```
import pandas as pd
import re
import nltk
from collections import defaultdict, Counter

# Download required NLTK models
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng') # Added this line

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

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

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

# Use only text column
tweets = df['clean_text'].dropna().astype(str)

print("Total Tweets:", len(tweets))
tweets.head()
```

```
Total Tweets: 162976
```

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

dtype: object

```
def preprocess_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"#\w+", "", text)            # Remove hashtags
    text = re.sub(r"[^a-zA-Z\s]", "", text)     # Keep only letters
    text = text.lower().strip()
    return text

clean_tweets = [preprocess_tweet(t) for t in tweets]
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']
```

```
tagged_tweets = []

for tweet in clean_tweets[:1000]:   # limit for faster processing
    tokens = word_tokenize(tweet)
    tagged = pos_tag(tokens)
    tagged_tweets.append(tagged)

tagged_tweets[0]
```

```
[('when', 'WRB'),
 ('modi', 'NN'),
 ('promised', 'VBD'),
 ('minimum', 'JJ'),
 ('government', 'NN'),
 ('maximum', 'JJ'),
 ('governance', 'NN'),
 ('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')]
```

```
transition_counts = defaultdict(Counter)
emission_counts = defaultdict(Counter)
tag_counts = Counter()

for tweet in tagged_tweets:
    prev_tag = "<START>"

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

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

```
transition_probs = {}
emission_probs = {}

for prev_tag, counter in transition_counts.items():
    total = sum(counter.values())
    transition_probs[prev_tag] = {
        tag: count/total for tag, count in counter.items()
    }

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

```
print("Transition Probabilities from NN:")
print(dict(list(transition_probs.get('NN', {}).items())[:10]))
```

```
Transition Probabilities from NN:
{'VBD': 0.05016025641025641, 'JJ': 0.04567307692307692, 'VBG': 0.02467948717948718, 'WRB': 0.009294871794871795, 'NN': 0.333
```

```
print("\nEmission Probabilities for VB:")
print(dict(list(emission_probs.get('VB', {}).items())[:10]))
```

```

Emission Probabilities for VB:
{'begin': 0.002372479240806643, 'take': 0.027283511269276393, 'get': 0.03558718861209965, 'exit': 0.002372479240806643, 'cor
```

```
word_freq = Counter()

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

rare_words = [w for w, c in word_freq.items() if c == 1]

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

```
Number of rare words: 3030
```

```
test_word = "lmaooo"

for tag in emission_probs:
    if test_word in emission_probs[tag]:
        print("Seen!")
```

```
test_tweet = clean_tweets[0]
tokens = word_tokenize(test_tweet)

tags = list(tag_counts.keys())

V = [{}]
path = {}

# Initialization
for tag in tags:
    trans_prob = transition_probs["<START>"].get(tag, 1e-6)
    emis_prob = emission_probs[tag].get(tokens[0], 1e-6)
    V[0][tag] = trans_prob * emis_prob
    path[tag] = [tag]

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

    for curr_tag in tags:
        (prob, prev_tag) = max(
            (V[t-1][pt] *
             transition_probs.get(pt, {}).get(curr_tag, 1e-6) *
             emission_probs[curr_tag].get(tokens[t], 1e-6), pt)
            for pt in tags
        )

        V[t][curr_tag] = prob
        new_path[curr_tag] = path[prev_tag] + [curr_tag]

    path = new_path

# Termination
(prob, final_tag) = max((V[-1][tag], tag) for tag in tags)

print("Tweet:", tokens)
print("Predicted Tags:", path[final_tag])
```

```
Tweet: ['when', 'modi', 'promised', 'minimum', 'government', 'maximum', 'governance', 'expected', 'him', 'begin', 'the', 'di
Predicted Tags: ['WRB', 'NN', 'VBD', 'JJ', 'NN', 'JJ', 'NN', 'VBD', 'PRP', 'VB', 'DT', 'JJ', 'NN', 'VBG', 'DT', 'NN', 'WRB',
```

```
"soooo happpyyyy rn!!!"
```

```
'soooo happpyyyy rn!!!'
```