Setup & Imports

Donikela Laxman(2403A51283)

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

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
```

Load Twitter Dataset

```
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

Text Preprocessing

```
def preprocess_tweet(text):
    text = str(text)
    text = re.sub(r"http\S+", "", text)      # remove URLs
    text = re.sub(r"@\w+", "", text)          # remove mentions
    text = re.sub(r"\s+", " ", text).strip()
    return text

df["processed_text"] = df["clean_text"].apply(preprocess_tweet)
df[["clean_text", "processed_text"]].head()
```

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

Tokenization & POS Tagging (Weak Supervision)

```
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
def pos_tag_tweet(text):
    tokens = nltk.word_tokenize(text)
    return nltk.pos_tag(tokens)
```

```
df["pos_tags"] = df["processed_text"].apply(pos_tag_tweet)
df["pos_tags"].head()
```

```
[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]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
```

	pos_tags
0	[(when, WRB), (modi, NN), (promised, VBD), (“,...
1	[(talk, NN), (all, PDT), (the, DT), (nonsense,...
2	[(what, WP), (did, VBD), (just, RB), (say, VB)...
3	[(asking, VBG), (his, PRP$), (supporters, NNS)...
4	[(answer, NN), (who, WP), (among, IN), (these,...

dtype: object

Build HMM Parameters

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

for tags in df["pos_tags"]:
    prev_tag = "<START>"
    for word, tag in tags:
        transition_counts[prev_tag][tag] += 1
        tag_counts[prev_tag] += 1
        prev_tag = tag

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

transition_probs["<START>"]
```

```
{'WRB': 0.030844464624276747,
 'NN': 0.3354113106596555,
 'WP': 0.017517594291289063,
 'VBG': 0.01973873934678701,
 'JJ': 0.13078985636186258,
 'DT': 0.07340209474840317,
 'IN': 0.03366077838249099,
 'NNS': 0.0696654170169163,
 'CD': 0.0141367906294676,
 'VBD': 0.012651936752587757,
 'VBZ': 0.00838758367642457,
 'RB': 0.09840531602230962,
 'VB': 0.04271102411967186,
 'PRP$': 0.015590965707238356,
 'JJR': 0.0007424269384399217,
 'MD': 0.018572944980641676,
 'PRP': 0.028641726848244252,
 'VBN': 0.012204026285595077,
 'VBP': 0.006651163646850208,
 'CC': 0.01644997208229281,
 'EX': 0.002534068806410642,
 'RBR': 0.0016075690733161941,
 'JJS': 0.003442161260039637,
 'UH': 0.0005767614232508483,
 'WDT': 0.002877671356432424,
 'RBS': 0.0006749335803999288,
 'PDT': 0.0019020855447634357,
 'FW': 9.203639732726302e-05,
 'WP$': 8.590063750544549e-05,
 'NNP': 3.067879910908767e-05}
```

```
emission_counts = defaultdict(Counter)
tag_word_counts = Counter()

for tags in df["pos_tags"]:
    for word, tag in tags:
        emission_counts[tag][word.lower()] += 1
        tag_word_counts[tag] += 1

emission_probs = {
    tag: {word: count / tag_word_counts[tag]
```

```
            for word, count in words.items()}
        for tag, words in emission_counts.items()
    }

    list(emission_probs.items())[:1]
```

```
[('WRB',
  {'when': 0.262271086100901,
   'why': 0.3527957583840224,
   'how': 0.25147187977751034,
   'where': 0.1224669030348372,
   'whenever': 0.005269492242136421,
   'walo': 6.505545977946199e-05,
   'mover': 3.2527729889730993e-05,
   'wont': 0.0008131932472432749,
   'whereever': 6.505545977946199e-05,
   'wow': 0.00013011091955892397,
   'waiver': 0.00019516637933838596,
   'wasn': 3.2527729889730993e-05,
   'wld': 3.2527729889730993e-05,
   'wiselythe': 3.2527729889730993e-05,
   'wherever': 0.00035780502878704094,
   'write': 9.758318966919298e-05,
   'whatsapp': 3.2527729889730993e-05,
   'wasnt': 0.000292749569007579,
   'wan': 0.00026022183911784795,
   'won': 6.505545977946199e-05,
   'withot': 3.2527729889730993e-05,
   'wil': 0.00016263864944865498,
   'wait': 6.505545977946199e-05,
   'wali': 3.2527729889730993e-05,
   'whatsoever': 6.505545977946199e-05,
   'wud': 0.00013011091955892397,
   'wicket': 3.2527729889730993e-05,
   'wada': 3.2527729889730993e-05,
   'whos': 9.758318966919298e-05,
   'jaiwere': 3.2527729889730993e-05,
   'mere': 0.00026022183911784795,
   'whichever': 6.505545977946199e-05,
   'wht': 3.2527729889730993e-05,
   'kongujratwhere': 3.2527729889730993e-05,
   'wouldn': 3.2527729889730993e-05,
   'workersthere': 3.2527729889730993e-05,
   'win': 0.00035780502878704094,
   'modi': 3.2527729889730993e-05,
   'watever': 9.758318966919298e-05,
   'wrk': 6.505545977946199e-05,
   'wah': 6.505545977946199e-05,
   'whatbieber': 3.2527729889730993e-05,
   'moreover': 3.2527729889730993e-05,
   'whereby': 6.505545977946199e-05,
   'wild': 9.758318966919298e-05,
   'wrongthen': 3.2527729889730993e-05,
   'waste': 3.2527729889730993e-05,
   'wohi': 3.2527729889730993e-05,
   'withoue': 3.2527729889730993e-05,
   'wer': 3.2527729889730993e-05,
   'wala': 3.2527729889730993e-05,
   'wrz✅promised✅jobs': 3.2527729889730993e-05,
   'wadhai': 3.2527729889730993e-05,
   'warna': 6.505545977946199e-05,
   'wot': 3.2527729889730993e-05,
   'manthere': 3.2527729889730993e-05,
   'wasa': 3.2527729889730993e-05,
```

Analyze Rare & Unknown Tokens

```
word_freq = Counter()

for tags in df["pos_tags"]:
    for word, tag in tags:
        word_freq[word.lower()] += 1

rare_words = [w for w, c in word_freq.items() if c == 1]
len(rare_words), rare_words[:10]
```

```
(68079,
 ['crustal',
  'maarkefir',
  'tax…the',
  'constituency2',
  'tuthukudi',
  'likly',
  'thuthukudi',
  'leadershipwho',
```

```
    'thiugh',
    'modiganga'])
```

Manual Viterbi Example (Conceptual)

```
sample_tweet = "I love NLP"
tokens = nltk.word_tokenize(sample_tweet)
nltk.pos_tag(tokens)
```

```
[('I', 'PRP'), ('love', 'VBP'), ('NLP', 'RB')]
```

Discussion – Why HMM Fails on Twitter

discussion = """ HMM-based POS tagging struggles on social media text because tweets are short, informal, and noisy. The Markov assumption is violated due to irregular grammar, slang, emojis, and abbreviations. Many words are rare or unseen, leading to zero or near-zero emission probabilities, causing incorrect Viterbi decoding paths. """

print(discussion)