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

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

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

data.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 |

```
def clean_tweet(text):
    text = re.sub(r"http\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) # remove special characters
    text = text.lower()
    return text.strip()
```

```
# Convert 'clean_text' column to string type and fill NaN values before cleaning
cleaned_tweets = [clean_tweet(str(t)) for t in data['clean_text'].fillna('')]
cleaned_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_sentences = []

nltk.download('punkt_tab') # Download the missing resource
nltk.download('averaged_perceptron_tagger_eng') # Download the missing resource for POS tagging

for tweet in cleaned_tweets[:200]:  # limit for experiment
    tokens = nltk.word_tokenize(tweet)
    if tokens:
        tagged = nltk.pos_tag(tokens)
        tagged_sentences.append(tagged)

tagged_sentences[:3]
```

```
  ('the', 'DT'),
  ('difficult', 'JJ'),
  ('job', 'NN'),
  ('reforming', 'VBG'),
```

```
 ('exit', 'VB'),
 ('psus', 'NN'),
 ('and', 'CC'),
 ('temples', 'NNS')],
[('talk', 'NN'),
 ('all', 'PDT'),
 ('the', 'DT'),
 ('nonsense', 'NN'),
 ('and', 'CC'),
 ('continue', 'VB'),
 ('all', 'PDT'),
 ('the', 'DT'),
 ('drama', 'NN'),
 ('will', 'MD'),
 ('vote', 'VB'),
 ('for', 'IN'),
 ('modi', 'NN')],
[('what', 'WP'),
 ('did', 'VBD'),
 ('just', 'RB'),
 ('say', 'VB'),
 ('vote', 'NN'),
 ('for', 'IN'),
 ('modi', 'JJ'),
 ('welcome', 'JJ'),
 ('bjp', 'NN'),
 ('told', 'VBD'),
 ('you', 'PRP'),
 ('rahul', 'VBP'),
 ('the', 'DT'),
 ('main', 'JJ'),
 ('campaigner', 'NN'),
 ('for', 'IN'),
 ('modi', 'NN'),
 ('think', 'VBP'),
 ('modi', 'NN'),
 ('should', 'MD'),
 ('just', 'RB'),
 ('relax', 'VB')]]
```

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

for sentence in tagged_sentences:
    prev_tag = "<START>"
    for word, tag in sentence:
        transition_counts[prev_tag][tag] += 1
```

```
        tag_counts[prev_tag] += 1
        prev_tag = tag

transition_probs = {}

for prev_tag, tags in transition_counts.items():
    transition_probs[prev_tag] = {}
    for tag, count in tags.items():
        transition_probs[prev_tag][tag] = count / tag_counts[prev_tag]

transition_probs["<START>"]
```

```
{'WRB': 0.04522613065326633,
 'NN': 0.38190954773869346,
 'WP': 0.01507537688442211,
 'VBG': 0.035175879396984924,
 'JJ': 0.10552763819095477,
 'DT': 0.06532663316582915,
 'IN': 0.06030150753768844,
 'NNS': 0.05025125628140704,
 'CD': 0.010050251256281407,
 'VBD': 0.01507537688442211,
 'VBZ': 0.010050251256281407,
 'RB': 0.10050251256281408,
 'VB': 0.020100502512562814,
 'PRP$': 0.01507537688442211,
 'JJR': 0.005025125628140704,
 'MD': 0.01507537688442211,
 'PRP': 0.010050251256281407,
 'VBN': 0.02512562814070352,
 'VBP': 0.010050251256281407,
 'CC': 0.005025125628140704}
```

```
emission_counts = defaultdict(Counter)
tag_totals = Counter()

for sentence in tagged_sentences:
    for word, tag in sentence:
        emission_counts[tag][word] += 1
        tag_totals[tag] += 1

emission_probs = {}

for tag, words in emission_counts.items():
```

```
        emission_probs[tag] = {}
        for word, count in words.items():
            emission_probs[tag][word] = count / tag_totals[tag]

    list(emission_probs.keys())[:5]
```

```
['WRB', 'NN', 'VBD', 'JJ', 'PRP']
```

```
print("Sample Transition Probabilities:")
for k in list(transition_probs.keys())[:5]:
    print(k, transition_probs[k])

print("\nSample Emission Probabilities:")
for k in list(emission_probs.keys())[:5]:
    print(k, list(emission_probs[k].items())[:5])
```

```
Sample Transition Probabilities:
<START> {'WRB': 0.04522613065326633, 'NN': 0.38190954773869346, 'WP': 0.01507537688442211, 'VBG': 0.035175879396984924, 'JJ':
WRB {'NN': 0.21875, 'VBZ': 0.03125, 'JJS': 0.03125, 'JJ': 0.15625, 'VBN': 0.0625, 'VBP': 0.09375, 'PRP': 0.0625, 'DT': 0.09375
NN {'VBD': 0.053621825023518345, 'JJ': 0.06302916274694262, 'VBG': 0.030103480714957668, 'WRB': 0.00940733772342427, 'NN': 0.3
VBD {'JJ': 0.20353982300884957, 'PRP': 0.04424778761061947, 'RB': 0.07079646017699115, 'WP': 0.02654867256637168, 'NN': 0.1858
JJ {'NN': 0.5871559633027523, 'JJ': 0.10550458715596331, 'WP': 0.009174311926605505, 'VB': 0.011467889908256881, 'NNS': 0.1605

Sample Emission Probabilities:
WRB [('when', 0.25), ('why', 0.40625), ('how', 0.15625), ('write', 0.03125), ('where', 0.15625)]
NN [('modi', 0.10379746835443038), ('government', 0.00590717299578059), ('governance', 0.002531645569620253), ('job', 0.004219
VBD [('promised', 0.008771929824561403), ('expected', 0.008771929824561403), ('did', 0.07017543859649122), ('told', 0.00877192
JJ [('minimum', 0.0022624434389140274), ('maximum', 0.0022624434389140274), ('difficult', 0.0022624434389140274), ('modi', 0.0
PRP [('him', 0.11842105263157894), ('you', 0.5), ('yogi', 0.013157894736842105), ('they', 0.21052631578947367), ('them', 0.065
```

```
word_freq = Counter()

for sentence in tagged_sentences:
    for word, tag in sentence:
        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))
print("Examples:", rare_words[:10])
```

```
Number of rare words: 1062
Examples: ['minimum', 'maximum', 'expected', 'begin', 'difficult', 'reforming', 'temples', 'nonsense', 'continue', 'drama']
```

```
test_tweet = "i love twitter"
tokens = nltk.word_tokenize(test_tweet)
tokens
```

```
['i', 'love', 'twitter']
```

```
tags = list(emission_probs.keys())

def viterbi(tokens):
    V = [{}]
    path = {}

    # Initialization
    for tag in tags:
        trans_p = transition_probs.get("<START>", {}).get(tag, 1e-6)
        emis_p = emission_probs.get(tag, {}).get(tokens[0], 1e-6)
        V[0][tag] = math.log(trans_p) + math.log(emis_p)
        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][prev_tag] +
                 math.log(transition_probs.get(prev_tag, {}).get(curr_tag, 1e-6)) +
                 math.log(emission_probs.get(curr_tag, {}).get(tokens[t], 1e-6)),
                 prev_tag)
                for prev_tag in tags
            )
            V[t][curr_tag] = prob
            new_path[curr_tag] = path[prev_tag] + [curr_tag]

        path = new_path

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

```
        return path[tag]

print("Tokens:", tokens)
print("Predicted POS tags:", viterbi(tokens))
```

```
Tokens: ['i', 'love', 'twitter']
Predicted POS tags: ['NN', 'NN', 'NN']
```