NLP PROJECT
Neural POS Tagger

Prepared by: Lemmatizer

1. Pratik Tiwari (2019201023)
2. Tirth Pandit (2019201017)

Contents

1 Introduction

1.1 Project Introduction

1. POS tagging is a very basic task in NLP and Computational Linguistics. However, as many of you already know, the state of POS tagging for Indian Languages is quite a mess at the moment. This project as two sub parts.
2. Scope of this project is to use statistical classification or sequence labeling techniques such as HMMs, CRFs or basic neural network models in order to create a POS Tagger for GUJARATI language.

1.2 POS Tagging

1. POS tagging is the process of tagging the words with their categories that best suits the definition of the word as well as the context of the sentence. It is often the first step for many NLP applications.
2. Aim of the project is to create the Neural network based POS Tagger for Gujrati Language.
3. In general and Gujrati Language in particular is not a very widely explored language in NLP Tasks. Also morphological complexity of the language makes it hard to develop NLP applications around it.
4. Previous Work has been done in the area of POS Tagging in Gujrati Language which uses different approaches like HMM (Hidden Markov Models) and CRF(Conditional Random Fields).
5. Purpose of this Project also includes the comparisons between these classical approaches and Neural POS Tagger

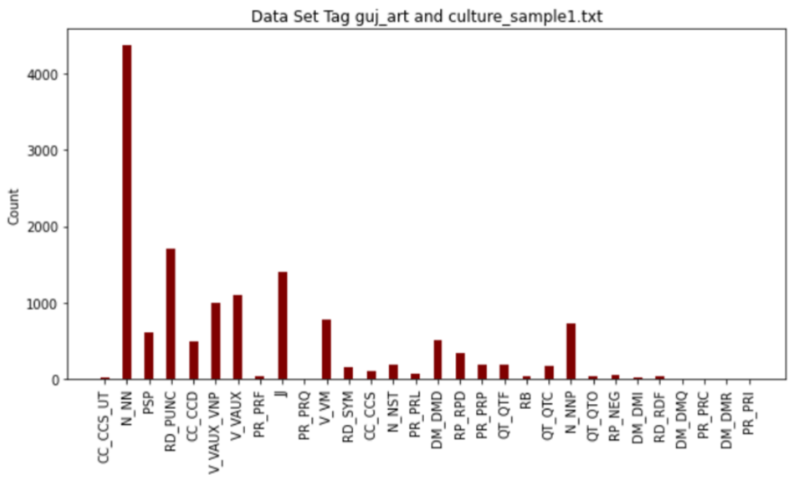
2 DataBase Details

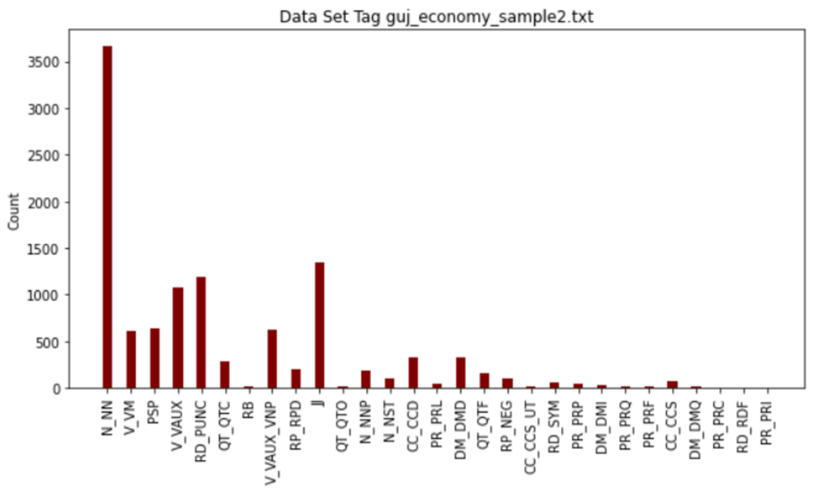
- Gujarati Monolingual Text Corpus ILCI-II 30,000 sentences of general domain. The translated sentences have been POS tagged according to BIS (Bureau of Indian Standards) tag set. `https://www.ldcil.org/Download/Tagset/LDCIL/5Gujrati.pdf`, It has eleven primary tags and similarly it divides in sub tags.

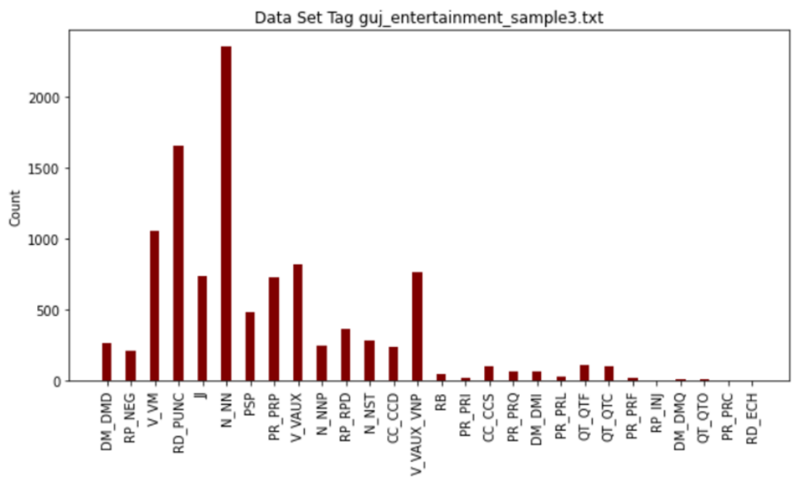
1. Art and culture sample1.txt
2. economy sample2.txt
3. entertainment sample3.txt
4. Philosophy sample4.txt

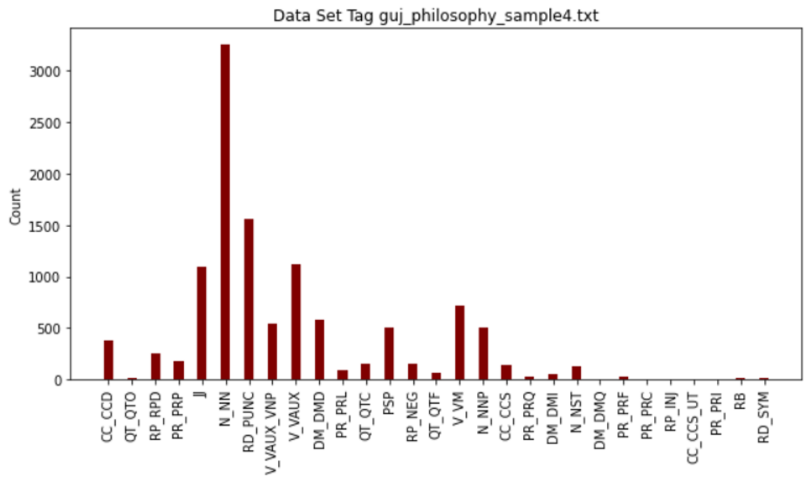
5. Religion sample5.txt
6. Science and technology sample6.txt
7. Sports sample7.txt

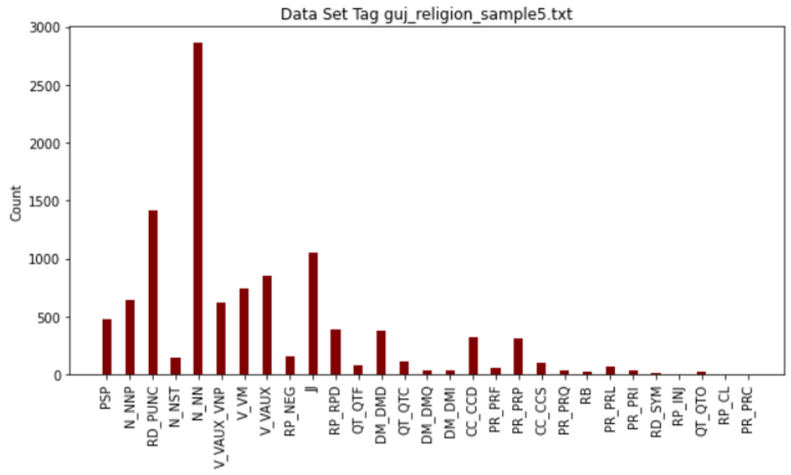
2.1 Histogram

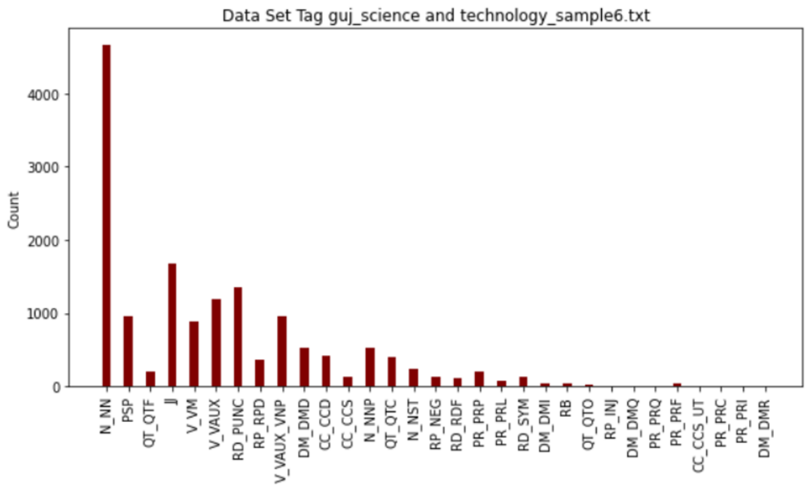


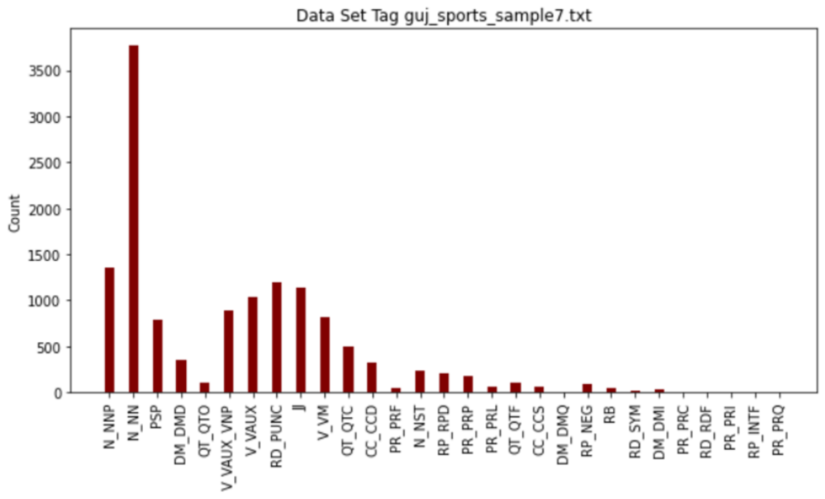



Data Set Tag guj_entertainment_sample3.txt
Count
0
500
1000
1500
2000
DM_DMD
RP_NEG
V_VM
RD_PUNC
JJ
N_NN
PSP
PR_PRP
V_VAUX
N_NNP
RP_RPD
N_NST
CC_CCD
V_VAUX_VNP
RB
PR_PRI
CC_CCS
PR_PRQ
DM_DMI
PR_PRL
QT_QTF
QT_QTC
PR_PRF
RP_INJ
DM_DMQ
QT_QTO
PR_PRC
RD_ECH


Data Set Tag guj_philosophy_sample4.txt
Count
0
500
1000
1500
2000
2500
3000
CC_CCD
QT_QTO
RP_RPD
PR_PRP
JJ
N_NN
RD_PUNC
V_VAUX_VNP
V_VAUX
DM_DMD
PR_PRL
QT_QTC
PSP
RP_NEG
QT_QTF
V_VM
N_NNP
CC_CCS
PR_PRQ
DM_DMI
N_NST
DM_DMQ
PR_PRF
PR_PRC
RP_INJ
CC_CCS_UT
PR_PRI
RB
RD_SYM


Data Set Tag guj_religion_sample5.txt
Count
0
500
1000
1500
2000
2500
3000
PSP
N_NNP
RD_PUNC
N_NST
N_NN
V_VAUX_VNP
V_VM
V_VAUX
RP_NEG
JJ
RP_RPD
QT_QTF
DM_DMD
QT_QTC
DM_DMQ
DM_DMI
CC_CCD
PR_PRF
PR_PRP
CC_CCS
PR_PRQ
RB
PR_PRL
PR_PRI
RD_SYM
RP_INJ
QT_QTO
RP_CL
PR_PRC





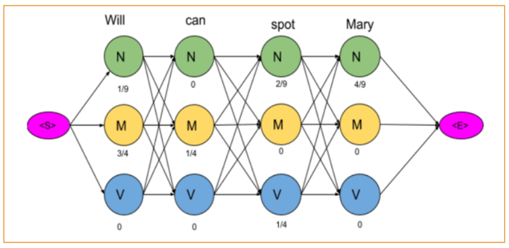
3 Baseline Mode

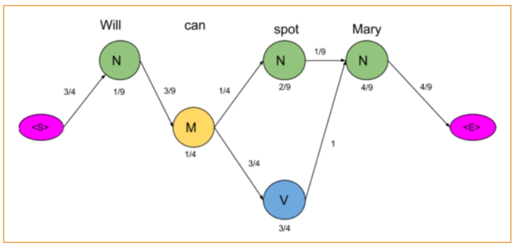
Reference Papers

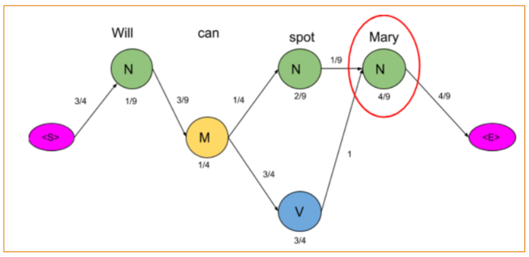
1. Part-Of-Speech Tagging for Gujarati Using Conditional Random Fields (Chirag Patel and Karthik Gali)
    - This paper describes a CRF machine learning algorithm for Gujarati Part of Speech Tagging.
    - This CRF model when provided with good features gives accuracy much better than other models.
    - The intuition presented in this paper is to convert the linguistic rules specific to Gujarati in to features which then provided to CRF. This method make use of advantages of both statistical and rule based approach.

2. A Statistical Method for Evaluating Performance of Part of Speech Tagger for Gujarati (Pooja M. Bhatt, Amit Ganatra)
    - This article presents POS tagging for Gujarati textual content the use of Hidden Markov Model. Using Gujarati text annotated corpus for training checking out statistics set are randomly separated. 80% accuracy is given by model. It discusses 2 approaches for POS tagging.
    - One is Supervised POS Tagging which requires tagged dataset this is used for studying details about rule sets, word-tag frequencies, tag gadgets,and so forth. The overall performance of supervised pos tagger fashions boom with enhancement of corpus's length.
    - Second is Unsupervised POS Tagging The Model does now not require tagged dataset. By applying computational techniques together with the transformation rules, Algorithm to generate tag clusters.

3.1 HMM Model Details



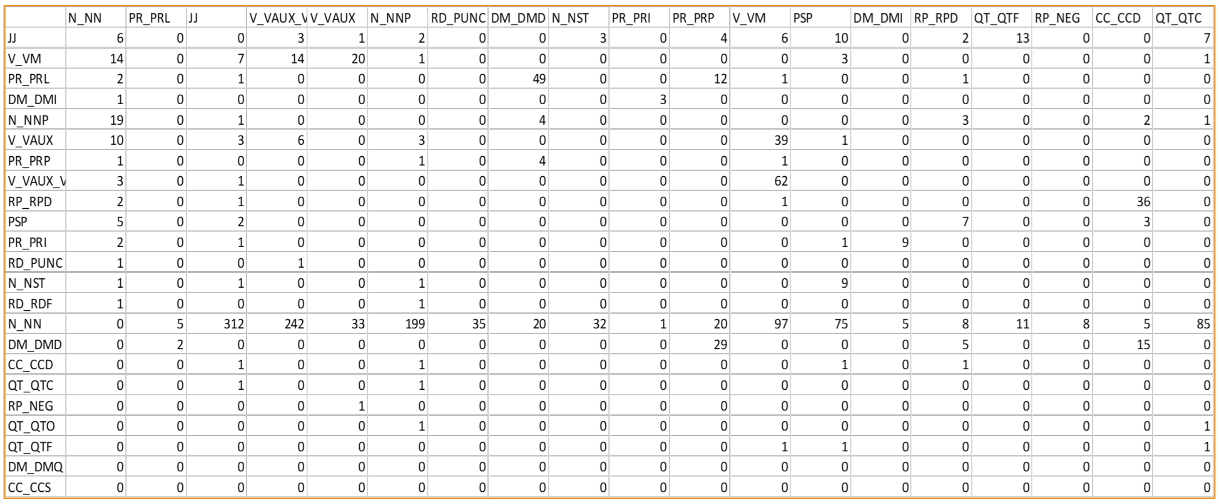




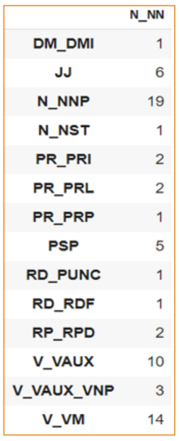
3.2 HMM Implementation Details

1. Data processing by extracting unique words from the training data.
2. Initially set count of each tag as zero and then count of occurrence of each tag.
3. Initialize and update Emission Transmission matrix.
4. For testing data compute viterbi decoding.
5. Remove the path with minimum probability and backtrack for given sentence

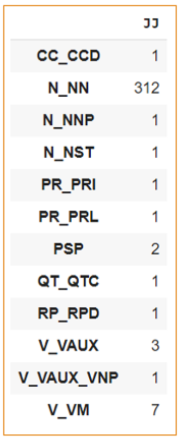
3.3 HMM Results



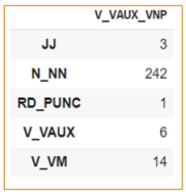
	N_NN	PR_PRL	JJ	V_VAUX_V	V_VAUX	N_NNP	RD_PUNC	DM_DMD	N_NST	PR_PRI	PR_PRP	V_VM	PSP	DM_DMI	RP_RPD	QT_QTF	RP_NEG	CC_CCD	QT_QTC
JJ	6	0	0	3	1	2	0	0	3	0	4	6	10	0	2	13	0	0	7
V_VM	14	0	7	14	20	1	0	0	0	0	0	0	3	0	0	0	0	0	1
PR_PRL	2	0	1	0	0	0	0	49	0	0	12	1	0	0	1	0	0	0	0
DM_DMI	1	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
N_NNP	19	0	1	0	0	0	0	4	0	0	0	0	0	0	3	0	0	2	1
V_VAUX	10	0	3	6	0	3	0	0	0	0	0	39	1	0	0	0	0	0	0
PR_PRP	1	0	0	0	0	1	0	4	0	0	0	1	0	0	0	0	0	0	0
V_VAUX_V	3	0	1	0	0	0	0	0	0	0	0	62	0	0	0	0	0	0	0
RP_RPD	2	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	36	0
PSP	5	0	2	0	0	0	0	0	0	0	0	0	0	0	7	0	0	3	0
PR_PRI	2	0	1	0	0	0	0	0	0	0	0	0	1	9	0	0	0	0	0
RD_PUNC	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N_NST	1	0	1	0	0	1	0	0	0	0	0	0	9	0	0	0	0	0	0
RD_RDF	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
N_NN	0	5	312	242	33	199	35	20	32	1	20	97	75	5	8	11	8	5	85
DM_DMD	0	2	0	0	0	0	0	0	0	0	29	0	0	0	5	0	0	15	0
CC_CCD	0	0	1	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0
QT_QTC	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
RP_NEG	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
QT_QTO	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
QT_QTF	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1
DM_DMQ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CC_CCS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



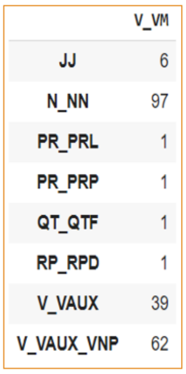
	N_NN
DM_DMI	1
JJ	6
N_NNP	19
N_NST	1
PR_PRI	2
PR_PRL	2
PR_PRP	1
PSP	5
RD_PUNC	1
RD_RDF	1
RP_RPD	2
V_VAUX	10
V_VAUX_VNP	3
V_VM	14



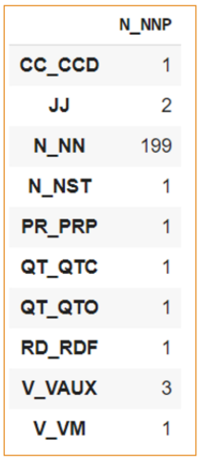
	JJ
CC_CCD	1
N_NN	312
N_NNP	1
N_NST	1
PR_PRI	1
PR_PRL	1
PSP	2
QT_QTC	1
RP_RPD	1
V_VAUX	3
V_VAUX_VNP	1
V_VM	7



	V_VAUX_VNP
JJ	3
N_NN	242
RD_PUNC	1
V_VAUX	6
V_VM	14



	V_VM
JJ	6
N_NN	97
PR_PRL	1
PR_PRP	1
QT_QTF	1
RP_RPD	1
V_VAUX	39
V_VAUX_VNP	62



	N_NNP
CC_CCD	1
JJ	2
N_NN	199
N_NST	1
PR_PRP	1
QT_QTC	1
QT_QTO	1
RD_RDF	1
V_VAUX	3
V_VM	1

3.4 CRF Model Details

1. A CRF is a sequence modeling algorithm which is used to identify entities or patterns in text, such as POS tags.
2. These models take into account previous data for which we use features which are generated from the data to feed into the CRF.
3. We define feature functions that express certain characteristic of the sequence that the data point represents , Ex : The tag sequence noun -¿ verb -¿ adjective
4. Y is the hidden state and X is the observed variable

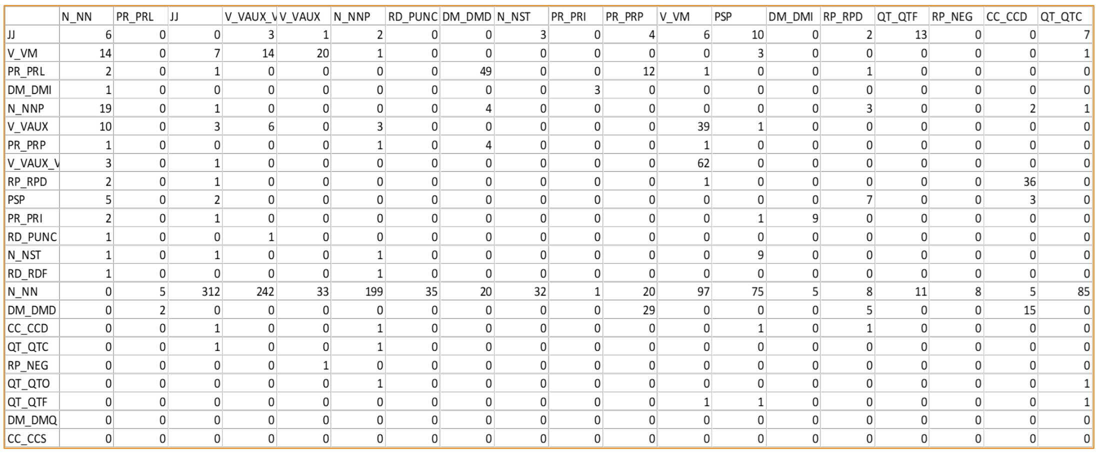
$$p(\mathbf{y}|\mathbf{x}) = \underbrace{\frac{1}{Z(\mathbf{x})}}_{\text{Normalization}} \prod_{t=1}^{T} \exp\left\{\sum_{k=1}^{K} \underbrace{\theta_k}_{\text{Weight}} \underbrace{f_k(y_t, y_{t-1}, \mathbf{x}_t)}_{\text{Feature}}\right\}$$

3.5 CRF Implementation Details

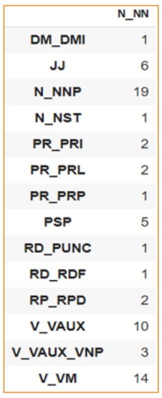
- The weight estimation is performed by maximum likelihood estimation(MLE) using the feature functions we define.

  1. Word
  2. Lowercase word
  3. Prefixes and suffixes of the word of varying lengths
  4. If word is a digit
  5. If word is a punctuation mark
  6. The length of the word - no. of characters (since shorter words are expected to be more likely to belong to a particular POS)
  7. Stemmed version of the word
  8. Features mentioned above for the previous word, the following word, and the words two places before and after

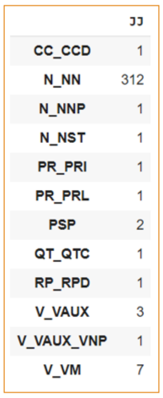
3.6 CRF Results



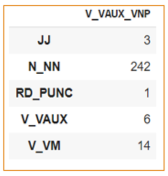
	N_NN	PR_PRL	JJ	V_VAUX_V	V_VAUX	N_NNP	RD_PUNC	DM_DMD	N_NST	PR_PRI	PR_PRP	V_VM	PSP	DM_DMI	RP_RPD	QT_QTF	RP_NEG	CC_CCD	QT_QTC
JJ	6	0	0	3	1	2	0	0	3	0	4	6	10	0	2	13	0	0	7
V_VM	14	0	7	14	20	1	0	0	0	0	0	0	3	0	0	0	0	0	1
PR_PRL	2	0	1	0	0	0	0	49	0	0	12	1	0	0	1	0	0	0	0
DM_DMI	1	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
N_NNP	19	0	1	0	0	0	0	4	0	0	0	0	0	0	3	0	0	2	1
V_VAUX	10	0	3	6	0	3	0	0	0	0	0	39	1	0	0	0	0	0	0
PR_PRP	1	0	0	0	0	1	0	4	0	0	0	1	0	0	0	0	0	0	0
V_VAUX_V	3	0	1	0	0	0	0	0	0	0	0	62	0	0	0	0	0	0	0
RP_RPD	2	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	36	0
PSP	5	0	2	0	0	0	0	0	0	0	0	0	0	0	7	0	0	3	0
PR_PRI	2	0	1	0	0	0	0	0	0	0	0	0	1	9	0	0	0	0	0
RD_PUNC	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N_NST	1	0	1	0	0	1	0	0	0	0	0	0	9	0	0	0	0	0	0
RD_RDF	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
N_NN	0	5	312	242	33	199	35	20	32	1	20	97	75	5	8	11	8	5	85
DM_DMD	0	2	0	0	0	0	0	0	0	0	29	0	0	0	5	0	0	15	0
CC_CCD	0	0	1	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0
QT_QTC	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
RP_NEG	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
QT_QTO	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
QT_QTF	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1
DM_DMQ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CC_CCS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



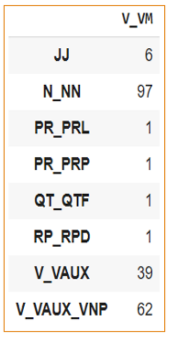
	N_NN
DM_DMI	1
JJ	6
N_NNP	19
N_NST	1
PR_PRI	2
PR_PRL	2
PR_PRP	1
PSP	5
RD_PUNC	1
RD_RDF	1
RP_RPD	2
V_VAUX	10
V_VAUX_VNP	3
V_VM	14



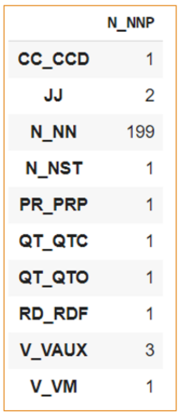
	JJ
CC_CCD	1
N_NN	312
N_NNP	1
N_NST	1
PR_PRI	1
PR_PRL	1
PSP	2
QT_QTC	1
RP_RPD	1
V_VAUX	3
V_VAUX_VNP	1
V_VM	7



	V_VAUX_VNP
JJ	3
N_NN	242
RD_PUNC	1
V_VAUX	6
V_VM	14

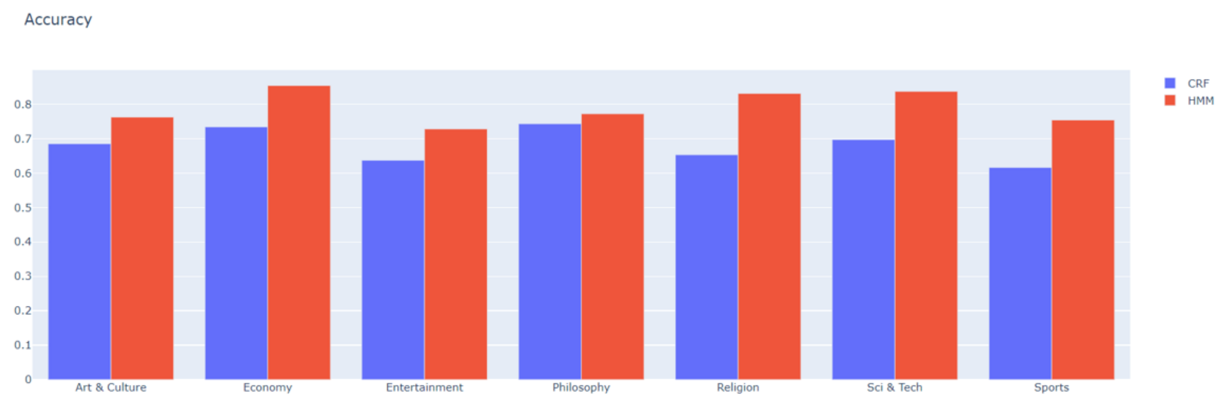


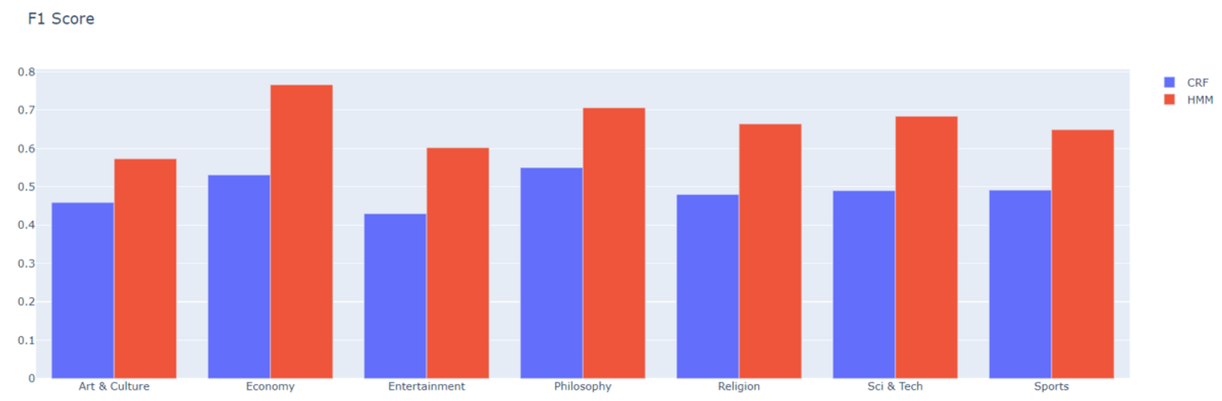
	V_VM
JJ	6
N_NN	97
PR_PRL	1
PR_PRP	1
QT_QTF	1
RP_RPD	1
V_VAUX	39
V_VAUX_VNP	62



	N_NNP
CC_CCD	1
JJ	2
N_NN	199
N_NST	1
PR_PRP	1
QT_QTC	1
QT_QTO	1
RD_RDF	1
V_VAUX	3
V_VM	1

3.7 Baseline Model Comparison





4 Neural Mode

4.1 Reference Papers

1. A Neural Network Model for Part-Of-Speech Tagging of Social Media Texts

   - This paper presents a neural network model for Part-Of-Speech (POS) tagging of User-Generated Content (UGC) such as Twitter, Facebook and Web forums. The proposed model uses both character and word level representations.
   - Character level representations are learned during the training of the model through a Convolutional Neural Network (CNN).
   - For word level representations paper describes combine several pre-trained embeddings (Word2Vec, FastText and GloVe).
   - To tackle the issue of the poor availability of annotated data on social media paper uses transfer learning . Neural model that we use for Transfer Learning experiments is based on bidirectional hierarchical Gated Recurrent Units (GRUs).

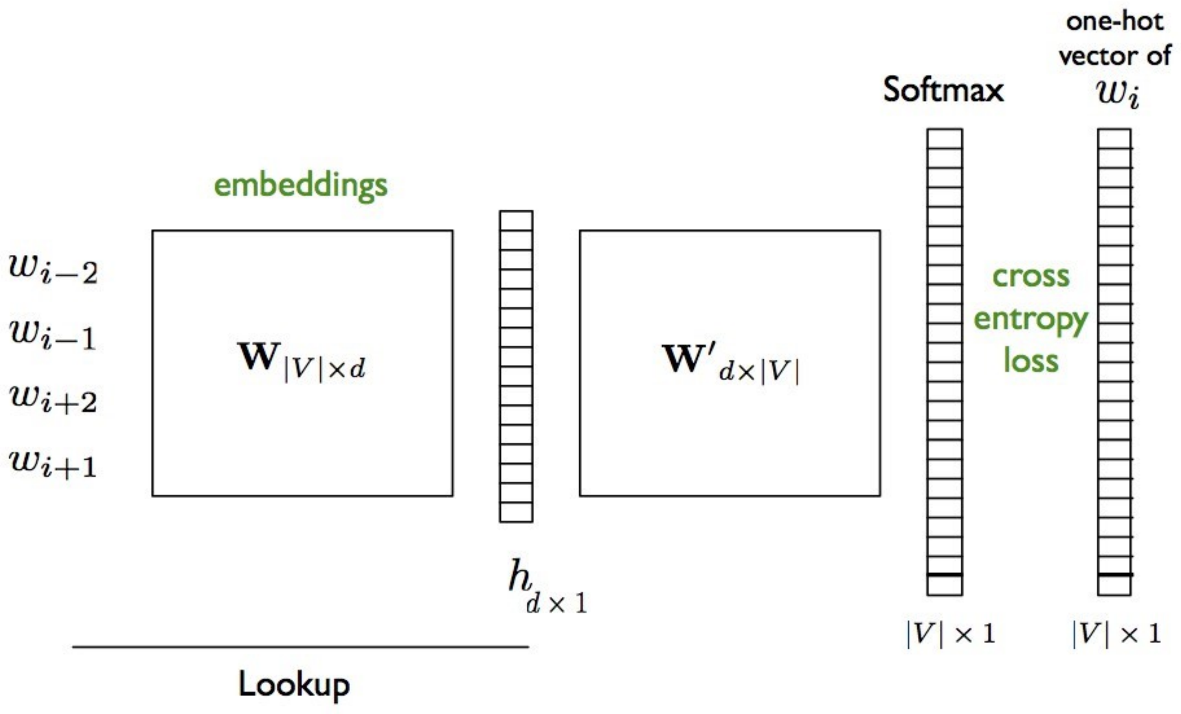
- It also uses 2 types of feature representations : character level and word level. CNN architecture is used for character level feature generation and Pretrained word embeddings are used for word level representation.

2. Part-Of-Speech Tagging using Neural network by Ankur Parikh

   - This paper presents two novel approaches of POS tagging using Neural network for Hindi language and compares them with two other machine learning approaches, HMM and CRF. In this paper, a single-neuro tagger, a Neural network based POS tagger with fixed length of context chosen is presented .
   - Then, a multineuro tagger which consists of multiple single-neuro taggers with fixed but different lengths of contexts is presented. Multineural tagger performs tagging by voting on the output of all single-neuro taggers.

3. Neural Network based Parts of Speech Tagger for Hindi

   - In this paper, Artificial Neural Network for Hindi parts of speech tagger has been used. Uses Rule base POS Tagger as the initial classifier and on top of it trains the Neural network to finally classify the words in tag classes.
   - The Rule based POS tagger tag the POS by simply using the Lexicon. The outcome of the Rule based POS Tagger is not perfect, for correction and accuracy it finally passes through the ANN based POS tagger. Here the ANN is used for pattern Recognition of corpus to identify and correct the POS tagging



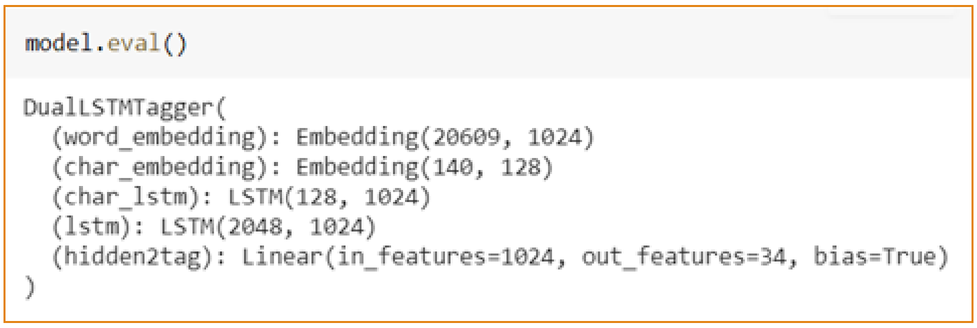
4.2 Word Embedding

1. Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation.
2. Embedding layer is learned jointly with a neural network model on a specific natural language processing task It requires that document text be cleaned and prepared such that each word is one-hot encoded.
3. The size of the vector space is specified as part of the model, such as 50, 100, or 300 dimensions.
4. The vectors are initialized with small random numbers. The embedding layer is used on the front end of a neural network and is fit in a supervised way using the Backpropagation algorithm

4.3 Model-1: LSTM

1. Model consist of the Word LSTM tagger , the Character LSTM Tagger and hidden states.
2. The LSTM takes in a sequence (words or characters), embeds the sequence into an embedding space (dimension of the space is a hyperparameter), and runs the LSTM model.
3. In current model, first the word level LSTM will take in a sequence of words and convert them into the word embedding space. Similarly, it will take the words sequentially and run the character level LSTM model, which will first take in the sequence of characters in each word and project it in the character embedding space and then run the LSTM model and take its hidden state and feed it back to the word LSTM model.
4. Using the character level hidden representation for every word as well as the word embedding, the word level model then runs LSTM on the sequence of words, and outputs the predictions for every tag.

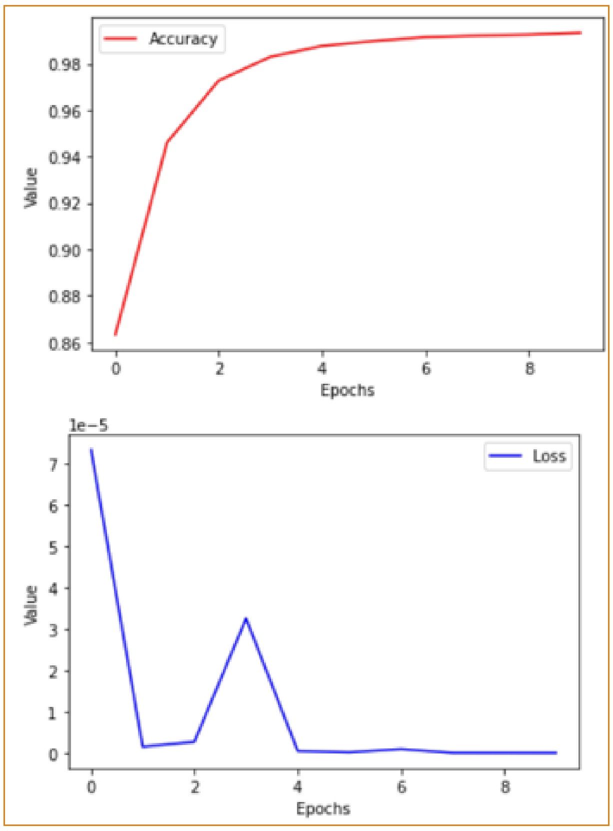
LSTM Model Details



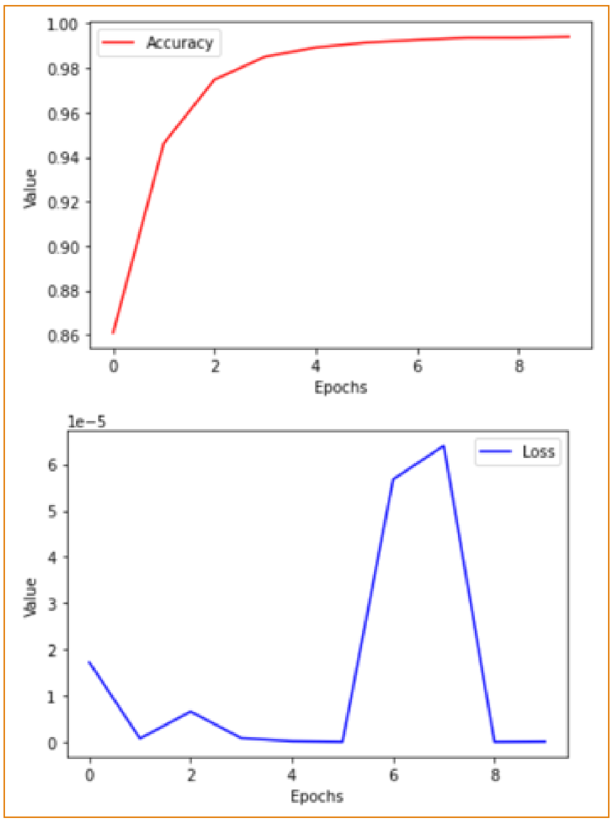
```
model.eval()

DualLSTMTagger(
  (word_embedding): Embedding(20609, 1024)
  (char_embedding): Embedding(140, 128)
  (char_lstm): LSTM(128, 1024)
  (lstm): LSTM(2048, 1024)
  (hidden2tag): Linear(in_features=1024, out_features=34, bias=True)
)
```

4.4 Hyper-Parameter Tuning Model-1

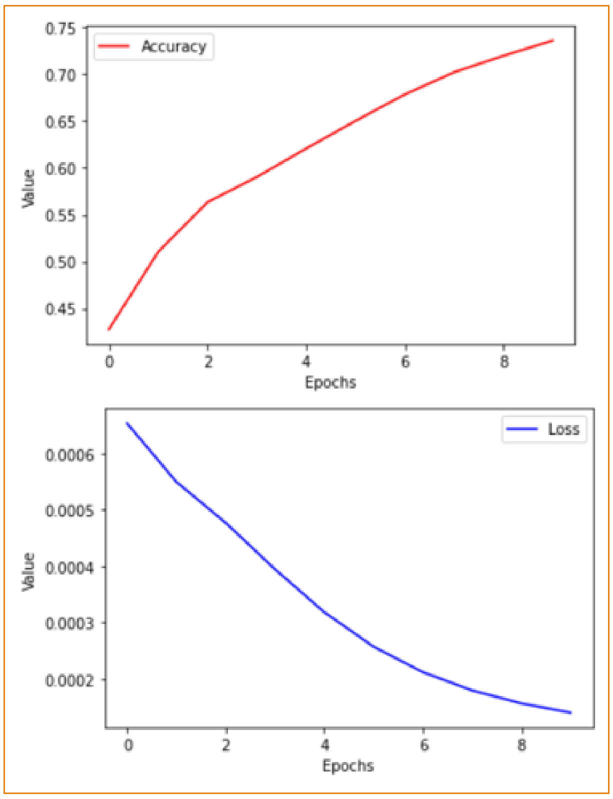
ADAM



RMSPROP

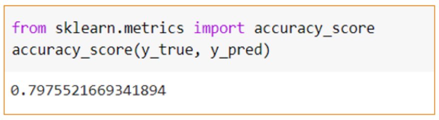


SGD



Validation Results

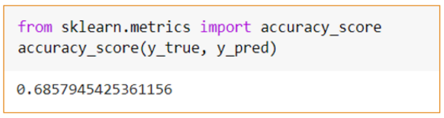
ADAM Optimizer



```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)
```

```
0.7975521669341894
```

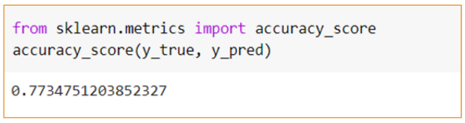
SGD Optimizer



```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)
```

```
0.6857945425361156
```

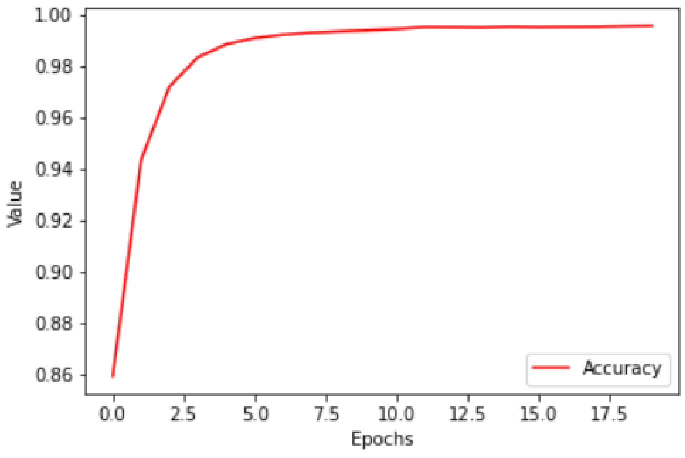
RMSProp Optimizer

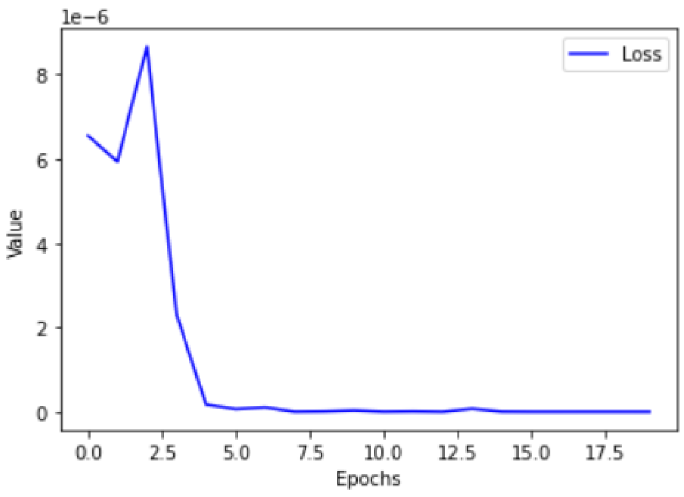


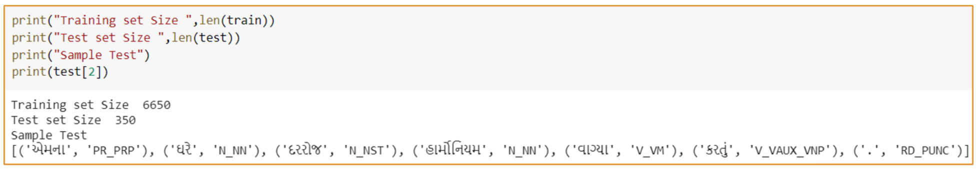
```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)
```

```
0.7734751203852327
```

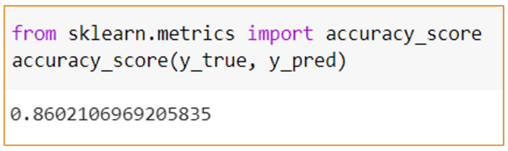
Testing Results





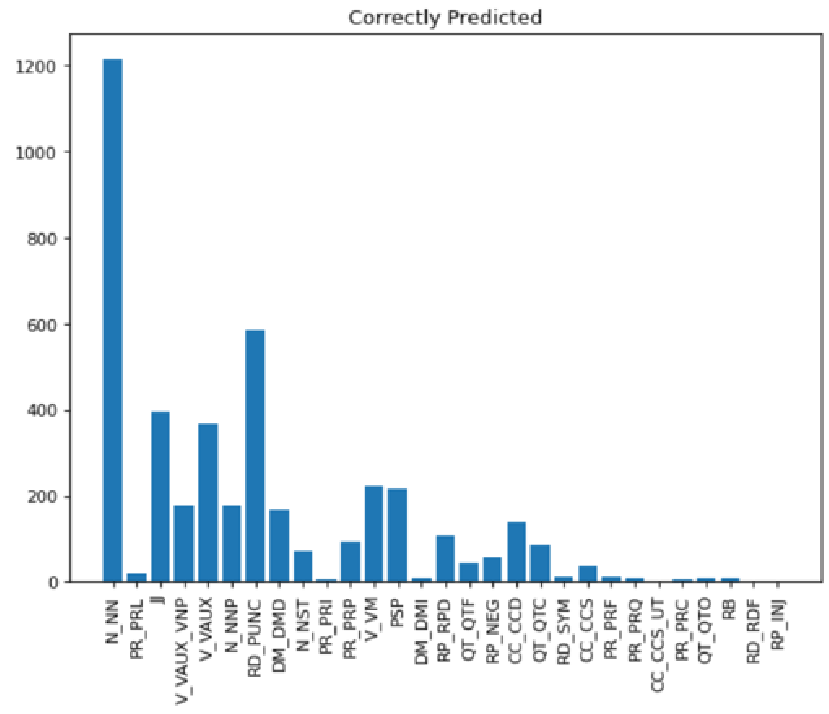


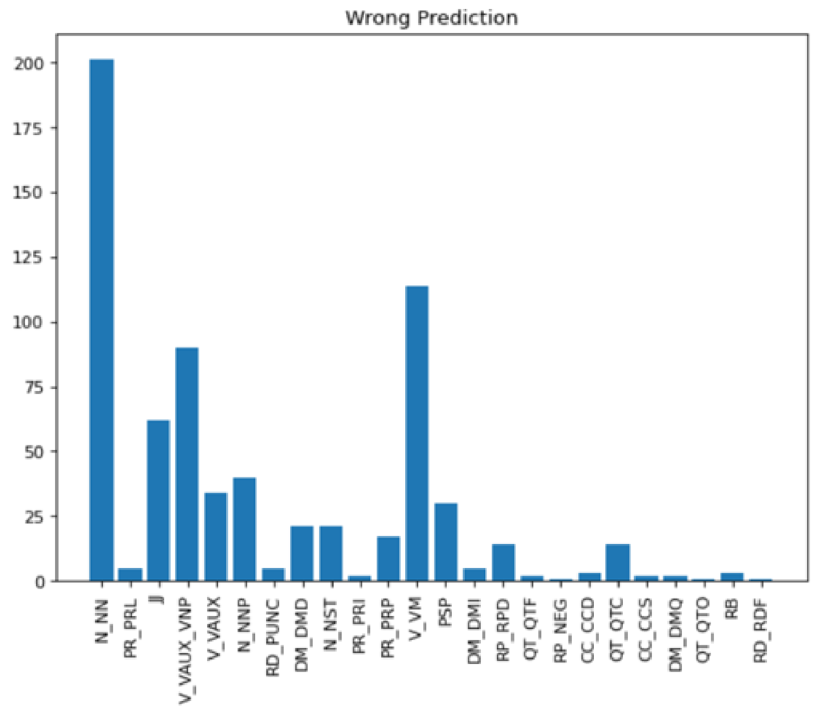
```
print("Training set Size ",len(train))
print("Test set Size ",len(test))
print("Sample Test")
print(test[2])

Training set Size  6650
Test set Size  350
Sample Test
[('એમના', 'PR_PRP'), ('ઘરે', 'N_NN'), ('દરરોજ', 'N_NST'), ('હાર્મોનિયમ', 'N_NN'), ('વાગ્યા', 'V_VM'), ('કરતું', 'V_VAUX_VNP'), ('.', 'RD_PUNC')]
```



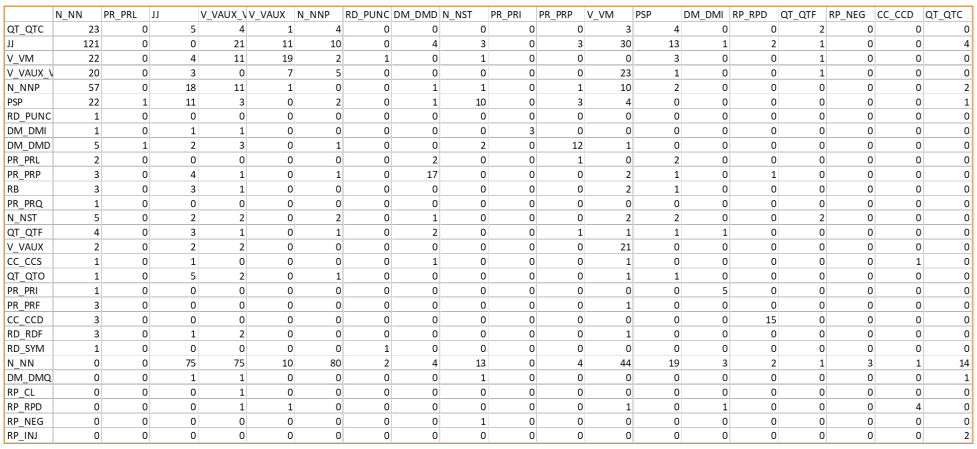
```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)

0.8602106969205835
```

4.5 Correct vs Wrong Prediction





4.6 Error Analysis

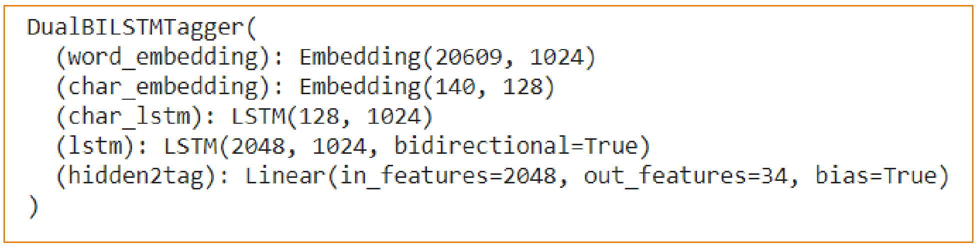


	N_NN	PR_PRL	JJ	V_VAUX_V	V_VAUX	N_NNP	RD_PUNC	DM_DMD	N_NST	PR_PRI	PR_PRP	V_VM	PSP	DM_DMI	RP_RPD	QT_QTF	RP_NEG	CC_CCD	QT_QTC
QT_QTC	23	0	5	4	1	4	0	0	0	0	0	3	4	0	0	2	0	0	0
JJ	121	0	0	21	11	10	0	4	3	0	3	30	13	1	2	1	0	0	4
V_VM	22	0	4	11	19	2	1	0	1	0	0	0	3	0	0	1	0	0	0
V_VAUX_V	20	0	3	0	7	5	0	0	0	0	0	23	1	0	0	1	0	0	0
N_NNP	57	0	18	11	1	0	0	1	1	0	1	10	2	0	0	0	0	0	2
PSP	22	1	11	3	0	2	0	1	10	0	3	4	0	0	0	0	0	0	1
RD_PUNC	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DM_DMI	1	0	1	1	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
DM_DMD	5	1	2	3	0	1	0	0	2	0	12	1	0	0	0	0	0	0	0
PR_PRL	2	0	0	0	0	0	0	2	0	0	1	0	2	0	0	0	0	0	0
PR_PRP	3	0	4	1	0	1	0	17	0	0	0	2	1	0	1	0	0	0	0
RB	3	0	3	1	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0
PR_PRQ	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N_NST	5	0	2	2	0	2	0	1	0	0	0	2	2	0	0	2	0	0	0
QT_QTF	4	0	3	1	0	1	0	2	0	0	1	1	1	1	0	0	0	0	0
V_VAUX	2	0	2	2	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0
CC_CCS	1	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0
QT_QTO	1	0	5	2	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0
PR_PRI	1	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
PR_PRF	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
CC_CCD	3	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0
RD_RDF	3	0	1	2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
RD_SYM	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
N_NN	0	0	75	75	10	80	2	4	13	0	4	44	19	3	2	1	3	1	14
DM_DMQ	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
RP_CL	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP_RPD	0	0	0	1	1	0	0	0	0	0	0	1	0	1	0	0	0	4	0
RP_NEG	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
RP_INJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2

4.7 Model-2: BILSTM

1. Bidirectional LSTM are just putting two independent LSTMs together.
2. This structure allows the networks to have both backward and forward information about the sequence at every time step Difference between unidirectional and Bidirectional LSTM is that in the LSTM that runs backward you preserve information from the future.
3. Using the two hidden states combined model is able to preserve information from both past and future at any point of time.

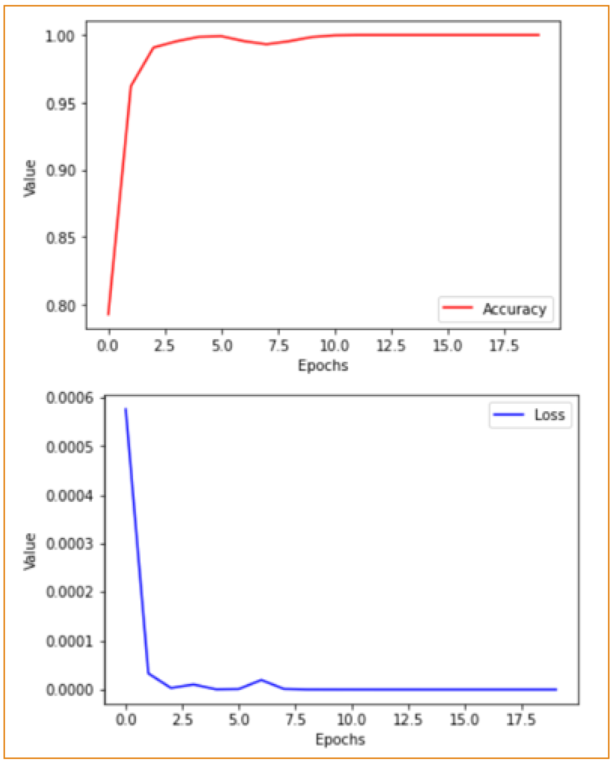
4.8 BILSTM Model Details



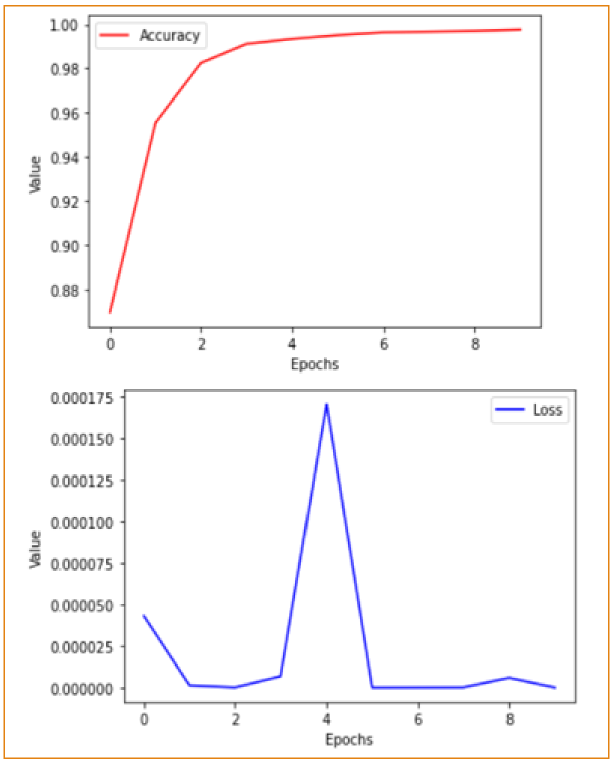
```
DualBILSTMTagger(
  (word_embedding): Embedding(20609, 1024)
  (char_embedding): Embedding(140, 128)
  (char_lstm): LSTM(128, 1024)
  (lstm): LSTM(2048, 1024, bidirectional=True)
  (hidden2tag): Linear(in_features=2048, out_features=34, bias=True)
)
```

4.9 BILSTM Hyper-Parameter Tuning Model-2

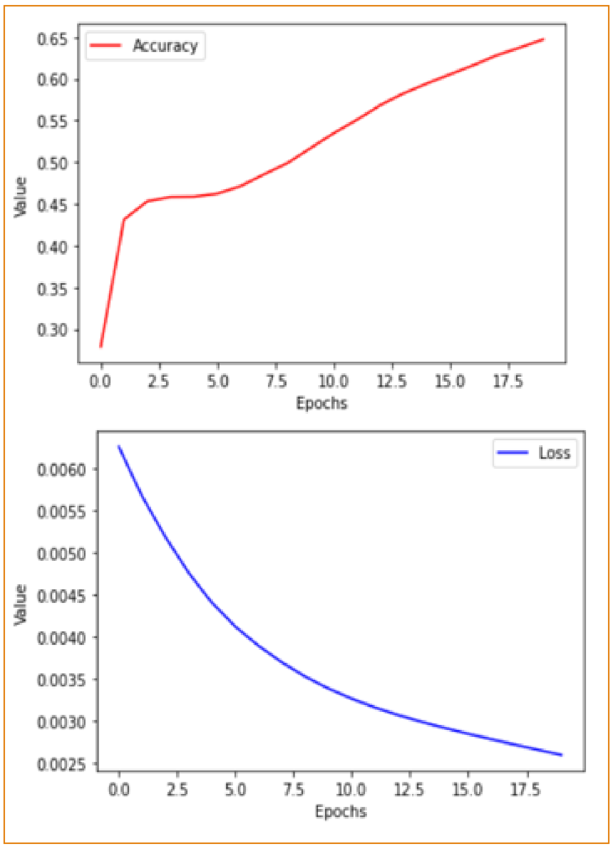
ADAM



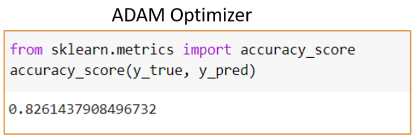
RMSPROP


Accuracy
Value
Epochs
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0
2
4
6
8
Loss
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000

SGD

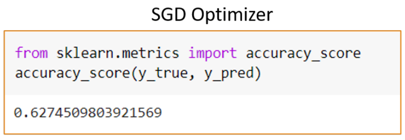


BILSTM Validation Results



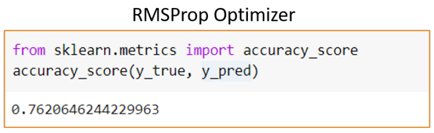
ADAM Optimizer

```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)

0.8261437908496732
```



SGD Optimizer

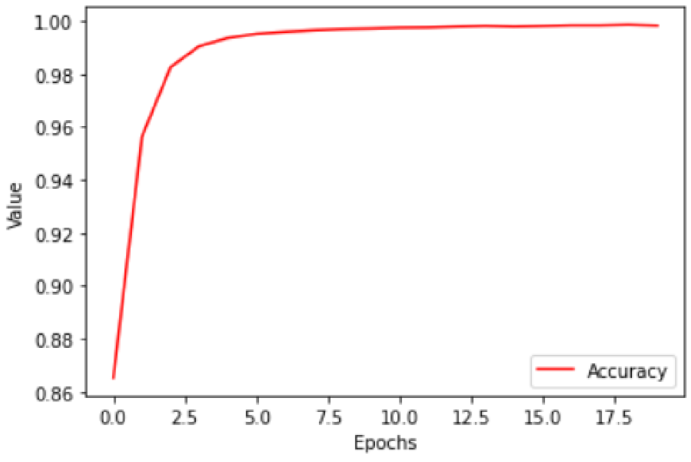
```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)

0.6274509803921569
```

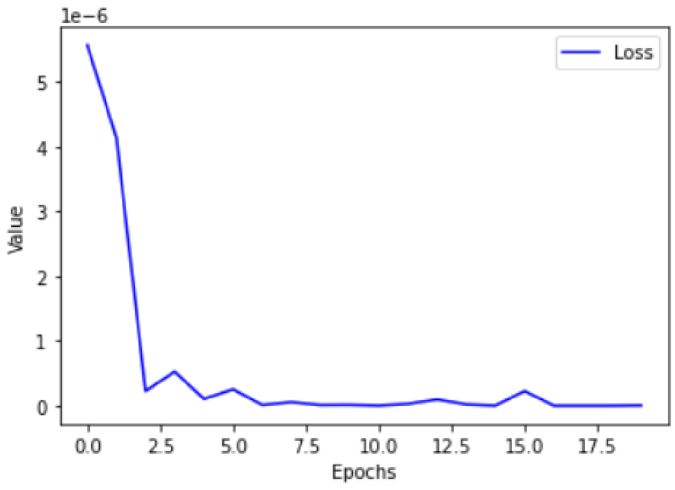


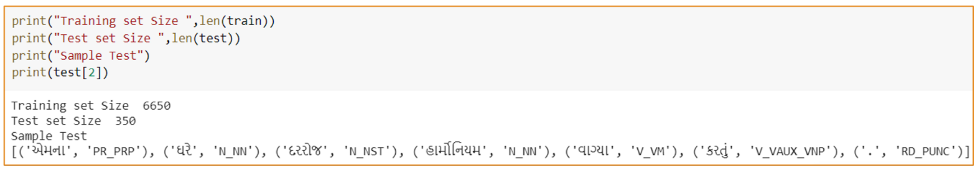
RMSProp Optimizer

```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)

0.7620646244229963
```

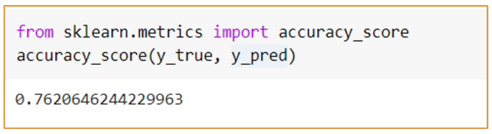
BILSTM Testing Results





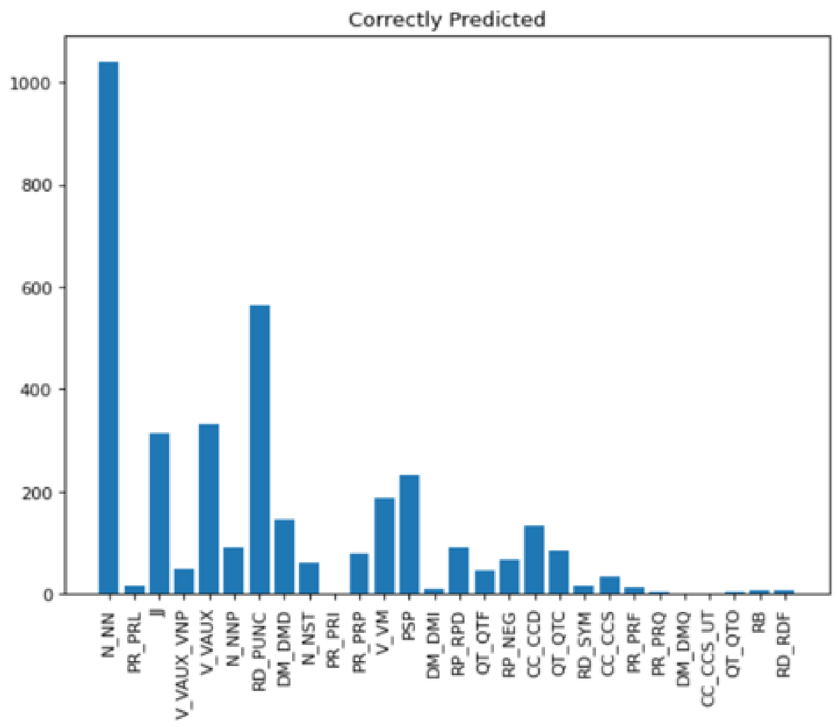


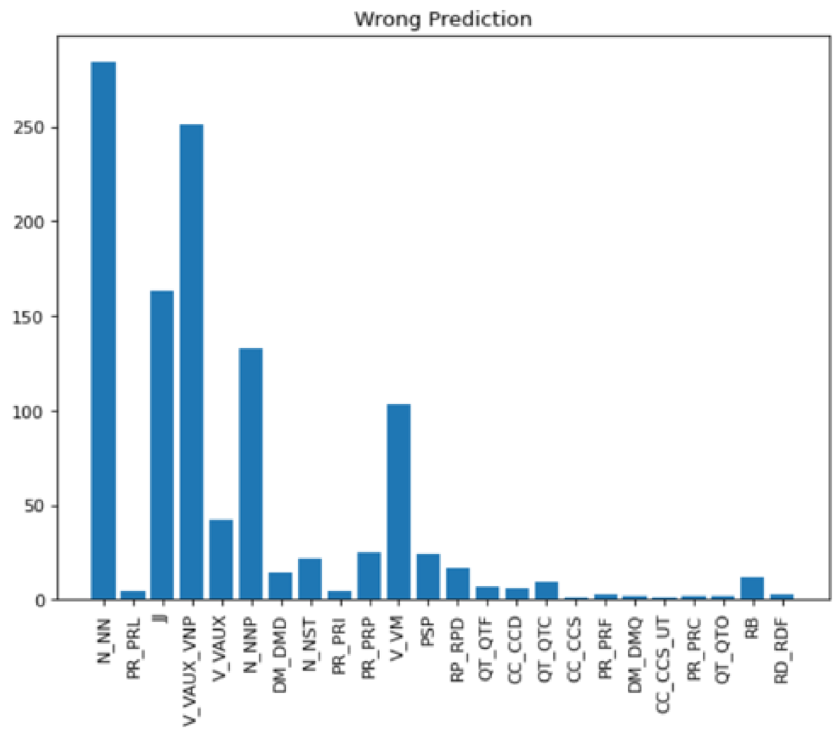
```
print("Training set Size ",len(train))
print("Test set Size ",len(test))
print("Sample Test")
print(test[2])

Training set Size  6650
Test set Size  350
Sample Test
[('એમના', 'PR_PRP'), ('ઘરે', 'N_NN'), ('દરરોજ', 'N_NST'), ('હાર્મોનિયમ', 'N_NN'), ('વાગ્યા', 'V_VM'), ('કરતું', 'V_VAUX_VNP'), ('.', 'RD_PUNC')]
```



```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)

0.7620646244229963
```

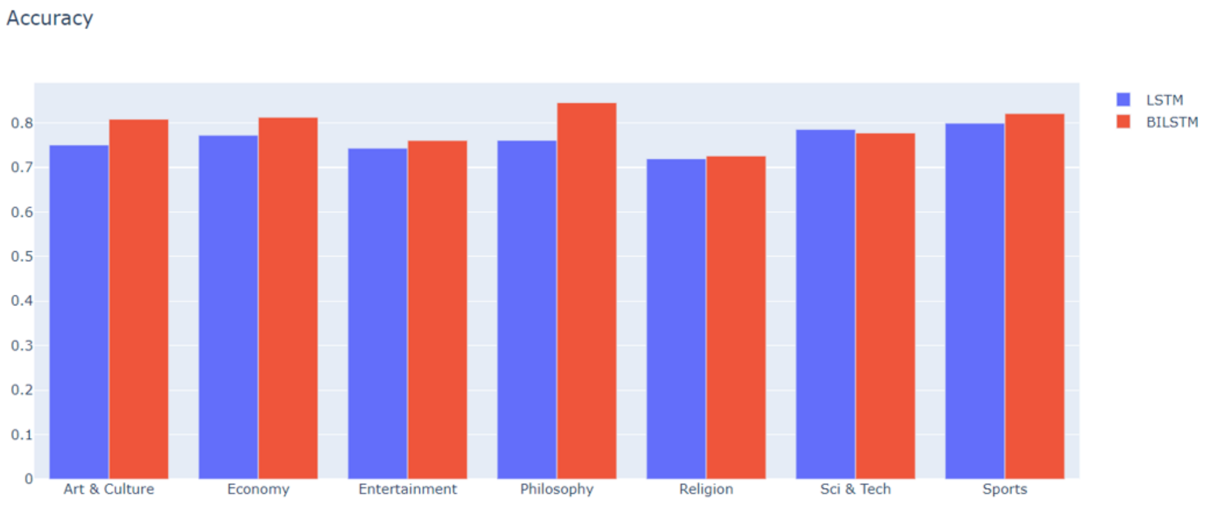
4.10 BILSTM Correct vs Wrong Prediction



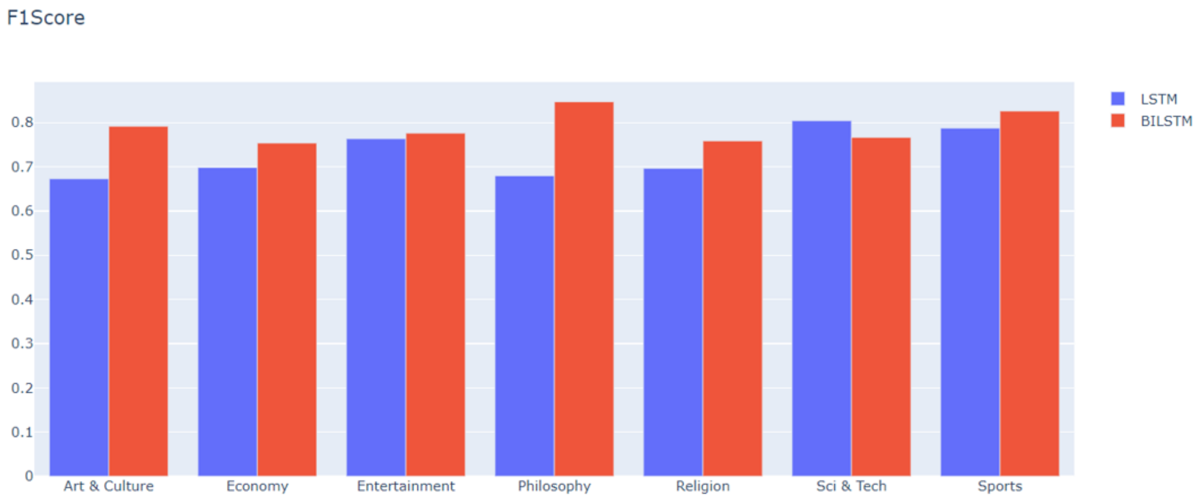


5 Results

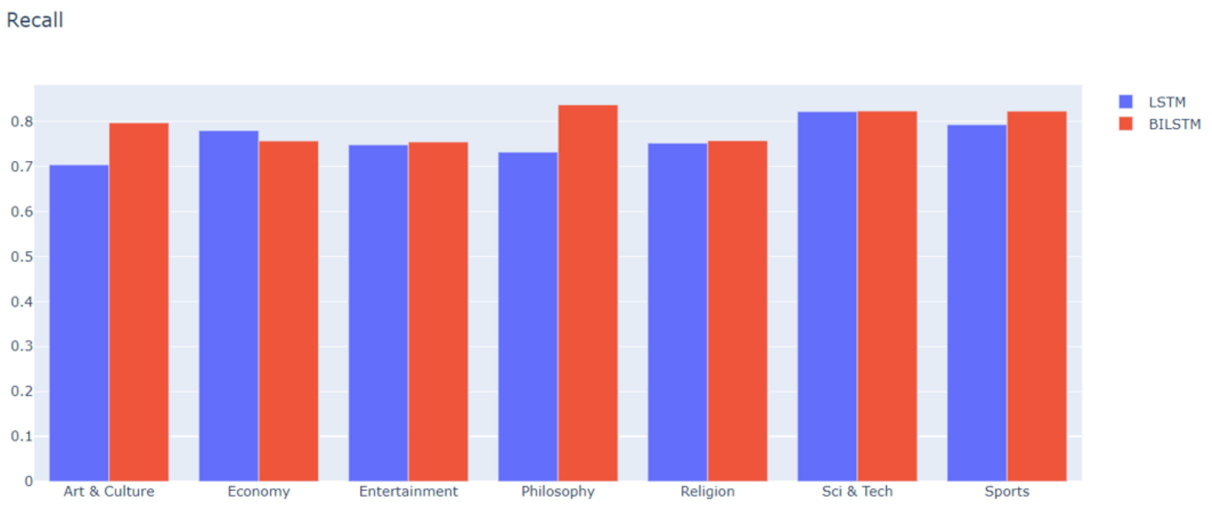
5.1 Accuracy



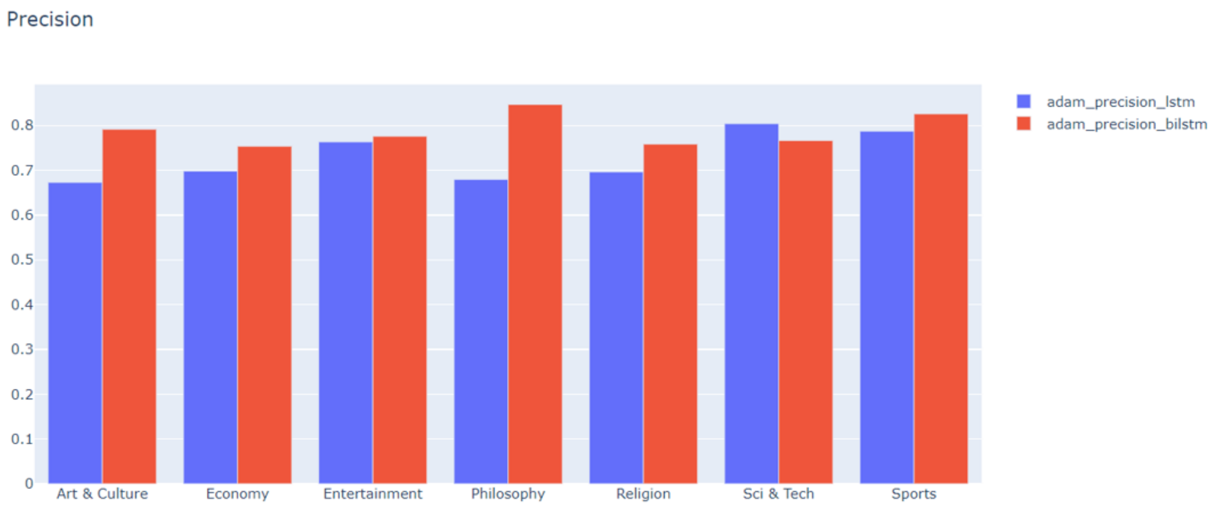
5.2 F1 Score



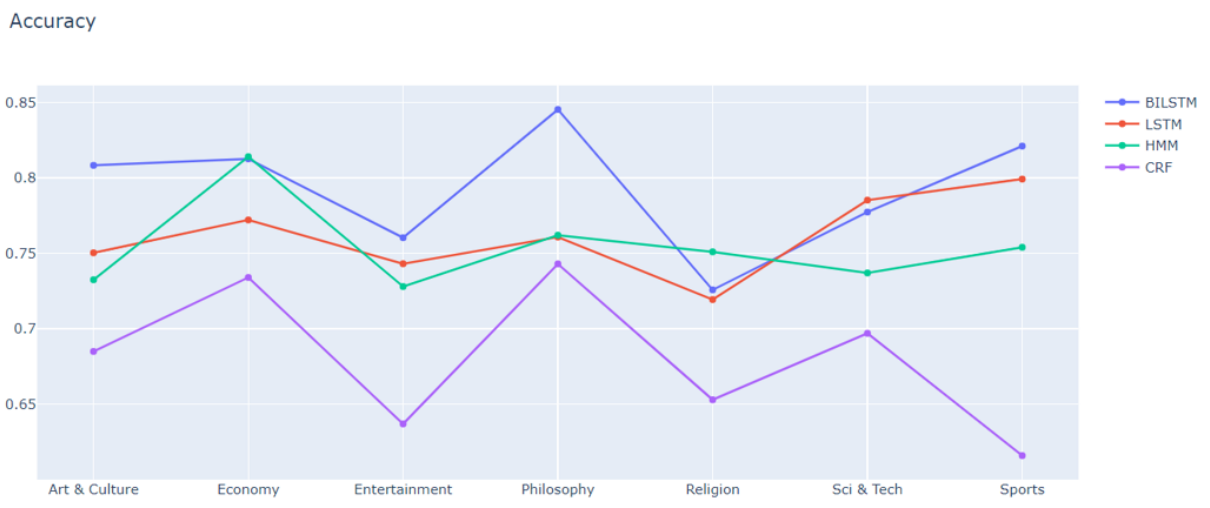
5.3 Recall

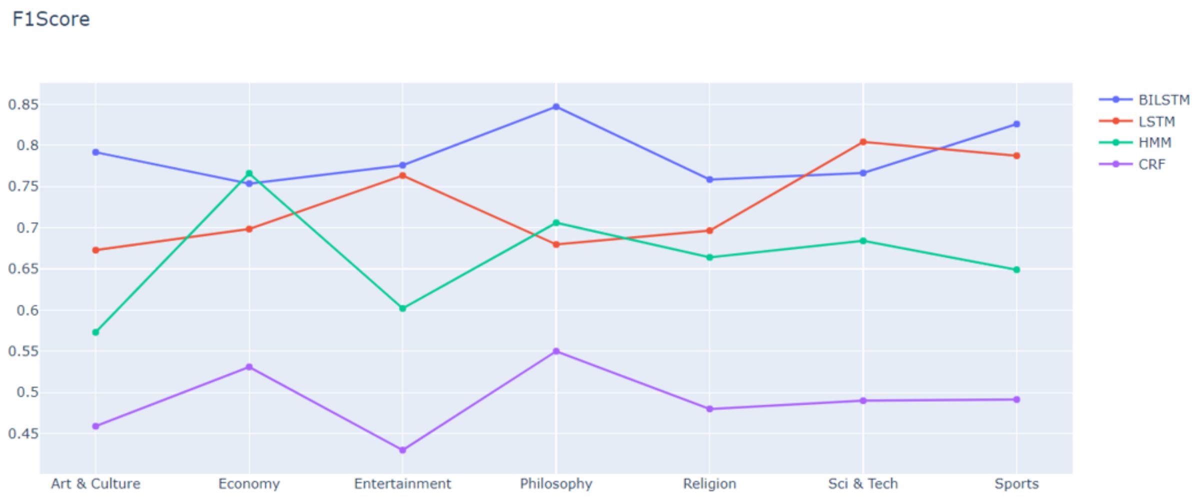


5.4 Precision



6 Comparison Statistical and Neural


Accuracy
Art & Culture
Economy
Entertainment
Philosophy
Religion
Sci & Tech
Sports
BILSTM
LSTM
HMM
CRF


F1Score
Art & Culture
Economy
Entertainment
Philosophy
Religion
Sci & Tech
Sports
BILSTM
LSTM
HMM
CRF