Name	**Shreya Rathod**
UID no.	**2022300087**

Experiment 4	
AIM :	Calculate emission and transition matrix which will be helpful for tagging Parts of Speech using Hidden Markov Model. Find POS tag of given sentence using HMM. Input: Text Corpus of sufficient length. For example movie reviews, newspaper articles, etc. Output: For the given corpus 1) Fill and display the emission and transition matrix 2) Find the POS tags for a given sentence.
THEORY:	The Hidden Markov Model (HMM) is a statistical model used for sequence prediction, particularly useful in natural language processing tasks such as Part-of-Speech (POS) tagging. It is based on the Markov process, where future states depend only on the current state and not on past states. **Components of HMM** HMM consists of: 1. States: Hidden variables representing different categories (e.g., POS tags like Noun, Verb, Adjective). 2. Observations: The observed sequence of words in the text. ● Transition Probabilities: The probability of transitioning from one hidden state to another. ● Emission Probabilities: The probability of a particular observation (word) being generated by a hidden state (POS tag). ● Initial Probabilities: The probability of starting in a particular state.

Working Principle

HMM operates on the assumption that the sequence of observed words is generated by an underlying sequence of hidden states. The model uses probabilities to determine the most likely sequence of hidden states (POS tags) given an observed sequence of words.

Applications

HMM is widely used in various fields, including:

- Natural Language Processing (NLP): POS tagging, speech recognition, and text analysis.
- Bioinformatics: Gene prediction and sequence alignment.
- Pattern Recognition: Handwriting and gesture recognition.

1. List a few ways for tagging parts of speech?

There are several methods for POS tagging:

- Rule-Based Tagging: Uses a set of predefined linguistic rules.
- Statistical Tagging (e.g., HMM-based): Uses probability models like Hidden Markov Models (HMMs) to determine the most likely tag sequence.
- Machine Learning-Based Tagging: Uses models like Conditional Random Fields (CRF), Decision Trees, and Neural Networks.
- Hybrid Tagging: Combines rule-based and statistical/machine learning approaches for improved accuracy.

2. How do you find the most probable sequence of POS tags from a sequence of text?

The HMM Algorithm, it finds the most probable sequence of tags for a given sentence by:

- Computing probabilities of tag transitions (transition matrix).
- Computing probabilities of words given a specific tag (emission matrix).
- Using dynamic programming to find the most likely sequence based on observed words.

In the provided code, the greedy_pos_tagging() function applies a simpler greedy algorithm, selecting the most probable tag for each word independently, based on emission and transition probabilities.

3. Differentiate between Markov chain and Markov model?

Markov Chain

- A stochastic process where the next state depends only on the current state.
- All states are observable (e.g., predicting weather conditions: sunny → rainy → cloudy).
- Used in applications like weather forecasting and simple text prediction.

Markov Model (e.g., Hidden Markov Model - HMM)

- A probabilistic model where states are hidden, and only observations (outputs) are visible.
- The goal is to infer the hidden states based on observed data (e.g., words in a sentence → POS tags).
- Used in applications like POS tagging, speech recognition, and bioinformatics.

4. How you can identify whether a system follows a Markov Process?

A system follows a Markov Process if:

- Memoryless Property: The future state depends only on the present state and not on past states.
- Transition Probabilities: Probabilities of moving from one state to another are well-defined.
- Stationarity: Probabilities remain consistent over time.
  In the given code, the transition matrix is an example of a Markov Process, where the probability of moving from one POS tag to another is based only on the current state.

5. Explain the use of Markov Chains in text generation algorithms

Markov Chains are widely used in text generation by modeling word sequences based on transition probabilities. Examples include:

- N-gram Models: Predict the next word based on the previous n words.
- Chatbots: Generate responses using probability-based word transitions.
- Poetry and Story Generation: Generate coherent sequences by predicting the next probable word.

<table>
<tr><td></td><td>

- Markov Chain Monte Carlo (MCMC): Used in probabilistic text modeling.

</td></tr>
<tr><td>CODE:</td><td>

```
    import nltk
from collections import defaultdict
import pandas as pd
import matplotlib.pyplot as plt
from colorama import Fore, Style, init

init(autoreset=True)

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

corpus = """
The artist can draw, but he can also sing.
She will watch a play and then play the piano.
The wind howled as they tried to wind the clock.
He used a bat to hit the ball, but a bat flew by.
The dove dove into the water gracefully.
She wore a bow on her dress and took a bow on stage.
"""

tokens = nltk.word_tokenize(corpus)
tokens = [word for word in tokens if word.isalpha()]
pos_tags = nltk.pos_tag(tokens)

# Step 1: Count transitions and emissions
transition_counts = defaultdict(lambda: defaultdict(int))
emission_counts = defaultdict(lambda: defaultdict(int))
tag_counts = defaultdict(int)

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

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

</td></tr>
</table>

```python
# Convert counts to probabilities
transition_probs = defaultdict(lambda: defaultdict(float))
emission_probs = defaultdict(lambda: defaultdict(float))

for prev_tag, next_tags in transition_counts.items():
    total_prev_tag_count = sum(next_tags.values())
    for next_tag, count in next_tags.items():
        transition_probs[prev_tag][next_tag] = count /
total_prev_tag_count

for tag, words in emission_counts.items():
    total_tag_count = tag_counts[tag]
    for word, count in words.items():
        emission_probs[tag][word] = count / total_tag_count

# Convert to DataFrame for better readability
transition_df = pd.DataFrame(transition_probs).fillna(0)
emission_df = pd.DataFrame(emission_probs).fillna(0)

# Step 2: Display transition and emission matrices using
matplotlib
print(Fore.CYAN + "\nTransition Matrix (Probabilities of tag
transitions):")
print(transition_df)

print(Fore.GREEN + "\nEmission Matrix (Probabilities of word
emission from tag):")
print(emission_df)

def greedy_pos_tagging(sentence):
    tagged_sentence = []
    prev_tag = "<START>"

    print(Fore.YELLOW + "\nGreedy POS Tagging Process:")
    results = []

    for word in sentence:
        max_prob = -1e6
```

```python
        best_tag = None

        for tag in tag_counts.keys():
            emission_prob = emission_probs[tag].get(word, 0)
            transition_prob =
transition_probs[prev_tag].get(tag, 0)
            prob = emission_prob * transition_prob

            results.append([word, tag,
f"{emission_prob:.4f}", f"{transition_prob:.4f}",
f"{prob:.6f}"])

            if prob > max_prob:
                max_prob = prob
                best_tag = tag

        tagged_sentence.append((word, best_tag))
        prev_tag = best_tag

    tagged_sentence.append(("<END>", "<END>"))

    # Display tagging process
    df_results = pd.DataFrame(results, columns=["Word",
"Tag", "Emission P", "Transition P", "Combined P"])
    pd.set_option('display.max_rows', None)
    print(df_results)

    return tagged_sentence

# Test sentence
test_sentence = "The dove dove into the water
gracefully".split()
tagged_sentence = greedy_pos_tagging(test_sentence)

# Output the final tagged sentence
print(Fore.MAGENTA + "\nFinal Tagged Sentence:")
print(pd.DataFrame(tagged_sentence, columns=["Word", "Tag"]))
```

OUTPUT:	

```
(venv) PS C:\Users\HP\Desktop\sem 6\nlp\exp 4> python test2.py
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!

Transition Matrix (Probabilities of tag transitions):
       <START>   DT      NN        MD   VB    CC  PRP        RB  VBG  VBD   IN   TO  PRP$
DT         1.0  0.0  0.0625  0.000000  0.8  0.25  0.0  0.000000  0.0  0.6  0.4  0.0   0.0
NN         0.0  1.0  0.1250  0.000000  0.0  0.00  0.0  0.000000  0.0  0.0  0.2  0.0   1.0
MD         0.0  0.0  0.0625  0.000000  0.0  0.00  0.4  0.000000  0.0  0.0  0.0  0.0   0.0
CC         0.0  0.0  0.1875  0.000000  0.2  0.00  0.0  0.000000  0.0  0.0  0.0  0.0   0.0
VBD        0.0  0.0  0.0625  0.000000  0.0  0.25  0.6  0.000000  0.0  0.0  0.0  0.0   0.0
PRP        0.0  0.0  0.0625  0.000000  0.0  0.25  0.0  0.333333  1.0  0.0  0.2  0.0   0.0
TO         0.0  0.0  0.0625  0.000000  0.0  0.00  0.0  0.000000  0.0  0.2  0.0  0.0   0.0
IN         0.0  0.0  0.2500  0.000000  0.0  0.00  0.0  0.000000  0.0  0.2  0.0  0.0   0.0
RB         0.0  0.0  0.0625  0.333333  0.0  0.25  0.0  0.000000  0.0  0.0  0.0  0.0   0.0
<END>      0.0  0.0  0.0625  0.000000  0.0  0.00  0.0  0.000000  0.0  0.0  0.0  0.0   0.0
VB         0.0  0.0  0.0000  0.666667  0.0  0.00  0.0  0.333333  0.0  0.0  0.0  1.0   0.0
VBG        0.0  0.0  0.0000  0.000000  0.0  0.00  0.0  0.333333  0.0  0.0  0.0  0.0   0.0
PRP$       0.0  0.0  0.0000  0.000000  0.0  0.00  0.0  0.000000  0.0  0.0  0.2  0.0   0.0
```

Emission Matrix (Probabilities of word emission from tag):

	DT	NN	MD	VB	CC	PRP	RB	VBG	VBD	IN	TO	PRP$
The	0.250000	0.0000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
a	0.416667	0.0000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
the	0.333333	0.0000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
artist	0.000000	0.0625	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
play	0.000000	0.0625	0.000000	0.2	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
piano	0.000000	0.0625	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
wind	0.000000	0.0625	0.000000	0.2	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
clock	0.000000	0.0625	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
bat	0.000000	0.1250	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
ball	0.000000	0.0625	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
flew	0.000000	0.0625	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
dove	0.000000	0.1250	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
water	0.000000	0.0625	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
bow	0.000000	0.1250	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
dress	0.000000	0.0625	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
stage	0.000000	0.0625	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
can	0.000000	0.0000	0.666667	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
will	0.000000	0.0000	0.333333	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
draw	0.000000	0.0000	0.000000	0.2	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
watch	0.000000	0.0000	0.000000	0.2	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
hit	0.000000	0.0000	0.000000	0.2	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
but	0.000000	0.0000	0.000000	0.0	0.5	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
and	0.000000	0.0000	0.000000	0.0	0.5	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
he	0.000000	0.0000	0.000000	0.0	0.0	0.2	0.000000	0.0	0.0	0.0	0.0	0.0
She	0.000000	0.0000	0.000000	0.0	0.0	0.4	0.000000	0.0	0.0	0.0	0.0	0.0
they	0.000000	0.0000	0.000000	0.0	0.0	0.2	0.000000	0.0	0.0	0.0	0.0	0.0
He	0.000000	0.0000	0.000000	0.0	0.0	0.2	0.000000	0.0	0.0	0.0	0.0	0.0
also	0.000000	0.0000	0.000000	0.0	0.0	0.0	0.333333	0.0	0.0	0.0	0.0	0.0
then	0.000000	0.0000	0.000000	0.0	0.0	0.0	0.333333	0.0	0.0	0.0	0.0	0.0
gracefully	0.000000	0.0000	0.000000	0.0	0.0	0.0	0.333333	0.0	0.0	0.0	0.0	0.0
sing	0.000000	0.0000	0.000000	0.0	0.0	0.0	0.000000	1.0	0.0	0.0	0.0	0.0
howled	0.000000	0.0000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.2	0.0	0.0	0.0
tried	0.000000	0.0000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.2	0.0	0.0	0.0
used	0.000000	0.0000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.2	0.0	0.0	0.0

```
howled    0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.2  0.0  0.0   0.0
tried     0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.2  0.0  0.0   0.0
used      0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.2  0.0  0.0   0.0
wore      0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.2  0.0  0.0   0.0
took      0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.2  0.0  0.0   0.0
as        0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.0  0.2  0.0   0.0
by        0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.0  0.2  0.0   0.0
into      0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.0  0.2  0.0   0.0
on        0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.0  0.4  0.0   0.0
to        0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.0  0.0  1.0   0.0
her       0.000000  0.0000  0.000000  0.0  0.0  0.0  0.000000  0.0  0.0  0.0  0.0   1.0

Greedy POS Tagging Process:
          Word   Tag Emission P Transition P Combined P
0          The    DT     0.2500       1.0000   0.250000
1          The    NN     0.0000       0.0000   0.000000
2          The    MD     0.0000       0.0000   0.000000
3          The    VB     0.0000       0.0000   0.000000
4          The    CC     0.0000       0.0000   0.000000
5          The   PRP     0.0000       0.0000   0.000000
6          The    RB     0.0000       0.0000   0.000000
7          The   VBG     0.0000       0.0000   0.000000
8          The   VBD     0.0000       0.0000   0.000000
9          The    IN     0.0000       0.0000   0.000000
10         The    TO     0.0000       0.0000   0.000000
11         The  PRP$     0.0000       0.0000   0.000000
12        dove    DT     0.0000       0.0000   0.000000
13        dove    NN     0.1250       1.0000   0.125000
14        dove    MD     0.0000       0.0000   0.000000
15        dove    VB     0.0000       0.0000   0.000000
16        dove    CC     0.0000       0.0000   0.000000
17        dove   PRP     0.0000       0.0000   0.000000
18        dove    RB     0.0000       0.0000   0.000000
19        dove   VBG     0.0000       0.0000   0.000000
20        dove   VBD     0.0000       0.0000   0.000000
21        dove    IN     0.0000       0.0000   0.000000
22        dove    TO     0.0000       0.0000   0.000000
```

Ln 105, Col 1 (3493 selected)   Spaces: 4

22	dove	TO	0.0000	0.0000	0.000000
23	dove	PRP$	0.0000	0.0000	0.000000
24	dove	DT	0.0000	0.0625	0.000000
25	dove	NN	0.1250	0.1250	0.015625
26	dove	MD	0.0000	0.0625	0.000000
27	dove	VB	0.0000	0.0000	0.000000
28	dove	CC	0.0000	0.1875	0.000000
29	dove	PRP	0.0000	0.0625	0.000000
30	dove	RB	0.0000	0.0625	0.000000
31	dove	VBG	0.0000	0.0000	0.000000
32	dove	VBD	0.0000	0.0625	0.000000
33	dove	IN	0.0000	0.2500	0.000000
34	dove	TO	0.0000	0.0625	0.000000
35	dove	PRP$	0.0000	0.0000	0.000000
36	into	DT	0.0000	0.0625	0.000000
37	into	NN	0.0000	0.1250	0.000000
38	into	MD	0.0000	0.0625	0.000000
39	into	VB	0.0000	0.0000	0.000000
40	into	CC	0.0000	0.1875	0.000000
41	into	PRP	0.0000	0.0625	0.000000
42	into	RB	0.0000	0.0625	0.000000
43	into	VBG	0.0000	0.0000	0.000000
44	into	VBD	0.0000	0.0625	0.000000
45	into	IN	0.2000	0.2500	0.050000
46	into	TO	0.0000	0.0625	0.000000
47	into	PRP$	0.0000	0.0000	0.000000
48	the	DT	0.3333	0.4000	0.133333
49	the	NN	0.0000	0.2000	0.000000
50	the	MD	0.0000	0.0000	0.000000
51	the	VB	0.0000	0.0000	0.000000
52	the	CC	0.0000	0.0000	0.000000
53	the	PRP	0.0000	0.2000	0.000000
54	the	RB	0.0000	0.0000	0.000000
55	the	VBG	0.0000	0.0000	0.000000
56	the	VBD	0.0000	0.0000	0.000000
57	the	IN	0.0000	0.0000	0.000000

```
59         the  PRP$    0.0000       0.2000   0.000000
60       water    DT    0.0000       0.0000   0.000000
61       water    NN    0.0625       1.0000   0.062500
62       water    MD    0.0000       0.0000   0.000000
63       water    VB    0.0000       0.0000   0.000000
64       water    CC    0.0000       0.0000   0.000000
65       water   PRP    0.0000       0.0000   0.000000
66       water    RB    0.0000       0.0000   0.000000
67       water   VBG    0.0000       0.0000   0.000000
68       water   VBD    0.0000       0.0000   0.000000
69       water    IN    0.0000       0.0000   0.000000
70       water    TO    0.0000       0.0000   0.000000
71       water  PRP$    0.0000       0.0000   0.000000
72  gracefully    DT    0.0000       0.0625   0.000000
73  gracefully    NN    0.0000       0.1250   0.000000
74  gracefully    MD    0.0000       0.0625   0.000000
75  gracefully    VB    0.0000       0.0000   0.000000
76  gracefully    CC    0.0000       0.1875   0.000000
77  gracefully   PRP    0.0000       0.0625   0.000000
78  gracefully    RB    0.3333       0.0625   0.020833
79  gracefully   VBG    0.0000       0.0000   0.000000
80  gracefully   VBD    0.0000       0.0625   0.000000
81  gracefully    IN    0.0000       0.2500   0.000000
82  gracefully    TO    0.0000       0.0625   0.000000
83  gracefully  PRP$    0.0000       0.0000   0.000000

Final Tagged Sentence:
          Word    Tag
0          The     DT
1         dove     NN
2         dove     NN
3         into     IN
4          the     DT
5        water     NN
6   gracefully     RB
7        <END>  <END>
(venv) PS C:\Users\HP\Desktop\sem 6\nlp\exp 4>
```

OBSERVATION:	1. **Tokenization and POS Tagging:**

- The given corpus is tokenized into words, removing punctuation.
- Each word is tagged with a corresponding Part-of-Speech (POS) using NLTK's `pos_tag()` function.

2. **Transition and Emission Matrices:**
   - A **transition matrix** is created to store probabilities of transitioning from one POS tag to another.
   - An **emission matrix** is generated to store probabilities of words occurring under specific POS tags.
   - These probabilities are computed using frequency counts and normalized.
3. **Greedy POS Tagging Approach:**
   - The tagging function processes a given sentence word by word.
   - It selects the POS tag with the **highest combined probability**.
   - The final output consists of the tagged words and their assigned POS labels.
4. **Performance Considerations:**
   - The model relies on the training corpus, so words not seen in training data may not be tagged accurately.
   - The greedy approach selects the best local option at each step, which may not always lead to the best overall sequence.

CONCLUSION:

The implementation successfully constructs **transition and emission matrices** for POS tagging using an **HMM-based approach**.

The accuracy of POS tagging depends on the size and diversity of the training corpus.

This approach provides a **probabilistic way to determine POS tags** and demonstrates how HMM principles can be applied in natural language processing.