# ELL884 - Deep Learning for Natural Language Processing
# Assignment 1 - Part-Of-Speech Tagging

Shashwat Bhardwaj
Entry Number: 2023AIY7528
Indian Institute of Technology Delhi

February 10, 2024

## 1 Introduction

This document presents the analysis and implementation of Hidden Markov Models (HMM) for Part-Of-Speech Tagging as part of Assignment 1. HMM is a statistical model that represents a system evolving over time with hidden states, observed states, and transition probabilities between states. In the context of Part-Of-Speech Tagging, HMM is employed to model the probability distribution of sequences of tags given the observed tokens in a sentence.

## 2 Abstract

This work introduces an effective approach to Part-Of-Speech (POS) tagging using Hidden Markov Models (HMM). POS tagging is a fundamental step in many Natural Language Processing (NLP) tasks, requiring precise identification of word types within context. HMMs offer a robust framework for addressing this challenge by modeling words and tags as sequences of observations and hidden states, respectively. Through the implementation of the Viterbi algorithm, we achieve accurate sequence prediction by maximizing the probability of tag sequences given sentence structures. This document details the methodology, from data preprocessing and model parameter estimation to the application of dynamic programming for decoding the most likely tag sequences. Our results demonstrate the effectiveness of HMMs in capturing linguistic patterns and significantly improving POS tagging accuracy.

## 3 Detailed Description of HMM

### 3.1 Data Preprocessing

The code begins by importing necessary libraries and reading the training data from the `train.csv` file. It then processes the tagged sentences, creating dictionaries for tokens and tags, and establishes mappings between indices and tokens/tags.

### 3.2 Implementation Details

#### 3.2.1 Data Structures Used

**df: DataFrame**

- **Description:** A pandas DataFrame representing the training data read from the `train.csv` file.

**data: List of Lists**

- **Description:** A list of lists containing the tagged sentences extracted from the DataFrame. Each inner list represents a sentence where each tuple consists of a token and its corresponding tag.

**dict_train_tokens: Dictionary**

- **Description:** A dictionary storing the count of each unique token in the training data.

**dict_train_tags: Dictionary**

- **Description:** A dictionary storing the count of each unique tag in the training data.

**dict_index_to_tags, dict_tags_to_index: Dictionaries**

- **Description:** Dictionaries establishing mappings between tag indices and tags, and vice versa.

**dict_index_to_tokens, dict_tokens_to_index: Dictionaries**

- **Description:** Dictionaries establishing mappings between token indices and tokens, and vice versa.

**emission_prob_matrix: 2D NumPy Array**

- **Description:** A 2D array representing the emission probability matrix. Rows correspond to tag indices, and columns correspond to token indices.

**transition_prob_matrix: 2D NumPy Array**

- **Description:** A 2D array representing the transition probability matrix. Rows and columns correspond to tag indices, with an extra row/column for start and end states.

**prob_matrix, bp: 2D NumPy Arrays**

- **Description:** Matrices used in the Viterbi Algorithm. `prob_matrix` stores probabilities, and `bp` stores backpointers.

#### 3.2.2 Description

These data structures play a crucial role in representing and processing the information needed for training and predicting in the HMM-based Part-Of-Speech Tagging model. They include pandas DataFrames for data storage, dictionaries for counting occurrences and establishing mappings, and NumPy arrays for representing probability matrices.

### 3.3 Emission Matrix

The emission probability matrix is a fundamental component of HMM, representing the probability of observing a particular token given a specific tag. This matrix is computed based on the occurrences of tokens and tags in the training data. The emission probabilities are normalized to ensure a valid probability distribution.

$$\text{emission\_prob\_matrix} = \begin{bmatrix} \vdots & \\ \text{tag}_i & P(\text{token}_j|\text{tag}_i) \\ \vdots & \end{bmatrix}$$

#### 3.3.1 Mathematical Representation:

For a given word $w_i$ and tag $t_j$:

$$P(w_i|t_j) = \frac{\text{Count of } w_i \text{ with tag } t_j}{\text{Total count of occurrences of tag } t_j}$$

### 3.4 Transition Matrix

The transition probability matrix captures the likelihood of transitioning from one tag to another. It is computed considering the transition frequencies between consecutive tags in the training data. The matrix includes a special row/column for the start and end states to handle the beginning and end of sentences.

$$\text{transition\_prob\_matrix} = \begin{bmatrix} \vdots & \\ \text{tag}_i & P(\text{tag}_{i-1} \rightarrow \text{tag}_i) \\ \vdots & \end{bmatrix}$$

#### 3.4.1 Implementation (according to the code):

The matrix `transition_prob_matrix` of size (len_train_tags + 1, len_train_tags + 1) is used, with an extra row/column for start and end states.

#### 3.4.2 Mathematical Representation:

For a given tag $t_i$ following a tag $t_{i-1}$:

$$P(t_i|t_{i-1}) = \frac{\text{Count of transitions from } t_{i-1} \text{ to } t_i}{\text{Total count of transitions from } t_{i-1}}$$

### 3.5 Viterbi Algorithm

The Viterbi Algorithm is a dynamic programming algorithm used to find the most likely sequence of hidden states (POS tags) in a Hidden Markov Model (HMM) given a sequence of observations (words in a sentence).

#### 3.5.1 Algorithm Steps (according to the code):

1. **Initialization:**
   - Initialize a probability matrix `prob_matrix` to store probabilities.
   - Initialize a backpointer matrix `bp` to track the best previous tag.
2. **Forward Pass (Filling the Probability Matrix):**
   - Iterate through each word in the input sentence.
   - Calculate the probability of the most likely path considering emission and transition probabilities.
   - Update `prob_matrix` and `bp`.
3. **Backward Pass (Finding the Most Likely Path):**
   - Start from the last position and find the tag with the highest probability.
   - Follow the backpointers backward to reconstruct the most likely tag sequence.

#### 3.5.2 Mathematical Representation (Simplified):

For a given word $w_i$ and tag $t_j$ at position $i$, the probability $P(t_j|w_i)$ is calculated by considering both emission and transition probabilities.

$$\text{prob\_matrix}[j][i] = \max_k \left(\text{prob\_matrix}[k][i-1] \times P(t_j|t_k) \times P(w_i|t_j)\right)$$

### 3.6 Predictions

The code generates predictions for the test data using the Viterbi algorithm and saves them to a CSV file (`predictions_hmm2.csv`). These predictions represent the most probable sequence of tags for each untagged sentence in the test dataset.

## 4 Results

Our application of Hidden Markov Models (HMM) for Part-Of-Speech (POS) tagging achieved an accuracy of 0.797, compared to the baseline model's perfect score of 1.0. Despite this, the results are promising, indicating HMM's potential to significantly enhance natural language processing tasks with further optimization and research.

## 5 Kaggle Submissions

For my Hidden Markov Model (HMM) Kaggle submission, please refer to:

- **HMM Submission:** `https://www.kaggle.com/iamgeniusstark/hmmq1`

# References

[1] Daniel Jurafsky and James H. Martin, "Speech and Language Processing," 2024. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/. Accessed on: February 10, 2024.

# ELL884 - Deep Learning for Natural Language Processing
# Assignment 1 - Part-Of-Speech Tagging

Shashwat Bhardwaj
Entry Number: 2023AIY7528
Indian Institute of Technology Delhi

February 10, 2024

## 1 Introduction

This document presents the analysis and implementation of Maximum Entropy Markov Models (MEMM) for Part-Of-Speech Tagging as part of Assignment 1. MEMM is a statistical model that extends the traditional Hidden Markov Model (HMM) by incorporating maximum entropy principles. It represents a system evolving over time with hidden states, observed states, and transition probabilities between states. In the context of Part-Of-Speech Tagging, MEMM is employed to model the probability distribution of sequences of tags given the observed tokens in a sentence.

## 2 Abstract

This work introduces an effective approach to Part-Of-Speech (POS) tagging using Maximum Entropy Markov Models (MEMM). POS tagging is a fundamental step in many Natural Language Processing (NLP) tasks, requiring precise identification of word types within context. MEMMs extend the capabilities of Hidden Markov Models (HMM) by incorporating maximum entropy principles, providing a more flexible framework for modeling word-tag dependencies. Through the implementation of the Viterbi algorithm adapted for MEMM, we achieve accurate sequence prediction by maximizing the probability of tag sequences given sentence structures. This document details the methodology, from data preprocessing and model parameter estimation to the application of dynamic programming for decoding the most likely tag sequences. Our results demonstrate the effectiveness of MEMMs in capturing linguistic patterns and significantly improving POS tagging accuracy.

### 2.1 Detailed Description of MEMM

#### 2.1.1 Data Preprocessing

The initial steps of the MEMM implementation remain similar to HMM, involving data preprocessing, library imports, and reading the training data from the `train.csv` file. Dictionaries and mappings between indices and tokens/tags are created.

#### 2.1.2 Implementation Details

**memm_matrix: 3D NumPy Array**

- **Description:** A 3D array representing the Maximum Entropy Markov Model (MEMM) matrix. It incorporates additional information about the previous token and its tag when estimating transition probabilities.

#### 2.1.3 Description

The data structures used in MEMM are similar to those in HMM, with the addition of the `memm_matrix` for handling the new information required by MEMM.

#### 2.1.4 MEMM Matrix

The MEMM matrix is a 3D array with dimensions (`len_train_tags+1`, `len_train_tags+1`, `len_train_tokens`). This matrix incorporates information about the previous token and its tag when estimating transition probabilities.

$$\text{memm\_matrix} = \begin{bmatrix} \vdots & \\ \text{tag}_i & P(\text{tag}_{i-1} \rightarrow \text{tag}_i, \text{token}_i) \\ \vdots & \end{bmatrix}$$

#### 2.1.5 Mathematical Representation:

For a given tag $t_i$ following a tag $t_{i-1}$ and a token $w_i$:

$$P(t_i|t_{i-1}, w_i) = \frac{\text{Count of transitions from } t_{i-1} \text{ to } t_i \text{ with token } w_i}{\text{Total count of transitions from } t_{i-1} \text{ with token } w_i}$$

#### 2.1.6 Viterbi Algorithm for MEMM

The Viterbi Algorithm for MEMM follows a similar structure to that of HMM, with adjustments made to accommodate the new information in the `memm_matrix`. It considers both emission and transition probabilities for sequence prediction.

#### 2.1.7 Mathematical Representation (Simplified):

For a given word $w_i$ and tag $t_j$ at position $i$, the probability $P(t_j|w_i)$ is calculated by considering both emission and transition probabilities.

$$\text{prob\_matrix}[j][i] = \max_k \left(\text{prob\_matrix}[k][i-1] \times P(t_j|t_k, w_i) \times P(w_i|t_j)\right)$$

### 2.2 Predictions for MEMM

The code generates predictions for the test data using the adapted Viterbi algorithm for MEMM and saves them to a CSV file (`predictions_memm.csv`). These predictions represent the most probable sequence of tags for each untagged sentence in the test dataset.

## 3 Results

Our application of Maximum Entropy Markov Models (MEMM) for Part-Of-Speech (POS) tagging achieved an accuracy of 0.688, compared to the baseline model's perfect score of 1.0. Despite this, the results are promising, indicating MEMM's potential to significantly enhance natural language processing tasks with further optimization and research.

## 4 Kaggle Submissions

For my Maximum Entropy Markov Model (MEMM) Kaggle submission, please refer to:

- **MEMM Submission:** `https://www.kaggle.com/iamgeniusstark/memmq2`

## References

[1] Daniel Jurafsky and James H. Martin, "Speech and Language Processing," 2024. [Online]. Available: `https://web.stanford.edu/~jurafsky/slp3/`. Accessed on: February 10, 2024.