

step 1: import libraries

```
import nltk
nltk.download('punkt_tab')
import pandas as pd
import spacy
import re
from collections import Counter, defaultdict
import matplotlib.pyplot as plt
import seaborn as sns

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

step 2: load dataset

```
# Load Tweet text and pre-process /content/Twitter_Data.csv
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 |

step 3: POS tags tweets using NLTK of the tweets

```
nltk.download('averaged_perceptron_tagger')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('universal_tagset')

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

```
print(df.columns)
```

```
Index(['clean_text', 'category'], dtype='object')
```

step 4: preprocessing the tweets

```
def preprocess_tweets(text):
    if not isinstance(text, str):
        return ""
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'@[A-Za-z0-9]+', '', text)
    text = re.sub(r'#', '', text)
    text = re.sub(r'^\w\s', '', text)
    # Remove whitespaces as well
    text = re.sub(r'\s+', ' ', text).strip()
    return text

df['preprocess_text'] = df['clean_text'].apply(preprocess_tweets)
df.head(10)
```

| | clean_text | category | preprocess_text |
|---|---|----------|---|
| 0 | when modi promised "minimum government maximum... | -1.0 | when modi promised minimum government maximum ... |
| 1 | talk all the nonsense and continue all the dra... | 0.0 | talk all the nonsense and continue all the dra... |
| 2 | what did just say vote for modi welcome bjp t... | 1.0 | what did just say vote for modi welcome bjp to... |
| 3 | asking his supporters prefix chowkidar their n... | 1.0 | asking his supporters prefix chowkidar their n... |
| 4 | answer who among these the most powerful world... | 1.0 | answer who among these the most powerful world... |
| 5 | kiya tho refresh maarkefir comment karo | 0.0 | kiya tho refresh maarkefir comment karo |
| 6 | surat women perform yagna seeks divine grace f... | 0.0 | surat women perform yagna seeks divine grace f... |
| 7 | this comes from cabinet which has scholars lik... | 0.0 | this comes from cabinet which has scholars lik... |
| 8 | with upcoming election india saga going import... | 1.0 | with upcoming election india saga going import... |
| 9 | gandhi was gay does modi | 1.0 | gandhi was gay does modi |

step 5:POS tagging using nltk

```
def pos_tag_tweet(text):
    tokens = nltk.word_tokenize(text.lower())

    pos_tags = nltk.pos_tag(tokens, tagset='universal')
    return pos_tags

df['pos_tags'] = df['preprocess_text'].apply(pos_tag_tweet)
df.head(10)
```

| | clean_text | category | preprocess_text | pos_tags |
|---|---|----------|---|---|
| 0 | when modi promised "minimum government maximum... | -1.0 | when modi promised minimum government maximum ... | [(when, ADV), (modi, NOUN), (promised, VERB), ... |
| 1 | talk all the nonsense and continue all the dra... | 0.0 | talk all the nonsense and continue all the dra... | [(talk, NOUN), (all, DET), (the, DET), (nonsen... |
| 2 | what did just say vote for modi welcome bjp t... | 1.0 | what did just say vote for modi welcome bjp to... | [(what, PRON), (did, VERB), (just, ADV), (say,... |
| 3 | asking his supporters prefix chowkidar their n... | 1.0 | asking his supporters prefix chowkidar their n... | [(asking, VERB), (his, PRON), (supporters, NOU... |
| 4 | answer who among these the most powerful world... | 1.0 | answer who among these the most powerful world... | [(answer, NOUN), (who, PRON), (among, ADP), (t... |
| 5 | kiya tho refresh maarkefir comment karo | 0.0 | kiya tho refresh maarkefir comment karo | [(kiya, NOUN), (tho, NOUN), (refresh, ADJ), (m... |
| 6 | surat women perform yagna seeks divine grace f... | 0.0 | surat women perform yagna seeks divine grace f... | [(surat, ADJ), (women, NOUN), (perform, VERB),... |
| 7 | this comes from cabinet which has scholars lik... | 0.0 | this comes from cabinet which has scholars lik... | [(this, DET), (comes, VERB), (from, ADP), (cab... |
| 8 | with upcoming election india saga going import... | 1.0 | with upcoming election india saga going import... | [(with, ADP), (upcoming, ADJ), (election, NOUN... |
| 9 | gandhi was gay does modi | 1.0 | gandhi was gay does modi | [(gandhi, NOUN), (was, VERB), (gay, NOUN), (do... |

step 6: simple Hmm

```
class SimpleHMM:
    """
    Simple HMM for POS tagging with parameter extraction.
    """
    def __init__(self):
        self.transition_counts = defaultdict(lambda: defaultdict(int))
        self.emission_counts = defaultdict(lambda: defaultdict(int))
        self.tag_counts = defaultdict(int)
        self.vocabulary = set()
        self.tagset = set()

    def train(self, tagged_sentences):
        """
        Train HMM from tagged sentences.
        """
        for sentence in tagged_sentences:
            if len(sentence) == 0:
```

```

        continue

    # Add start state
    prev_tag = '<START>'
    self.tag_counts[prev_tag] += 1

    for word, tag in sentence:
        # Emission counts: P(word|tag)
        self.emission_counts[tag][word] += 1
        self.tag_counts[tag] += 1
        self.vocabulary.add(word)
        self.tagset.add(tag)

        # Transition counts: P(tag|prev_tag)
        self.transition_counts[prev_tag][tag] += 1
        prev_tag = tag

    # Add end state
    self.transition_counts[prev_tag]['<END>'] += 1

def get_transition_prob(self, prev_tag, tag, smoothing=1e-6):
    """
    Calculate transition probability with Laplace smoothing.
    """
    count = self.transition_counts[prev_tag][tag]
    total = sum(self.transition_counts[prev_tag].values())
    vocab_size = len(self.tagset) + 1 # +1 for <END>
    return (count + smoothing) / (total + smoothing * vocab_size)

def get_emission_prob(self, tag, word, smoothing=1e-6):
    """
    Calculate emission probability with Laplace smoothing.
    """
    count = self.emission_counts[tag][word]
    total = self.tag_counts[tag]
    vocab_size = len(self.vocabulary)
    return (count + smoothing) / (total + smoothing * vocab_size)

# Train HMM
hmm = SimpleHMM()
hmm.train(df['pos_tags'].tolist())

print("HMM Training Complete!")
print(f"\nVocabulary size: {len(hmm.vocabulary)}")
print(f"Tagset size: {len(hmm.tagset)}")
print(f"Tags: {sorted(hmm.tagset)}")

```

HMM Training Complete!

Vocabulary size: 107055

Tagset size: 12

Tags: ['.', 'ADJ', 'ADP', 'ADV', 'CONJ', 'DET', 'NOUN', 'NUM', 'PRON', 'PRT', 'VERB', 'X']

step 7: Emission Probability Snapshots

```

print("\n Emission Probability Examples")
print("="*60)

sample_tags = list(hmm.tagset)[:5] # First 5 tags

for tag in sample_tags:
    # Get top 10 most likely words for this tag
    words_probs = [(word, hmm.get_emission_prob(tag, word))
                    for word in list(hmm.emission_counts[tag].keys())[:20]]
    words_probs.sort(key=lambda x: x[1], reverse=True)

    print(f"\n{tag}:")
    for word, prob in words_probs[:10]:
        print(f" {word:20s} → {prob:.6f}")

```

Emission Probability Examples

=====

```

.:
modi          → 0.398743
modis         → 0.041428
india         → 0.036249
ndtv          → 0.025892
till          → 0.020714
yes           → 0.015535

```

| | |
|-----------|------------|
| baar | → 0.010357 |
| modiin | → 0.010357 |
| kiya | → 0.005178 |
| vikas | → 0.005178 |
| VERB: | |
| will | → 0.041686 |
| should | → 0.010409 |
| did | → 0.010109 |
| get | → 0.007220 |
| say | → 0.006028 |
| take | → 0.005720 |
| think | → 0.005308 |
| vote | → 0.004674 |
| does | → 0.004115 |
| rahul | → 0.002000 |
| ADP: | |
| for | → 0.250188 |
| with | → 0.100317 |
| that | → 0.084863 |
| from | → 0.078367 |
| like | → 0.057043 |
| about | → 0.045517 |
| against | → 0.032225 |
| after | → 0.031409 |
| than | → 0.025333 |
| because | → 0.024379 |
| ADV: | |
| not | → 0.130469 |
| now | → 0.049626 |
| why | → 0.046662 |
| just | → 0.035301 |
| when | → 0.034690 |
| how | → 0.033257 |
| again | → 0.024714 |
| modi | → 0.015198 |
| more | → 0.014695 |
| well | → 0.013435 |
| ADJ: | |
| modi | → 0.045280 |
| narendra | → 0.013841 |
| great | → 0.008997 |
| important | → 0.002990 |

Start coding or [generate](#) with AI.