International Institute of Information Technology, Hyderabad

Part-Of-Speech Tagging for Gujarati Language

Team Members:
Smitkumar Khanpara (2019201021)
Dharmeshgiri Gusai (2019201025)

Project Mentor : Ujwal Narayan

April 30, 2021

1 INTRODUCTION

Parts of Speech tagging is the process of tagging the words of a running text with their categories that best suits the definition of the word as well as the context of the sentence in which it is used. This process is often the first step for many NLP applications. Work in this field is usually either statistical or machine learning based, or rule based. Some of the models that use the first approach are Hidden Markov Models (HMMs), Conditional Random Fields (CRFs), Maximum Entropy Markov Models (MEMMs), etc.

In the rule-based methods, A dictionary is constructed with possible tags for each word. Rules guide the tagger to disambiguate. Rules are either hand-crafted, learned or both. These rules are directly applied to the test corpus. Where In the statistical methods, A text corpus is used to derive useful probabilities. Given a sequence of words, the most probable sequence of tags is selected. These are also called stochastic or probabilistic taggers. The statistical learning-based methods solve the problem mostly as a classification problem. The neural network-based methods use a supervised learning approach where each word converted to some features with the corresponding tag is given as input while training. In this method, we can train models as per our requirements.

This project contains the implementation of two statistical baseline models Conditional Random Field (CRF) and Hidden Markov Model (HMM). The statistical models are not language specific. Hence, they fail when semantic knowledge is needed while tagging a word with more than one sense. Even for those words which have not appeared in the training corpus, these methods go by the probabilities but are not guaranteed to give the correct tag as they lack the semantic knowledge of the language. Also, they need a large annotated corpus. But the bright side of these methods is they can tag any word (known or unknown) with a high accuracy based on the probabilities of similar tags occurring in a particular context and some features provided for learning from the training data.

On the other hand, RNN, LSTM and BiLSTM are Neural models implemented in this project. In low-resource settings, neural POS taggers sometimes perform poorly compared to log-linear models. However, neural POS taggers have other advantages, including being easily integrable into multi-task learning architectures, sidestepping feature engineering, and providing compact word-level and sentence-level representations.

Dataset used in the project contains around 7000 sentences as Gujarati is currently a less privileged language in the sense of being resource poor for manually tagged data. The dataset comprises sentences from different topics like art and culture, economy, entertainment, philosophy, religion, science and technology and sports. For training, testing and validation, the dataset is split with ratio training(65%)-testing(20%)-validation(15%).

2 LITERATURE SURVEY

Part-Of-Speech Tagging for the Gujarati language was performed by Chirag Patel and Karthik Gali [1]. Where they used the Conditional Random Field(CRF) model to perform the task. The dataset consist of tagged (600 sentences) and untagged (5000 sentences) and the tagset contains 26 different tags which are in the standard Indian Language (IL) tagset. The features given to CRF are properly chosen keeping the linguistic aspect of Gujarati in mind. The authors also tried tagging the word on the basis of possible tags between the two surrounding words. The algorithm has achieved an accuracy of 92% for Gujarati texts where the training corpus is of 10,000 words and the test corpus is of 5,000 words.

Manish Shrivastava and Pushpak Bhattacharyya [3] present a simple HMM based POS tagger, which employs a naive(longest suffix matching) stemmer as a pre-processor to achieve a reasonably good accuracy of 93.12%. This method does not require any linguistic resource apart from a list of possible suffixes for the language. This list can be easily created using existing machine learning techniques. This method aims to demonstrate that even without employing tools like morphological analyzer or resources like a pre-compiled structured lexicon, it is possible to harness the morphological richness of Indian Languages.

Character-level Supervision for Low-resource POS Tagging presented by Katharina Kann et al. [5] in a workshop by Association for Computational Linguistics. Neural part-of-speech (POS) taggers are known to not perform well with little training data. As a step towards overcoming this problem, the authors present a novel architecture for inducing more robust neural POS taggers from small samples of annotated data in low-resource languages, combining a hierarchical, deep biLSTM sequence tagger with a character-based sequence-to-sequence model. Experiments with minimal amounts of labelled data on 34 languages show that this paper's new architecture outperforms a single-task baseline and, surprisingly, that, on average, raw text autoencoding can be as beneficial for low resource POS tagging as using lemma information. The results confirmed that additional subword-level supervision improves POS taggers for resource-poor languages.

Pruthwik Mishra et al. [4] present an approach for POS tagging without any labelled data. The authors used feature transfer from a resource rich language to resource poor languages. They try to leverage these similarities and the availability of Hindi corpus for creating resources for other languages. Across 8 different Indian Languages, the authors achieved encouraging accuracies without any knowledge of the target language and any human annotation. Languages that are close to Hindi gave better results than other languages. Gujarati, Urdu and Punjabi have similar syntactic structure to Hindi with minor variations. Where, Other languages like Bengali, Konkani, Marathi, Telugu, Malayalam are morphologically richer than Hindi. So, the word level alignment is less accurate. To overcome these shortcomings, they combined the post positions marking the case, the auxiliary verbs with the preceding head categories. By incorporating these changes, they were able to capture the inherent syntactic behaviour of these languages. They achieve a reasonably good average accuracy of 80% for POS Tagging for 8 different languages.

3 Dataset Description

Dataset contains around 7000 sentences containing 97000 words. As Gujarati is currently a less privileged language in the sense of being resource poor for manually tagged data. The dataset comprises sentences from different topics like art and culture, economy, entertainment, philosophy, religion, science and technology and sports. Figure 3.1 shows details of the dataset. The tagset contains 34 different tags, which are in the standard Indian Language(IL) tagset. A detailed description of each Tag is in figure 3.3. The frequencies of each tag are shown in figure 3.2.

Topics	Size (No. of sentences)	Size (No. of words)
Art and Culture	1000	16050
Economy	1000	12467
Entertainment	1000	11879
Philosophy	1000	12904
Religion	1000	12247
Science and Technology	1000	16857
Sports	1000	14836
Total	7000	97240

Figure 3.1: Dataset Details

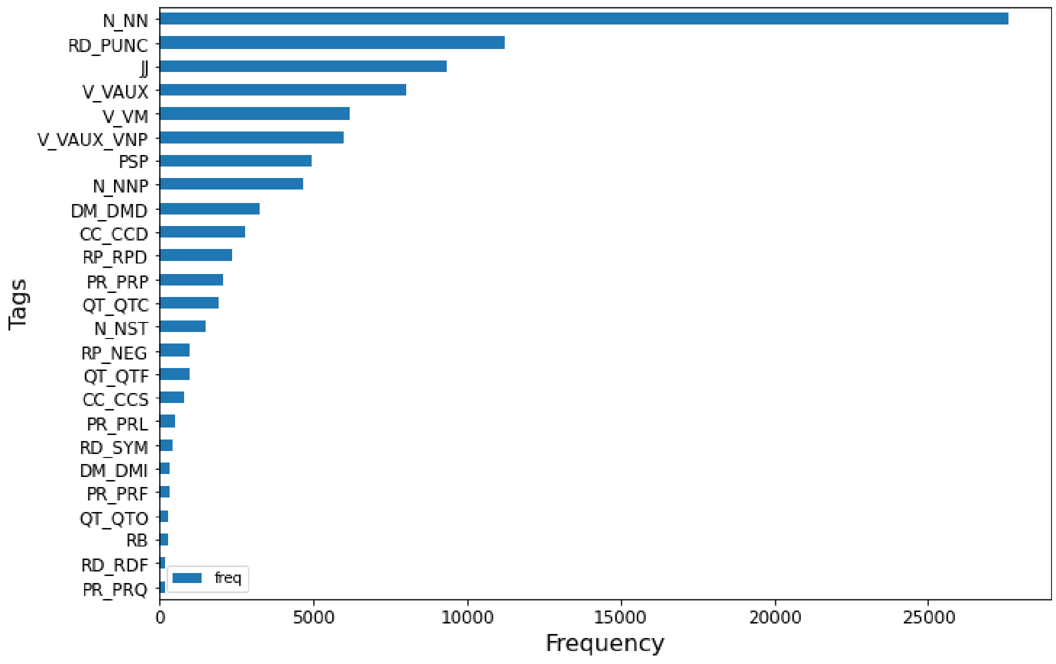
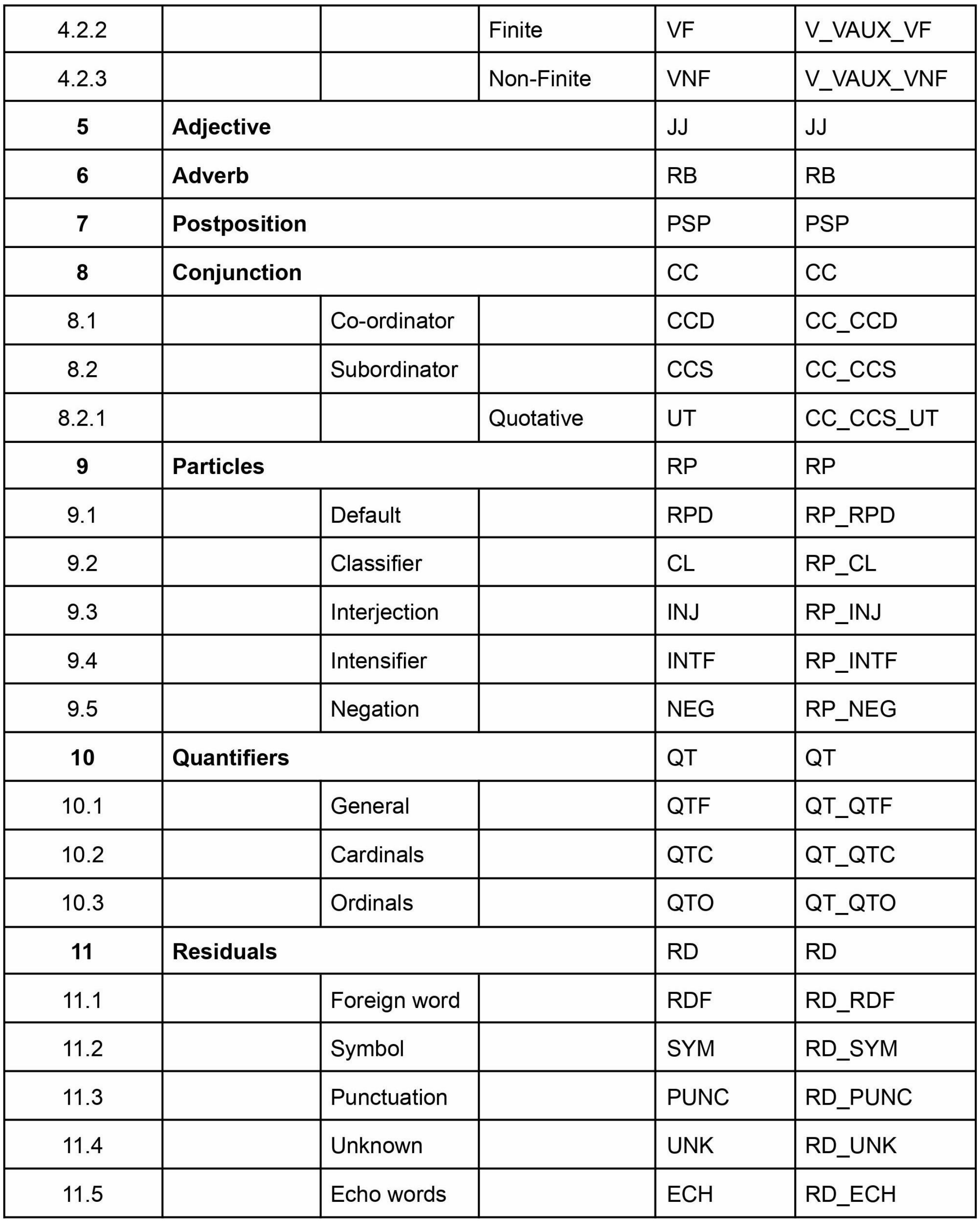


Figure 3.2: Tag Frequency in dataset

Sl No.	Category			Label	Annotation Convention
	Top Level	Subtype (Level1)	Subtype (Level2)		
1	**Noun**			N	N
1.1		Common		NN	N_NN
1.2		Proper		NNP	N_NNP
1.3		locative case		NST	N_NST
2	**Pronoun**			PR	PR
2.1		Personal		PRP	PR_PRP
2.2		Reflexive		PRF	PR_PRF
2.3		Relative		PRL	PR_PRL
2.4		Reciprocal		PRC	PR_PRC
2.5		Wh-word		PRQ	PR_PRQ
2.6		Indefinite		PRI	PR_PRI
3	**Demonstrative**			DM	DM
3.1		Deictic		DMD	DM_DMD
3.2		Relative		DMR	DM_DMR
3.3		Wh-word		DMQ	DM_DMQ
3.4		Indefinite		DMI	DM_DMI
4	**Verb**			V	V
4.1		Main		VM	V_VM
4.1.1			Finite	VF	V_VM_VF
4.1.2			Non-Finite	VNF	V_VM_VNF
4.1.3			Infinitive	VINF	V_VM_VINF
4.2		Auxiliary		VAUX	V_VAUX
4.2.1			Participle Noun	VNP	V_VAUX_VNP



4.2.2			Finite	VF	V_VAUX_VF
4.2.3			Non-Finite	VNF	V_VAUX_VNF
5	**Adjective**			JJ	JJ
6	**Adverb**			RB	RB
7	**Postposition**			PSP	PSP
8	**Conjunction**			CC	CC
8.1		Co-ordinator		CCD	CC_CCD
8.2		Subordinator		CCS	CC_CCS
8.2.1			Quotative	UT	CC_CCS_UT
9	**Particles**			RP	RP
9.1		Default		RPD	RP_RPD
9.2		Classifier		CL	RP_CL
9.3		Interjection		INJ	RP_INJ
9.4		Intensifier		INTF	RP_INTF
9.5		Negation		NEG	RP_NEG
10	**Quantifiers**			QT	QT
10.1		General		QTF	QT_QTF
10.2		Cardinals		QTC	QT_QTC
10.3		Ordinals		QTO	QT_QTO
11	**Residuals**			RD	RD
11.1		Foreign word		RDF	RD_RDF
11.2		Symbol		SYM	RD_SYM
11.3		Punctuation		PUNC	RD_PUNC
11.4		Unknown		UNK	RD_UNK
11.5		Echo words		ECH	RD_ECH

set

Figure 3.3: Tag Description

4 Baseline Models

The Stochastic technique based Hidden Markov Model(HMM) and a sequence modelling algorithm Conditional Random Field (CRF) are two models implemented here as standard baseline models. A stochastic approach includes frequency., probability or statistics.

4.1 Hidden Markov Model (HMM)

Hidden Markov Model (HMM) is a Stochastic technique for POS tagging. Also, HMM are known for their applications to reinforcement learning and temporal pattern recognition such as speech, handwriting, gesture recognition, musical score following, partial discharges, and bioinformatics. HMM is a commonly used generative stochastic method regularly used in NL, Speech and Image Processing domains. The allure of HMM is its malleability and the ability to perform well if trained on data closely resembling the test data. By malleability, we mean the ability to modify a model. HMMs are very simple stochastic models and present themselves with ease to modifications.

4.1.1 Model Description

Hidden Markov Model (HMM) implemented here in the project is optimized HMM where standard HMM is optimized using the Viterbi algorithm. The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states called the Viterbi path that results in a sequence of observed events, especially in the context of Markov information sources and HMM.

The basic idea of the model can be obtained from Figure 4.1. It consists of steps like data loading followed by required preprocessing then the calculation of emission and transition probabilities for words. These probabilities are feed to HMM model to create states. At last Viterbi algorithm applied to HMM.

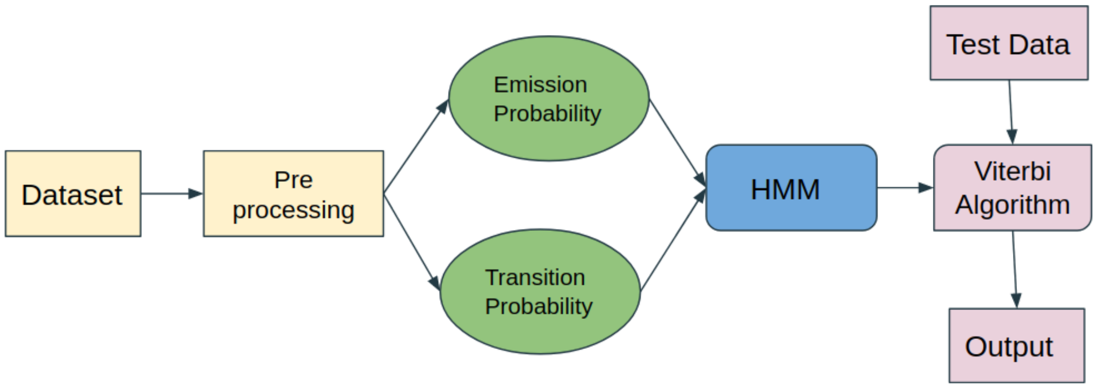


Figure 4.1: Hidden Markov Model (HMM) for POS Tagging

Standard Hidden Markov Models are simple three tuple models described as $\lambda(\pi, A, B)$, where,

- π = Initial Probabilities
- A = Transition Probabilities
- B = Emission Probabilities

For a given input sequence $W = (w_1, w_2, ..., w_n)$ we wish to determine a tag sequence $T = (t_1, t_2, ..., t_n)$ such that P(W, T) is maximized. This probability term when broken down using chain rule results in a term implausible to compute.

$$P(W,T) = \pi_i^N [P(w_i | t_{1,i}, w_{1,i-1}) P(t_i | t_{1,i-1}, w_{1,i-1})]$$

where,

- W is word sequence
- T is the tag sequence
- w_i is the word at i^{th} position
- t_i is the tag at i^{th} position
- N is the length of the sequence.

This term is restricted by HMM using two simplifying assumptions:

- Word w_i depends only on the current tag (Lexical Independence).
- Tag t_i depends on previous K tags (Markov Property).

4.1.2 Data Preprocessing & Feature Selection

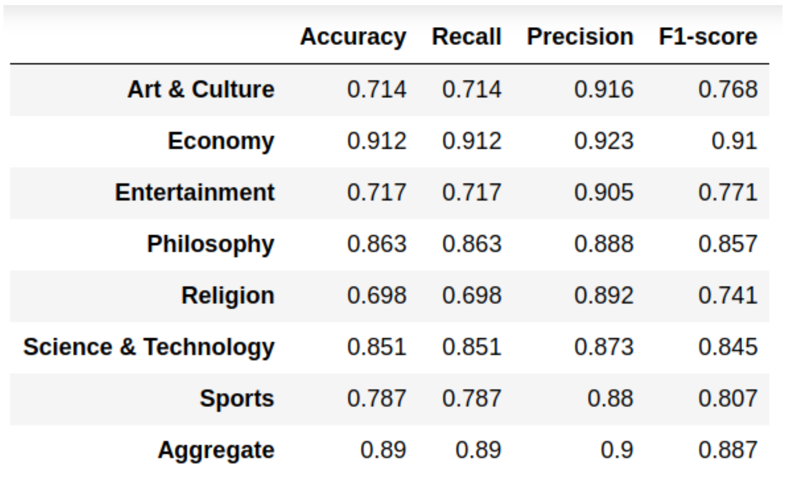
- **Data preprocessing :** Data cleaning, Remove non useful characters, words and output list of sentence.
- **Handling of Unknown Word :** The minimum word frequency from the training set is considered as the frequency of missing word.
- **Train-Test Split :** Train set (80%) - Test set(20%)
- **Features :** Emission Probability and Transition Probability

4.1.3 Results

HMM model trained with 80% of the dataset and the remaining 20% dataset used as test dataset. The accuracy and F1-score achieved by the model are 89% and 87.7%. The model performed best in sentences from the economy field. Figure 4.2 has more details of model performance.

4.1.4 Error Analysis

Error analysis of HMM is given in Figure 4.3 contains Actual tag, Assigned tag by HMM and Error count. We can see here that the model is struggling to make differences between common (N_NN) and proper noun (N_NNP). Also, In many cases, an adjective(JJ) is tagged as a



	Accuracy	Recall	Precision	F1-score
Art & Culture	0.714	0.714	0.916	0.768
Economy	0.912	0.912	0.923	0.91
Entertainment	0.717	0.717	0.905	0.771
Philosophy	0.863	0.863	0.888	0.857
Religion	0.698	0.698	0.892	0.741
Science & Technology	0.851	0.851	0.873	0.845
Sports	0.787	0.787	0.88	0.807
Aggregate	0.89	0.89	0.9	0.887

Figure 4.2: HMM performance

noun (N_NN) because, in this language, adjectives may or may not occur before the nouns. Hence the probability of this unknown word to be a noun(N_NN) or an adjective(JJ) is equal or will depend on the number of instances of both in the training corpus. Moreover, The model able to precisely assigned tags of categories Conjunction(CC), Demonstrative (DM), Quantifiers(QT) and Residuals like symbol, punctuation, etc.

4.2 Conditional Random Field (CRF)

A Conditional Random Field (CRF) is a sequence modeling algorithm which is used to identify entities or patterns in text, such as POS tags. This model not only assumes that features are dependent on each other, but also considers future observations while learning a pattern. Since these models take into account previous data, we use features which are modelled from the data to feed into the CRF.

4.2.1 Model Description

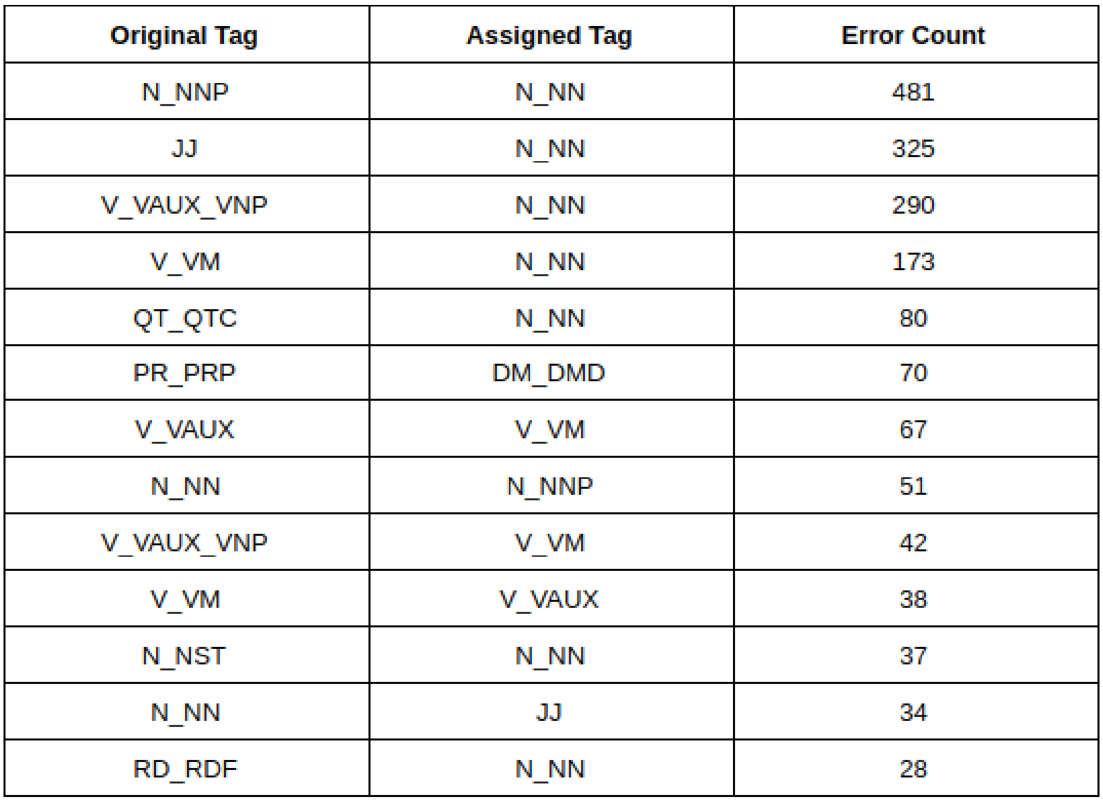
CRF is a discriminant model for sequences data. It models the dependency between each state and the entire input sequences. CRF overcomes the label bias issue by using global normalizer.

Conditional Random Fields is described as:

$$P_w(y|x) = \frac{1}{Z_w(x)} \exp\left(\Sigma_{j=1}^{n} \Sigma_{i=1}^{m} w_i f_i(y_{j-1}, y_j, x, j)\right)$$

with Z(x) defined as:

$$Z_w(y|x) = \Sigma_{y \epsilon Y} \exp\left(\Sigma_{j=1}^{n} \Sigma_{i=1}^{m} w_i f_i(y_{j-1}, y_j, x, j)\right)$$



Original Tag	Assigned Tag	Error Count
N_NNP	N_NN	481
JJ	N_NN	325
V_VAUX_VNP	N_NN	290
V_VM	N_NN	173
QT_QTC	N_NN	80
PR_PRP	DM_DMD	70
V_VAUX	V_VM	67
N_NN	N_NNP	51
V_VAUX_VNP	V_VM	42
V_VM	V_VAUX	38
N_NST	N_NN	37
N_NN	JJ	34
RD_RDF	N_NN	28

Figure 4.3: Error Analysis of HMM

- The summation of j=1 to n is the sum of all data points. This is needed in comparison to the Maximum Entropy Model. The whole label sequence is considered in the prediction instead of a single label. The variable j specifies the position of the input sequence x.
- The summation of i=1 to m is the sum of all feature functions.
- The summation of y is the sum of all possible label sequences. It is performed to get the feasible probability.
- f_i is the feature function. More details of feature function is given in section 4.2.2.
- To overcome the label bias problem, CRF uses the global normalizer Z(x) instead of local normalizer as in the MEMM. So Z(x) in CRF takes a sum of all the possible sequences of tag y ϵ Y.

The basic idea of the model can be obtained from Figure 4.4. It consists of steps like data loading followed by required preprocessing then the most important part feature extraction for words. Then the model trained using these features. Furthermore, the model evaluated using a test dataset.

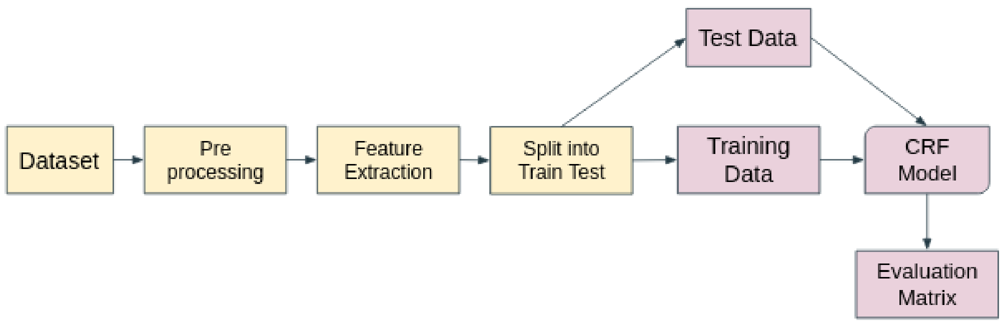


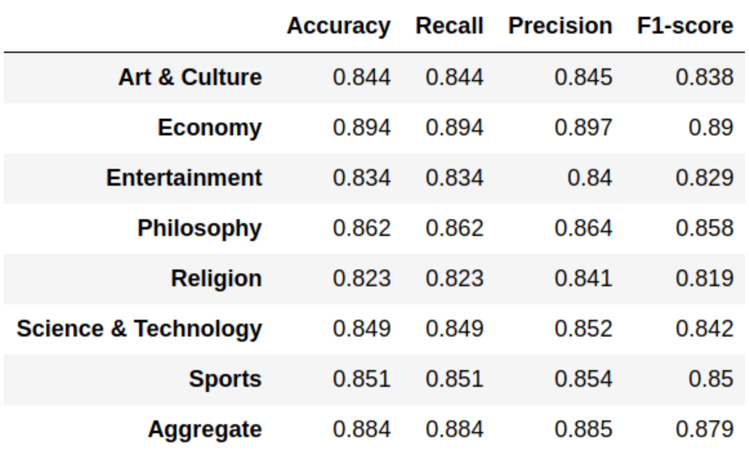
Figure 4.4: Conditional Random Field (CRF) Model for POS Tagging

4.2.2 Data Preprocessing & Feature Selection

- **Data preprocessing :** Data cleaning, Remove non useful characters, words and output list of sentence.
- **Train-Test Split :** Train set (80%) - Test set(20%)
- **Features :**
  - 'word' : Word
  - 'is_first' : Is it first word of sentence ? (True / False)
  - 'is_last' : Is it last word of sentence ? (True / False)
  - 'prefix-1' : Prefix of word of sizes - 1
  - 'prefix-2' : Prefix of word of sizes - 2
  - 'prefix-3' : Prefix of word of sizes - 3
  - 'suffix-1' : Suffix of word of sizes - 1
  - 'suffix-2' : Suffix of word of sizes - 2
  - 'suffix-3' : Suffix of word of sizes - 3
  - 'prev_word' : Previous word
  - 'pprev_word' : Previous of previous word
  - 'next_word' : Next word
  - 'nnext_word' : Next to next word
  - 'Is_numeric' : Is it numeric ? (True / False)

4.2.3 Results

CRF model trained with 80% of the dataset and the remaining 20% dataset used as test dataset. The accuracy and F1-score achieved by the model are 88.4% and 87.9%. The model performed best in sentences from the economy field. Also, The balance performance was found by CRF while Tagging words from different topics. Figure 4.5 has more details of model performance. From the experiments, we observed that if the language specific rules can be formulated into features for CRF then the accuracy can be reached to very high extents.

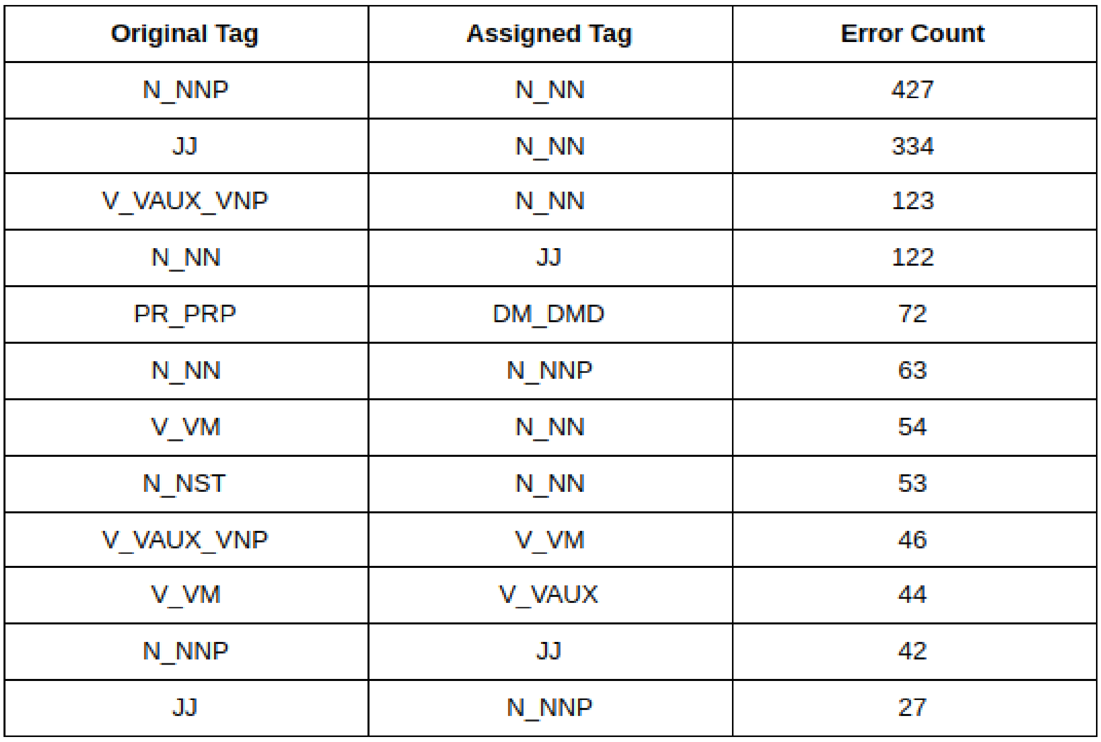


	Accuracy	Recall	Precision	F1-score
Art & Culture	0.844	0.844	0.845	0.838
Economy	0.894	0.894	0.897	0.89
Entertainment	0.834	0.834	0.84	0.829
Philosophy	0.862	0.862	0.864	0.858
Religion	0.823	0.823	0.841	0.819
Science & Technology	0.849	0.849	0.852	0.842
Sports	0.851	0.851	0.854	0.85
Aggregate	0.884	0.884	0.885	0.879

Figure 4.5: CRF Performance

4.2.4 Error Analysis

Error analysis of CRF is given in Figure 4.6 contains Actual tag, Assigned tag by CRF and Error count. We can see here that the model is struggling to make differences between common (N_NN) and proper noun (N_NNP) like HMM. Also, In many cases, an adjective(JJ) is tagged as a noun (N_NN) because, in this language, adjectives may or may not occur before the nouns. Hence the probability of this unknown word to be a noun(N_NN) or an adjective(JJ) is equal or will depend on the number of instances of both in the training corpus. There are very less instances where two adjectives come together in the training corpus. Again the chances of it being a noun increase as the quantifier(Q_QF) mostly precede nouns instead of adjectives. Here we also have a Q_QF before the unknown word. The CRF model confuses between a verb and noun Tagging. Moreover, The model able to precisely assigned tags of categories Conjunction(CC), Demonstrative (DM), Quantifiers(QT) and Residuals like symbol, punctuation, etc.



Original Tag	Assigned Tag	Error Count
N_NNP	N_NN	427
JJ	N_NN	334
V_VAUX_VNP	N_NN	123
N_NN	JJ	122
PR_PRP	DM_DMD	72
N_NN	N_NNP	63
V_VM	N_NN	54
N_NST	N_NN	53
V_VAUX_VNP	V_VM	46
V_VM	V_VAUX	44
N_NNP	JJ	42
JJ	N_NNP	27

Figure 4.6: Error Analysis of CRF

5 Neural Models

Recurrent neural networks, or RNNs, are a type of artificial neural network that add additional weights to the network to create cycles in the network graph in an effort to maintain an internal state. The promise of adding state to neural networks is that they will be able to explicitly learn and exploit context in sequence prediction problems, such as POS Tagging.

5.1 Model Description

The project contains the experiments with three different RNN variants described below.

- **Recurrent Neural Network (RNN) :** RNNs have feedback loops in the recurrent layer. This lets them maintain information in 'memory' over time. But, it can be difficult to train standard RNNs to solve problems that require learning long-term temporal dependencies. This is because the gradient of the loss function decays exponentially with time (called the vanishing gradient problem).

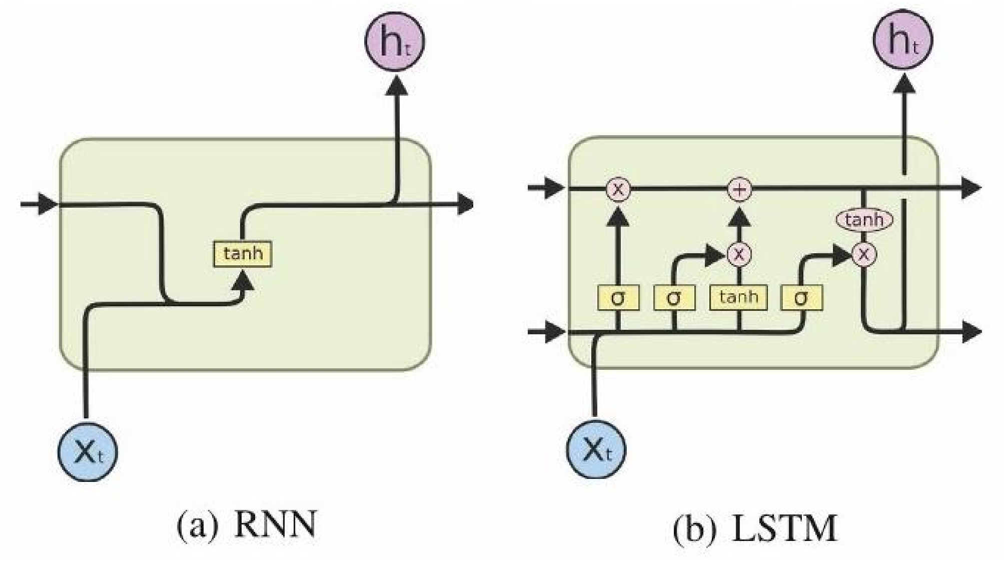


Figure 5.1: RNN vs LSTM Architecture

- **Long Short-Term Memory (LSTM) :** LSTM networks are a type of RNN that uses special units in addition to standard units. LSTM units include a 'memory cell' that can maintain information in memory for long periods of time. A set of gates is used to control when information enters the memory, when it's output, and when it's forgotten. This architecture lets them learn longer-term dependencies.
- **Bidirectional LSTM (BiLSTM) :** Bidirectional LSTM manage inputs in two ways, one from past to future and one from future to past and it differs this approach from unidirectional is that in the LSTM which runs backwards you preserve information from the

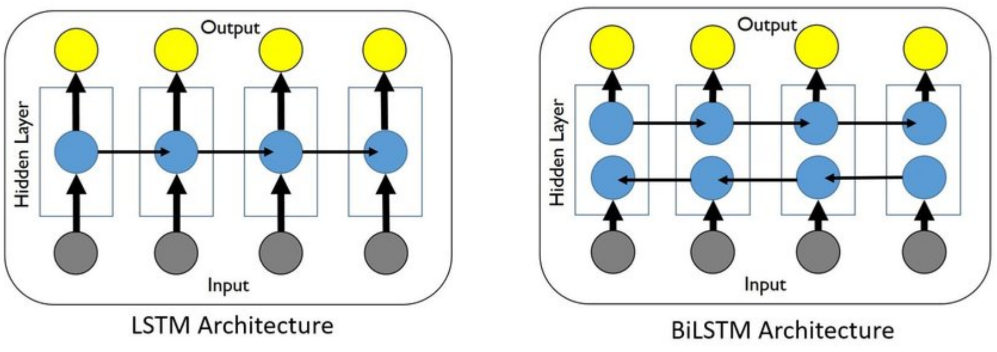


Figure 5.2: LSTM vs BiLSTM Architecture

future and using the two hidden states combined, you will be able at any point in time to preserve information from both past and future.

The basic idea of the model can be obtained from Figure 5.3. It consists of steps like data loading followed by required preprocessing then the most important part feature extraction for words. Then the model trained using these features. Furthermore, the model evaluated using a test dataset.

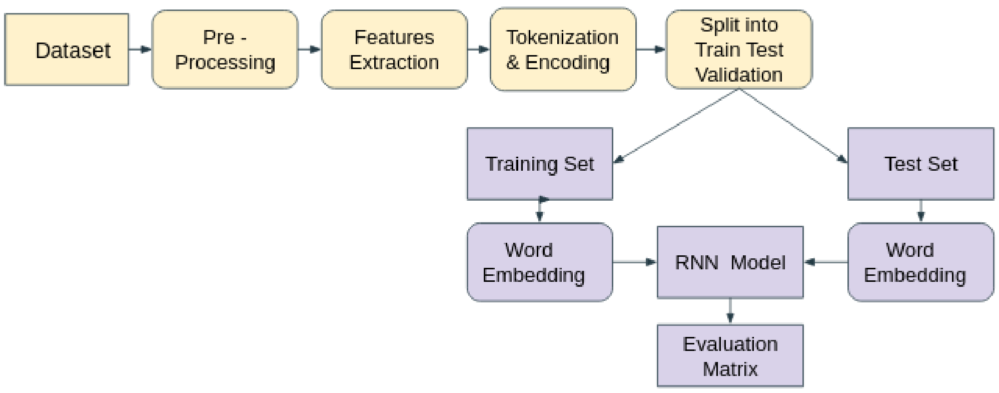


Figure 5.3: RNN Model for POS Tagging

5.2 DATA PREPROCESSING & FEATURE SELECTION

- **Data preprocessing :** Data cleaning, Remove non useful characters, words and output list of sentence.
- **Features :**
  - ‘word’ : Word
  - ’is_first’ : Is it first word of sentence ? (True / False)
  - ‘is_last’ : Is it last word of sentence ? (True / False)

  - ‘prefix-1’ : Prefix of word of sizes - 1
  - ‘prefix-2’ : Prefix of word of sizes - 2
  - ‘prefix-3’ : Prefix of word of sizes - 3
  - ‘suffix-1’ : Suffix of word of sizes - 1
  - ‘suffix-2’ : Suffix of word of sizes - 2
  - ‘suffix-3’ : Suffix of word of sizes - 3
  - ‘prev_word’ : Previous word
  - ‘pprev_word’ : Previous of previous word
  - ‘next_word’ : Next word
  - ‘nnext_word’ : Next to next word
  - ‘Is_numeric’ : Is it numeric ? (True / False)
- **Tokenization & Encoding :** encoding of features of using inbuilt Tokenizer of nltk.
- **Train-Test-Validation Split :** Training set (65%) - Test set(20%) - Validation set(15%).
- **Word Embedding :** More meaningful vector representation for neural model.

5.3 Results

We have performed experiments with three different RNN models; Standard RNN, LSTM and BiLSTM. All three models perform almost equally in every aspect. Validation accuracy and loss during the training phase of all model is given in figure 5.4 as a comparison plot. But LSTM performs slightly better on the test corpus than others. The accuracy and f1-score achieved by LSTM is 90.8% and 90.7%, and this is better than baseline models CRF and HMM. From the errors, we conclude that as the training data increases, the fewer numbers of unknown word will be encountered in the test corpus, which will increase the accuracy. More data can be found in tables given in figures 5.5, 5.6 and 5.7.

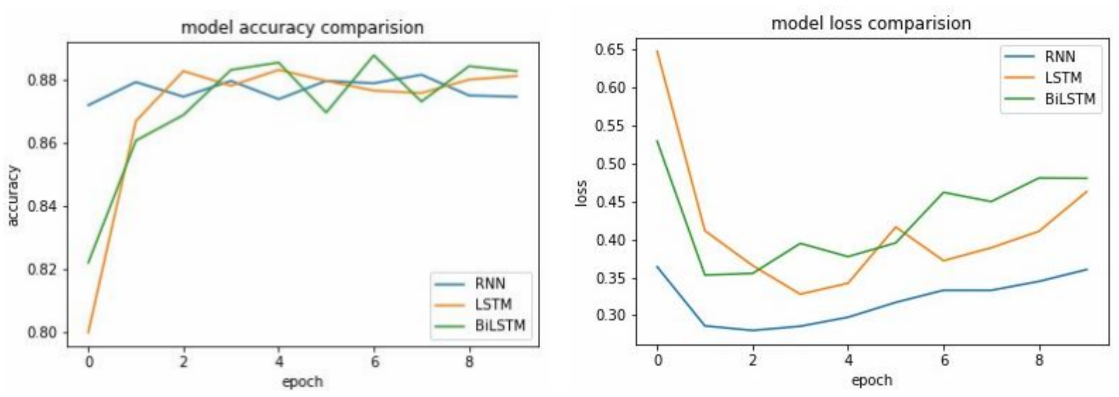
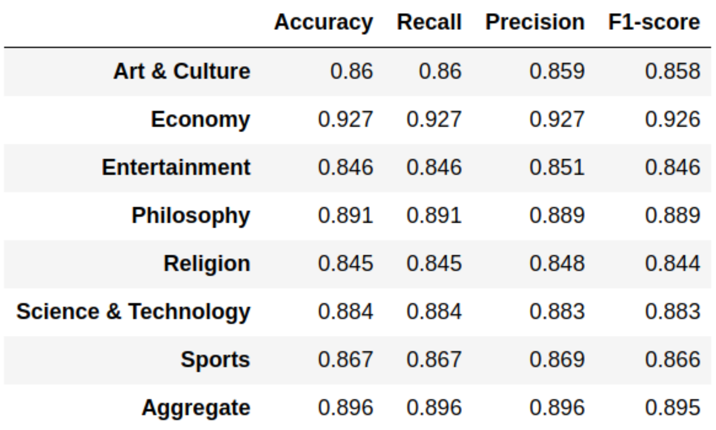
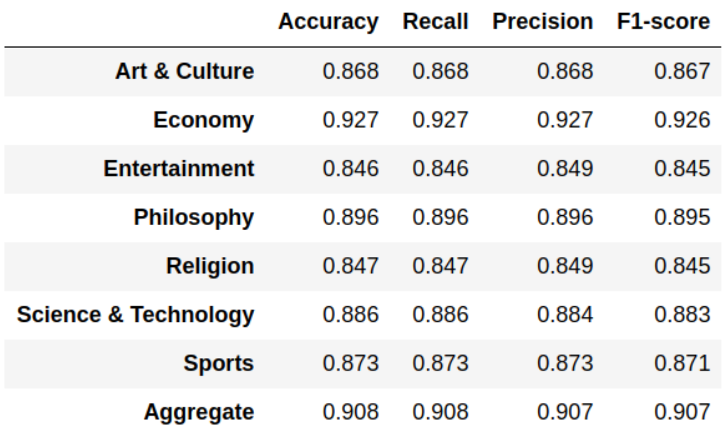


Figure 5.4: Validation Accuracy & Loss Comparison of RNN Models



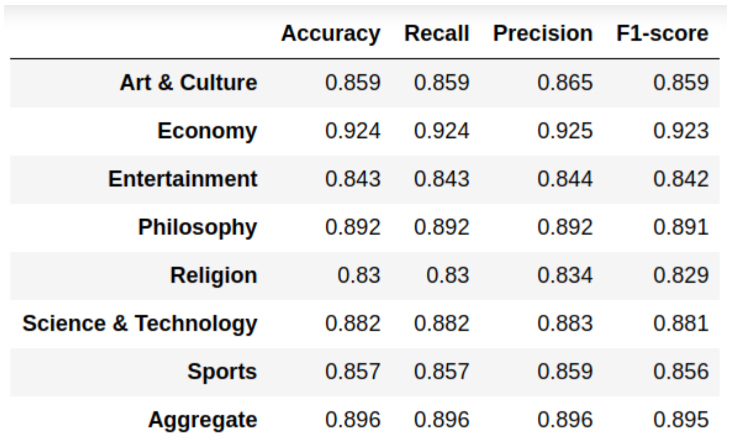
	Accuracy	Recall	Precision	F1-score
Art & Culture	0.86	0.86	0.859	0.858
Economy	0.927	0.927	0.927	0.926
Entertainment	0.846	0.846	0.851	0.846
Philosophy	0.891	0.891	0.889	0.889
Religion	0.845	0.845	0.848	0.844
Science & Technology	0.884	0.884	0.883	0.883
Sports	0.867	0.867	0.869	0.866
Aggregate	0.896	0.896	0.896	0.895

Figure 5.5: RNN Performance



	Accuracy	Recall	Precision	F1-score
Art & Culture	0.868	0.868	0.868	0.867
Economy	0.927	0.927	0.927	0.926
Entertainment	0.846	0.846	0.849	0.845
Philosophy	0.896	0.896	0.896	0.895
Religion	0.847	0.847	0.849	0.845
Science & Technology	0.886	0.886	0.884	0.883
Sports	0.873	0.873	0.873	0.871
Aggregate	0.908	0.908	0.907	0.907

Figure 5.6: LSTM Performance

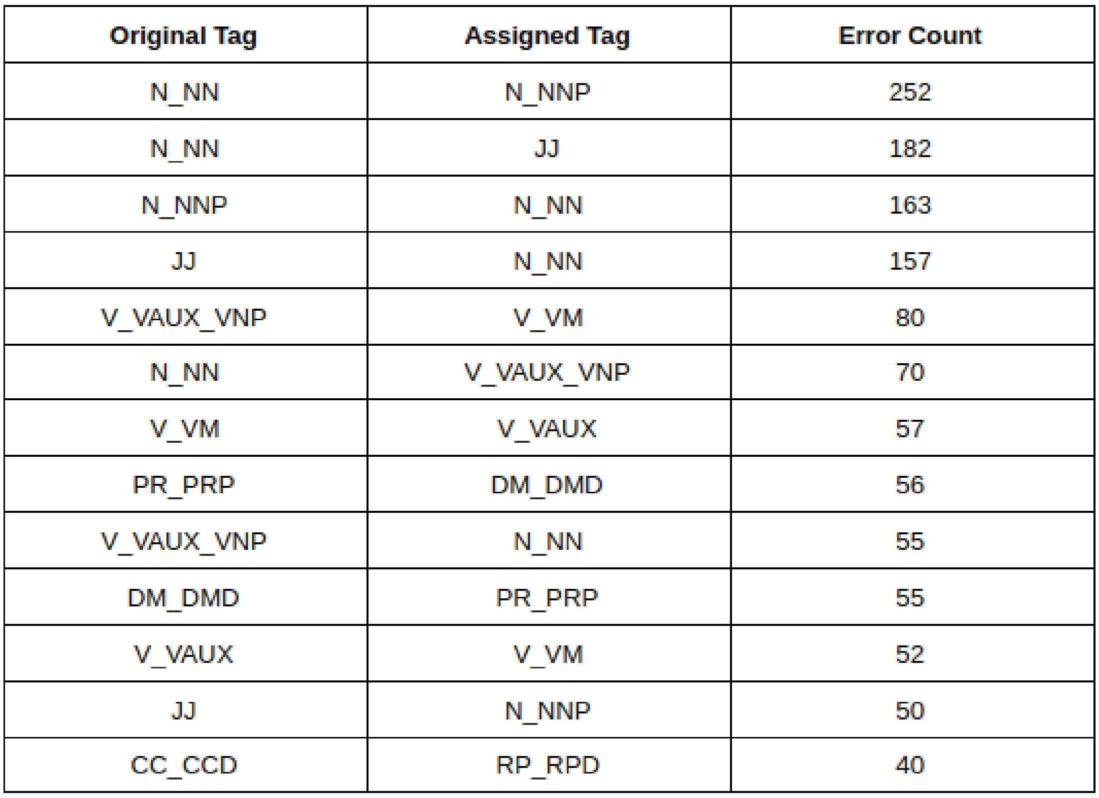


	Accuracy	Recall	Precision	F1-score
Art & Culture	0.859	0.859	0.865	0.859
Economy	0.924	0.924	0.925	0.923
Entertainment	0.843	0.843	0.844	0.842
Philosophy	0.892	0.892	0.892	0.891
Religion	0.83	0.83	0.834	0.829
Science & Technology	0.882	0.882	0.883	0.881
Sports	0.857	0.857	0.859	0.856
Aggregate	0.896	0.896	0.896	0.895

Figure 5.7: BiLSTM Performance

5.4 Error Analysis

Error analysis of LSTM is given in Figure 5.8 contains the Actual tag, Assigned tag by CRF and Error count. LSTM having lesser error counts in POS Tagging than CRF and HRF but, making the same types of errors as CRF and HMM. The model is struggling to make differences between common(N_NN) and proper noun (N_NNP). Also, In many cases, an adjective(JJ) is tagged as a noun (N_NN) because, in this language, adjectives may or may not occur before the nouns. Hence, the probability of this unknown word to be a noun(N_NN) or an adjective(JJ)is equal or will depend on the number of instances of both in the training corpus. More, The model confuses between a verb and noun Tagging. Moreover, The model able to precisely assigned tags of categories Conjunction(CC), Demonstrative (DM), Pronoun(PR), Preposition(PSP), Quantifiers(QT) and Residuals like symbol, punctuation, etc.



Original Tag	Assigned Tag	Error Count
N_NN	N_NNP	252
N_NN	JJ	182
N_NNP	N_NN	163
JJ	N_NN	157
V_VAUX_VNP	V_VM	80
N_NN	V_VAUX_VNP	70
V_VM	V_VAUX	57
PR_PRP	DM_DMD	56
V_VAUX_VNP	N_NN	55
DM_DMD	PR_PRP	55
V_VAUX	V_VM	52
JJ	N_NNP	50
CC_CCD	RP_RPD	40

Figure 5.8: Error Analysis of LSTM

6 Comparison & Conclusion

Comparison plot of accuracy (figure 6.1), precision (figure 6.3), recall(figure 6.2) and f1-score (figure 6.4) of all models is given in respective figures. From analysis, we can conclude few points like:

- HMM model has more precision but less recall compare to other models.
- CRF accuracy does not fluctuate much with change in dataset.
- LSTM model has a good F1 score overall compare to other models.
- Neural models have more accuracy than baseline models and there is not much difference in the accuracy of RNN, LSTM and BiLSTM.
- LSTM has at least 40% lesser numbers of error count for tag prediction than HMM and CRF in major misclassification.
- All models are struggling to make differences between common(N_NN) and proper noun (N_NNP).
- An adjective (JJ) is tagged as a noun (N_NN) with a high error count because, in this language, adjectives may or may not occur before the nouns.
- All models can precisely assign almost all tags of categories Conjunction(CC), Demonstrative (DM), Quantifiers(QT) and Residuals.
- As the training data increases, the less number of unknown words will be encountered in the test corpus, which will increase the accuracy.
- Unlike English, Gujarati has a very complex grammatical structure and low resource availability, these are the reasons for the model get confused between some tags and having a higher error count in few tags.

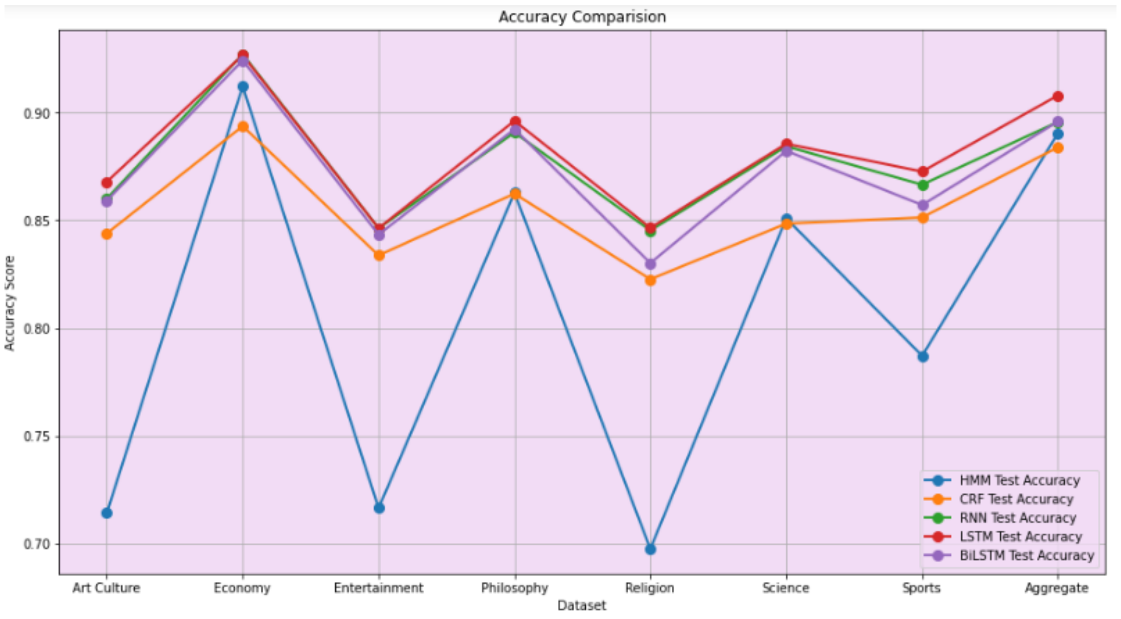


Figure 6.1: Accuracy Comparison Plot

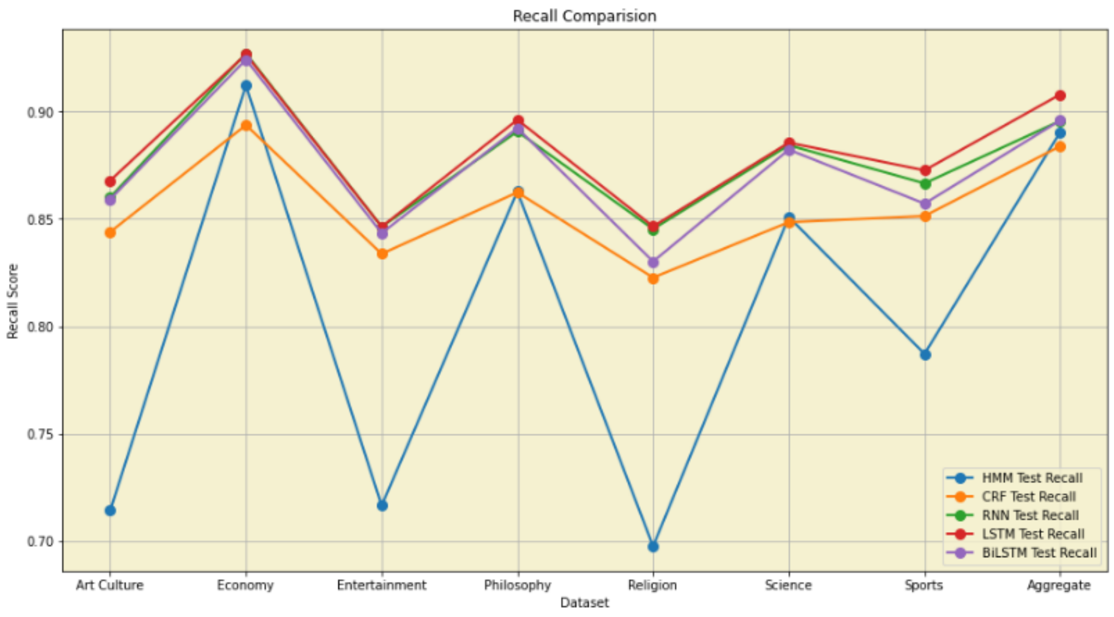


Figure 6.2: Recall Comparison Plot

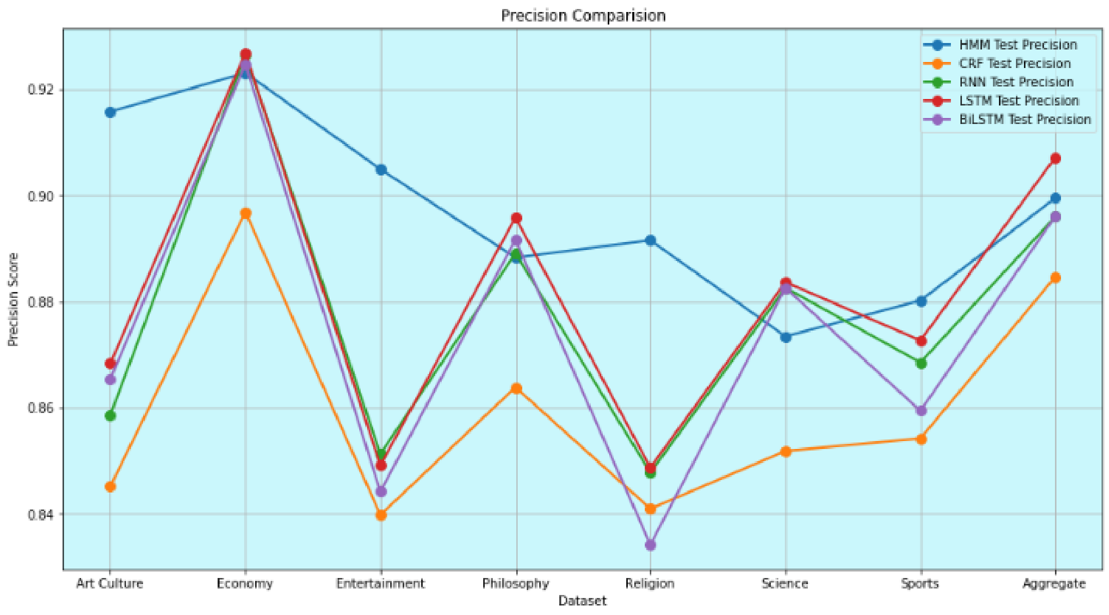


Figure 6.3: Precision Comparison Plot

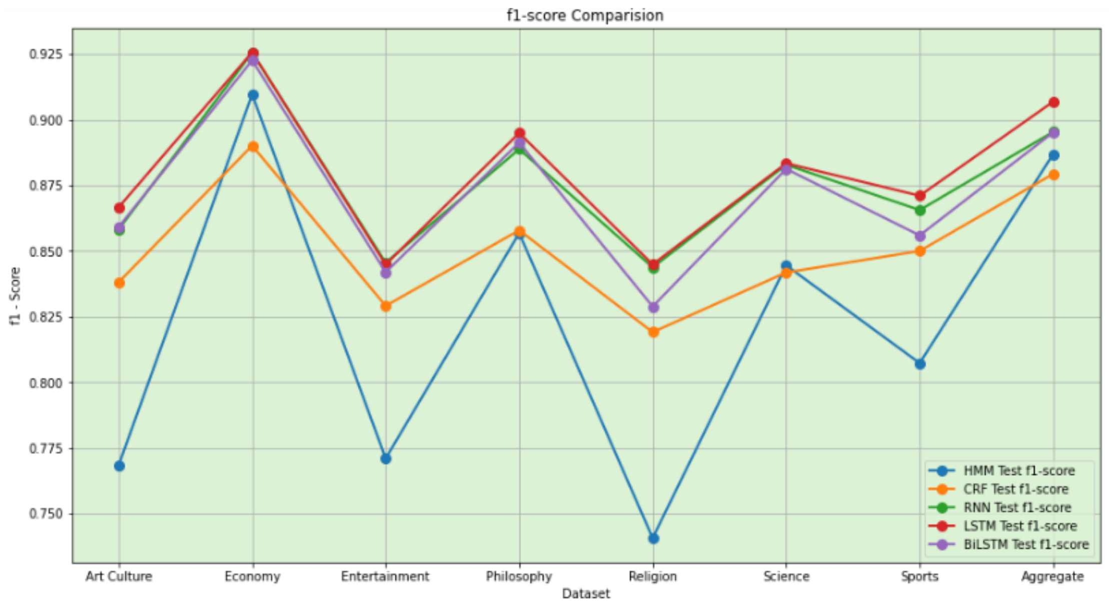


Figure 6.4: F1-Score Comparison Plot

7 REFERENCES

1. Part-Of-Speech Tagging for Gujarati Using Conditional Random Fields
`https://www.aclweb.org/anthology/I08-3019.pdf`

2. A Statistical Method for Evaluating Performance of Part of Speech Tagger for Gujarati
`https://www.ijrte.org/wp-content/uploads/papers/v8i2/B1492078219.pdf`

3. Hindi POS Tagger Using Naive Stemming : Harnessing Morphological Information Without Extensive Linguistic Knowledge
`https://www.cse.iitb.ac.in/~pb/papers/icon08-hindi-pos-tagger`

4. POS Tagging For Resource Poor Indian Languages Through Feature Projection
`https://www.researchgate.net/publication/323174678_POS_Tagging_For_Resource_Poor_Indian_Languages_Through_Feature_Projection`

5. Character-level Supervision for Low-resource POS Tagging
`https://pdfs.semanticscholar.org/a93a/3799ba977aa2393cad8cd260d6f778a495e0.pdf?_ga=2.249332687.936938462.1614443610-381175129.1614443610`

6. Character-level Supervision for Low-resource POS Tagging
`https://pdfs.semanticscholar.org/a93a/3799ba977aa2393cad8cd260d6f778a495e0.pdf?_ga=2.249332687.936938462.1614443610-381175129.1614443610`

7. Part-of-Speech Tagging using Conditional Random Fields: Exploiting Sub-Label Dependencies for Improved Accuracy
`https://www.aclweb.org/anthology/P14-2043.pdf`

8. Part of Speech (POS) tagging with Hidden Markov Model
`https://www.mygreatlearning.com/blog/pos-tagging/`

9. POS Tagging Using CRFs
`https://towardsdatascience.com/pos-tagging-using-crfs-ea430c5fb78b`

10. POS Tagging Using RNN
`https://towardsdatascience.com/pos-tagging-using-rnn-7f08a522f849`