Assignment-1: POS Tagging

Aditya Sharma 170050043
Yash Jain 170050055
Debabrata Mandal 170050073

Performance Comparison of the three models

Method	Test Accuracy
SVM	83.5%
HMM	96.57%
Bi-LSTM	97.00%

References:

- **SVM**
  1. https://www.nltk.org/book/ch02.html
  2. http://cs229.stanford.edu/notes/cs229-notes3.pdf - SVM
  3. http://cs229.stanford.edu/materials/smo.pdf - SMO
  4. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
  5. https://people.csail.mit.edu/dsontag/courses/ml16/slides/lecture6_notes.pdf - Pegasos Algorithm
  6. https://xavierbourretsicotte.github.io/SVM_implementation.html - Using cvxopt
  7. https://rare-technologies.com/word2vec-tutorial/

- **HMM**
  1. https://www.nltk.org/book/ch02.html
  2. https://www.scss.tcd.ie/Martin.Emms/MLforNLP/hmm_slides_slides.pdf
  3. Assignment and lecture material from offering of AIML course in 2019 Fall term.

- **Bi-LSTM**
  1. https://pytorch.org/

SVM

Accuracy :

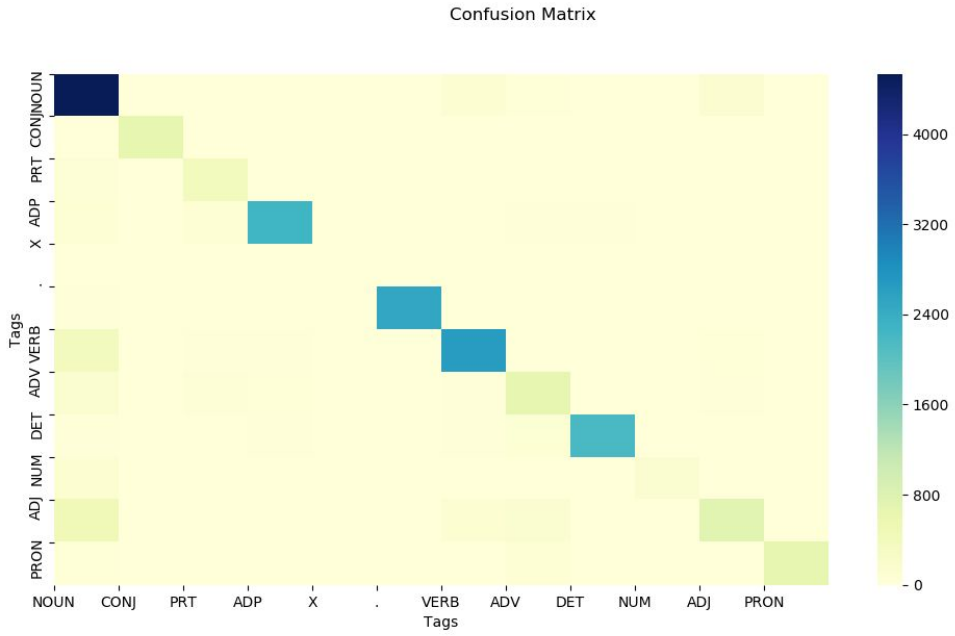
The model's accuracy as a whole in predicting tags from input sentences. Trained on 100 sentences (equivalent to ~ 2000 word vectors) and tested on 1000 test sentences (equivalent to ~ 20000 words) (time to train model 5-6 mins).

Training set has been kept larger since we want to demonstrate that no overfitting takes place even when training on a small dataset (since larger datasets take huge time to train).

(Feature length - 768)

	Our training accuracy	Our test accuracy	sklearn's SVC training accuracy	sklearn's SVC test accuracy
1st fold	99.95 %	83.05 %	98.96 %	86.52 %
2nd fold	98.64 %	84.82 %	98.75 %	86.29 %
3rd fold	99.61 %	83.69 %	98.06 %	86.04 %
4th fold	98.42 %	83.87 %	98.75 %	86.29 %
5th fold	99.08 %	83.98 %	98.01 %	86.68 %
Avg accuracy	**~ 99 %**	**~ 83.5 %**	**~ 98.5 %**	**~ 86 %**

Confusion Matrix : For final fold



per-POS accuracy :
Averaged across every fold

.	99.99 %
NOUN	90.37 %
VERB	96.49 %
NUM	92.44 %
DET	99.35 %
PRON	99.5 %
PRT	98.94 %
ADP	97.93 %
X	0 %
ADV	97.52 %
ADJ	92.16 %
CONJ	99.87 %

Strengths of using SVMs for POS tagging:
Since we are using Linear SVM for POS tagging, the feature vectors need to be quite robust in order for the algorithm to converge under relatively tight margin bounds.

Thus for training feature vectors of size approx 450-500, usually suffices to get an average accuracy above 80%. The strengths with this method lies in the fact that if the feature vectors are not correct (i.e. incorrectly chose features) then the accuracy will be very low. But if features are chosen properly then even a small subset (approx 200 sentences) will result in an accuracy of above 80% on a test dataset of 10000 sentences.

This is quite useful since for languages for which a large corpus is not available. SVMs show just capturing the essential features from the language will give quite good results.

Further for small datasets where the vocabulary size is small feature vector construction using a one-hot like scheme could result in far better performance due to the better separation (hence easier to find the hyperplane) in higher dimensions.

Weakness witnessed for SVM for POS tagging:

One primary weakness of using Linear SVM for POS tagging lies in the fact that if we are unable to design the feature vectors correctly the impact of this on the algorithm could be huge since it tries to classify data using a hyperplane (which can be relaxed to some extent by using a soft margin classifier or using kernels).

Another weakness is the time required to train the model. Since we are solving a quadratic optimization problem here, scaling with an increasing number of examples for training is not feasible due to the fact with more examples the size of the feature vector increases (faster when using one hot encoding). This puts a constraint on the size of the dataset to be trained upon. (There are alternatives like using kernels or solving the quadratic optimization problem by using an algorithm called SMO which solves the dual version of the problem, allowing for kernels also, but we left it due to its complicated implementation.)

To train 2000 examples, we need to solve an optimization problem involving a 2000x2000 matrix with each vector and that too for each tag individually since it is a multi classification problem.
This makes the training too time consuming for this model.

Error Analysis:

Overall Model Test Accuracy -

- Lower accuracy than sklearn.svm's SVC since they use probabilistic estimates rather than binary class values to choose the best tag class among all valid tags.
- Increasing the training dataset size only increases the time taken to train the model without any notable increase in accuracy (not improvement either on adjusting the regularization parameter for large datasets).
- Due to smaller dataset rare tag words of 'X' are not seen leading to a zero probability for that tag class.
- Prefix and suffix play an important role as features, hugely impacting the accuracy of the model.
- Using word embeddings (from word2vec) only poorly increases the accuracy. Using a one-hot encoding of words hugely increases the accuracy (above 95 %) but leads to an extremely large feature vector (of size of 10K+).
- Using word embeddings (from word2vec) for previous and next word make no significant impact on the test accuracy.
- Using an optimal regularization (for soft margins) of '20' gives the best results. This ensures almost exact fitting since we are working with a very small dataset.

per-POS test accuracy -

- Although the number of true positives is maximum for NOUN, due to presence of higher number of false positives and false negatives lower overall test accuracy.
- Higher number of false negatives for NOUN, VERB, ADV etc. The reason is an indicative of the fact that Linear SVMs do not handle outliers well. Popular POS tag words like for NOUN etc. will tend to more outliers than the rest of the tags thus adding up to the problem.

Error Pattern :-

- Case 1:
  - For the following sentence, Noun and Adjective are mis predicted even in the presence of capitalisation. This signifies a single bit in a vector (for capitalisation) of size 768 does not successfully predict a tag.
  - Ex :- The_DET_DET integrity_NOUN_NOUN .... be_VERB_VERB ,_._. as_ADP_ADP the_DET_DET Charter_NOUN_ADJ puts_VERB_VERB ...
- Case 2:
  - Maximum discrepancy happens with the pairs (NOUN, ADJ) & (NOUN, VERB). (Visualised on an output of 400 predictions with 14 (NOUN, VERB) mismatch and 10 (NOUN, ADJ) mismatch.
  - Due to a linear model the separation between NOUN and VERB is not properly learnt.

Learnings:

1. Theoretical & mathematical background behind SVM.
2. Converting the SVM problem to its dual version and finding a solution for that.
3. Sequential Minimal Optimization algorithm
4. Pegasos algorithm using SGD for solving SVM problem
5. Using cvxopt library to get the solution to the SVM problem
6. Calculate word embeddings using word2vec
7. Word stemming algorithms like Snowball stemmer
8. Efficient feature extraction from english words

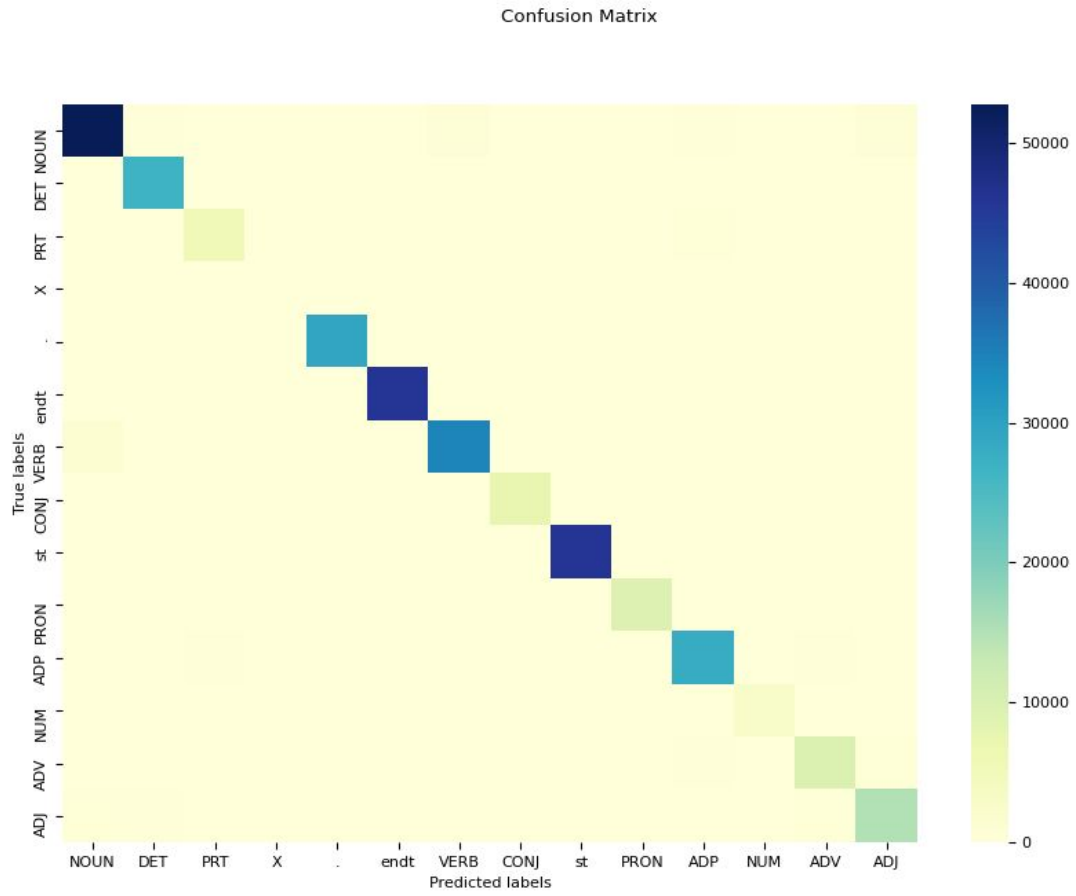
HMM

Accuracy :

The HMM was implemented for 5-fold cross validation and the following are the reported accuracies:

	Accuracy
1st fold	95.94 %
2nd fold	96.87 %
3rd fold	96.55 %
4th fold	96.43 %
5th fold	97.06 %
Avg accuracy	**96.57 %**

Confusion Matrix : Plotted for the last fold



Per-pos accuracy:

For the last cross-validation fold

DET	98.68 %
VERB	95.13 %
NOUN	95.29 %
ADP	96.58 %
NUM	91.81 %
.	99.89 %
PRT	90.31 %
X	57.18 %
ADV	89.53 %
PRON	98.37 %
ADJ	91.79%
CONJ	99.48%

Strengths of using HMMs for POS tagging:

1. The learning algorithm is efficient. Decoding is also efficient -> Complexity: $O(n^2L)$
2. Faster training as compared to SVM and Bi-LSTM.
3. No need for feature engineering because we don't have to design features.
4. Very less parameters as compared to SVM.

Weakness witnessed for HMM for POS tagging:

1. HMM assumes Markov condition holds, which may not always be true. Therefore this is limited by first order markov property.
2. The effect of far off terms on is not considered while deciding the tags for the given word.
3. It is sometimes difficult to tag words where the uncommon tag is shadowed by the dominant tag in Emission probabilities.(example given below)

Learnings:

There was a lot to be learnt from HMM. Firstly, all the theoretical aspects like probability, Likelihood Maximization and Markov principle.Then we also learnt the implementation and the aspects of Viterbi algorithm, and how using Dynamic programming we can reduce complexity to $O(n^2L)$. We also understood the backtracking mechanism to find the best path for the POS tagging.

Then there were other aspects to learn like between which tags there can be error and why that occurs(examples mentioned below). Aspects of Machine Learning like accuracy, recall and confusion matrix were also understood during the assignment.

We also got exposed to various new libraries like NLTK which has a very good database of corpuses. We also got to know about various corpuses and tagsets available and some hindsight thinking about how the tagset was chosen and engineered.

Error Analysis:

- Before using any smoothing techniques, the model was giving an accuracy of around 75%. Then after using a simple Linear Interpolation technique, the accuracy jumped to 95% +. The following gives an idea about the interpolation technique used:

$$P_{smooth}(w_k|Tag_i) = (1-\lambda) * P(w_k|Tag_i) + \frac{\lambda}{|V|}$$

```
Running 1 iteration
Accuracy 74.65649509278704
Running 2 iteration
Accuracy 74.69807166210715
Running 3 iteration
Accuracy 75.8498606549398
Running 4 iteration
Accuracy 75.53833159437572
Running 5 iteration
Accuracy 75.51219331062076
```

```
Running 1 iteration
Accuracy 95.93384912292794
Running 2 iteration
Accuracy 96.30020725143618
Running 3 iteration
Accuracy 96.59385879328781
Running 4 iteration
Accuracy 96.89182558417102
Running 5 iteration
Accuracy 97.06547038873909
```

- There were errors in individual tags which couldn't be avoided because the probabilities are trained from the given data set.

Actual Tag	Predicted Tag	Sample Sentence	Explanation
NOUN	VERB	Sentence: cap would find him and take care of him . Actual Tags: NOUN VERB VERB PRON CONJ VERB NOUN ADP PRON . Given Tags: NOUN VERB VERB PRON CONJ VERB VERB ADP PRON .	Notice here that "care" also behaves as a VERB in english and hence emission probability will be high for VERB. Also there are many instances of VERB followed by VERB for example: "he was playing", "they were dancing". Hence, VERB to VERB transition probability is also high.
VERB	NOUN	Sentence: there was much shouting and screaming . Actual Tags: PRT VERB ADJ NOUN CONJ NOUN . Given Tags: PRT VERB ADJ VERB CONJ VERB .	Notice that "shouting" and "screaming" also behave as continuous tense forms of the verbs and hence emission probability will be high for VERB.
NOUN	ADJ	Sentence: around the billiard tables were always at least a couple of dozen beribboned generals . Actual Tags: ADP DET NOUN NOUN VERB ADV ADP ADJ DET NOUN ADP NOUN ADJ NOUN . Actual Tags: ADP DET ADJ NOUN VERB ADV ADP ADJ DET NOUN ADP NOUN ADP NOUN .	This is one of the many cases where the phrase is a COMPOUND NOUN and the Model incorrectly classifies it as an ADJECTIVE followed by a Noun. This is also a problem because of high transition probabilities of ADJ to NOUN and DET to ADJ. Whenever there is a word which occurs rarely, smoothing will reduce the advantage of Emission Probability.

- Also in case of smoothing, if you remove smoothing for **'st'** and **'endt'** tags, you see a slight improvement in accuracy.

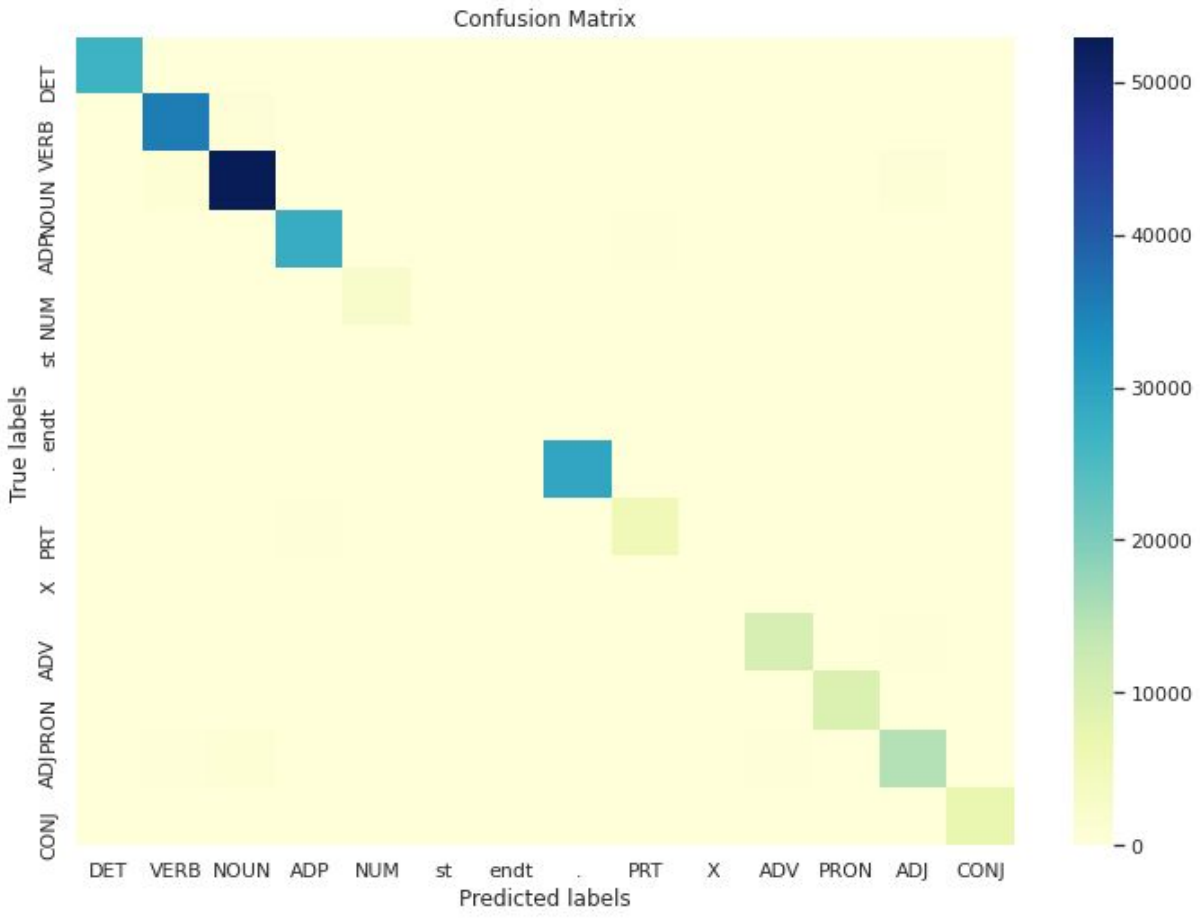
Bi-LSTM

Accuracy :

The Bi-LSTM model was implemented for 5-fold cross-validation and the following are the reported accuracies:

	Accuracy
1st fold	96.98 %
2nd fold	96.91 %
3rd fold	96.99 %
4th fold	97.07 %
5th fold	97.03 %
Avg accuracy	**97.0 %**

Confusion Matrix: Plotted for the last fold



Per-pos accuracy: For the last cross-validation fold

DET	99 %
VERB	97 %
NOUN	97 %
ADP	98 %
NUM	88 %
.	100 %
PRT	95 %
X	10 %
ADV	93%
PRON	98 %
ADJ	91 %
CONJ	100 %

Strengths of using Bi-LSTM for POS tagging:

1. Achieved the highest accuracy among the other methods
2. Does not require any type of feature engineering, model handles everything on its own
3. Method is able to utilise the context of far-off words from both sides of the sentence giving better POS tagging

Weakness witnessed for Bi-LSTM for POS tagging:

1. Slowest in training compared to other methods
2. Per-POS tag accuracy varies quite a lot with tags (depends on support/data) and hence, is less robust compared to HMM or SVM
3. Does not have a good explainability of the reason behind performance compared to other methods
4. Hidden dimension of Bi-LSTM: Tuning this hyperparameter showed us the fragility of our system with respect to data available. For eg. when kept to 64: the average accuracy was 95% but the accuracy of

POS-tag ‘X’ was 0%. While when increased to 256, the average accuracy came out to be 97% with the accuracy of POS-tag ‘X’ being 10%.

Error Analysis:

1. To+infinitive: In English language, the word “to” can be followed by either a VERB or a NOUN depending on its use as a preposition or adposition. But the Bi-LSTM seems to have been biased in predicting the word followed by “to” as VERB always. Here are some examples to bolster the hypothesis.
   Eg. From the fusty panaceas of spinach, eggs and prunes. The US has progressed to curds, concentrates and capsules.
   Actual tags: ADP DET ADJ NOUN ADP NOUN . NOUN CONJ NOUN . DET NOUN VERB VERB ADP NOUN . NOUN CONJ NOUN
   Predicted tags: ADP DET NOUN NOUN ADP NOUN . NOUN CONJ VERB . DET NOUN VERB VERB PRT VERB . NOUN CONJ NOUN

   Much to Damon Runyon’s amazement, as well as my own. I got along splendidly with hetman.
   Actual tags: ADJ ADP NOUN NOUN NOUN …
   Predicted tags: ADJ ADP VERB ADJ NOUN …

2. ADJ vs NOUN: In longer sentences, it has been observed that NOUN is often tagged as ADJ, increasing the number of false negatives of ADJ and thereby its per-pos accuracy.
   Eg. A coat of paste wax or a rubdown with a piece of wax paper will protect the polished surface of the table.
   Actual tags: DET NOUN ADP NOUN NOUN CONJ DET NOUN ADP DET NOUN ADP NOUN NOUN VERB VERB DET VERB NOUN ADP DET NOUN .
   Predicted tags: DET NOUN ADP NOUN NOUN CONJ DET ADJ ADP DET NOUN ADP NOUN NOUN VERB VERB DET VERB NOUN ADP DET NOUN .

   The private detective is militant against injustice, a humorous and ironic explorer of the underworld.
   Actual tags: DET ADJ NOUN VERB NOUN ADP NOUN . DET ADJ CONJ ADJ NOUN ADP DET NOUN .
   Predicted tags: DET ADJ NOUN VERB ADJ ADP NOUN . DET ADJ CONJ ADJ NOUN ADP DET NOUN .

3. Input embeddings: Earlier we tried to give one-hot vector of words as input to the model but due to vocabulary being large (~43000), the model dimensions

became unfeasible and started giving OOM errors.
To circumvent that, we tried to use binary representation of the word count in the vocabulary as an input embedding but expectedly that also didn't work.
Finally, we used the method torch.embeddings() which generate random embeddings of the vocabulary initially but update them as the model trains, giving us appropriate task-specific input embeddings.

Learnings:

From Bi-LSTM we could experience the powerful nature of Deep learning models, which while serving as a black box could model a complex problem very well. Apart from that, we learnt the usage of the deep learning framework "Pytorch" and its various caveats. Overall, it was a great learning experience to compare the performances of different methods.