```
import pandas as pd

data = pd.read_csv("Twitter_Data.csv")
tweets = data['clean_text']
```

```
import re

def clean_tweet(text):
    text = str(text) # Ensure text is a string
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"@\w+", "", text)
    return text.lower()

# Fill NaN values with an empty string and then apply the cleaning function
cleaned_tweets = tweets.fillna('').apply(clean_tweet)
```

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
import nltk
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
True
```

```
import nltk

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

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

```
import pandas as pd

# Load dataset
df = pd.read_csv("Twitter_Data.csv")

# View first few rows
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

```
# Extract tweet text
tweets = df['clean_text'].dropna().tolist()
print("Total tweets:", len(tweets))
```

```
Total tweets: 162976
```

```
import re

def clean_tweet(text):
    text = re.sub(r"http\S+|www\S+", "", text)   # Remove URLs
    text = re.sub(r"@\w+", "", text)             # Remove mentions
    text = re.sub(r"#", "", text)                # Remove hashtag symbol
    text = text.lower()
    return text.strip()
```

```
import re
```

```
cleaned_tweets = [clean_tweet(tweet) for tweet in tweets]
cleaned_tweets[:3]
```

```
['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']
```

```
tokenized_tweets = [nltk.word_tokenize(tweet) for tweet in cleaned_tweets]
```

```
import nltk
nltk.download('averaged_perceptron_tagger_eng')

# POS tagging using NLTK
tagged_tweets = [nltk.pos_tag(tokens) for tokens in tokenized_tweets]
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
```

```
# Sample tagged tweet
tagged_tweets[0]
```

```
[('when', 'WRB'),
 ('modi', 'NN'),
 ('promised', 'VBD'),
 ('“', 'NNP'),
 ('minimum', 'JJ'),
 ('government', 'NN'),
 ('maximum', 'JJ'),
 ('governance', 'NN'),
 ('”', 'NNP'),
 ('expected', 'VBD'),
 ('him', 'PRP'),
 ('begin', 'VB'),
 ('the', 'DT'),
 ('difficult', 'JJ'),
 ('job', 'NN'),
 ('reforming', 'VBG'),
 ('the', 'DT'),
 ('state', 'NN'),
 ('why', 'WRB'),
 ('does', 'VBZ'),
 ('take', 'VB'),
 ('years', 'NNS'),
 ('get', 'VB'),
 ('justice', 'NN'),
 ('state', 'NN'),
 ('should', 'MD'),
 ('and', 'CC'),
 ('not', 'RB'),
 ('business', 'NN'),
 ('and', 'CC'),
 ('should', 'MD'),
 ('exit', 'VB'),
 ('psus', 'NN'),
 ('and', 'CC'),
 ('temples', 'NNS')]
```

```
from collections import defaultdict, Counter

transition_counts = defaultdict(Counter)
tag_counts = Counter()

for tweet in tagged_tweets:
    prev_tag = "<START>"
    tag_counts[prev_tag] += 1

    for _, tag in tweet:
        transition_counts[prev_tag][tag] += 1
        tag_counts[tag] += 1
        prev_tag = tag

    transition_counts[prev_tag]["<END>"] += 1
```

```
# Convert to probabilities
transition_probs = defaultdict(dict)

for prev_tag in transition_counts:
    total = sum(transition_counts[prev_tag].values())
    for tag in transition_counts[prev_tag]:
        transition_probs[prev_tag][tag] = transition_counts[prev_tag][tag] / total
```

```
emission_counts = defaultdict(Counter)

for tweet in tagged_tweets:
    for word, tag in tweet:
        emission_counts[tag][word] += 1
```

```
emission_probs = defaultdict(dict)

for tag in emission_counts:
    total = sum(emission_counts[tag].values())
    for word in emission_counts[tag]:
        emission_probs[tag][word] = emission_counts[tag][word] / total
```

```
# Show transitions from START
transition_probs["<START>"]
```

```
{'WRB': 0.030845032397408221,
 'NN': 0.3353683978009032,
 'WP': 0.017517916748478305,
 'VBG': 0.01973910268996662,
 'JJ': 0.1307738562733163,
 'DT': 0.0734341252699784,
 'IN': 0.03366139799725113,
 'NNS': 0.06966669939132142,
 'CD': 0.01413705085411349,
 'VBD': 0.01265216964461025,
 'VBZ': 0.008387738071863342,
 'RB': 0.09840099155703907,
 'VB': 0.04271181032790104,
 'PRP$': 0.015591252699784017,
 'JJR': 0.0007424406047516199,
 'MD': 0.018591694482623208,
 'PRP': 0.02864225407421952,
 'VBN': 0.01220425093265266,
 'VBP': 0.006651286078931867,
 'CC': 0.01645027488709994,
 'EX': 0.0025341154525819753,
 'RBR': 0.001607598664834086,
 'JJS': 0.0034422246220302375,
 'UH': 0.0005767720400549774,
 'WDT': 0.002877724327508345,
 'RBS': 0.0006749460043196544,
 'PDT': 0.001902120557628117,
 'FW': 9.20380914981347e-05,
 'WP$': 8.590221873159239e-05,
 'NNP': 3.067936383271157e-05,
 '<END>': 6.135872766542313e-06}
```

```
word_frequency = Counter()

for tweet in tokenized_tweets:
    for word in tweet:
        word_frequency[word] += 1
```

```
# Rare words (occur only once)
rare_words = [word for word, freq in word_frequency.items() if freq == 1]
len(rare_words)
```

```
68064
```

```
test_tweet = "love this movie so much"
tokens = nltk.word_tokenize(test_tweet)
```

```
tokens
```

```
['love', 'this', 'movie', 'so', 'much']
```

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

# Initialization
for state in states:
    V[0][state] = transition_probs["<START>"].get(state, 1e-6) * \
                  emission_probs[state].get(tokens[0], 1e-6)
    path[state] = [state]
```

```
# Recursion
for t in range(1, len(tokens)):
    V.append({})
    new_path = {}

    for curr_state in states:
        prob_state = []
        for prev_state in states:
            prob = V[t-1][prev_state] * \
                   transition_probs[prev_state].get(curr_state, 1e-6) * \
                   emission_probs[curr_state].get(tokens[t], 1e-6)
            prob_state.append((prob, prev_state))

        best_prob, best_prev = max(prob_state)
        V[t][curr_state] = best_prob
        new_path[curr_state] = path[best_prev] + [curr_state]

    path = new_path
```

```
# Termination
final_state = max(V[-1], key=V[-1].get)
best_path = path[final_state]

list(zip(tokens, best_path))
```

```
[('love', 'VB'), ('this', 'DT'), ('movie', 'NN'), ('so', 'NN'), ('much', 'RB')]
```

```
Start coding or generate with AI.
```