2403A52006

MD.Mustafa

Import Required Libraries

```
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
```

Loading the Twitter Dataset

```
df = pd.read_csv("Twitter_Data.csv")
df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

Select Tweet Text Column

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

Preprocess Tweets (Noise Removal)

```
def clean_tweet(text):
    text = re.sub(r"http\S+|www\S+", "", text)
    text = re.sub(r"@\w+", "", text)
    text = re.sub(r"#\w+", "", text)
    text = re.sub(r"[^a-zA-Z\s]", "", text)
    return text.lower().strip()

cleaned_tweets = [clean_tweet(tweet) for tweet 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']
```

POS Tagging Using NLTK (Weak Supervision)

```
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
tagged_sentences = []

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

tagged_sentences[:2]
```

```
[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.

[[('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')],
 [('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')]]
```

Build HMM Transition Probabilities

```python
print("Sample HMM Transition Probabilities:\n")

for tag in list(transition_counts.keys())[:3]:   # show only 3 tags
    print(f"Transitions from '{tag}':")
    for next_tag, count in transition_counts[tag].most_common(3):
```

```
        print(f"  {tag} → {next_tag} : {count}")
    print()
```

```
Sample HMM Transition Probabilities:

Transitions from 'WRB':
  WRB → NN : 7
  WRB → JJ : 5
  WRB → VBP : 3

Transitions from 'NN':
  NN → NN : 395
  NN → IN : 113
  NN → NNS : 69

Transitions from 'VBD':
  VBD → JJ : 23
  VBD → NN : 21
  VBD → DT : 11
```

Short Emission Probability Snapshot

```
print("Sample HMM Emission Probabilities:\n")

for tag in list(emission_counts.keys())[:3]:   # show only 3 POS tags
    print(f"Words emitted by '{tag}':")
    for word, count in emission_counts[tag].most_common(3):
        print(f"  {tag} → '{word}' : {count}")
    print()
```

```
Sample HMM Emission Probabilities:

Words emitted by 'WRB':
  WRB → 'why' : 13
  WRB → 'when' : 8
  WRB → 'how' : 5

Words emitted by 'NN':
  NN → 'modi' : 123
  NN → 'vote' : 27
  NN → 'india' : 27

Words emitted by 'VBD':
  VBD → 'was' : 17
  VBD → 'did' : 8
  VBD → 'were' : 7
```

Observing Irregularities in HMM Parameters

```
# Show transitions from a sample tag
sample_tag = list(transition_counts.keys())[0]
transition_counts[sample_tag]
```

```
Counter({'NN': 7,
         'VBZ': 1,
         'JJS': 1,
         'JJ': 5,
         'VBN': 2,
         'VBP': 3,
         'PRP': 2,
         'DT': 3,
         'RB': 3,
         'VB': 2,
         'NNS': 1,
         'MD': 1,
         'PRP$': 1})
```

Rare and Unknown Token Analysis

```
word_freq = Counter()

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

# Rare words (frequency = 1)
rare_words = [word for word, freq in word_freq.items() if freq == 1]
rare_words[:10]
```

```
['minimum',
 'maximum',
 'expected',
 'begin',
 'difficult',
 'reforming',
 'temples',
 'nonsense',
 'continue',
 'drama']
```

```
from collections import Counter

print("========== FINAL OUTPUT ==========\n")

print("1. Sample HMM Transition Probabilities:\n")

sample_transition_tag = list(transition_counts.keys())[0]
print(f"Transitions from tag '{sample_transition_tag}':")
```

```
for next_tag, count in
transition_counts[sample_transition_tag].most_common(5):
    print(f"  {sample_transition_tag} -> {next_tag} : {count}")

print("\n" + "-"*50 + "\n")

print("2. Sample HMM Emission Probabilities:\n")

sample_emission_tag = list(emission_counts.keys())[0]
print(f"Words emitted by tag '{sample_emission_tag}':")
for word, count in
emission_counts[sample_emission_tag].most_common(5):
    print(f"  {sample_emission_tag} -> '{word}' : {count}")

print("\n" + "-"*50 + "\n")

print("3. Rare Tokens (Frequency = 1):\n")

word_freq = Counter()
for sentence in tagged_sentences:
    for word, tag in sentence:
        word_freq[word] += 1

rare_words = [word for word, freq in word_freq.items() if freq == 1]

print("Sample rare words:")
print(rare_words[:10])

print("\nTotal number of rare words:", len(rare_words))

print("\n" + "-"*50 + "\n")

print("4. Viterbi Decoding Example (Using NLTK POS Tagger):\n")

example_tweet = cleaned_tweets[0]
tokens = nltk.word_tokenize(example_tweet)
tagged_output = nltk.pos_tag(tokens)

print("Tweet:")
print(example_tweet)
print("\nPOS Tags:")
print(tagged_output)

print("\n========== END OF FINAL OUTPUT ==========")
```

```
========== FINAL OUTPUT ==========

1. Sample HMM Transition Probabilities:

Transitions from tag 'WRB':
  WRB -> NN : 7
```

```
  WRB -> JJ : 5
  WRB -> VBP : 3
  WRB -> DT : 3
  WRB -> RB : 3

--------------------------------------------------

2. Sample HMM Emission Probabilities:

Words emitted by tag 'WRB':
  WRB -> 'why' : 13
  WRB -> 'when' : 8
  WRB -> 'how' : 5
  WRB -> 'where' : 5
  WRB -> 'write' : 1

--------------------------------------------------

3. Rare Tokens (Frequency = 1):

Sample rare words:
['minimum', 'maximum', 'expected', 'begin', 'difficult', 'reforming',
'temples', 'nonsense', 'continue', 'drama']

Total number of rare words: 1065

--------------------------------------------------

4. Viterbi Decoding Example (Using NLTK POS Tagger):

Tweet:
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

POS Tags:
[('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')]

========== END OF FINAL OUTPUT ==========
```

Discussion

This experiment demonstrates the limitations of HMM-based POS tagging when applied to noisy Twitter text. Social media language contains informal grammar, spelling variations, abbreviations, and rare words, which violate the core assumptions of Hidden Markov Models. As a result, transition and emission probabilities become sparse, leading to incorrect POS tag sequences during Viterbi decoding. This highlights why traditional HMM approaches are less effective for real-world noisy text and motivates the use of modern neural NLP models.