```
import pandas as pd
import numpy as np
import re
import nltk
import matplotlib.pyplot as plt

from nltk.tokenize import word_tokenize
from nltk import pos_tag
from collections import defaultdict, Counter
```

```
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('averaged_perceptron_tagger_eng')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
True
```

```
df = pd.read_csv('/content/Twitter_Data.csv')
df.head()
```

	clean_text	category
0	when modi promised “minimum government maximum...	-1.0
1	talk all the nonsense and continue all the dra...	0.0
2	what did just say vote for modi welcome bjp t...	1.0
3	asking his supporters prefix chowkidar their n...	1.0
4	answer who among these the most powerful world...	1.0

```
df.columns
```

```
Index(['clean_text', 'category'], dtype='object')
```

```
def clean_tweet(text):
    text = str(text).lower()
    text = re.sub(r'http\S+|www\S+', '', text)
    text = re.sub(r'@\w+', '', text)
    text = re.sub(r'#', '', text)
    text = re.sub(r'[^a-z\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text

clean_tweets = df['clean_text'].dropna().apply(clean_tweet).tolist()
clean_tweets[:5]
```

```
['when modi promised minimum government maximum governance expected him begin the difficult job reforming the state why
does take years get justice state should and not business and should exit psus and temples',
 'talk all the nonsense and continue all the drama will vote for modi',
 'what did just say vote for modi welcome bjp told you rahul the main campaigner for modi think modi should just relax',
 'asking his supporters prefix chowkidar their names modi did great service now there confusion what read what not now
crustal clear what will crass filthy nonsensical see how most abuses are coming from chowkidars',
 'answer who among these the most powerful world leader today trump putin modi may']
```

```
nltk.download('punkt_tab')
tokenized_tweets = [word_tokenize(tweet) for tweet in clean_tweets if tweet != '']
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

```
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag

# Download required NLTK resources
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
# Example cleaned tweets (replace with your own cleaned list if available)
clean_tweets = [
    "i love this movie",
    "this product is amazing",
    "bad service not recommended"
]

# Tokenization
tokenized_tweets = [word_tokenize(tweet) for tweet in clean_tweets if tweet.strip() != ""]

# POS tagging
pos_tagged_tweets = [pos_tag(tokens) for tokens in tokenized_tweets if len(tokens) > 0]

# Example output
pos_tagged_tweets[0]
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[('i', 'RB'), ('love', 'VBP'), ('this', 'DT'), ('movie', 'NN')]
```

```
transition_counts = defaultdict(Counter)

for sent in pos_tagged_tweets:
    for i in range(len(sent) - 1):
        tag1 = sent[i][1]
        tag2 = sent[i+1][1]
        transition_counts[tag1][tag2] += 1

transition_probs = {}
for tag1 in transition_counts:
    total = sum(transition_counts[tag1].values())
    transition_probs[tag1] = {tag2: c/total for tag2, c in transition_counts[tag1].items()}
```

```
emission_counts = defaultdict(Counter)

for sent in pos_tagged_tweets:
    for word, tag in sent:
        emission_counts[tag][word] += 1

emission_probs = {}
for tag in emission_counts:
    total = sum(emission_counts[tag].values())
    emission_probs[tag] = {word: c/total for word, c in emission_counts[tag].items()}
```

```
list(transition_probs.items())[:3]
```

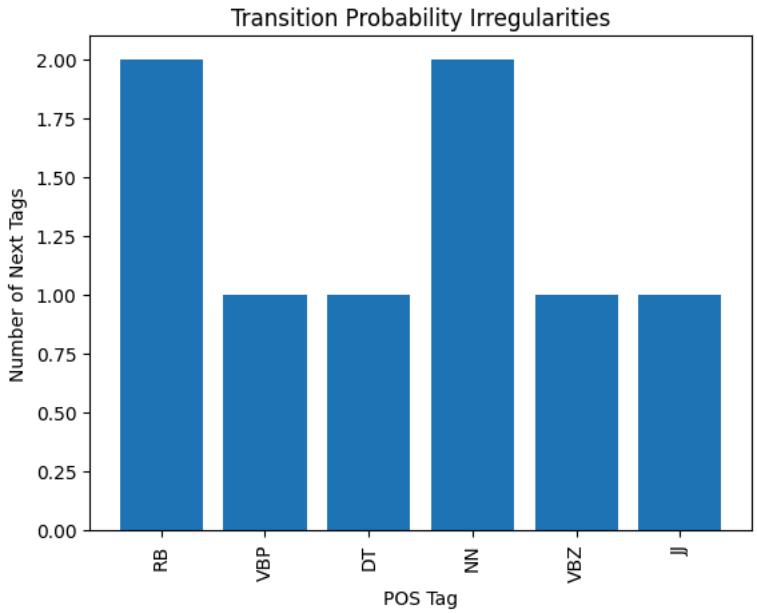
```
[('RB', {'VBP': 0.5, 'VBD': 0.5}), ('VBP', {'DT': 1.0}), ('DT', {'NN': 1.0})]
```

```
list(emission_probs.items())[:3]
```

```
[('RB', {'i': 0.5, 'not': 0.5}), ('VBP', {'love': 1.0}), ('DT', {'this': 1.0})]
```

```
transition_size = {tag: len(next_tags) for tag, next_tags in transition_probs.items()}

plt.figure()
plt.bar(transition_size.keys(), transition_size.values())
plt.xticks(rotation=90)
plt.title("Transition Probability Irregularities")
plt.xlabel("POS Tag")
plt.ylabel("Number of Next Tags")
plt.show()
```




```
word_freq = Counter()

for sent in tokenized_tweets:
    word_freq.update(sent)

rare_words = [word for word, count in word_freq.items() if count == 1]

print("Number of rare words:", len(rare_words))
```

```
Number of rare words: 10
```

```
test_tweet = "i love this movie"
tokens = word_tokenize(test_tweet)
```

```
states = list(emission_probs.keys())
viterbi = [{}]

# Initialization
for state in states:
    viterbi[0][state] = emission_probs[state].get(tokens[0], 1e-6)

# Recursion
for t in range(1, len(tokens)):
    viterbi.append({})
    for curr_state in states:
        emission = emission_probs[curr_state].get(tokens[t], 1e-6)
        viterbi[t][curr_state] = max(
            viterbi[t-1][prev_state] *
            transition_probs.get(prev_state, {}).get(curr_state, 1e-6) *
            emission
            for prev_state in states
        )

# Best path
best_tags = [max(step, key=step.get) for step in viterbi]
list(zip(tokens, best_tags))
```

```
[('i', 'RB'), ('love', 'VBP'), ('this', 'DT'), ('movie', 'NN')]
```