Report for Assignment - 1

This report was prepared by four team members, each contributing a specific role in the analysis of the given datasets and the development of our HMM–based Part-of-Speech tagging system:

1. Deepansh Pandey (Roll No. MT2024038)
2. Eshwar Chawda (Roll No. MT2024047)
3. Himanshu Shrivastava (Roll No. MT2024059)
4. Shivam Pandey (Roll No. MT2024145)

Our primary reference files are **TRAIN.csv** and **TEST.csv** which contain tokenized sentences labeled with POS tags. The goal of our project is to use Viterbi algorithm to train, validate, and evaluate an HMM for accurate POS classification, using these datasets as the main source of training and testing data. The sections that follow describe our methods, findings, and concluding remarks. We have attached our Python Notebook with this report zip.

1. Preprocessing

We began by collecting and examining our tagged text data. Our primary goal was to understand how many tokens (words) and how many unique tags (NOUN, VERB, ADJ, etc.) we had in total. This led us to perform frequency counts for each tag, which quickly showed that certain classes—particularly NOUN and VERB—accounted for a large fraction of the data, while tags like INTJ or PRT were underrepresented.

Given that some tags were much rarer than others, we checked for potential imbalances that could affect model performance. We also looked for out-of-vocabulary words in the test set that weren't present in the training data. Identifying these OOV words helped us anticipate problems the model might face in correctly tagging unseen tokens.

We took note of any tokens that can appear in multiple parts of speech—for example, "like," which can be a verb ("I like apples"), a preposition ("something like a tree"), or even a filler word in informal speech. Observations like these would prove relevant later, since our approach must handle a certain level of linguistic ambiguity.

The data also contained many stopwords that could be removed, as well as various words that could be converted to lowercase for consistency.

2. Viterbi

Once the dataset was ready for modeling, we implemented the Viterbi algorithm to perform sequence decoding. We first established dynamic programming matrices: one to store the maximum log-likelihood for each state (i.e., each tag) at each position in the sentence, and another for backpointers to reconstruct the most likely tag sequence.

We made sure to handle unknown words by assigning them a low default probability, preventing numerical underflow with small constants or log-space arithmetic. Because Viterbi accumulates probabilities across a sentence, any small misestimation early on could lead to large downstream errors.

During development, we ran the Viterbi procedure on a few example sentences to confirm that it tagged common words correctly—particularly boundary cases like the first word in the sentence, where an initial probability distribution is used instead of a transition from a previous tag. After verifying these steps, we integrated the code to evaluate our entire test set.

3. HMM

We used a standard Hidden Markov Model (HMM) framework to estimate the probability of each tag sequence. First, we derived emission probabilities from the training set by counting how often a given tag produces a given word. Then, we computed transition probabilities to capture which tag is most likely to follow another (for instance, ADJ often precedes NOUN).

To initialize the model, we incremented all counts by one (Laplace smoothing) to avoid zero probabilities for any unseen (tag, word) or (tag, tag) pair. This provided a more robust set of estimates, especially for tags that appeared infrequently.

Finally, we stored the resulting probabilities as log-probabilities to avoid numerical instability—important because multiplying many small probabilities over long sentences can quickly approach floating-point limits. Armed with these emission and transition distributions, we handed the data off to be decoded by the Viterbi procedure.We used a standard Hidden Markov Model (HMM) framework to estimate the probability of each tag sequence. First, we derived emission probabilities from the training set by counting how often a given tag produces a given word. Then, we computed transition probabilities to capture which tag is most likely to follow another (for instance, ADJ often precedes NOUN).

To initialize the model, we incremented all counts by one (Laplace smoothing) to avoid zero probabilities for any unseen (tag, word) or (tag, tag) pair. This provided a more robust set of estimates, especially for tags that appeared infrequently.

Finally, we stored the resulting probabilities as log-probabilities to avoid numerical instability—important because multiplying many small probabilities over long sentences can quickly approach floating-point limits. Armed with these emission and transition distributions, we handed the data off to be decoded by the Viterbi procedure.

4. Confusion Matrix

After we obtained predicted tags on the test set, we generated the confusion matrix to visualize the model's performance. On the rows, we placed the actual (ground truth) tags, and on the columns, we placed the predicted tags. Because the matrix was normalized, each row sums to 1, making it easier to compare how well each tag was correctly identified versus how it was misclassified.

Many tags, particularly NOUN, VERB, and ADJ, had strong diagonal scores near 0.9 or higher. This indicates that the HMM + Viterbi pipeline handled common word classes effectively. However, the matrix revealed potential confusion between PROPN (proper nouns) and NOUN—likely due to capitalization inconsistencies or ambiguous contexts. There were also occasional mix-ups between AUX and VERB, which makes sense because words like "have," "do," or modal auxiliaries can appear in full verb contexts.

By analyzing the largest off-diagonal cells, we discovered that rare tags—INTJ, PRT, X—were more prone to errors. This finding led us to suggest targeted data augmentation and additional linguistic rules to better handle ambiguous or low-frequency categories.

5. Conclusion

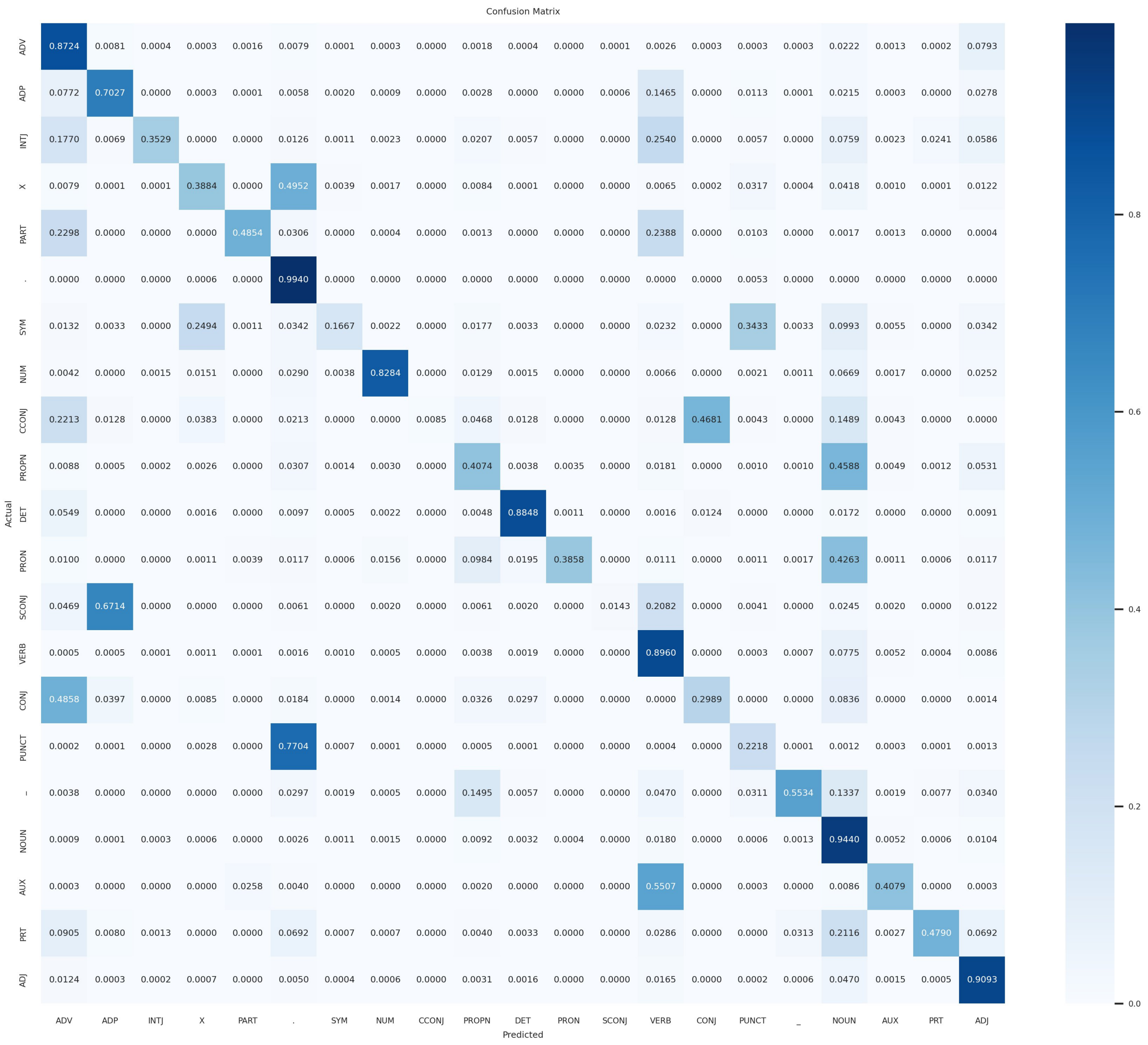
Overall, the **HMM** approach with **Viterbi decoding** proved quite robust for most frequent tags, achieving high accuracy on classes like NOUN, VERB, and ADJ. The **confusion matrix** confirmed that while common categories are well-modeled, the system often confuses certain classes—particularly those that are inherently ambiguous or underrepresented in the training corpus.

Results

Initial accuracy: 68.78%
Initial Dataset: TRAIN (1056973 rows) & TEST (1318475 rows)
Initial Train-Test time: 8.5 Minutes

After Lowercasing and Stopword Removal.
Final Accuracy: 85.89%
Final Dataset: TRAIN (633439 rows) & TEST (790497 rows)
Final Train-Test time: 5 Minutes

Confusion Matrix:



	ADV	ADP	INTJ	X	PART	.	SYM	NUM	CCONJ	PROPN	DET	PRON	SCONJ	VERB	CONJ	PUNCT	_	NOUN	AUX	PRT	ADJ
ADV	0.8724	0.0081	0.0004	0.0003	0.0016	0.0079	0.0001	0.0003	0.0000	0.0018	0.0004	0.0000	0.0001	0.0026	0.0003	0.0003	0.0003	0.0222	0.0013	0.0002	0.0793
ADP	0.0772	0.7027	0.0000	0.0003	0.0001	0.0058	0.0020	0.0009	0.0000	0.0028	0.0000	0.0000	0.0006	0.1465	0.0000	0.0113	0.0001	0.0215	0.0003	0.0000	0.0278
INTJ	0.1770	0.0069	0.3529	0.0000	0.0000	0.0126	0.0011	0.0023	0.0000	0.0207	0.0057	0.0000	0.0000	0.2540	0.0000	0.0057	0.0000	0.0759	0.0023	0.0241	0.0586
X	0.0079	0.0001	0.0001	0.3884	0.0000	0.4952	0.0039	0.0017	0.0000	0.0084	0.0001	0.0000	0.0000	0.0065	0.0002	0.0317	0.0004	0.0418	0.0010	0.0001	0.0122
PART	0.2298	0.0000	0.0000	0.0000	0.4854	0.0306	0.0000	0.0004	0.0000	0.0013	0.0000	0.0000	0.0000	0.2388	0.0000	0.0103	0.0000	0.0017	0.0013	0.0000	0.0004
.	0.0000	0.0000	0.0000	0.0006	0.0000	0.9940	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0053	0.0000	0.0000	0.0000	0.0000	0.0000
SYM	0.0132	0.0033	0.0000	0.2494	0.0011	0.0342	0.1667	0.0022	0.0000	0.0177	0.0033	0.0000	0.0000	0.0232	0.0000	0.3433	0.0033	0.0993	0.0055	0.0000	0.0342
NUM	0.0042	0.0000	0.0015	0.0151	0.0000	0.0290	0.0038	0.8284	0.0000	0.0129	0.0015	0.0000	0.0000	0.0066	0.0000	0.0021	0.0011	0.0669	0.0017	0.0000	0.0252
CCONJ	0.2213	0.0128	0.0000	0.0383	0.0000	0.0213	0.0000	0.0000	0.0085	0.0468	0.0128	0.0000	0.0000	0.0128	0.4681	0.0043	0.0000	0.1489	0.0043	0.0000	0.0000
PROPN	0.0088	0.0005	0.0002	0.0026	0.0000	0.0307	0.0014	0.0030	0.0000	0.4074	0.0038	0.0035	0.0000	0.0181	0.0000	0.0010	0.0010	0.4588	0.0049	0.0012	0.0531
DET	0.0549	0.0000	0.0000	0.0016	0.0000	0.0097	0.0005	0.0022	0.0000	0.0048	0.8848	0.0011	0.0000	0.0016	0.0124	0.0000	0.0000	0.0172	0.0000	0.0000	0.0091
PRON	0.0100	0.0000	0.0000	0.0011	0.0039	0.0117	0.0006	0.0156	0.0000	0.0984	0.0195	0.3858	0.0000	0.0111	0.0000	0.0011	0.0017	0.4263	0.0011	0.0006	0.0117
SCONJ	0.0469	0.6714	0.0000	0.0000	0.0000	0.0061	0.0000	0.0020	0.0000	0.0061	0.0020	0.0000	0.0143	0.2082	0.0000	0.0041	0.0000	0.0245	0.0020	0.0000	0.0122
VERB	0.0005	0.0005	0.0001	0.0011	0.0001	0.0016	0.0010	0.0005	0.0000	0.0038	0.0019	0.0000	0.0000	0.8960	0.0000	0.0003	0.0007	0.0775	0.0052	0.0004	0.0086
CONJ	0.4858	0.0397	0.0000	0.0085	0.0000	0.0184	0.0000	0.0014	0.0000	0.0326	0.0297	0.0000	0.0000	0.0000	0.2989	0.0000	0.0000	0.0836	0.0000	0.0000	0.0014
PUNCT	0.0002	0.0001	0.0000	0.0028	0.0000	0.7704	0.0007	0.0001	0.0000	0.0005	0.0001	0.0000	0.0000	0.0004	0.0000	0.2218	0.0001	0.0012	0.0003	0.0001	0.0013
_	0.0038	0.0000	0.0000	0.0000	0.0000	0.0297	0.0019	0.0005	0.0000	0.1495	0.0057	0.0000	0.0000	0.0470	0.0000	0.0311	0.5534	0.1337	0.0019	0.0077	0.0340
NOUN	0.0009	0.0001	0.0003	0.0006	0.0000	0.0026	0.0011	0.0015	0.0000	0.0092	0.0032	0.0004	0.0000	0.0180	0.0000	0.0006	0.0013	0.9440	0.0052	0.0006	0.0104
AUX	0.0003	0.0000	0.0000	0.0000	0.0258	0.0040	0.0000	0.0000	0.0000	0.0020	0.0000	0.0000	0.0000	0.5507	0.0000	0.0003	0.0000	0.0086	0.4079	0.0000	0.0003
PRT	0.0905	0.0080	0.0013	0.0000	0.0000	0.0692	0.0007	0.0007	0.0000	0.0040	0.0033	0.0000	0.0000	0.0286	0.0000	0.0000	0.0313	0.2116	0.0027	0.4790	0.0692
ADJ	0.0124	0.0003	0.0002	0.0007	0.0000	0.0050	0.0004	0.0006	0.0000	0.0031	0.0016	0.0000	0.0000	0.0165	0.0000	0.0002	0.0006	0.0470	0.0015	0.0005	0.9093

In these results, the confusion matrix illustrates how often each actual POS tag (rows) is classified as each predicted tag (columns). Most common tags—such as NOUN, ADJ, and VERB—show high diagonal values, indicating strong model accuracy. Rarer or ambiguous tags, including PRT and INTJ, exhibit more confusion, likely due to fewer examples or overlapping linguistic functions. Notable misclassifications include occasional mix-ups between PROPN and NOUN, as well as AUX and VERB, which arise from shared grammatical contexts. Overall, the matrix confirms robust performance on frequent classes while highlighting improvement opportunities for less frequent or context-dependent tags.