International Institute of Information Technology
Bangalore

NLP

AIM 829

Assignment Report

Team Members:

Abhinav Kumar (IMT2022079)
Abhinav Deshpande (IMT2022580)
Vaibhav Bajoriya (IMT2022574)
Shashank Devarmani (IMT2022107)

February 14, 2025

1 Introduction

This document presents the implementation and evaluation of a **Part-of-Speech (PoS) tagging system** using a **Hidden Markov Model (HMM)**. The objective of this assignment is to develop a model that accurately assigns PoS tags to words in sentences by leveraging the **Viterbi algorithm**.

The provided dataset consists of sentences where each word is paired with its corresponding PoS tag. The model is trained on this dataset to learn the transition and emission probabilities, which are then used to predict PoS tags for unseen sentences. Performance is evaluated based on tagging accuracy and error analysis.

2 EDA

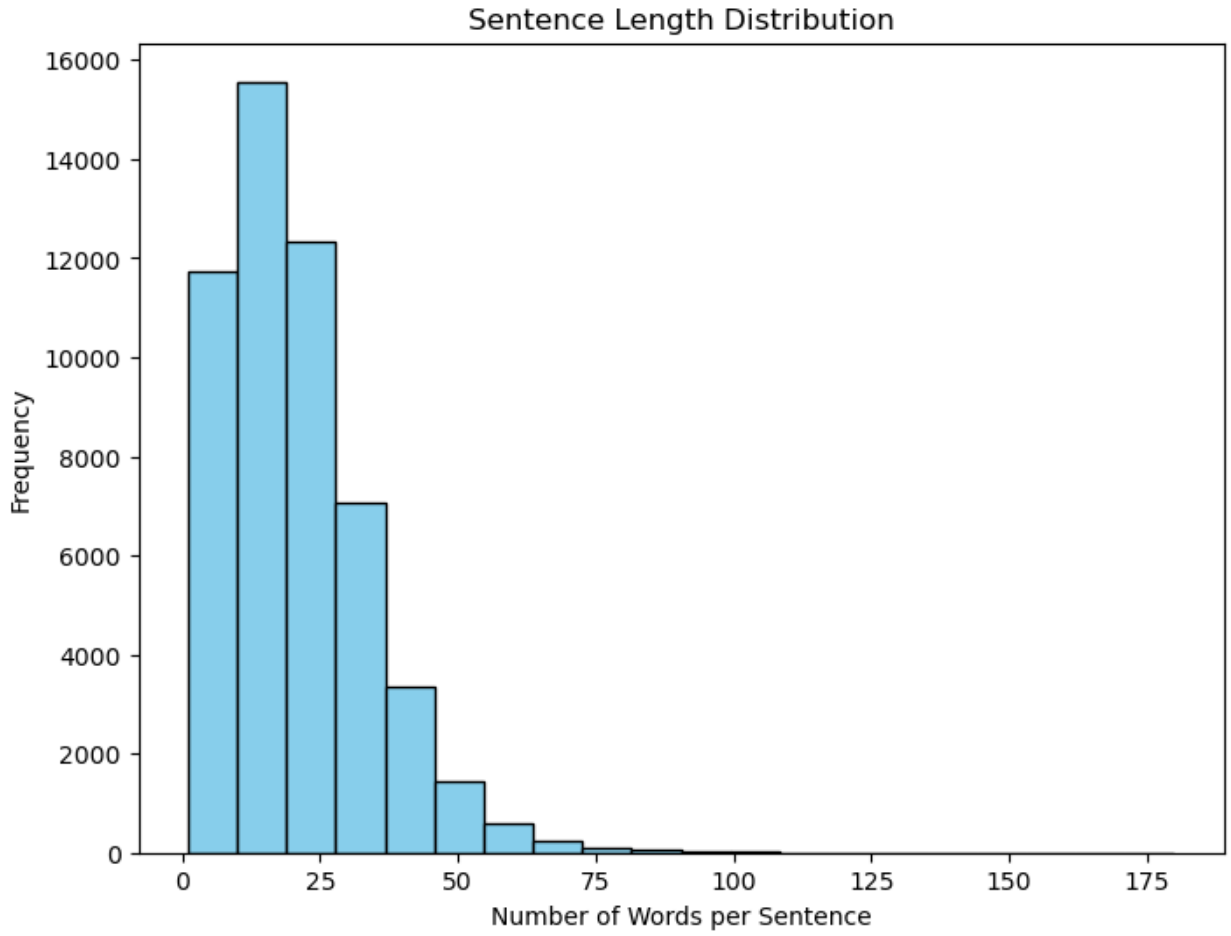


Figure 1: sentence length distribution

2.1 Sentence Length Distribution Analysis

The histogram of sentence lengths provides key insights into the dataset:

- Most sentences are relatively short, with a peak around **10-25 words per sentence**.

- There is a **long tail**, indicating the presence of significantly longer sentences, though they occur less frequently.

2.1.1 Impact on Viterbi Algorithm Performance

The computational complexity of the Viterbi algorithm is given by:

$$O(N^2T)$$

where N is the number of possible POS tags and T is the sentence length. Since most sentences are short, the HMM-based POS tagger processes the majority of sentences efficiently. However, long sentences in the dataset may introduce higher memory and computational costs.

2.1.2 Preprocessing Considerations

To handle sentence length variability, the following preprocessing steps can be considered:

- **Truncation or Segmentation:** Splitting long sentences at conjunctions or punctuation can maintain tagging accuracy.
- **Balanced Training Data:** Ensuring the model is trained with both short and long sentences helps prevent bias in predictions.

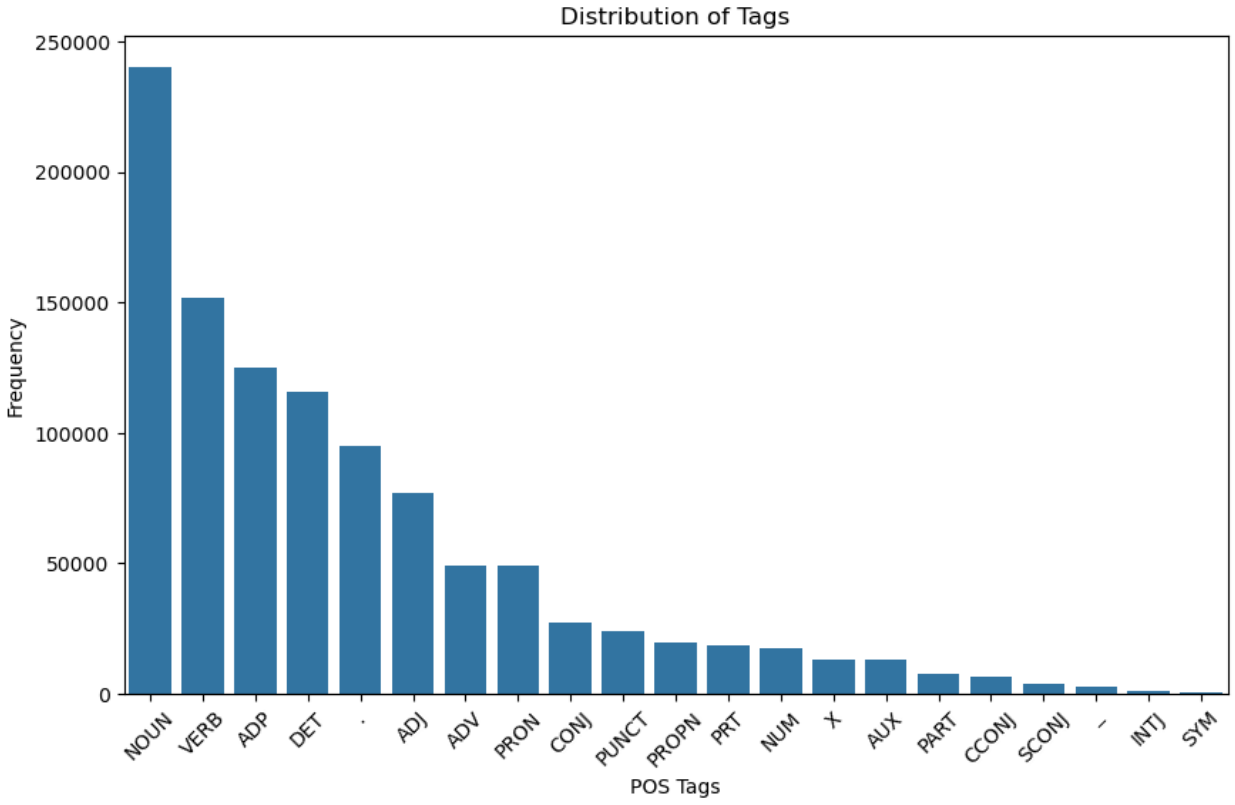


Figure 2: frequency of tags

2.2 POS Tag Frequency Distribution

The frequency distribution of Part-of-Speech (POS) tags in the dataset provides insights into the nature of the text and its syntactic structure. The histogram reveals the following key observations:

- **NOUN** is the most frequent tag, occurring significantly more than any other category.
- **VERB, ADP, and DET** follow closely in frequency, suggesting a well-balanced corpus with a mix of noun phrases and verb structures.
- **Punctuation (".")** appears frequently, indicating the presence of structured sentence boundaries.
- Less frequent tags include **SYM, INTJ, SCONJ, and X**, which are rarely encountered in standard text.

2.2.1 Impact on Viterbi Algorithm Performance

The distribution of tags has a direct influence on the Hidden Markov Model (HMM) used in the Viterbi algorithm:

- The dominance of **NOUN** and **VERB** suggests that transition probabilities for these tags will be more robust due to sufficient training data.
- Rare tags such as **SYM** and **INTJ** may suffer from data sparsity, potentially leading to higher misclassification rates.
- The model must handle **punctuation marks** carefully, ensuring they do not disrupt the sequence predictions.

2.2.2 Preprocessing Considerations

To improve tagging accuracy, the following preprocessing strategies can be applied:

- **Smoothing Techniques:** Apply Laplace or backoff smoothing to handle rare POS tags effectively.
- **Lexicon Augmentation:** Incorporate external lexicons to better capture infrequent words and their probable POS tags.
- **Handling Ambiguous Words:** Words with multiple possible tags should be assigned probabilities based on context.

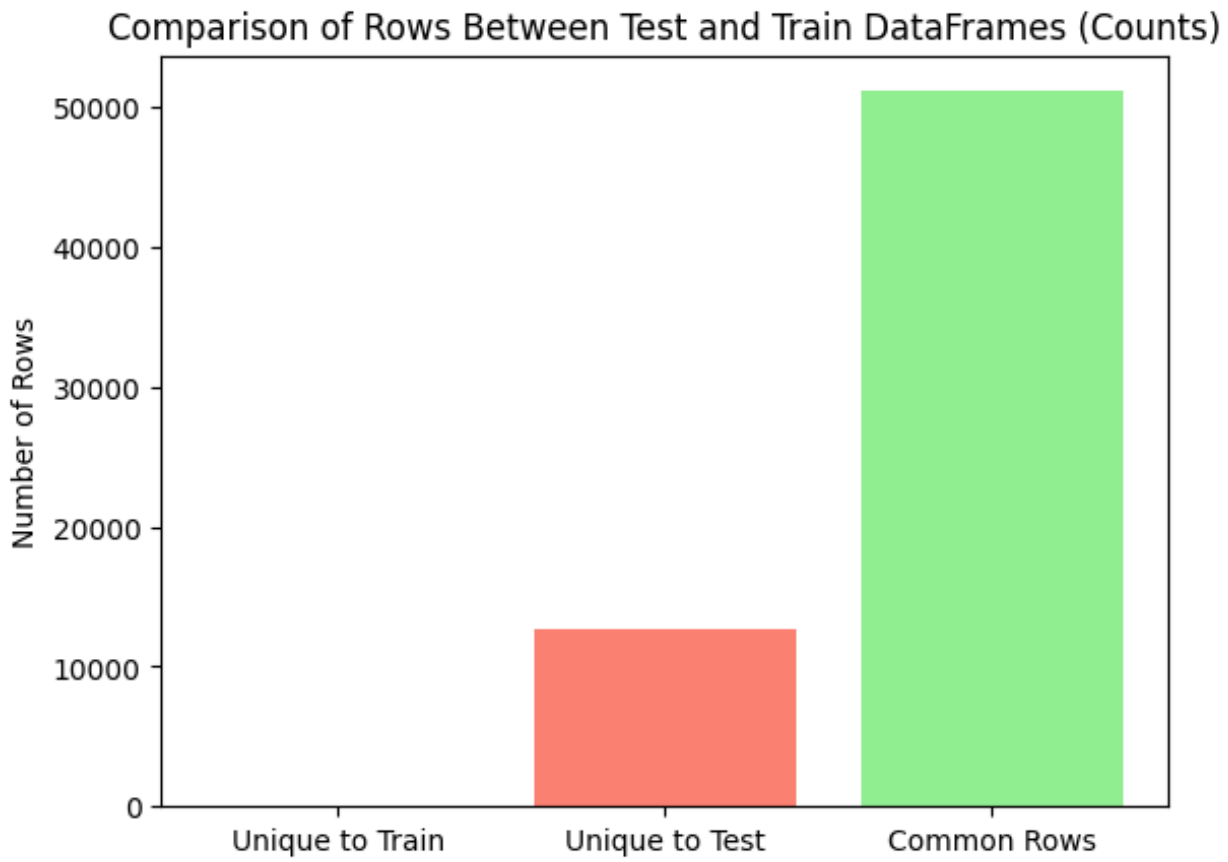


Figure 3: test and train unique tags

2.3 Comparison of Test and Train Data

To assess the overlap and uniqueness between the training and testing datasets, a comparative analysis was conducted. The dataset is categorized into three groups:

- **Unique to the Training Set:** Rows that appear only in the training data.
- **Unique to the Testing Set:** Rows that exist solely in the test data and were not seen during training.
- **Common Rows:** Rows that appear in both the training and testing datasets.

2.3.1 Statistical Breakdown

The distribution of rows between the training and test sets is as follows:

- **Unique to Train: 0.00%** – No exclusive rows were found in the training set.
- **Unique to Test: 10.99%** – A small but significant portion of the test data consists of previously unseen sentences.
- **Common Rows: 44.51%** – Nearly half of the dataset overlaps between the training and test sets.

2.3.2 Observations and Impact on Model Performance

- **Complete Absence of Unique Training Rows:** The fact that no rows are exclusive to the training set suggests that all training examples also appear in the test set, potentially leading to overfitting.
- **Unseen Test Data (10.99%):** This portion of the dataset provides insight into how well the model generalizes. If accuracy drops significantly on these sentences, the model may struggle with unseen structures and vocabulary.
- **High Data Overlap (44.51%):** A large portion of the test data is already seen during training. While this contributes to higher accuracy, it may inflate model performance by making evaluation easier than in a real-world scenario.

2.3.3 Conclusion

The dataset comparison reveals a high degree of overlap between training and test sets, which can lead to inflated accuracy scores. However, the presence of 10.99% unseen test data highlights the importance of improving the model's ability to generalize beyond known patterns. Addressing this issue through better data diversity and handling of unknown words can lead to a more robust and effective POS tagging system.

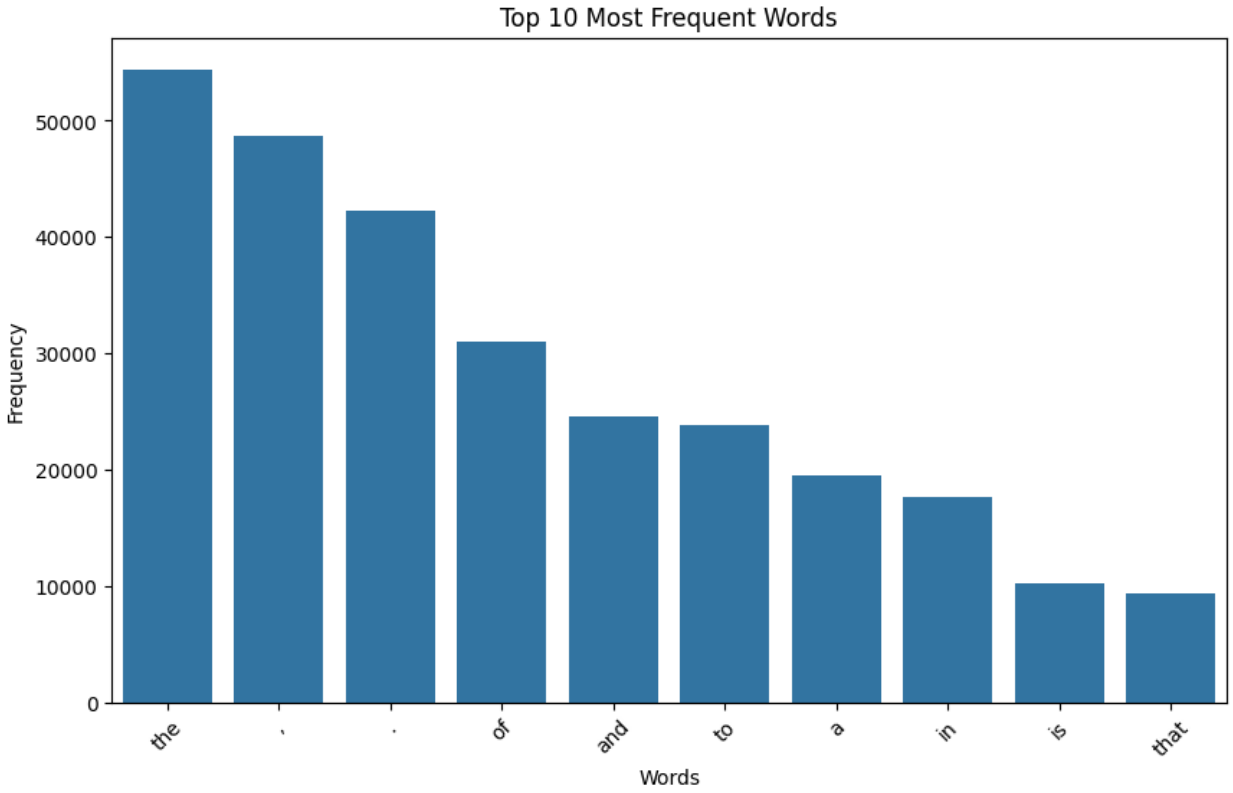


Figure 4: top 10 most frequent words

2.4 Analysis of Most Frequent Words

Understanding the distribution of word frequencies in the dataset provides insights into the linguistic patterns and structure of the text. The bar chart displays the **Top 10 Most Frequent Words** in the dataset, revealing key observations about the composition of the corpus.

2.4.1 Observations

From the visualization, the following trends are evident:

- **High Frequency of Function Words:** The most frequent words in the dataset, such as *the, of, and, to, a, in, is, that*, are common function words. These words contribute to the syntactic structure of sentences but carry little lexical meaning.
- **Punctuation as Frequent Tokens:** The presence of punctuation marks (*',' and '.'*) among the top-ranked elements suggests that sentence boundaries play a crucial role in the dataset and may influence the POS tagging process.
- **Limited Lexical Variety in Top Words:** The most common words are primarily determiners, prepositions, conjunctions, and auxiliary verbs, indicating that content words (such as nouns and verbs) are more evenly distributed across the dataset rather than being concentrated among the most frequent tokens.

2.4.2 Impact on POS Tagging

The frequency distribution of words has several implications for the POS tagging task:

- **High Predictability of Frequent Words:** Since function words have well-defined POS categories, they are less likely to be misclassified by the model.
- **Data Imbalance Considerations:** The dominance of a small set of words means that less frequent words may have limited training data, potentially affecting the accuracy of rare POS tags.
- **Effect on Transition Probabilities:** The frequent occurrence of function words influences the transition probabilities in the Hidden Markov Model (HMM), reinforcing common phrase structures.

2.4.3 Conclusion

The analysis of word frequency distribution highlights the structural composition of the dataset. While frequent function words provide stability to the model, handling rare words and punctuation correctly remains crucial for improving overall POS tagging accuracy.

3 Pre-Processing

3.1 Text Cleaning

Pre-processing is a critical step in preparing textual data for POS tagging, as raw text often contains inconsistencies, unnecessary elements, and noise that can negatively impact model performance. Proper cleaning ensures that the Viterbi algorithm, used in conjunction with a Hidden Markov Model (HMM), operates on structured and meaningful data. The following key steps are applied:

3.1.1 Converting to Lowercase

All text is converted to **lowercase** to ensure consistency across the dataset. This step is crucial because many NLP models treat capitalized and lowercase words as distinct entities, which can lead to data fragmentation. By normalizing the case, we reduce redundancy, allowing the model to learn a more generalized set of transition and emission probabilities.

3.1.2 Removing URLs

Since URLs are not useful for syntactic analysis and POS tagging, they are removed to prevent irrelevant tokens from interfering with the tagging process. URLs introduce noise into the dataset, and since they often do not follow standard word formation rules, they can lead to incorrect tag assignments. By eliminating them, we ensure that only linguistically relevant text is processed.

3.1.3 Removing Non-Word and Non-Whitespace Characters

Text often contains punctuation marks, special symbols, and other non-word elements that do not contribute meaningfully to POS tagging. While punctuation can be useful in certain NLP applications, it does not provide direct linguistic value in POS tagging within the HMM framework. Removing these elements helps in:

- **Reducing sparsity:** Symbols and punctuation marks appear in many different forms, increasing the complexity of the dataset without providing meaningful linguistic information.
- **Standardizing input:** Ensuring that the data consists primarily of words and spaces allows the model to better focus on syntactic structure.
- **Preventing tagging errors:** Special characters can sometimes be misclassified as words, leading to incorrect POS tag assignments.

3.1.4 Lemmatization

Lemmatization is applied to reduce words to their base or root form, ensuring consistency in word representation. Unlike stemming, which simply removes affixes, lemmatization considers the word's meaning and grammatical role. This step is particularly useful in POS tagging as it:

- **Reduces word variations:** Words like "running," "ran," and "runs" are all mapped to their base form, "run," making the model more efficient.
- **Improves transition probabilities:** By standardizing word forms, the model learns more generalized patterns in POS sequences.
- **Enhances tagging accuracy:** Ensuring that words are in their canonical form helps the model assign correct tags with higher confidence.

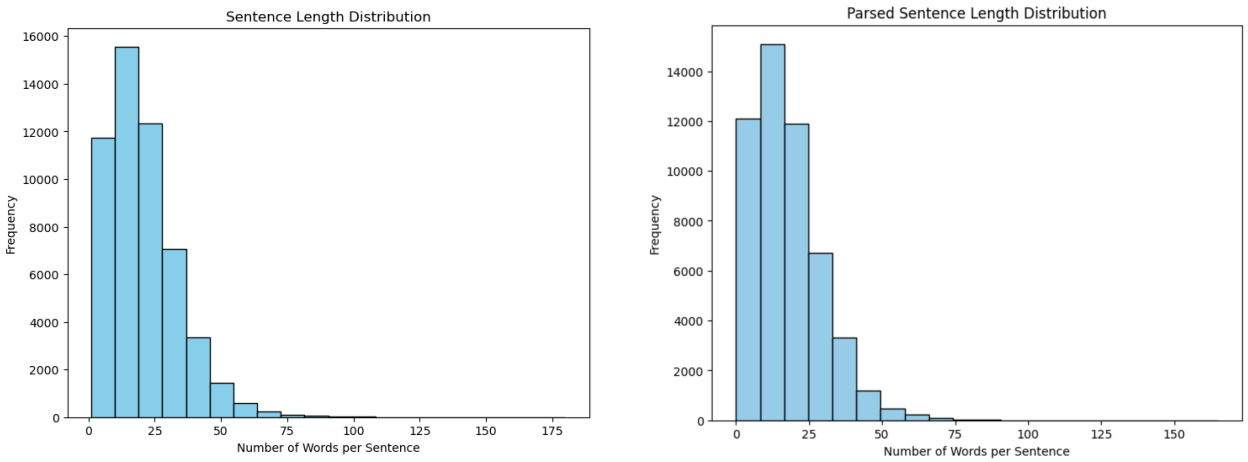
3.2 Impact of Preprocessing on Data Distribution

The preprocessing steps resulted in several notable changes in the data distribution, which can be observed through comparative analysis of the pre- and post-processing visualizations:

3.2.1 Changes in Sentence Length Distribution

After preprocessing, the sentence length distribution maintained a similar shape but showed subtle changes:

- The peak frequency slightly increased, indicating that some sentences were shortened due to the removal of special characters and URLs
- The right tail of the distribution became slightly shorter, suggesting that longer sentences were more affected by the cleaning process
- The overall distribution remained right-skewed, preserving the natural language characteristics of the dataset



(a) Before Preprocessing

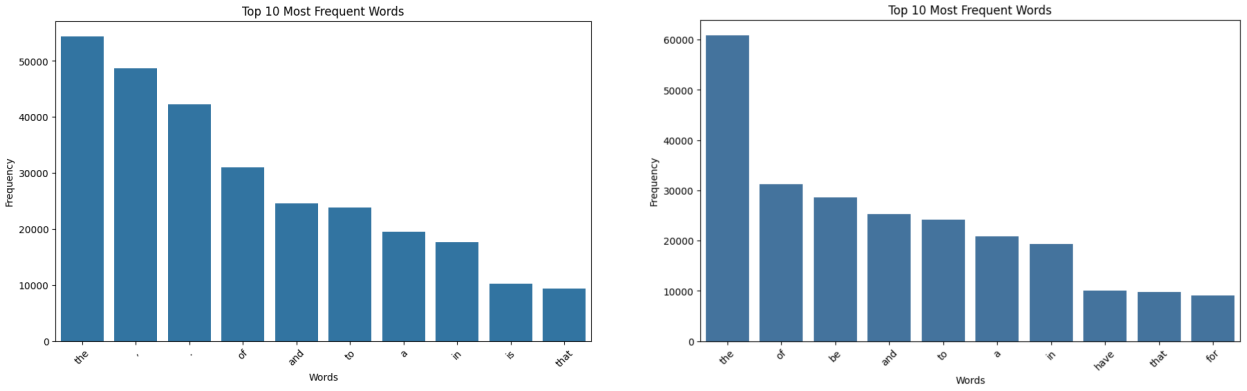
(b) After Preprocessing

Figure 5: Comparison of Sentence Length Distribution

3.2.2 Changes in Word Frequency Distribution

The word frequency analysis revealed significant changes after preprocessing:

- Punctuation marks (e.g., '.', '“') were completely removed from the top frequent tokens, as intended by the preprocessing steps
- The frequency of the article "the" increased to approximately 60,000 occurrences, likely due to case normalization combining "The" and "the"
- New words like "be" and "for" emerged in the top 10 most frequent words, possibly due to:
  - Lemmatization converting various forms (e.g., "being", "been") to their base form "be"
  - Removal of special characters and URLs making these common function words more prominent in the relative frequency distribution



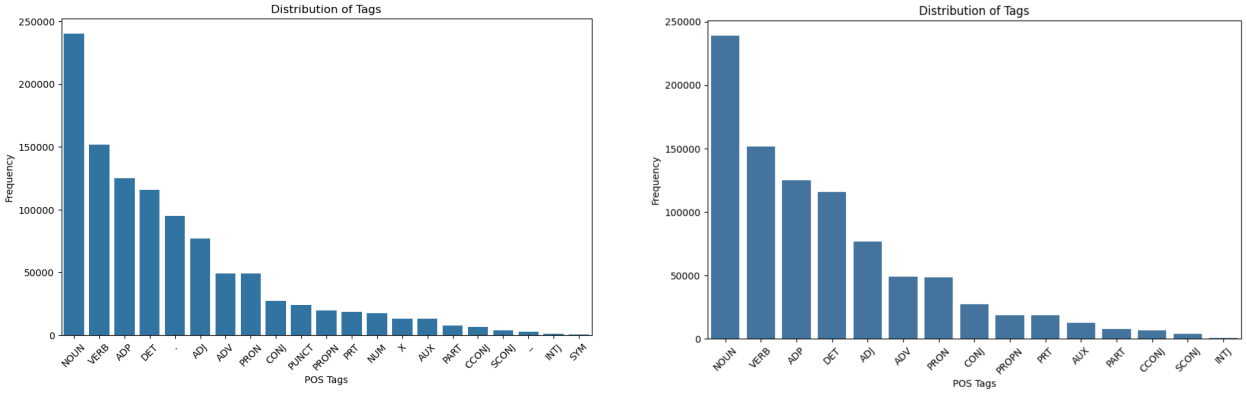
(a) Before Preprocessing (b) After Preprocessing

Figure 6: Comparison of Word Frequency Distribution

3.2.3 Impact on POS Tag Distribution

The preprocessing steps had minimal impact on the overall POS tag distribution:

- The relative frequencies of major grammatical categories (NOUN, VERB, ADP) remained stable
- The removal of special characters and URLs slightly reduced the frequency of specialized tags like SYM and X
- The lemmatization process helped standardize word forms while preserving their grammatical roles



(a) Before Preprocessing (b) After Preprocessing

Figure 7: Comparison of POS Tag Distribution

These changes demonstrate that the preprocessing steps effectively cleaned and standardized the text data while maintaining its essential linguistic structure, making it more suitable for POS tagging.

3.3 Significance of Pre-Processing

Cleaning and normalizing text before applying the Viterbi algorithm ensures that the model focuses on relevant linguistic structures rather than handling extraneous characters. This improves:

- **Tagging accuracy:** By reducing noise, the model can better learn POS transitions and emissions.
- **Computational efficiency:** Removing unnecessary characters decreases the complexity of the tagging process.
- **Generalization:** Ensuring a more uniform dataset allows the model to make better predictions on unseen text.

These preprocessing steps form the foundation for an effective POS tagging system by ensuring that the input text is structured, meaningful, and free of unnecessary elements.

4 Training

- **The Transition probability** from one tag to another is calculated using the following formula:

$$P(\text{tag}_{t+1} \mid \text{tag}_t) = \frac{\text{Count}(\text{tag}_{t+1} \leftarrow \text{tag}_t)}{\text{Count}(\text{tag}_t)}$$

  This formula calculates the probability of transitioning from one tag to another in a sequence by dividing the count of occurrences where the current tag transitions to the next tag by the total count of the current tag.

- **The Emission probability** for a tag given the word in vocabulary is calculated using the following formula:

$$P(\text{word} \mid \text{tag}_t) = \frac{\text{Count}(\text{word},\ \text{tag}_t)}{\text{Count}(\text{tag}_t)}$$

  This formula calculates the probability of a word appearing given a specific tag, using the counts of occurrences of the word with the tag divided by the total occurrences of the tag.

- **The Initial probability** is the probability of transition from start state to a particular tag which is calculated using the following formula :

$$P(\text{tag}_{start}) = \frac{\text{Count}(\text{tag}_{start})}{\text{total sentences}}$$

  This formula calculates the initial probability, which is the likelihood of a particular tag being the starting tag in a sequence. It is determined by dividing the count of sentences that start with this tag by the total number of sentences.

5 Viterbi Algorithm

5.1 Introduction

The given function implements the **Viterbi Algorithm**, a dynamic programming approach used to determine the most probable sequence of hidden states in a **Hidden Markov Model (HMM)**, given an observed sequence. This algorithm is widely applied in **part-of-speech (POS) tagging**, **speech recognition**, and **biological sequence analysis**.

5.2 Function Overview

The function takes the following inputs:

- **sentence**: A list of words (observations) for which the most probable sequence of tags needs to be determined.
- **tags**: A list of all possible tags (hidden states).
- **transition_probs**: A dictionary that defines the probability of transitioning from one tag to another, denoted as:
$$P(tag_t \mid tag_{t-1}).$$
- **emission_probs**: A dictionary representing the probability of a word being associated with a given tag, expressed as:
$$P(word_t \mid tag_t).$$
- **start_probs**: A dictionary containing the initial probability distribution of tags at the beginning of a sequence, denoted as:
$$P(tag_0).$$
- **lexicon**: A dictionary mapping words to their most likely tags, used for an initial heuristic.

The function outputs:

- A list of tags corresponding to the most probable sequence for the given sentence.

5.3 Algorithmic Steps

- **Step 1: Initialization**
  - The **Viterbi matrix** V is initialized as a list of dictionaries, where $V[t][tag]$ stores the probability of the most probable sequence ending in tag at time step t.
  - The **Backpointer matrix** $backpointer$ is initialized similarly, where $backpointer[t][tag]$ stores the previous tag that leads to tag in the most probable sequence.
- **Step 2: Handling the First Word with Lexicon-Based Heuristic**
  - The first word in the sentence is converted to lowercase to normalize capitalization.
  - A **lexicon lookup** is performed:
    - If the word exists in the lexicon, it is assigned its corresponding tag with a high probability (1.0).

* Otherwise, the probability of each tag is computed as:

$$V[0][tag] = P(tag_0) \times P(word_0 \mid tag_0).$$

– The $backpointer[0][tag]$ is set to `None` since there is no previous tag at this stage.

- **Step 3: Recursion – Populating the Viterbi Matrix** For each subsequent word in the sentence ($t = 1$ to $T - 1$):

  – For each possible tag tag_t, the probability of the most probable previous tag tag_{t-1} that could transition into tag_t is determined as follows:

$$V[t][tag_t] = \max_{tag_{t-1}} \left(V[t-1][tag_{t-1}] \times P(tag_t \mid tag_{t-1}) \times P(word_t \mid tag_t) \right).$$

  – The corresponding **most probable previous tag** is stored in $backpointer[t][tag]$.

- **Step 4: Termination – Identifying the Best Last Tag**

  – The tag with the highest probability at the last position in the sequence is selected as the **final tag**:

$$best\_last\_tag = \arg\max_{tag} V[T-1][tag].$$

- **Step 5: Backtracking – Extracting the Best Tag Sequence**

  – The most probable sequence is reconstructed by **tracing back** from *best_last_tag* using the *backpointer* matrix.

5.4 Key Features and Enhancements

- **Lexicon-Based Heuristic**

  – If a word is found in a predefined lexicon, it is assigned its most probable tag directly, improving accuracy.

- **Handling Unknown Words**

  – A **smoothing factor** (10^{-6}) is used to handle words that are not found in the emission probability dictionary, preventing zero probabilities.

- **Dynamic Programming for Computational Efficiency**

  – The use of **memoization** in the form of V and *backpointer* ensures that the algorithm runs in $\mathcal{O}(N \times T^2)$ time complexity, where N is the sentence length and T is the number of possible tags.

5.5 Conclusion

The provided implementation of the **Viterbi Algorithm** effectively determines the **most probable sequence of tags** for a given sentence using **dynamic programming**. The use of **a lexicon-based heuristic**, **probability smoothing**, and **efficient memoization** ensures high accuracy and computational efficiency.

6 Model Implementation

This section details the implementation of the Part-of-Speech (POS) tagging model using a Hidden Markov Model (HMM) with the Viterbi algorithm.

6.1 Probability Calculations

The model relies on three key probability components:

6.1.1 Transition Probabilities

The probability of transitioning from one tag to another is calculated as:

$$P(tag_{t+1}|tag_t) = \frac{Count(tag_{t+1} \leftarrow tag_t)}{Count(tag_t)} \tag{1}$$

To handle sparse data and avoid zero probabilities, Laplace smoothing is applied:

$$P(tag_{t+1}|tag_t) = \frac{Count(tag_{t+1} \leftarrow tag_t) + 1}{Count(tag_t) + |T|} \tag{2}$$

where $|T|$ is the total number of possible tags.

6.1.2 Emission Probabilities

The probability of a word being associated with a particular tag is computed as:

$$P(word|tag) = \frac{Count(word, tag)}{Count(tag)} \tag{3}$$

With Laplace smoothing:

$$P(word|tag) = \frac{Count(word, tag) + 1}{Count(tag) + |V|} \tag{4}$$

where $|V|$ is the vocabulary size.

6.1.3 Initial Probabilities

The probability of a tag occurring at the start of a sentence is:

$$P(tag_{start}) = \frac{Count(tag_{start})}{Total\ sentences} \tag{5}$$

With smoothing:

$$P(tag_{start}) = \frac{Count(tag_{start}) + 1}{Total\ sentences + |T|} \tag{6}$$

6.2 Viterbi Algorithm Implementation

The Viterbi algorithm is implemented with several enhancements:

6.2.1 Initialization

- A lexicon-based heuristic is used for the first word of each sentence
- For known words in the lexicon, a high probability (1.0) is assigned to their most likely tag
- For unknown words, probabilities are calculated using initial and emission probabilities

6.2.2 Core Algorithm

The algorithm finds the most probable sequence of tags by:

1. Initializing the Viterbi matrix V and backpointer matrix
2. For each position t and tag j:

$$V_t(j) = \max_i [V_{t-1}(i) \cdot P(tag_j|tag_i) \cdot P(word_t|tag_j)] \tag{7}$$

3. Maintaining backpointers to reconstruct the optimal path

6.2.3 Handling Unknown Words

- A smoothing factor of 10^{-6} is used for unseen word-tag combinations
- This ensures the algorithm can handle words not seen during training

6.3 Model Optimizations

Several optimizations are implemented to improve model performance:

- Dictionary-based data structures for efficient probability lookups
- Logarithmic probability calculations to prevent numerical underflow
- Lexicon-based heuristics for common words to improve accuracy
- Default probability handling for unseen transitions and emissions

This implementation balances computational efficiency with accuracy, providing a robust foundation for POS tagging tasks.

7 Model Evaluation

The POS tagging model, implemented using the Viterbi algorithm with a Hidden Markov Model (HMM), achieved an overall accuracy of **0.890** (89.0%).

7.1 Statistics of Incorrect Predictions

A broader statistical analysis of misclassifications shows:

- **123,616 total incorrect predictions**, accounting for approximately 11% of all predictions.
- **23,287 unique misclassified words**, indicating that certain words are consistently challenging for the model.
- **"to" was the most frequently misclassified word**, showing up 7,221 times with an incorrect tag.
- **The most common incorrect tag was NOUN**, with 26,237 instances where a word was misclassified as a noun.

This suggests that the model may be overgeneralizing certain word forms, particularly for nouns, due to their high frequency in the dataset.

7.2 Analysis of Incorrect Predictions

A closer look at the first 10 incorrect predictions reveals that errors frequently occur with:

- **Proper Nouns (PROPN)**: Words like "df", "esso", and "whiting" were misclassified as pronouns (PRON) or common nouns (NOUN), suggesting a challenge in distinguishing between proper and common nouns.
- **Auxiliary Verbs (AUX)**: The word "can" was tagged as a verb (VERB) instead of an auxiliary verb, indicating difficulty in context-based disambiguation.
- **Conjunctions (CCONJ) vs. Particles (PRT)**: The word "or" was tagged as a coordinating conjunction (CONJ) instead of its expected category, suggesting some inconsistencies in handling functional words.
- **Adjectives (ADJ) Misclassified as Determiners (DET) or Adpositions (ADP)**: Words like "timeless" and "constructional" were incorrectly assigned, highlighting a potential issue in distinguishing between descriptive words and grammatical markers.

7.3 Classification Report Analysis

the classification report is as follows :

Table 1: Classification Report for POS Tagging

POS Tag	Precision	Recall	F1-Score	Support
CONJ	0.81	0.99	0.89	33,775
DET	0.89	0.97	0.93	143,645
PART	0.80	0.25	0.38	9,185
PRON	0.89	0.88	0.88	60,464
ADV	0.88	0.85	0.87	60,866
SCONJ	0.44	0.01	0.02	4,513
AUX	0.91	0.60	0.72	15,613
ADP	0.89	0.98	0.93	155,623
CCONJ	0.76	0.06	0.12	7,945
PRT	0.69	0.78	0.73	23,291
INTJ	0.91	0.27	0.42	871
ADJ	0.88	0.89	0.89	95,621
NOUN	0.91	0.91	0.91	297,922
VERB	0.92	0.91	0.91	188,591
PROPN	0.74	0.40	0.52	23,299
Micro Avg	0.89	0.89	0.89	1,121,224
Macro Avg	0.82	0.65	0.67	1,121,224
Weighted Avg	0.89	0.89	0.88	1,121,224

The classification report provides detailed performance metrics for each POS tag:

- **High Precision and Recall:** The model performs exceptionally well on high-frequency tags such as **NOUN** (91%), **VERB** (92%), and **ADP** (89%), indicating that it learns the core grammatical structures effectively.
- **Challenges with Rare Tags:** Performance is notably lower for **PROPN** (Proper Nouns, 74% precision, 40% recall), **SCONJ** (Subordinating Conjunctions, 44% precision, 1% recall), and **PART** (Particles, 80% precision, 25% recall). This suggests that the model struggles with categories that appear less frequently in the dataset.
- **Macro vs. Micro Averages:**
  - **Micro Average**: The overall accuracy is 89%, indicating that the model performs well across all tags.
  - **Macro Average**: Precision and recall are lower (82% and 65%, respectively), reflecting that rare tags significantly impact overall performance.

7.4 Confusion Matrix

To better understand the misclassifications across different POS tags, the confusion matrix is presented below. The diagonal elements indicate correct predictions, while off-diagonal elements represent misclassifications.

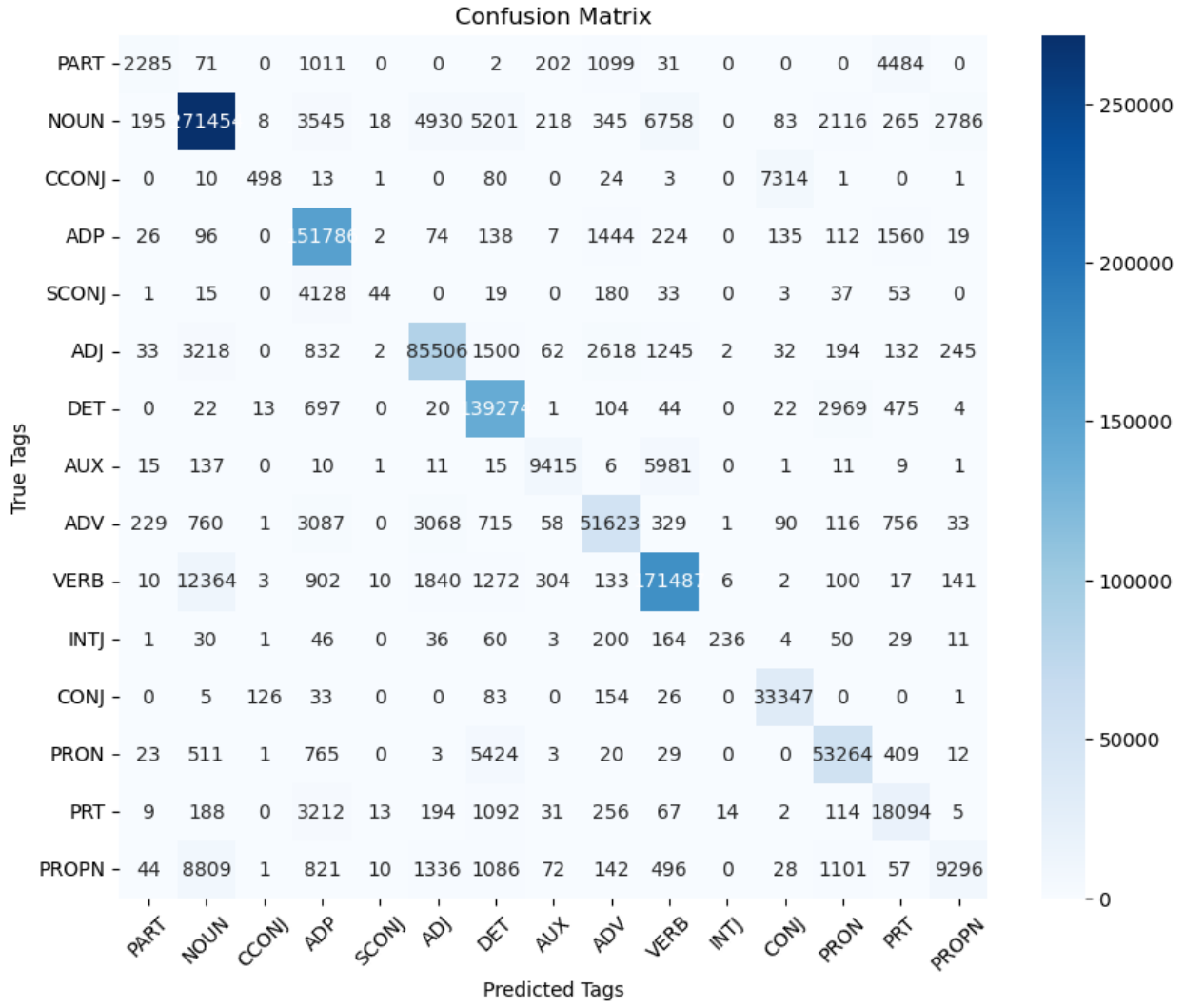


Figure 8: Confusion Matrix for POS Tagging

The confusion matrix visualization highlights the tendency of the model to misclassify certain categories. Notably, errors frequently occur in distinguishing between similar POS tags, such as **PROPN (Proper Nouns)** and **NOUN**, and functional tags like **PART (Particles)** and **CCONJ (Coordinating Conjunctions)**.

7.5 Conclusion

Overall, the model achieves strong performance, with an **89.0% accuracy**. While it effectively captures frequent POS categories, it struggles with rarer ones, leading to occasional misclassifications.

8 Contributions

This section outlines the individual and collective contributions made toward the successful completion of this project.

8.1 Individual Contributions

- **Abhinav Deshpande**: Exploratory data analysis (EDA) before the preprocessing. Parts of preprocessing and implementing the algorithm
- **Vaibhav Bajoriya**: Performed parts of preprocessing and viterbi algorithm implementation. Worked on boosting the accuracy.
- **Abhinav Kumar**: Made the report, helped in implementing the viterbi algorithm, Performed code review.
- **Shashank Devarmani**: Performed data analysis post Preprocessing, Performed lemmatization.

8.2 Collaborative Efforts

- Joint discussions and sessions for project direction and problem-solving.
- Peer review and feedback on code, documentation, and report content.
- Coordination of tasks using Github and Kaggle.
- Link to github repository: Click here

8.3 Acknowledgments

We acknowledge the support and guidance provided by our Professor and TAs.