step1:Import Libraries and Download NLTK Resources

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

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

step 2: Load Twitter Dataset

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

# Keep only tweet text
tweets = df['clean_text'].dropna().head(500)  # limit for lab
```

Step 3: Preprocess Tweets (Remove URLs, Mentions, Special Characters)

```
def preprocess_tweet(tweet):
    tweet = re.sub(r"http\S+", "", tweet)       # remove URLs
    tweet = re.sub(r"@\w+", "", tweet)           # remove mentions
    tweet = re.sub(r"#\w+", "", tweet)           # remove hashtags
    tweet = re.sub(r"[^a-zA-Z\s]", "", tweet)    # remove emojis/symbols
    tweet = tweet.lower().strip()
    return tweet

clean_tweets = tweets.apply(preprocess_tweet)
```

Step 4: Tokenize and POS-Tag Tweets Using NLTK

```
tagged_sentences = []

for tweet in clean_tweets:
    tokens = nltk.word_tokenize(tweet)
    if tokens:
        tagged = nltk.pos_tag(tokens)
        tagged_sentences.append(tagged)

# Example output
print(tagged_sentences[0])
```

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

Step 5: Build HMM Parameters

5.1 Transition Probabilities

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

for sentence in tagged_sentences:
    prev_tag = "<START>"
```

```
        tag_counts[prev_tag] += 1

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

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

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

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

5.2 Emission Probabilities

```
emission_counts = defaultdict(Counter)

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

```
emission_probs = defaultdict(dict)

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

Step 6: Display HMM Parameter Snapshots

```
print("Transition Probabilities (sample):")
for prev_tag in list(transition_probs.keys())[:3]:
    print(prev_tag, "→", transition_probs[prev_tag])

print("\nEmission Probabilities (sample):")
for tag in list(emission_probs.keys())[:3]:
    print(tag, "→", list(emission_probs[tag].items())[:5])
```

```
Transition Probabilities (sample):
<START> → {'WRB': 0.034, 'NN': 0.378, 'WP': 0.008, 'VBG': 0.02, 'JJ': 0.114, 'DT': 0.062, 'IN': 0.06, 'NNS': 0.066, 'CD':
WRB → {'NN': 0.23711340206185566, 'VBZ': 0.010309278350515464, 'JJS': 0.010309278350515464, 'JJ': 0.20618556701030927, 'V
NN → {'VBD': 0.05525040387722133, 'JJ': 0.04975767366720517, 'VBG': 0.02294022617124394, 'WRB': 0.010339256865912763, 'NN

Emission Probabilities (sample):
WRB → [('when', 0.2268041237113402), ('why', 0.38144329896907214), ('how', 0.24742268041237114), ('write', 0.010309278350
NN → [('modi', 0.09402261712439418), ('government', 0.006138933764135703), ('governance', 0.0012924071082390954), ('job',
VBD → [('promised', 0.003236245954692557), ('expected', 0.006472491909385114), ('did', 0.05177993527508091), ('told', 0.0(
```

Step 7: Manual Viterbi Decoding (Single Tweet

Example Tweet

```
test_tweet = "this movie lit"
words = test_tweet.split()
```

Possible Tags (from training data)

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

Viterbi Initialization

```
viterbi = [{}]
backpointer = [{}]

for tag in possible_tags:
    trans_p = transition_probs["<START>"].get(tag, 1e-6)
    emis_p = emission_probs[tag].get(words[0], 1e-6)
    viterbi[0][tag] = trans_p * emis_p
    backpointer[0][tag] = None
```

Viterbi Recursion

```
for t in range(1, len(words)):
    viterbi.append({})
    backpointer.append({})

    for curr_tag in possible_tags:
        max_prob = 0
        best_prev = None

        for prev_tag in possible_tags:
            prob = (
                viterbi[t-1][prev_tag]
                * transition_probs[prev_tag].get(curr_tag, 1e-6)
                * emission_probs[curr_tag].get(words[t], 1e-6)
            )

            if prob > max_prob:
                max_prob = prob
                best_prev = prev_tag

        viterbi[t][curr_tag] = max_prob
        backpointer[t][curr_tag] = best_prev
```

Step 8: Backtracking Best POS Tag Sequence

```
# Find best last tag
last_tag = max(viterbi[-1], key=viterbi[-1].get)

best_path = [last_tag]

for t in range(len(words)-1, 0, -1):
    last_tag = backpointer[t][last_tag]
    best_path.insert(0, last_tag)

print("Tweet:", test_tweet)
print("Predicted POS Tags:", best_path)
```

```
Tweet: this movie lit
Predicted POS Tags: ['DT', 'NN', 'NN']
```