CS 626 Assignment 1(POS Tagging) Report

-Drumil Trivedi(Roll No.170020016:)
-Sarvesh Mehtani (Roll No.:170050107)
-Anshul Nasery(Roll No.:170070015)

Method 1: **HMM**
Test Accuracy achieved: 94.2%
Method 2: **SVM**
Test Accuracy achieved: 86.4%
Method 3: **Bi-LSTM**
Test Accuracy achieved: 96.5 %

Note: All accuracies are reported using 5-fold cross validation.

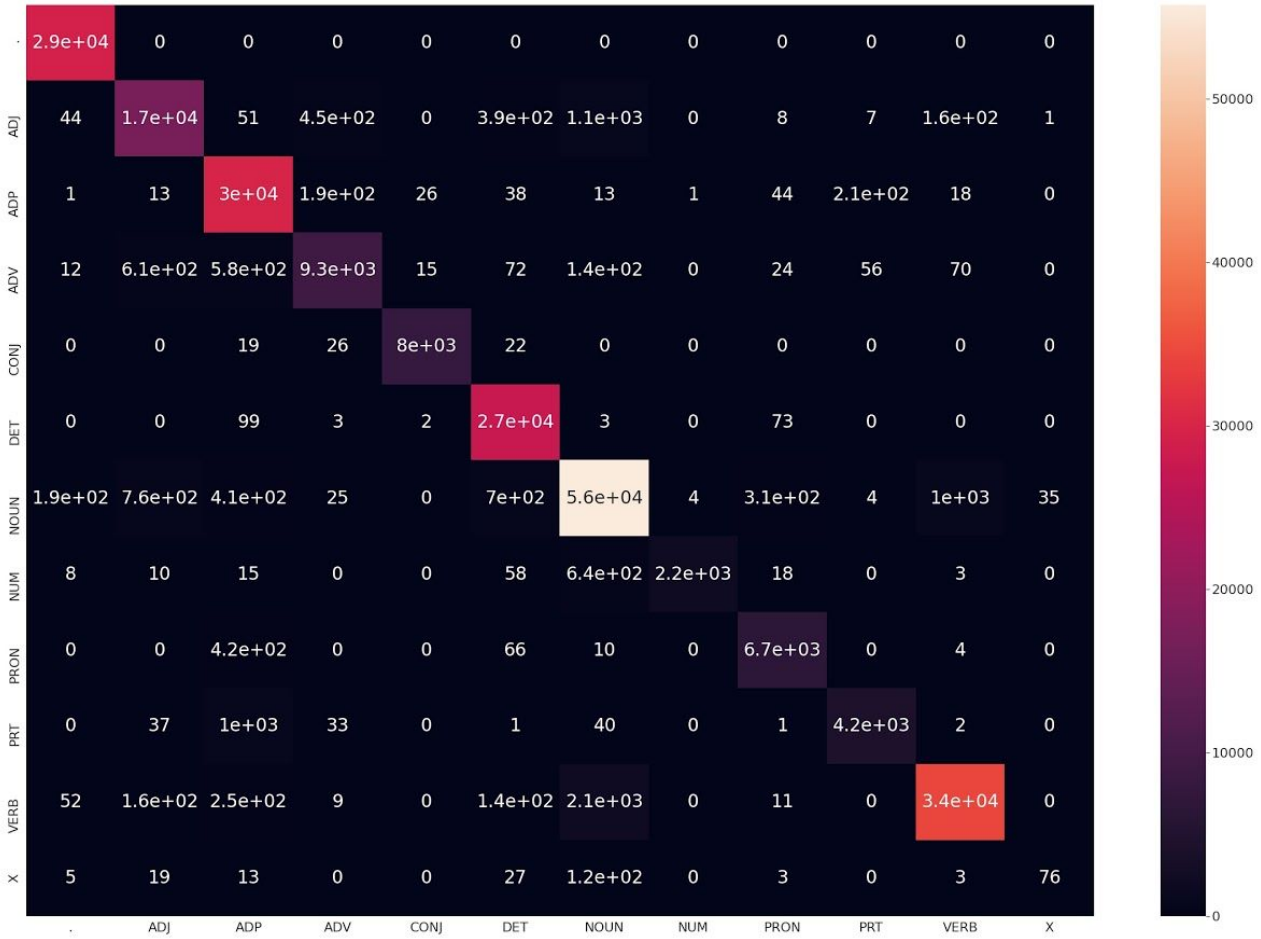
Details of each method

HMM:
The HMM is trained on the tagged Brown corpus. We append a beginning of sentence tag to each sentence. We do not apply any discounting or interpolation while computing the bigram probabilities. However, we apply add-one discounting while computing the emission probabilities to account for OOV words. We use Viterbi decoding during inference, starting from the start-of-sentence tag.]

Results -
Class Wise Precision, Recall, f-1 score:

	.	ADJ	ADP	ADV	CONJ	DET	NOUN	NUM	PRON	PRT	VERB	X
Precision	0.99	0.90	0.90	0.93	0.99	0.94	0.94	1.00	0.91	0.94	0.96	0.73
Recall	1.00	0.88	0.98	0.86	0.99	0.99	0.95	0.81	0.94	0.84	0.94	0.25
F1	1.00	0.88	0.92	0.89	0.99	0.96	0.93	0.89	0.95	0.83	0.96	0.21

Confusion Matrix -



Analysis-

1. We find that the unknown tag has a poor recall. This is due to the fact that the HMM has a high number of false negatives for this tag, due to lot of OOV words. We find that the SVM handles this aspect better due to its use of embeddings which lends a semantic meaning to OOV words.
2. The highest confusion can be seen in the noun <-> verb and adv <-> adj tags, similar to SVM. However, the second confusion is less than that in the SVM, due to previous word's tag being given.
3. Further, NOUN is the most frequent wrong prediction by the model, because of the class imbalance in the data, (i.e. NOUN occurs most frequently in language). This can be seen across models as well.

SVM:

We use a multi-class SVM trained with a multi-class hinge loss. This loss tries to maximize the prediction between the positive and negative class for a given sample. The features used are-

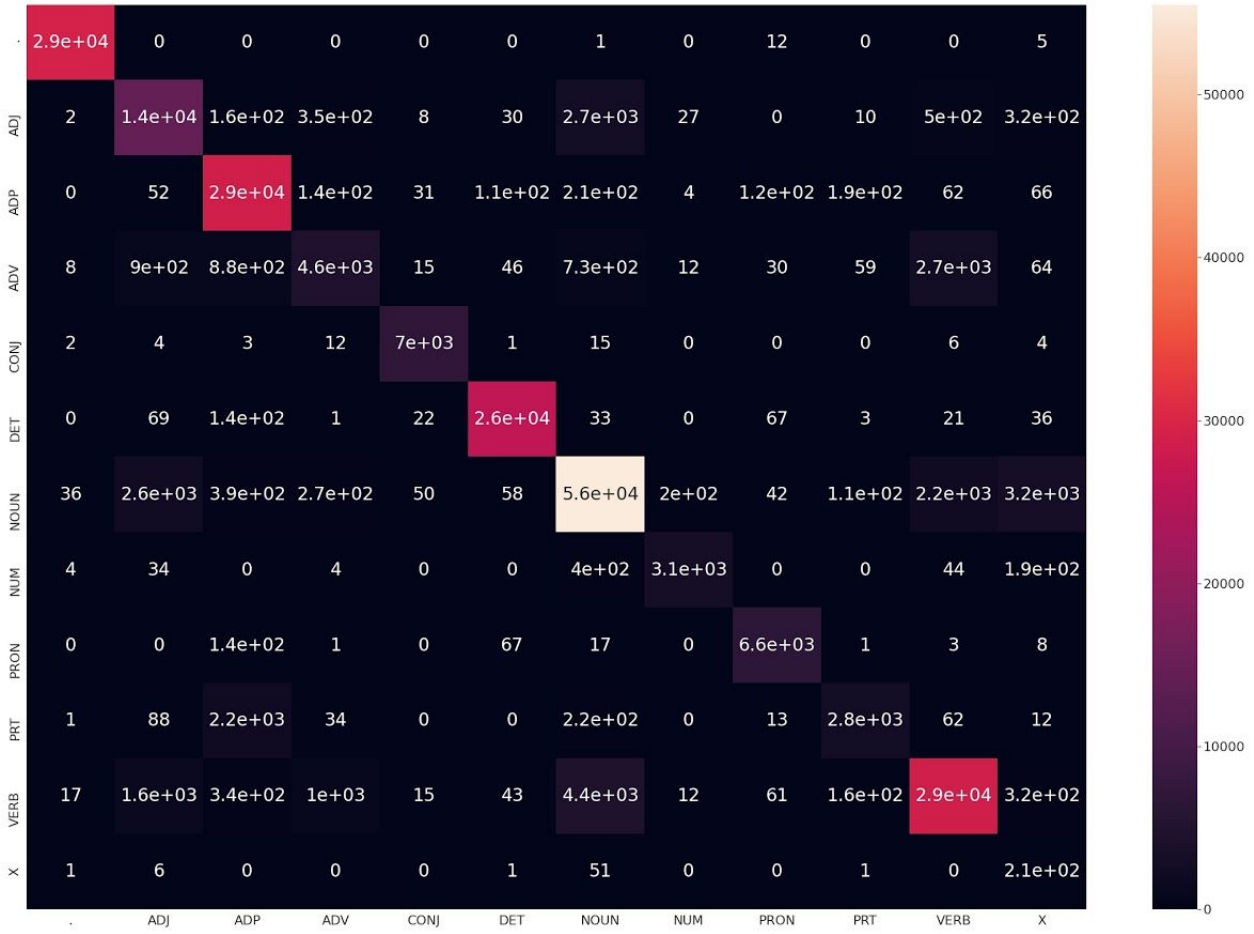
1. Prefix in the word (one-hot vector of length 40)
2. Suffix in the word (one-hot vector of length 30)
3. Capitalization
4. Word2Vec embeddings of the word, two words before the word and two words after.

Notice that the only sequentiality in the algorithm appears from the embeddings. Further, we used 100-dim embeddings trained using the Brown corpus. We also tried using pre-trained GloVe vectors of 100-dimension, which led to a slight increase in the accuracy. The final feature vector size for each word was 572. We implemented the SVM algorithm in both torch and numpy, but found that the torch variant was easier to train and debug.

Class Wise Precision, Recall, f-1 score: -

	.	ADJ	ADP	ADV	CONJ	DET	NOUN	NUM	PRON	PRT	VERB	X
Precision	1.00	0.73	0.87	0.72	0.98	0.99	0.86	0.92	0.95	0.84	0.84	0.05
Recall	1.00	0.78	0.97	0.46	0.99	0.99	0.86	0.82	0.97	0.51	0.78	0.78
F1	1.00	0.75	0.92	0.56	0.99	0.99	0.86	0.87	0.96	0.64	0.81	0.09

Confusion Matrix-



Analysis-

1. We find that word embeddings played the most significant role in predictions (through ablations). This might be due to sparsity of other metrics, and due to the rich nature of word embeddings.
2. Substituting Word-2-vec for GloVe trained on larger vocabulary makes the recall on unknown tags much higher (0.79 vs 0.20). This is due to the fact that a smaller vocabulary has a much higher OOV tags, leading to a much higher number of words being tagged unknown, and a lower recall.
3. We also notice that ADJ are often confused for ADV and vice-versa. They are also the classes the model has the worst metrics on. The HMM performs better on these classes. We suspect this occurs because HMM has direct information about the tag of the previous word, and this helps in these two tags' cases, since they depend heavily on which tag comes before/after them (NOUN/VERB), while their own embeddings might be similar. We also find that prefix/suffix are important to predict these classes.
4. The other common confusion for the model is VERB <-> NOUN.

5. For the NUM tag, it is often confused as NOUN. This happens because their embeddings might be similar across larger pieces of text.

Tell me when do

Bi-LSTM:

Architecture Details:

```
  Embedding Layer( Dimension: Vocab Size, Embedding dimension=100)
  Bi-LSTM Layer(100, 128, num_layers=2, dropout=0.25, bidirectional=True)
  Dropout(p=0.25)
  Linear Layers(in_features=256, out_features=13)
)
Loss: Cross Entropy Loss
```

Other settings tried:

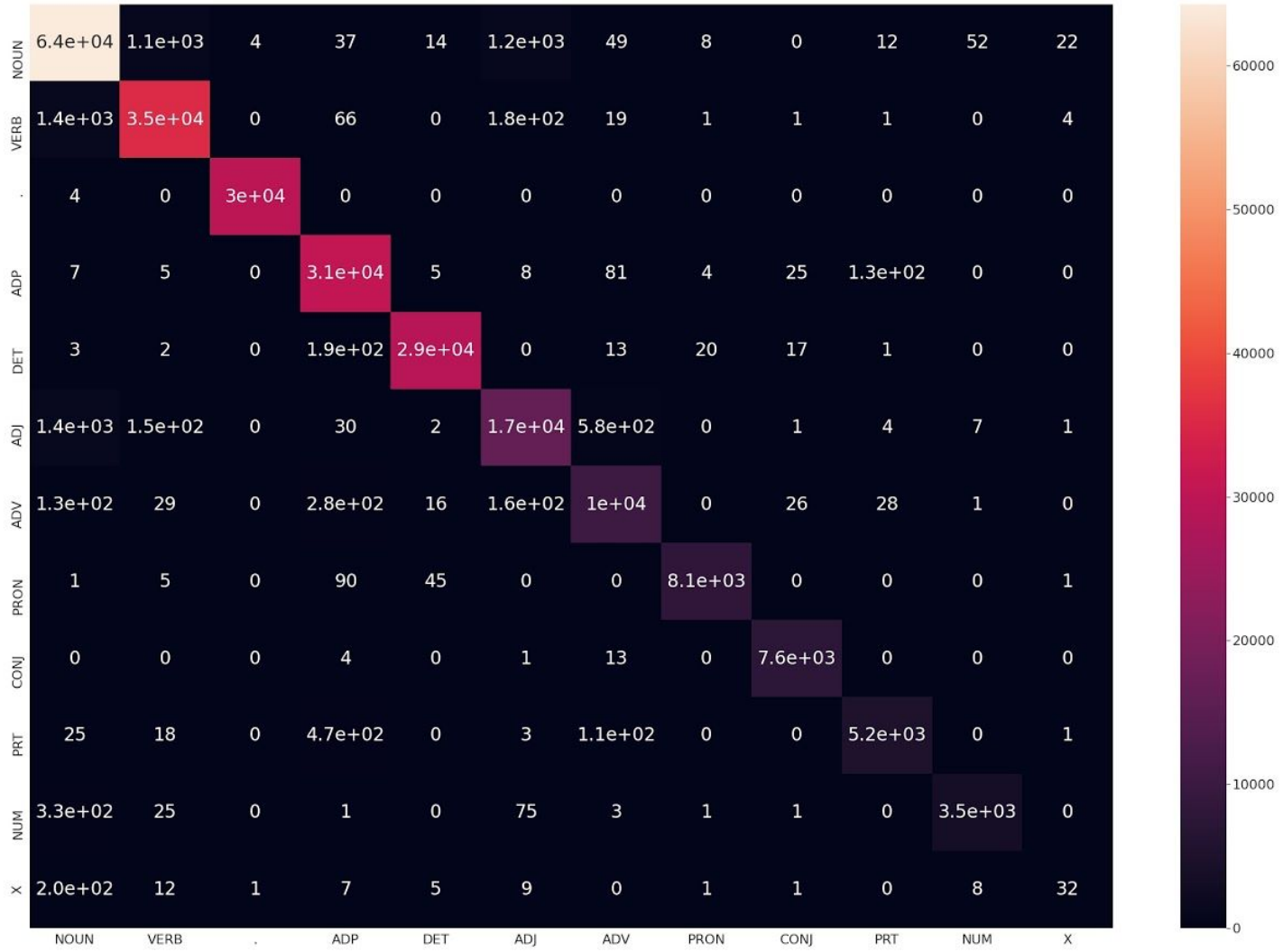
- We also tried setting embedding dimension to 300 and then initialising the embeddings with pretrained Word2Vec embedding.. We finally didn't do that as that resulted in an average test accuracy of 95.6% which is approximately 0.9% less than the accuracy obtained with the above architecture.
- We also experimented with using a 2-layer Feed forward network with ReLu activation over the Bi-LSTM but no increase in accuracy was seen, hence was removed from the final model.

Class Wise Precision, Recall, F1, Accuracy:

	NOUN	VERB	.	ADP	DET	ADJ	ADV	PRON	CONJ	PRT	NUM	X
Precision	0.948	0.962	0.999	0.963	0.997	0.911	0.922	0.995	0.990	0.966	0.981	0.516
Recall	0.962	0.954	0.999	0.991	0.991	0.886	0.938	0.982	0.997	0.891	0.889	0.113
F1	0.955	0.958	0.999	0.977	0.994	0.898	0.930	0.989	0.994	0.927	0.933	0.187
Accuracy	0.975	0.987	0.999	0.994	0.998	0.984	0.993	0.999	0.999	0.996	0.997	0.998

Note: Class wise Accuracy is not a good metric in general for Multi-class problems

Confusion Matrix:



Analysis:

- Using embeddings didn't help much as
  - There were about 40% tokens in the vocabulary of the corpus which were not present in word2vec embeddings. This lead to the model having the learn these OOV words. Hence it didn't give much advantage over learning the whole embedding matrix from scratch
  - As this was a small corpus and the model was able to learn it very well with a less (100d) embedding dimension itself instead of using a higher dimension(300d).

Strengths and weaknesses:

- The model tends to classify a large number of tokens as NOUN(as the Precision is low) because of the class imbalance tilted towards NOUN.
- Model generally performs really well for **.** (SYM) tags and the (DET) Determiner tags.
- The model tries to classify the **X** (No class) tokens to one of the other classes (as can be seen from the very low recall).
- I feel there is very high scope of improvement wrt ADJ (Adjective), ADV(Adverb), and X tags as can be seen from their low precision and recall.
- Model does miss out in identifying token with NUM tag and misclassifies them
- PRT tokens are often misclassified as ADP (adpositions) and ADV(adverb).
- Model also gets confused between ADP and ADV tag in general, misclassifying one class into the other.

Learnings: For me(Sarvesh), it was one of the first times writing a pytorch model from scratch and training it. I learned a lot about using pytorch and TorchText libraries in the process of doing the Bi-LSTM part of this assignment.