```
!pip install nltk spacy pandas numpy
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: spacy in /usr/local/lib/python3.12/dist-packages (3.8.11)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.15)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.13)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (8.3.10)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.5.2)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.4.3)
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.21.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.32.4)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.12.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (25.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.3)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (2.41.4)
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (4.15.0)
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (0.4.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2026.1.4)
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (1.3.3)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (0.1.5)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2->spacy) (0.23.0)
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2->spacy) (7.5.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->spacy) (3.0.3)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.4.2->spacy) (2.0.1)
```

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

```
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
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 averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
True
```

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

```
tweets = data['clean_text'].dropna().astype(str).tolist()
```

```
def clean_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)       # remove special chars
    return text.lower().strip()
```

```
cleaned_tweets = [clean_tweet(t) for t in tweets]
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']
```

```
tokenized_tweets = [nltk.word_tokenize(t) for t in cleaned_tweets]
```

```
pos_tagged_tweets = [nltk.pos_tag(tokens) for tokens in tokenized_tweets if len(tokens) > 0]
```

```
pos_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)

for sentence in pos_tagged_tweets:
    tags = [tag for (_, tag) in sentence]
    for i in range(len(tags) - 1):
        transition_counts[tags[i]][tags[i+1]] += 1
```

```
transition_probs = {}
for tag in transition_counts:
    total = sum(transition_counts[tag].values())
    transition_probs[tag] = {t: c/total for t, c in transition_counts[tag].items()}
```

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

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

```
emission_probs = {}
for tag in emission_counts:
    emission_probs[tag] = {w: c/tag_counts[tag] for w, c in emission_counts[tag].items()}
```

```
print("Sample Transition Probabilities:")
list(transition_probs.items())[:3]
```

```
Sample Transition Probabilities:
[('WRB',
  {'NN': 0.2648639790232711,
   'VBZ': 0.028515240904621434,
   'JJS': 0.0008194034742707309,
   'JJ': 0.1857423795476893,
   'VBN': 0.016191412651589642,
   'VBP': 0.05109800065552278,
   'PRP': 0.08066207800721074,
   'DT': 0.0704031465093412,
   'RB': 0.0471976401179941,
   'VB': 0.013044903310390037,
   'NNS': 0.0450671910848902,
   'MD': 0.06542117338577516,
   'PRP$': 0.0189446083251393,
   'EX': 0.006063585709603409,
   'VBD': 0.06640445755490003,
   'VBG': 0.016519174041297935,
   'IN': 0.008849557522123894,
   'CC': 0.003736479842674533,
   'CD': 0.0025893149786955097,
   'WDT': 0.00013110455588331694,
   'WP': 0.0009832841691248771,
   'PDT': 0.0022287774500163882,
   'WRB': 0.0021304490331039002,
   'JJR': 0.0004260898066207801,
   'RBS': 0.0008521796132415602,
   'RP': 0.00039331366764995085,
   'TO': 0.00013110455588331694,
   'RBR': 0.0004260898066207801,
   'WP$': 3.2776138970829235e-05,
   'FW': 9.832841691248771e-05,
   'NNP': 3.2776138970829235e-05}),
 ('NN',
  {'VBD': 0.06108914362899798,
   'JJ': 0.05632319596985964,
   'VBG': 0.027592209868567183,
   'WRB': 0.00918011493747442,
   'NN': 0.3647342423248173,
   'MD': 0.03085858090441383,
   'CC': 0.053401840307492135,
   'PDT': 0.0008320959007438126,
   'IN': 0.095076540420355,
   'VBP': 0.025586385197261977,
   'RB': 0.054571059073171446,
   'WP': 0.009952453273530669,
   'WDT': 0.009606310398831006,
   'VBZ': 0.046752965610085274,
   'DT': 0.029981384954849213,
   'NNS': 0.06058176807976395,
   'VB': 0.012670858716204561,
   'FW': 0.0027138954377918115,
   'PRP': 0.019885739026312497,
   'PRP$': 0.007484353101812233,
   'CD': 0.0023350550276970677,
   'VBN': 0.010409091267841298,
   'RP': 0.0006246356761681194,
   'RBR': 0.001752136896681908,
```

```python
print("Sample Emission Probabilities:")
list(emission_probs.items())[:3]
```

```
Sample Emission Probabilities:
[('WRB',
  {'when': 0.26217155477605636,
   'why': 0.35282081262788656,
   'how': 0.25122608723894896,
   'write': 0.000129916528630355,
   'where': 0.1226736821592127,
   'whenever': 0.005294098541686966,
   'walo': 6.49582643151775e-05,
   'mover': 3.247913215758875e-05,
   'wont': 0.0008119783039397188,
   'whereever': 6.49582643151775e-05,
   'wow': 0.000129916528630355,
   'wada': 6.49582643151775e-05,
   'waiver': 0.0001948747929455325,
   'sincere': 3.247913215758875e-05,
   'wld': 3.247913215758875e-05,
   'wasn': 3.247913215758875e-05,
   'wherever': 0.0003572704537334762,
   'whatsapp': 3.247913215758875e-05,
   'wasnt': 0.00029231218941829873,
   'wan': 0.00025983305726071,
   'won': 3.247913215758875e-05,
   'withot': 3.247913215758875e-05,
   'wil': 0.00016239566078794375,
   'wait': 6.49582643151775e-05,
   'wali': 3.247913215758875e-05,
   'whatsoever': 6.49582643151775e-05,
   'wah': 9.743739647276625e-05,
   'wud': 0.00016239566078794375,
   'wicket': 3.247913215758875e-05,
   'whos': 9.743739647276625e-05,
   'jaiwere': 3.247913215758875e-05,
   'mere': 0.00029231218941829873,
   'whichever': 6.49582643151775e-05,
   'wht': 3.247913215758875e-05,
   'workersthere': 3.247913215758875e-05,
   'modi': 3.247913215758875e-05,
   'whereby': 9.743739647276625e-05,
   'win': 0.0004547078502062425,
   'watever': 9.743739647276625e-05,
   'wrk': 6.49582643151775e-05,
   'whatbieber': 3.247913215758875e-05,
   'withdrew': 3.247913215758875e-05,
   'moreover': 3.247913215758875e-05,
   'wild': 9.743739647276625e-05,
   'wrongthen': 3.247913215758875e-05,
   'waste': 3.247913215758875e-05,
   'wohi': 3.247913215758875e-05,
   'withoue': 3.247913215758875e-05,
   'wer': 3.247913215758875e-05,
   'wala': 3.247913215758875e-05,
   'wrzpromisedjobs': 3.247913215758875e-05,
   'wadhai': 3.247913215758875e-05,
    'warna': 6.49582643151775e-05,
    'wot': 3.247913215758875e-05,
    'manthere': 3.247913215758875e-05,
    'wasa': 3.247913215758875e-05,
```

```
word_freq = Counter()
for sentence in tokenized_tweets:
    word_freq.update(sentence)
```

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

```
62172
```

```
test_tweet = "flight delayed again"
tokens = nltk.word_tokenize(test_tweet)
tokens
```

```
['flight', 'delayed', 'again']
```

```
states = list(tag_counts.keys())
V = [{}]

# Initialization
for state in states:
    emission = emission_probs.get(state, {}).get(tokens[0], 1e-6)
    V[0][state] = emission
```

```
# Recursion
for t in range(1, len(tokens)):
    V.append({})
    for curr_state in states:
        max_prob = 0
        for prev_state in states:
            trans = transition_probs.get(prev_state, {}).get(curr_state, 1e-6)
            emit = emission_probs.get(curr_state, {}).get(tokens[t], 1e-6)
            prob = V[t-1][prev_state] * trans * emit
            max_prob = max(max_prob, prob)
        V[t][curr_state] = max_prob
```

```
# Best final tag
best_tags = [max(V[t], key=V[t].get) for t in range(len(tokens))]
list(zip(tokens, best_tags))
```

```
[('flight', 'NN'), ('delayed', 'VBD'), ('again', 'RB')]
```

Start coding or generate with AI.