Automated Error Correction and Validation for POS Tagging of Hindi

Sachi Angle
MIT, Manipal
sachiangle@gmail.com

Pruthwik Mishra
IIIT, Hyderabad
pruthwikmishra@gmail.com

Dipti Misra Sharma
IIIT, Hyderabad
dipti@iiit.ac.in

Abstract

The Part-Of-Speech tag of a word can provide crucial information for a large number of tasks, and so, it is of utmost importance that the POS tagged data is accurate. However, manually checking the data is a tedious and time consuming task. Thus, there is a need for an Automatic Error Correction and Validation model for any POS Tagged Data. In this paper, we work towards achieving the aforementioned goal for Hindi POS Tagging. This is achieved by using an ensemble model consisting of three POS Tagging Models. Based on the predictions made by the three models, and the POS tag present in the dataset, the ensemble model predicts the presence of an error. The POS tagging models explored were the Hidden Markov Model, Support Vector Machine, Conditional Random Fields, Long Short Term Memory (LSTM) Networks, Bidirectional LSTM Networks, and Logistic Regression. A Fully Connected Neural Network was used to build the ensemble model, and it achieved an accuracy of 94.02%.

1 Introduction

Part-Of-Speech (POS) tagging refers to categorizing words into the POS categories (e.g. noun, verb, or conjunction) they belong to. It forms an essential component of a large number of Natural Language Processing tasks, ranging from speech modeling, where they affect the pronunciation, to Information Retrieval, where they are helpful in stemming. Although extensive work has been done on POS tagging of English, there is still a lot of scope for advancement for the same with respect to Indian Languages.

Common models used for POS tagging include the Hidden Markov Model (HMM), Long-Short Term Memory (LSTM) Networks, and Conditional Random Fields(CRF). These models can efficiently tag the data as they take into account the previous states of data realized through words and tags. This greatly increases the accuracy with which the data is tagged. Significant work has been undertaken for Hindi POS Tagging, however not enough has been done yet to build an Automatic Error Correction and Validation model. Models explored to achieve this involve a lot of manual tagging and checking of data. Therefore, there is a need to build a model that reduces human effort. The method to be followed includes the predictions of three models for a particular context, and based on these predictions, check for the presence of an error. This model will automate error checking, thus ending the need for manual checking.

2 Literature Survey

Traditionally, rule-based tagging, stochastic tagging, and transformation-based tagging have been used for POS Tagging. Rule-based taggers generally involve a large database of hand-written disambiguation rules. On the other hand, a Stochastic Tagger relies on available data and on probability to resolve ambiguities.

Hidden Markov Model is one such Stochastic Tagger, used for POS tagging of data. An HMM uses the conditional probability of a tag being assigned to a given word, given the previous few tags that have already been encountered. Logistic regression is a predictive analysis of data. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. One such model, Support Vector Machine (SVM) is a machine learning model used for classification and regression. Conditional Random Field (Kudo, 2005) (CRF) is a probabilistic framework for labeling and segmenting data, by taking the context of the label to be predicted into account. Long Short Term Memory (LSTM) Networks are very efficient solutions for sequential prediction problems. LSTMs are a special kind of Recurrent Neural Network, capable of learning long-term dependencies. In Bidirectional LSTMs (BiLSTMs), the learning algorithm is fed with the original data from the beginning to the end, and then from the end to beginning.

Brants (2000) established that an HMM using trigrams performs at least as well as other known methods at POS tagging by building the Trigrams'n'Tags (TnT) model. TnT tagger relies on all possible trigrams having occurred in the dataset. For unseen trigrams the probability will be wrongly set to zero. To handle sparse data, different methods of smoothing are discussed, like Deleted Interpolation, which are used to redistribute the probabilities of the data such that possible trigrams have probabilities greater than 0. If a word has not occurred in the training data, the probability of that word being assigned a tag will be set to zero. In this case, the TnT uses the suffix of the unknown word to predict a POS tag for it based on the suffixes of the words in the training data.

In (Gadde and Yeleti, 2008), they used the TnT tagger, and Conditional Random Fields to generate POS tags for Hindi and Telegu. The effects of introducing features like Root, Gender, Number and Person of the word to the TnT are subsequently shown. The addition of these morph features enabled the model to achieve an accuracy of 91.35% with Hindi data, and 91.23% with Telegu Data.

Yao et al. (2002) built a model to detect the errors in Chinese data. This model built rules on when to correct the data based on the errors found in the training data, which consisted of predicted data as well as the actual tags, and the context in which the errors occurred. The rules generated were then added to a library of candidate rules.

Park et al. (1998) emphasized that one of the main reasons for inaccurate prediction of tags is a large number of unknown or previously unseen words. It built rules for error detection and correction by training on errors and their context. The data was then manually verified for errors, and detected errors, along with their contexts were added to the already built set of rules. This new set of rules consisting of both, the learnt rules as well as the manually formed rules, were then used to correct more data.

Agarwal (2012) suggested a hybrid model, which was a combination of a rule based model and a statistical model. Rules for rule based correction module were formed using detailed analysis of the development data. The statistical model initially predicted a list of possible tags for a word, along with their probabilities of being the tag. The probabilities of the two most likely tags were compared to a threshold value to detect the presence of errors.

In (Agrawal, 2016), a dependency parser was used to check for errors. If the parser performed well in some constructions, it would mostly give a correct decision and this decision could be compared to the annotator's decision . If the decision made by the annotator was different, it could be inferred that the annotation may be incorrect.

Through the Literature Survey, it becomes clear that there is a need for an Error Detection system that does not require manual effort during the training. Creation of such a system would greatly reduce the time spent on checking Hindi POS Tagged Data.

3 Methodology

To analyze if there are errors present in the Hindi data, we explore the following models - HMM, SVM, Logistic Regression, LSTMs, BiLSTMs, and CRF. The three models that perform the best are used in the ensemble model, and if two or more of the models find an error, then that row of data is said to contain an error. For this, the three models have to be built to achieve high accuracy. Smoothing methods such as Add One Smoothing, Good Turing Smoothing, and Deleted Interpolation can be used to improve the accuracy of the HMM. The accuracies of the SVM, Logistic Regression, and the CRF can be improved by experimenting with different sets of features. The LSTM and BiLSTM are built using the word sequences as features and the POS Tags as labels. Once the models have been built, the error analysis model has to be built by combining the three models. Four different methods are explored for the ensemble model. The first is a Voting Model which checks if the tag in the dataset is correct, by checking if it matches the tag predicted by the majority of the three POS Tagging models. The second uses an average of the probabilities the three POS tagging models predict for a certain tag to obtain the possible correct tag for that word. The third and the fourth use neural networks to predict the presence of an error or not. The ensemble model that performs the best can then be used to identify and correct errors in the Hindi data.

The Hindi data used consists of 1,187 documents each containing an average of 20 sentences from the publicly available hindi treebank [1]. The sentences are stored in Shakti Standard Format (Bharati et al., 2007), with each line containing a word in a sentence, along with additional features about the word. After extraction of each word and its features from these files, a data set of 457701 words (20882 sentences) is obtained. Two additional features, the previous tag $t_i - 1$ and the tag that appeared before that $t_i - 2$, are required to be extracted for each word. Additional features of the previous two tokens (words) and the following two tokens, i.e, a context window of size 5, are also extracted. Thus the final Hindi dataset consists of 'Root', 'Category', 'Gender', 'Person', 'Number', 'Case', 'Word', 'POS', 'Previous POS', 'Previous2 POS', 'Previous word', 'Previous2 word', 'Next Word' and 'Next2 word' features. A sample consisting of 3 rows from the data set can be seen in Table 1 and Table 2. SS in the tables below denotes 'start of the sentence'.

For the implementation of the Automatic Error Correction and Validation model, three submodels, from HMM, SVM, LSTM, BiLSTM, Logistic Regression, and CRF, are built.

3.1 Hidden Markov Model

The HMM is built using trigrams. To handle the zero probabilities due to missing bigrams and trigrams, various smoothing techniques have been applied. These include Add One Smoothing, Good Turing Smoothing, and Deleted Interpolation. To handle zero probabilities because of unknown words, an SVM model was trained on the features about the words. The structure of the word could give information about the most likely tag for a word. If the word has not been seen before, the probabilities associated with it in the HMM are all set to zero. The zero probabilities are replaced by the probabilities provided by the SVM.

3.2 Support Vector Machines

The second model used for Hindi POS tagging is Support Vector Machine (SVM). To implement this, the Sklearn Library (Pedregosa et al., 2011) is used. First, just the features about the word, i.e, the word, word category, gender, number, person, case, and the root are used. Subsequently, all the above features except the root and the word itself are used. The previous two encountered POS tags are also added to the feature set. Finally, all the features mentioned above are tried.

YAMCHA - Yet Another Multipurpose Chunk Annotator (Kudo and Matsumoto, 2005) - is a moderately high performance chunker based on Support Vector Machines. The feature sets (window-size), parsing-direction (for-

[1] http://ltrc.iiit.ac.in/treebank_H2014

root	category	gender	number	person	direct oblique	word
यह	pn	any	sg	3	d	यह
एशिया	n	m	sg	3	o	एशिया
का	psp	f	pl	any	o	का

Table 1: Dataset.

POS	POS-1	POS-2	word-1	word-2	word+1	word+2
DM_DMD	SS	SS	SS	SS	एशिया	का
N_NNP	DM_DMD	SS	यह	SS	का	सबसे
PSP	N_NNP	DM_DMD	एशिया	यह	सबसे	बड़ा

Table 2: Dataset - Continuation.

ward/backward) and algorithms of multi-class problem (pair wise/one versus rest) can be redefined according to the the task at hand.

3.3 Conditional Random Fields

The CRF++ toolkit (Kudo, 2005) is used with different sets of features for predicting the most probable pos tag sequence for a sentence. The features included the word, the prefixes of the word upto the length of 3, the suffixes of the word upto the length of 7, a binary feature stating whether the word is long or short (where long is greater than 4), and a context window of sizes 3 (the previous word and the next word) or 5 (the previous two words and the next two words). To use the CRF++ Tool, a template needs to be created describing the features in terms of the row and column of the dataset containing the mentioned features. The data can then be trained and tested.

3.4 Long Short Term Memory Network

The LSTM (Hochreiter and Schmidhuber, 1997) network was built, using the Keras library (Chollet and others, 2015), to process a sentence at a time, with the length of the lookback equal to the length of the longest sentence found in the entire dataset. The network used softmax activation, and categorical cross entropy loss. The BiLSTM (Graves and Schmidhuber, 2005) had a similar structure, and the Keras Library was used to build both the models.

3.5 Logistic Regression

The Logistic Regression POS Tagger was implemented using the Sklearn Library. First, the feature set - 'word, third person, category, case, gender, number, root, previous POS tag, the tag before the previous POS tag' - was used. Following this, to obtain a higher accuracy, word embeddings of all the words were used in the feature set. Word embeddings were obtained from FastText (Joulin et al., 2016).

After building all the models, an ensemble model of the three is built. Based on predictions made by each model for a particular data point, the data point is analyzed for errors. If majority of the models predict that the tag is different from the given tag of that word, that particular tag can be considered erroneous. Experiments are also conducted to take into account the confidence level of predictions, i.e, the probability with which a classifier has made the prediction, for the model's prediction to be considered as the correct tag. If all models have low confidence in the predictions, then this data point is clearly ambiguous and can be seen as a possible error. Three ensemble models were explored for the Error Detection - Voting Model, Probability Average Model, and a Neural Network Model to learn from the data of the errors.

3.6 Voting Model

The Voting Model comprises of three models, and three from a set of five classifiers consisting of HMM, SVM, LSTM, BiLSTM, and CRF were

used. If majority of the models‘ predictions for the a point match the tag in the dataset, the tag is deemed accurate. Otherwise, the tag is erroneous. If none of the models agree upon a tag, the pos tag of a token is clearly ambiguous, and thus a source of possible error.

3.7 Probability Average Model

Every model makes its predictions with probabilities. Each data point can be tagged with a POS tag with a certain probability. Thus, if you average the probabilities as scored by all models for a certain data point, it results in the average probability (AP) with which that data point could take on the corresponding tag. Following this, the tag with the maximum average probability (MAP) is the most likely tag for that data point. If this tag matches the tag in the dataset, the tag is not an error. Otherwise, it is.

$$AP = predicted\_probabilities(X)$$
$$MAP = max(AP)$$

3.8 Neural Network

Half of the datapoints in the dataset were randomly picked and their tags were set to any other tag. A separate binary error list is maintained, with 1 for every correct tag, and 0 for the presence of an error. Then for any token, the confidence probabilities of each model are fetched for the given tag. The confidence probability is the probability with which that tag would be predicted by that model. Three models are picked from the HMM, CRF, SVM, LSTM, and the BiLSTM. Thus the features are given POS tag of the word, Model-1 probability, Model-2 probability, Model-3 probability. Thus, the neural network trains on POS tagged data, and the probabilities with which a particular tag could be present, and is used to predict the presence of errors. The confidence of a model can be assessed by getting the difference between the probability of the most likely tag and the next most likely tag. The higher the difference, the more accurate the model. The difference between the probability of the most likely tag and the tag present in the data provides information on the tag-confidence on how accurate the present tag could be. The Neural Network is then provided with additional features of confidence and tag-confidence of each model and retrained. This model can accurately capture trends in the level of probabilities required for accurate predictions.

Of the various methods explored for POS Tagging, Logistic Regression proved to be the worst, and was therefore, omitted from the ensemble model. Using the remaining models, various ensemble models were experimented with. The Voting model only considered the tag predicted by majority models, the Probability Average Model considered the predicted probabilities for all tags, by all models, and then took into consideration the tag with the highest average probability. Neither model took into consideration the actual tag present while making its prediction. They only predict the most likely tag and then check for an error. The ensemble model built using Neural Networks, on the other hand, learned from the errors present in the data, and the corresponding model probabilities to predict whether an error has occurred or not.

4 Result Analysis

The HMM has been built, using trigrams and the three methods of Smoothing were tried. The TNT Tagger (Brants, 2000) achieved an accuracy of 95.1% on this dataset, while the HMM, using Add One Smoothing, with SVM probabilities for unknown words, achieved an accuracy percentage of 96.1%.

The SVM model was used with different sets of features. The first being just the features about the word, i.e, word, word category, gender, number, person, case, and the root (Feature Set A). The second, all the above features except the root and the word, along with the previous two encountered POS tags (Feature Set B). The third feature set included all the features mentioned above (Feature Set C). Feature set B achieved an accuracy of 88.42%, Feature set C achieved an accuracy of 91.84%, and Feature set A performed the best with 93.29% accuracy.

The CRF++ Tool was used with features including the word, the prefixes of the word upto

the length of 3, the suffixes of the word of lengths varying between 4 to 7, a binary feature stating whether the word is long or short (where long is greater than 4), and a context window of sizes 3 (the previous word and the next word) and 5 (the previous two words and the next two words). The experiments with different context windows and suffix lengths were conducted, and the highest accuracy of 97.2% was obtained with suffix length 7, and context window size 5. Remaining work includes building the ensemble model of the three described models, and analyzing their results to be able to identify errors in the data.

Logistic Regression was attempted initially with features about the words of the data - i.e. the word, third person, category, case, gender, singular or plural, root, the previous POS tag, and the tag before the previous POS tag - but this model could achieve an accuracy of only 26%. To improve this, fast text word embeddings were used, and the model trained on this achieved an accuracy of 81.64%.

The LSTM and BiLSTM performed the best, with the LSTM was 98.22% accurate and the BiLSTM, 98.59% accurate.

Therefore, HMM, SVM, LSTM, BiLSTM, and CRF++ were used for the ensemble model. Experiments were conducted with both, sklearn‘s SVM model and YamCha‘s SVM model.

Three methods were experimented with, for the ensemble model, the first being - The Voting Model (VM). The POS tag predicted by majority of the three models is considered to be the correct tag, and if it does not match the tag in the dataset, that tag is marked as an error.

Voting Model 1: Using the sklearn library for SVM,the HMM, and the CRF, 1232 data points were tagged as errors (No errors were intentionally introduced into the data). Therefore, 90.02% accuracy was achieved (Assuming no errors were present).

Voting Model 2: Replacing the SVM built using sklearn in the previous model with the SVM built using the YamCha Toolkit, 1094 data points were tagged as errors (No errors were intentionally introduced into the data). 91.14% accuracy was achieved.

Voting Model 3: The Voting model was used with the LSTM and the BiLSTM too. In this model, the HMM is replaced with the LSTM, and this ensemble model tagged 1102 data points as errors. This model has as accuracy of 91.07%.

Voting Model 4: The Voting Model was built with the CRF, SVM and BiLSTM models. 1145 errors were tagged, with the model achieving 90.2% accuracy.

The second method tried, the Probability Average Model (PAM) involved averaging the probabilities made by all models for a certain data point, resulting in the average probability with which that data point could take on that tag.

Average Probability Model 1: Using Sklearn library for the SVM, the HMM, and the CRF, 1137 data points tagged as errors (No errors were intentionally introduced into the data). 90.79% accuracy was obtained.

Average Probability Model 2: Replacing the Sklearn-SVM with the YAMCHA-SVM, 1740 data points tagged as errors (No errors were intentionally introduced into the data). 85.91% accuracy was obtained.

Average Probability Model 3: Using the LSTM instead of the HMM, 1648 data points tagged as errors (No errors were intentionally introduced into the data). 86.65% accuracy was obtained.

Average Probability Model 4: When the BiLSTM is used instead of the LSTM, 1670 data points were tagged as errors, and thus, 86.47% accuracy was obtained.

The third model was built using a neural network (NN1). The neural network trained on data including whether an error was present or not, and the predicted probabilities with which an error could be present.

Neural Network 1: For this model, only YamCha was used as it had performed better individually than the Sklearn model, along with the HMM and CRF models. Errors were introduced into the data at random. The POS Tag present, and the probabilities with which each model would predict the tag present were used as features. The output class, 0 and 1, are used

Voting Model : Sub-Models Used	Accuracy
SVM(sklearn), HMM, CRF	90.02%
SVM(YAMCHA toolkit), HMM, CRF	91.14%
SVM(YAMCHA toolkit), LSTM, CRF	91.07%
SVM(YAMCHA toolkit), Bi-LSTM, CRF	90.20%

Table 3: Voting Model Accuracies.

Average Probability Model : Sub-Models Used	Accuracy
SVM(sklearn), HMM, CRF	90.79%
SVM(YAMCHA Toolkit), HMM, CRF	85.91%
SVM(YAMCHA Toolkit), LSTM, CRF	86.65%
SVM(YAMCHA Toolkit), Bi-LSTM, CRF	86.47%

Table 4: Average Probability Model Accuracies.

for error and no error, respectively. This model predicted the presence of an error with 91.37% accuracy.

Neural Network 2: When this model is tried with the CRF, SVM, and LSTM, and the model had 93.19% accuracy.

Neural Network 3: By replacing the LSTM with the BiLSTM, 90.86% accuracy was obtained.

The model was improved by adding more features (NN2). The confidence of the model (difference in probabilities of the highest probability tag and second highest probability tag) and the difference in probabilities of highest probability tag and the probability of the particular tag present in the dataset (the one being checked for error or not) were added for each model.

Neural Network (with additional Features) 1: Using HMM, CRF, and YAMCHA-SVM models, 93.64% accuracy was achieved.

Neural Network (with additional Features) 2: Using an LSTM instead of the HMM, 94.02% accuracy was obtained.

Neural Network (with additional Features) 3: By replacing the LSTM with the BiLSTM, 93.97% was achieved.

Thus, the Improved Neural Network model 2, the neural network with additional features comprising of the LSTM, CRF, and SVM performed the best with 94.02% accuracy, resulting in this being the best model of the ones under consideration.

5 Conclusion

Error Detection in Hindi POS Tagged data is achieved by using an ensemble model consisting of three POS Tagging Models. Based on the predictions made by the three models and on the probabilities with which the models predict the tag present, if the predicted tag fails to match the tag in the dataset, then that data point is tagged as an error. To find three models that could accurately predict the POS tags, the Hidden Markov Model, Support Vector Machine, Conditional Random Fields, and Logistic Regression were explored. The ensemble model was built using a Fully Connected Neural Network.

The Error Detection and Correction model, built with the Neural Network training on the results of the individual models and their predicted probabilities, has achieved an accuracy of 94.02% and can accurately identify majority of the errors present in a dataset, thus, reducing the amount of human effort required in cleansing the data to a minimum.

References

Rahul Agarwal. 2012. *Automatic Error Detection for Treebank Validation.* Ph.D. thesis, PhD thesis, International Institute of Information Technology Hyderabad.

Neural Network : Sub-Models Used	Accuracy
SVM(YAMCHA Toolkit), HMM, CRF	91.37%
SVM(YAMCHA Toolkit), LSTM, CRF	93.19%
SVM(YAMCHA Toolkit), Bi-LSTM, CRF	90.86%

Table 5: Neural Network Model Accuracies.

Neural Network Model 2 : Sub-Models Used	Accuracy
SVM(YAMCHA Toolkit), HMM, CRF	93.64%
SVM(YAMCHA Toolkit), LSTM, CRF	94.02%
SVM(YAMCHA Toolkit), Bi-LSTM, CRF	93.97%

Table 6: Neural Network (with additional Features) Accuracies.

Bhasha Agrawal. 2016. *Error Detection and Dependency Parsing*. Ph.D. thesis, International Institute of Information Technology Hyderabad.

Akshar Bharati, Rajeev Sangal, and Dipti M Sharma. 2007. Ssf: Shakti standard format guide. *Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India*, pages 1–25.

Thorsten Brants. 2000. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. Association for Computational Linguistics.

François Chollet et al. 2015. Keras.

Phani Gadde and Meher Vijay Yeleti. 2008. Improving statistical pos tagging using linguistic feature for hindi and telugu. *Proc. of ICON*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Taku Kudo and Yuji Matsumoto. 2005. Yamcha: Yet another multipurpose chunk annotator. *URL: http://chasen. org/~ taku/software/YamCha/(accessed 2011 May 9)*.

Taku Kudo. 2005. Crf++: Yet another crf toolkit. *http://crfpp. sourceforge. net/*.

Junsik Park, Jung-Goo Kang, Wook Hur, and Key-Sun Choi. 1998. Machine aided error-correction environment for korean morphological analysis and part-of-speech tagging. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 1015–1019. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Tianfang Yao, Wei Ding, and Gregor Erbach. 2002. Correcting word segmentation and part-of-speech tagging errors for chinese named entity recognition. In *The Internet Challenge: Technology and Applications*, pages 29–36. Springer.