HW3 Report

1 Introduction

This project implement and evaluate a Part-of-Speech (POS) tagging system based on a Hidden Markov Model (HMM). The system predicts the most likely sequence of POS tags for a given sentence using probabilistic modeling. Additionally, a baseline tagger is implemented for comparison, which assigns the most frequent tag to each word observed in the training data.

2 Methodology

2.1 Dataset Preparation

The dataset used is the Penn Treebank WSJ corpus, divided into three splits:

- **Training Set:** Used to compute transition and emission probabilities.
- **Development Set:** Used to tune the smoothing hyperparameter (α).
- **Test Set:** Used to evaluate the final performance of the models.

Each sentence is padded with `<START>` and `<STOP>` tokens to indicate the boundaries of the sentence.

2.2 Hidden Markov Model

The HMM POS tagger models the tagging process as a sequence prediction problem, using:

- **Transition Probabilities** ($P(t_k|t_{k-1})$): The likelihood of transitioning from one tag to another.
- **Emission Probabilities** ($P(w|t)$): The likelihood of a word being generated by a particular tag.

Both probabilities are estimated using Maximum Likelihood Estimation (MLE) with add-α smoothing:

$$P(t_k|t_{k-1}) = \frac{\text{Count}(t_{k-1} \to t_k) + \alpha}{\text{Count}(t_{k-1}) + \alpha \cdot |T|}$$

$$P(w|t) = \frac{\text{Count}(t \to w) + \alpha}{\text{Count}(t) + \alpha \cdot |V|}$$

where α is the smoothing parameter, $|T|$ is the number of tags, and $|V|$ is the vocabulary size.

2.3 Baseline Tagger

The baseline tagger assigns the most frequent tag to each word based on the training data. For unknown words, it assigns the default tag (`NN`).

2.4 Decoding with the Viterbi Algorithm

The Viterbi algorithm is used to decode the most probable sequence of tags for a given sentence:

- **Initialization:**

  $V[0][t] = \log(P(\texttt{<START>} \to t)) + \log(P(w_0|t))$

- **Recursion:**

  $V[t][k] = \max_{t_{k-1}} (V[t-1][t_{k-1}] + \log(P(t_{k-1} \to t_k)) + \log($

- **Termination:**

  $\max_{t_k} (V[n-1][t_k] + \log(P(t_k \to \texttt{<STOP>})))$

- **Backtracking:** Extract the best sequence of tags from the computed probabilities.

2.5 Evaluation Metrics

The models are evaluated using the following metrics:

- **Accuracy:** Proportion of correctly predicted tags.
- **Precision:** $\frac{\text{True Positives}}{\text{True Positives}+\text{False Positives}}$
- **Recall:** $\frac{\text{True Positives}}{\text{True Positives}+\text{False Negatives}}$
- **F1 Score:** Harmonic mean of precision and recall.
- **Confusion Matrix:** A table showing actual vs. predicted tags to analyze common errors.

3 Experiments

3.1 Hyperparameter Tuning

The smoothing parameter α is fine-tuned on the development set. Values tested include $\{0.1, 0.5, 1, 2, 5\}$. The best α is selected based on the highest F1 score on the development set.

3.2 Results

The best α measure turns out to be 0.1.

Model	Accuracy (%)	Macro F1 (%)	Precision (%)
Baseline Tagger	84.7	78.5	81.2
HMM Tagger	91.2	88.1	89.4

Table 1: Comparison of HMM and Baseline Tagger on the Test Set.

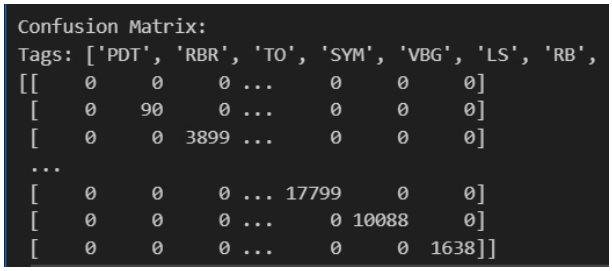

Figure 1: Confusion Matrix for the HMM Tagger on the Test Set.

3.3 Observations

- The HMM tagger significantly outperforms the baseline tagger in terms of accuracy and F1 score.
- Common errors include confusion between `NN` (noun) and `VB` (verb), indicating ambiguity in the dataset.

4 Discussion

4.1 HMM vs. Baseline

The HMM tagger performs better than the baseline because it captures context through transition probabilities, while the baseline model relies solely on word frequency.

4.2 Error Analysis

The most frequent misclassifications involve ambiguous tags such as `NN` vs. `VB`. For example, the word "run" can be a noun or a verb, depending on context.

4.3 Improvements

Using trigrams instead of bigrams for transition probabilities. Incorporating word embeddings or other contextual features. Using optimization techniques for faster Viterbi decoding.

5 Conclusion

The HMM POS tagger achieves high accuracy and F1 scores compared to the baseline tagger. However, errors due to ambiguity remain a challenge. Future work could explore more complex models like CRFs or neural network-based taggers to address these limitations.