
6. **Text Summarization:** Creating a short summary of a document.

#### Example: Text Classification with Scikit-Learn

We will classify text using a simple Naïve Bayes model.

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn.pipeline import make_pipeline
```

```
# Sample dataset
```

```
texts = ["I love this movie", "This is a terrible film", "Great acting and story", "Worst movie ever!"]
```

```
labels = [1, 0, 1, 0] # 1 = Positive, 0 = Negative
```

```
# Build and train the model
```

```
model = make_pipeline(CountVectorizer(), MultinomialNB())
```

```
model.fit(texts, labels)
```

```
# Prediction
test_text = ["I hate this movie"]
print(model.predict(test_text)) # Output: [0] (Negative sentiment)
```

---

## 2. Text Preprocessing

Text preprocessing is crucial to clean and normalize text before applying NLP models.

### 2.1 Tokenization

Splitting a sentence into words or subwords.

```
import nltk

from nltk.tokenize import word_tokenize

nltk.download('punkt')

text = "Natural Language Processing is amazing!"
tokens = word_tokenize(text)

print(tokens) # Output: ['Natural', 'Language', 'Processing', 'is', 'amazing', '!']
```

### 2.2 Stemming

Reduces words to their root form.

Mathematical Intuition:

$$\text{stem}(\text{word}) = \arg\min_{\text{root}} \text{rootdistance}(\text{word}, \text{root}) \quad \text{stem}(\text{word}) = \arg\min_{\text{root}} \{\text{distance}(\text{word}, \text{root})\}$$

Example:

```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

words = ["running", "flies", "easily", "happily"]

stemmed_words = [stemmer.stem(word) for word in words]

print(stemmed_words) # Output: ['run', 'fli', 'easili', 'happili']
```

### 2.3 Lemmatization

Lemmatization maps words to their base forms using a dictionary.

Example:

```
from nltk.stem import WordNetLemmatizer
```

```
from nltk.corpus import wordnet
```

```
nltk.download('wordnet')
```

```
lemmatizer = WordNetLemmatizer()
```

```
words = ["running", "flies", "easily", "happily"]
```

```
lemmatized_words = [lemmatizer.lemmatize(word, pos="v") for word in words]
```

```
print(lemmatized_words) # Output: ['run', 'fly', 'easily', 'happily']
```

## 2.4 Stopword Removal

Stopwords are common words (e.g., "the", "is", "in") that don't add much meaning.

```
from nltk.corpus import stopwords
```

```
nltk.download('stopwords')
```

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

```
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
```

```
print(filtered_tokens) # Output: ['Natural', 'Language', 'Processing', 'amazing', '!']
```

---

## 3. Regular Expressions (Regex)

Regex helps find patterns in text.

### 3.1 Extracting Email Addresses

```
import re
```

```
text = "My email is example@gmail.com and support@nlp.com"
```

```
emails = re.findall(r"[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}", text)
```

```
print(emails) # Output: ['example@gmail.com', 'support@nlp.com']
```

### 3.2 Extracting Phone Numbers

```
text = "Call me at +1-800-555-1234 or (123) 456-7890"
```




```
phones = re.findall(r"\+?\d{1,3}[-.\s]?(\d{3})?[-.\s]?d{3}[-.\s]?d{4}", text)
print(phones) # Output: ['+1-800-555-1234', '(123) 456-7890']
```

### 3.3 Removing Special Characters

```
text = "Hello! How's it going? @user #NLP"
clean_text = re.sub(r"^[a-zA-Z0-9\s]", "", text)
print(clean_text) # Output: 'Hello Hows it going NLP'
```

---

## Conclusion

In this class, we covered:  **NLP tasks** (Text classification, Sentiment Analysis, etc.)  
 **Text preprocessing** (Tokenization, Stemming, Lemmatization, Stopword Removal)  
 **Regex applications** (Pattern Matching, Text Extraction)