Twitter Data Analysis for Stemming and Lemmatization

In [ ]:
```
import http.client
import json
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer
from nltk.stem import WordNetLemmatizer
import spacy
```

Initialize stemmers and lemmatizers

In [ ]:
```
porter = PorterStemmer()
snowball = SnowballStemmer("english")
lancaster = LancasterStemmer()
wordnet_lemmatizer = WordNetLemmatizer()
spacy_nlp = spacy.load("en_core_web_sm")
```

Define stemming and lemmatization functions

In [ ]:
```
nlp = spacy.load("en_core_web_sm")
def porter_stemmer(text):
    tokens = word_tokenize(text)
    stemmed = [porter.stem(token) for token in tokens]
    return ' '.join(stemmed)

def snowball_stemmer(text):
    tokens = word_tokenize(text)
    stemmed = [snowball.stem(token) for token in tokens]
    return ' '.join(stemmed)

def lancaster_stemmer(text):
    tokens = word_tokenize(text)
    stemmed = [lancaster.stem(token) for token in tokens]
    return ' '.join(stemmed)

def wordnet_lemmatizer(text):
    wordnet = WordNetLemmatizer()
    tokens = word_tokenize(text)
    lemmatized = [wordnet.lemmatize(token) for token in tokens]
    return ' '.join(lemmatized)

def spacy_lemmatizer(text):
    doc = nlp(text)
    lemmatized = [token.lemma_ for token in doc]
    return ' '.join(lemmatized)
```

Function to fetch tweets using RapidAPI

In [ ]:
```
def fetch_tweets(query, min_retweets=20, min_likes=20, limit=5, start_date="2022-01-01", lang
    conn = http.client.HTTPSConnection("twitter154.p.rapidapi.com")
    headers = {
        'x-rapidapi-key': "5c75308bf6msh2b302bcfa9a8e0ep109e11jsn949c33074b98",
        'x-rapidapi-host': "twitter154.p.rapidapi.com"
    }

    endpoint = f"/search/search/continuation?query={query}&section=top&min_retweets={min_retw
    conn.request("GET", endpoint, headers=headers)
    res = conn.getresponse()

    if res.status != 200:
        raise Exception(f"API request failed with status {res.status}")

    data = res.read()
    return json.loads(data)
```

Converting Data from API response to Dataframe

In [ ]:
```
# Fetch tweets containing the query NeetUG24Controversy
tweets_data = fetch_tweets("NeetUG24Controversy")

# Extract the tweet texts
tweets = [tweet['text'] for tweet in tweets_data.get('results', [])]

# Check if tweets are retrieved
if not tweets:
    raise Exception("No tweets were retrieved from the API")

# Convert to DataFrame for convenience
tweets_df = pd.DataFrame(tweets, columns=['Tweet'])
```

Adding New Columns for Stemmed and Lemmatized Text

In [ ]:
```
# Apply stemming and lemmatization functions to the DataFrame
tweets_df['Porter Stemmed'] = tweets_df['Tweet'].apply(porter_stemmer)
tweets_df['Snowball Stemmed'] = tweets_df['Tweet'].apply(snowball_stemmer)
tweets_df['Lancaster Stemmed'] = tweets_df['Tweet'].apply(lancaster_stemmer)
tweets_df['WordNet Lemmatized'] = tweets_df['Tweet'].apply(wordnet_lemmatizer)
tweets_df['Spacy Lemmatized'] = tweets_df['Tweet'].apply(spacy_lemmatizer)

# Display the DataFrame
tweets_df
```

Out[ ]:

	Tweet	Porter Stemmed	Snowball Stemmed	Lancaster Stemmed	WordNet Lemmatized	Spacy Lemmatized
0	No Hindus #Sanatani will pass without liking ...	no hindu # sanatani will pass without like thi...	no hindus # sanatani will pass without like th...	no hind # sanatan wil pass without lik thi twe...	No Hindus # Sanatani will pas without liking t...	no Hindus # Sanatani will pass without like ...
1	Submitted an urgent hearing application to the...	submit an urgent hear applic to the suprem cou...	submit an urgent hear applic to the suprem cou...	submit an urg hear apply to the suprem court s...	Submitted an urgent hearing application to the...	submit an urgent hearing application to the Su...
2	No Hindus #Sanatani will pass without liking ...	no hindu # sanatani will pass without like thi...	no hindus # sanatani will pass without like th...	no hind # sanatan wil pass without lik thi twe...	No Hindus # Sanatani will pas without liking t...	no Hindus # Sanatani will pass without like ...
3	Ab ye omr ki sheet pe questions ki speed kha s...	ab ye omr ki sheet pe question ki speed kha se...	ab ye omr ki sheet pe question ki speed kha se...	ab ye omr ki sheet pe quest ki spee kha se bta...	Ab ye omr ki sheet pe question ki speed kha se...	ab ye omr ki sheet pe question ki speed kha se...
4	This is really heart wrenching !\nसरकार इतनी क...	thi is realli heart wrench ! सरकार इतनी कमजोर ...	this is realli heart wrench ! सरकार इतनी कमजोर...	thi is real heart wrench ! सरकार इतनी कमजोर क्...	This is really heart wrenching ! सरकार इतनी कम...	this be really heart wrench ! \n सरकार इतनी कम...

Saving Data To A CSV file for Ease of Access

In [ ]:
```
# Save the DataFrame to a CSV file
tweets_df.to_csv('stemming_lemmatization_comparison.csv', index=False)
```

Conclusion and Observations

Introduction

In this analysis, we explored various methods for stemming and lemmatization on a dataset of tweets related to the hashtag #NeetUG24Controversy. The goal was to understand how different preprocessing techniques affect the text data and to compare the outputs generated by each method.

Stemming and Lemmatization Methods

1. **Porter Stemmer**: A classic and widely-used stemming algorithm that reduces words to their root forms.
2. **Snowball Stemmer**: An improvement over the Porter Stemmer, known for being more aggressive in its approach.
3. **Lancaster Stemmer**: The most aggressive stemmer among the three, often resulting in shorter stems.
4. **WordNet Lemmatizer**: Uses a dictionary-based approach to reduce words to their base or dictionary form.

5. **Spacy Lemmatizer**: Utilizes Spacy's natural language processing capabilities to lemmatize words based on context.

Observations

Porter Stemmer

- **Output**: "Standing up for neet 2024 aspir 💯. Head to delhi with my team to fight for justic in the suprem court."
- **Comments**: The Porter Stemmer effectively reduced words to their root forms, but it sometimes produced non-intuitive stems (e.g., "justic" instead of "justice").

Snowball Stemmer

- **Output**: "Stand up for neet 2024 aspir 💯. Head to delhi with my team to fight for justic in the suprem court."
- **Comments**: The Snowball Stemmer produced results similar to the Porter Stemmer but was slightly more aggressive in certain cases.

Lancaster Stemmer

- **Output**: "Stand up for neet 2024 aspir 💯. Head to delhi with my team to fight for justic in the suprem court."
- **Comments**: The Lancaster Stemmer was the most aggressive, often resulting in very short and sometimes confusing stems (e.g., "suprem" instead of "supreme").

WordNet Lemmatizer

- **Output**: "Standing up for neet 2024 aspirant 💯. Heading to delhi with my team to fight for justice in the supreme court."
- **Comments**: The WordNet Lemmatizer provided more contextually accurate results, preserving the meaning of the original words better than the stemmers.

Spacy Lemmatizer

- **Output**: "Standing up for neet 2024 aspirant 💯. Heading to delhi with my team to fight for justice in the supreme court."
- **Comments**: Similar to the WordNet Lemmatizer, the Spacy Lemmatizer produced contextually accurate and readable results.

Conclusion

- **Stemmers** (Porter, Snowball, Lancaster) are useful for reducing words to their root forms, which can be beneficial for certain text processing tasks where the exact meaning of words is not as important.
- **Lemmatizers** (WordNet, Spacy) are more sophisticated and provide contextually accurate base forms of words, preserving their meanings and making the text more readable and understandable.
- The choice between stemming and lemmatization should be based on the specific requirements of the text processing task. For tasks requiring more accurate and readable text, lemmatization is preferred. For simpler, more computationally efficient tasks, stemming may suffice.