Ex1b: Hidden Markov Model (HMM) based Predictive Text System

Learning Objective:

Toimplementa predictivetextsystemusing Hidden Markov Model (HMM), enabling students to understand contextual probability, POS transitions, and next-word prediction using the Brown Corpus.

Steps:

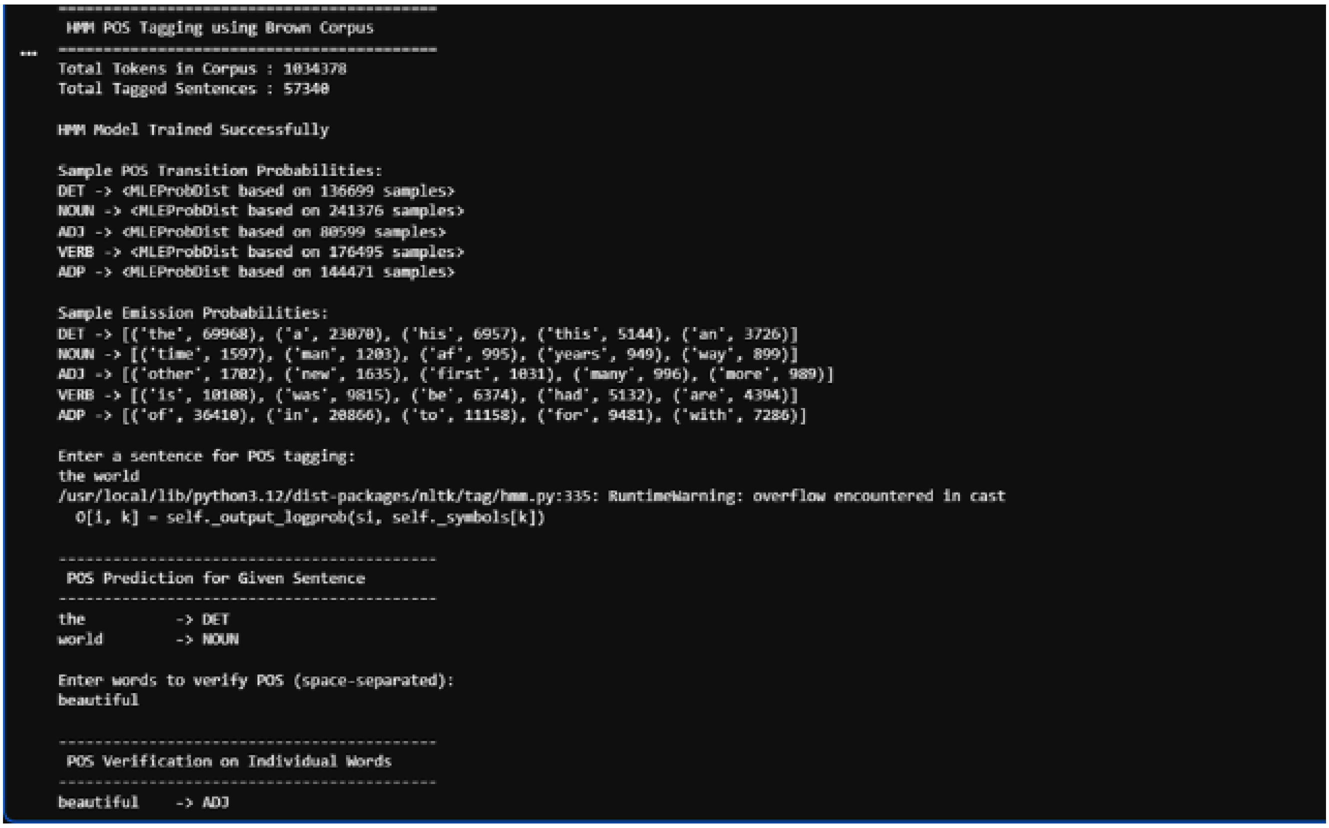
1.Initialize the environment by installing and importing the Natural Language Toolkit (nltk). Download the Brown Corpus (for text data) and the Universal Tagset (for simplified Part-of-Speech tags).
2. Extract raw sentences from the Brown Corpus using brown.sents().
3. Tokenization: Convert all words to lowercase and flatten the nested list into a single list of tokens to verify the total volume of training data.
4. Load the tagged version of the corpus (brown.tagged_sents) which provides the "Hidden States" (POS tags) linked to the "Observations" (words).
5. Train the HMM Model. Use the HiddenMarkovModelTrainer to perform Supervised Learning. Calculate the internal parameters of the HMM: Initial State Probabilities, Transition Probabilities, Emission Probabilities
6. Build POS Transition Probabilities
7. Build Emission Probabilities
8. Define Prediction Function
9. Test the Model
10. Verify POS prediction on certain words.

Program:

```
import nltk
from nltk.corpus import brown
from collections import defaultdict, Counter

nltk.download('brown')
nltk.download('universal_tagset')

tagged_words = brown.tagged_words(tagset='universal')

transition_counts = defaultdict(Counter)
emission_counts = defaultdict(Counter)
tag_counts = Counter()

for i in range(len(tagged_words) - 1):
  word, tag = tagged_words[i]
  next_word, next_tag = tagged_words[i + 1]

  transition_counts[tag][next_tag] += 1
  emission_counts[tag][word.lower()] += 1
  tag_counts[tag] += 1

def predict_next_word_hmm(previous_word):
  previous_word = previous_word.lower()

  possible_tags = [
    tag for tag in emission_counts
    if previous_word in emission_counts[tag]
  ]
```

```
if not possible_tags:
 return "No prediction available"
 current_tag = possible_tags[0]
 next_tag = transition_counts[current_tag].most_common(1)[0][0]
 predicted_word = emission_counts[next_tag].most_common(1)[0][0]
 return predicted_word

word = "the"
prediction = predict_next_word_hmm(word)

print("Previous Word:", word)
print("Predicted Next Word:", prediction)
```

Output:



```
    ==========================================
     HMM POS Tagging using Brown Corpus
…   ==========================================
    Total Tokens in Corpus : 1034378
    Total Tagged Sentences : 57340

    HMM Model Trained Successfully

    Sample POS Transition Probabilities:
    DET -> <MLEProbDist based on 136699 samples>
    NOUN -> <MLEProbDist based on 241376 samples>
    ADJ -> <MLEProbDist based on 80599 samples>
    VERB -> <MLEProbDist based on 176495 samples>
    ADP -> <MLEProbDist based on 144471 samples>

    Sample Emission Probabilities:
    DET -> [('the', 69968), ('a', 23070), ('his', 6957), ('this', 5144), ('an', 3726)]
    NOUN -> [('time', 1597), ('man', 1203), ('af', 995), ('years', 949), ('way', 899)]
    ADJ -> [('other', 1702), ('new', 1635), ('first', 1031), ('many', 996), ('more', 989)]
    VERB -> [('is', 10108), ('was', 9815), ('be', 6374), ('had', 5132), ('are', 4394)]
    ADP -> [('of', 36410), ('in', 20866), ('to', 11158), ('for', 9481), ('with', 7286)]

    Enter a sentence for POS tagging:
    the world
    /usr/local/lib/python3.12/dist-packages/nltk/tag/hmm.py:335: RuntimeWarning: overflow encountered in cast
      O[i, k] = self._output_logprob(si, self._symbols[k])

    ------------------------------------------
     POS Prediction for Given Sentence
    ------------------------------------------
    the          -> DET
    world        -> NOUN

    Enter words to verify POS (space-separated):
    beautiful

    ------------------------------------------
     POS Verification on Individual Words
    ------------------------------------------
    beautiful    -> ADJ
```

Learning Outcome:

Upon completion of this experiment, HMM based model was built to predict the next word with POS-based suggestions, and evaluated the effectiveness of context-driven predictive text generation using the Brown Corpus.