Part-Of-Speech Tagging using Neural network

by

Ankur Parikh

in

International Conference on Natural Language Processing
(ICON-2009)


Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2009
Part-Of-Speech Tagging using Neural network

Ankur Parikh
Language Technologies Research Centre
IIIT Hyderabad, Gachibowli, Hyderabad, Andhra Pradesh, India 500032
ankur.parikh85@gmail.com

Abstract
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. To the best of my knowledge, this is the first time Neural network is used for POS tagging of Hindi language. In this paper, a single-neuro tagger, a Neural network based POS tagger with fixed length of context chosen empirically is presented first. Then, a multi-neuro tagger which consists of multiple single-neuro taggers with fixed but different lengths of contexts is presented. Multi-neuro tagger performs tagging by voting on the output of all single-neuro taggers. The experiments carried out are discussed. It is shown that multi-neuro tagger performs as well as the HMM based tagger and the CRF based tagger and it performs slightly better than any of the single neuro tagger with fixed context. It is also shown that training cost of multi-neuro tagger is same as that of single-neuro tagger.

1 Introduction
POS (Parts of Speech) tagging is the task of assigning grammatical classes to words in a natural language sentence based on both its definition and context. POS tagging can be handled either at word-level or sentence-level. At word-level POS tagging is posed as a classification problem in which an appropriate tag for a word is found where as at sentence level a series of tags corresponding to the sequence of words are obtained. The intricacies in POS tagging arise from insufficient training data, inherent POS ambiguities and unknown words which are ubiquitous. POS tagging is an important step in many natural language processing applications such as speech recognition, speech synthesis, information extraction, machine translation, question answering, information retrieval etc. It is also the primary step in the syntactic analysis of a language.

Hindi, the national language of India is a relatively free word order language. Therefore, many permutations of the same sentence can convey similar meaning. Hindi is also morphologically rich which allows generation of several words by combining various morphemes that may not be present in the training data. These phenomemons further increase the complexity of POS tagging.

For corpus rich languages like English, there have been many implementations of POS tagger, using machine learning techniques such as Second Order Markov Models based tagger (Church, 1985) and (Kempe, 1993), Neural network based tagger (Schmid, 1994), Probabilistic Triclass Model based tagger (Merialdo, 1994), Transformation Based Error-Driven Learning based tagger (Eric, 1995), Maximum Entropy Markov Model based tagger (Adwait, 1996), Memory Based Learning based tagger (Daelemans and Gillis, 1996), Markov Models based tagger with smoothing techniques and methods to handle unknown words (Brants, 2000) SVM based tagger (Tetsuji Nakagawa and Matsumoto, 2001).


There has been some previous work towards building a POS tagger for Indian languages, such as the partial POS tagger (P. R. Ray and Sarkar,
(M. Shrivastava and Bhattacharya., 2005) propose harnessing morphological characteristics of Hindi for POS tagging in (M. Shrivastava and Bhattacharya., 2005). This was further enhanced in (Smriti Singh and Bhattacharyya, 2006), which suggests a methodology that makes use of detailed morphological analysis and lexicon lookup for tagging, HMM based tagger for Bengali (Sandipan Dandapat, 2006), CRF based tagger with TBL as post processing unit (PVS and Gali, 2007), Decision Forests (Pammi and Prahallad, 2007), Hybrid which uses HMM and rule based approach (Pattabhi R K Rao T and L, 2007). Gadde and Vijay improve accuracy of HMM based tagger by adding linguistic features and handling compound words effectively in (Phani Gedde, 2008).

HMM and CRF based tagger are the two major statistical POS tagging approaches used for Indian languages. It was proved that CRF performs better than HMM as it takes linguistic features along with words (PVS and Gali, 2007) (Phani Gedde, 2008) (Agarwal and Mani, 2006). However training time of CRF based tagger is in the order of hours whereas training time of HMM based tagger is in the order of seconds. As POS tagging is the primary step in many applications, less accuracy at this stage affects later stages. To build a POS tagger for new corpus or a new language, tuning of various parameters (length of context) is a nontrivial task. Thus, the need of the hour is an approach which minimizes the training time and yields higher accuracy.

Hindi, unlike English, belongs to the category of inflectionally rich languages which suffer from data sparseness problem. Neural network is one of the most efficient approaches for learning from a sparse data. As mentioned above, Hindi is a relatively free word order language, it requires an approach which provides variable lengths of contexts. Most of the previous approaches used for POS tagging of Hindi were unable to capture this, but Neural network is quite capable of handling these issues. Corpus based features play an important role to achieve adequate accuracy. In a HMM based tagger it is cumbersome to add these features while in a CRF based tagger such features lead to huge training time. Interestingly, in Neural network with clever encoding scheme such features can be provided in input implicitly so that it requires less training time.

This paper shows a Neural network based approach which learns the parameters of POS tagger from a representative training data set whose training time is in the order of minutes (10-13) and performance is better than HMM based tagger and very close to CRF based tagger. Single-neuro tagger with fixed length of context chosen empirically is a Neural network based POS tagger. It can produce an alternative tag without any additional cost, if the decision between two tags is difficult. Unlike most of the statistical models proposed so far for Indian languages in which length of context is fixed and chosen empirically, multi-neuro tagger is composed of multiple single-neuro taggers with fixed but different lengths of contexts and a voting based selection rule to obtain the final output.

The system presented here performs POS tagging locally i.e. at word level. Most of the statistical taggers which use Markov models perform POS tagging globally i.e. at sentence level. Advantage of sentence level POS tagging is that it is linguistically more plausible. But drawback is that tagging errors done at the start-of-sentence is propagated through out the whole sentence which is not the case in word-level POS tagging. On the other hand, drawback of word-level tagging is that it allows quite implausible combinations of tags which are unacceptable in the sentence-level model. Experiments done by (Merialdo, 1994) shows that the tagging accuracy do not seem to be seriously affected by above drawback.

A MLP network with three layers is used as a single-neuro tagger and is trained in supervised manner with well known Error Backpropagation Learning algorithm. Java programs have been implemented in order to prepare lexicons, implement MLP network and error backpropagation learning algorithm and to achieve the results shown in this paper.

The paper is arranged as follows; section 2 describes the data, tools and resources used, section 3 describes MLP and error backpropagation learning algorithm. Section 4 describes the approach followed. Section 5 describes Single-neuro tagger. Multi-neuro tagger is described in section 6. Results are discussed in section 7. Error analysis of the results and analysis of the Neural network based approach is discussed in section 8 and 9 respectively. The paper is concluded in section 10.
2 Experimental Setup

2.1 Data used for experiments

The training, development and testing corpus for the current experiments were provided by ILMT. Size of the training, development and testing corpus were 187,095, 23,565 and 23,281 words respectively. Percentage of unknown words in the development and testing corpus were 5.33% and 8.15% respectively. Tag set of 25 tags built by IIIT-Hyderabad are used.

2.2 Tools used for experiments

1. Morfessor Categories-MAP, a tool which finds stem, suffix and prefix from unannotated text (Creutz and Lagus, 2005) was used to handle unseen words.
2. Brant’s TnT (Brants, 2000) a HMM based tagger and CRF++, a CRF based tagger were used to compare performance of the presented taggers.

2.3 Resources used for experiments

The following resources were used to handle unseen words.
1. Universal Word - Hindi Dictionary, a lexicon which provides the grammatical, morphological and semantic attributes of the Hindi words. (Nitin Verma, 2002).
2. Wordnet which gives different relations between synonym sets which represent unique concepts. (Nitin Verma, 2002).

3 Neural Network

3.1 Multilayer Perceptron Network

The MLP Network consists of at least three layers, input layer, hidden layer and output layer, where each layer intern consists of elementary processing units. The elementary processing unit represents the model of an artificial neuron and incorporate a nonlinear activation function, the sigmoid function in this case. An activation value is associated with every processing element. The MLP network presented here is fully connected i.e., all neurons of the one layer are connected to the every neuron of the adjacent layer by weighted direct links. Working of an MLP can be described as successive projections of the data to be classified into different spaces. The neurons of the input layer receive inputs from the external environment while, the neurons of the hidden layer participate in the projection. The neurons of the output layer participate in the separation of classes.

3.2 Error Back-propagation Learning Algorithm

The Error BackPropagation Learning Algorithm is used to train multilayer perceptron network. It is based on an error-correction learning rule and specifically on the minimization of the mean squared error that is a measure of the difference between the actual and the desired output. Each application of the input algorithm performs two passes, forward pass and backward pass where input can be a pattern or a set of patterns. In the forward pass weights of the directed links remain unchanged and at each processing unit j of the hidden layer weighted outputs from all the neurons of input layer are added and a bias parameter \( \theta_h \) is subtracted. The resulted value is then passed through a sigmoid function in order to restrict the value range to the interval [0, 1].

\[
Net_h = \sum_{i=1}^{n} a_i W_{ih} - \theta_h \quad (1)
\]

\[
Y_h = \frac{1}{1 + e^{-Net_h}} \quad (2)
\]

Then at each processing unit j of the output layer weighted outputs from all the neurons of hidden layer are added and a bias parameter \( \theta_o \) is subtracted. The resulted value is then passed through a sigmoid function in order to restrict the value range to the interval [0, 1].

\[
Net_o = \sum_{h=1}^{n} a_h W_{ho} - \theta_o \quad (3)
\]

\[
Y_o = \frac{1}{1 + e^{-Net_o}} \quad (4)
\]

Subscript i, h and o is used to represent neurons of input, hidden and output layer respectively. \( Y_o \) is obtained output after performing forward pass and \( D_o \) is desired output.

The backward pass, on the other hand, starts at the output layer. First, for each output neuron local gradient is calculated.

\[
Grad_o = error_o (Y_o) (1 - Y_o) \quad (5)
\]

\[
error_o = D_o - Y_o
\]

Then for each hidden neuron local gradient is calculated.

\[
Grad_h = error_h (Y_h) (1 - Y_h) \quad (6)
\]

\[
error_h = (\sum_{o=1}^{n} Grad_o * W_{ho})
\]

To speedup the training process momentum term \( \alpha \) is introduced into the weight update formula and bias update formula. \( \eta \) is a learning rate.

\[
\Delta W_{ho}(t) = \eta * Grad_o * Y_h + \alpha * \Delta W_{ho}(t-1) \quad (7)
\]

\[
W_{ho}(t) = W_{ho}(t-1) + \Delta W_{ho}(t) \quad (8)
\]
\[
\Delta W_{ih}(t) = \eta * \text{Grad}_h * Y_i + \alpha * \Delta W_{ih}(t-1) \tag{9}
\]
\[
W_{ih}(t) = W_{ih}(t-1) + \Delta W_{ih}(t) \tag{10}
\]
\[
\Delta \theta_h(t) = \alpha * \Delta \theta_h(t-1) + \eta * \theta_h * \text{Grad}_h \tag{11}
\]
\[
\theta_h(t) = \theta_h(t-1) + \Delta \theta_h(t) \tag{12}
\]
\[
\Delta \theta_o(t) = \alpha * \Delta \theta_o(t-1) + \eta * \theta_o + \text{Grad}_o \tag{13}
\]
\[
\theta_o(t) = \theta_o(t-1) + \Delta \theta_o(t) \tag{14}
\]

For detailed introduction to MLP networks see e.g. (Haykin, 2001).

4 Approach

4.1 Handling of Compound Words

(Gedde and Vijay, 2008) improved accuracy of the HMM and CRF based tagger by handling compound words effectively. The same approach is also followed here. In this way XC tag used for compound words is removed from the corpus and size of the tagset is reduced by 1.

4.2 Lexicon

First Information such as root of the word is added to the training corpus. Then two lexicons from the training corpus, one for words and lexicon for the roots of the words are prepared. For each unique word \(w\) of the training corpus, there is one entry in the lexicon which is an \(n\)-element vector \(\text{INPUT} = (t_1, t_2, \ldots, t_n)\), where \(n\) is the total number of tags.

\[
t_j = \frac{c(pos_j, w)}{c(w)}
\]

Where \(c(pos_j, w)\) is the number of occurrences of \(w\) tagged as \(pos_j\) in the training data and \(c(w)\) is the number of occurrences of \(w\) in the same. Same approach was followed to prepare lexicon for root of the words.

4.3 Single-neuro tagger

Initially basic features like contextual features and corpus based features are used in single-neuro tagger. Later, word features like root of the word and length of the word are added. Then single-neuro tagger is trained using these features. The unseen words of the development and testing data are handled using strategy discussed in section 5.5.

4.4 Multi-neuro tagger

Multi-neuro tagger is composed of seven single-neuro taggers with fixed but different contexts. These taggers are trained using the features described in section 4.3. Strategy discussed in section 5.5 is capable of handling unseen words of development and testing data. Voting based selection rule is applied on output of all the taggers to obtain the final output. To reduce the training cost of multi-neuro tagger and to make the training cost comparable to that of a single-neuro tagger new learning method in which the weights of the trained single-neuro tagger with short context is used as initial weights for the single-neuro tagger with long context was used.

4.5 HMM and CRF based tagger

Finally, to compare the performance of presented taggers against HMM and CRF based tagger various experiments mentioned in (Gedde and Vijay, 2008) for HMM and (Himanshu and Mani, 2006) for CRF are carried out. To handle unseen words strategy discussed in section 5.5 is followed.

5 Single-Neuro tagger

5.1 Structure

The input to the MLP network (see Fig.1) consist of words which fall into a context of fixed length, root and length of the target word to be tagged. Each neuron in the output layer corresponds to one of the tags in tag set. The well known error backpropagation learning algorithm is used to assign correct POS tag to the target word by adapting weights of the direct links between the units of the adjacent layers.

5.2 Representation of the INPUT and OUTPUT data

As input to the Neural network takes numerical values encoding of input word into a suitable form, which the network can identify and use is essential. A single-neuro tagger takes numerical values as input, which is obtained by encoding the words using prior tag probabilities. The contextual probabilities are left for being learned from the training corpus. Each word \(w\) from the corpus is encoded as an \(n\)-element vector \(\text{INPUT} = (t_1, t_2, \ldots, t_n)\) where \(n\) corresponds to the total number of tags. Here, \(t_j\) is the prior probability of the word \(w\) that corresponds to the tag \(pos_j\) which is estimated from the training data. If the word \(w\) appears in the training data, the vector \(\text{INPUT}\) comes from the lexicon of words as mentioned in section 4.2. Else, \(N(w)\) is obtained by the method discussed in section 5.5.

Where, \(N(w) = \text{number of possible POS tags that can be assigned to the word } w\).
Figure 1: Structure of single-neuro tagger. Legend $n_i$, $n_o$, and $n_h$ indicate the nodes in the input, hidden and output layers. $l$ and $r$ are lengths of the left and right context for the target word. $i_p$, $i_r$, and $i_l$ indicate root and length of the target word respectively

$$t_j = 1/N(w) \text{ if } pos_j \text{ is a candidate}$$
$$= 0 \text{ otherwise}$$

Similar approach is followed for root (rt) of the word.

Instead of actual length of the word, length indicator (lt) is used.

Value of length indicator
$$= 1, \text{if } length \geq 3$$
$$= 0, \text{otherwise}$$

In this way, the number of neurons in the output layer, $n_o = n$ and number of neurons in the input layer, $n_i = n^* (l + 1 + r)$ for context + n for root + 1 for length indicator. Here $l =$ number of previous words and $r =$ number of next words.

The encoding scheme for the desired output $D = (d_1, d_2, . . . . d_n)$, an n-element vector is as follows.

$$d_j = 1, \text{if } POS_j \text{ is a correct answer}$$
$$= 0, \text{otherwise}$$

The activation value of each neuron is passed to sigmoid function which restricts the range to the $[0, 1]$. The decoding scheme to obtain POS of the target word is described below.

Result $= POS_j$, if $o_j = \max(OUTPUT)$, where

$$OUTPUT = (o_1, o_2, . . . . o_n).$$

5.3 Training

A single-neuro tagger is trained using error back-propagation supervised learning algorithm. Estimating the best weights which correctly predict the output units with respect to the input is the prime objective. The network is trained in sequential mode. The weights are updated according to equations (8) and (10) of section 3.2 after each sequence. The presentation of entire training set during the learning process is called an epoch. Cross validation is used as a stopping criteria for training. In sequential mode, though shuffling of training patterns from one epoch to the next is preferred, it is avoided as it decreases the accuracy of this system. The momentum term is introduced to speed up and stabilize the convergence. The value of momentum rate and learning rate is empirically chosen as 0.1 and 0.4 respectively.

5.4 Tagging

To perform tagging, first prior tag probabilities of the word, its neighbours, root and the value of the length indicator are passed to the input units. After that only forward pass of the error backpropagation learning algorithm is allowed and output neuron with largest value is found. The tag corresponding to this output neuron is finally attached to the current word.

If the second largest value in the output layer is close to the largest one, the tag corresponding to the second largest value may be given as an alternative output. So, multiple outputs or a sorted list of all tags as output may be given without any additional computation and the final decision can be delayed to a later processing stage like chunker, parser or a rule based post processing system can be used to select the most appropriate tag.

5.5 Experiments

A crucial aspect of the POS tagging by machine learning is to identify the appropriate facts about the data called features. Features can be contextual features, word features, corpus based features, or specialized features which capture lexical and morphological properties of a language. In this section best features along with best structure which can capture these features are found. Experiments using training and development data and final model are tested on testing data. In this section the results are reported over development data.

**Contextual and Corpus-based features**

Corpus based features play a crucial role in POS tagging. The features include all possible tags of the word, an indicator which indicates whether
word has only a single tag or word is a proper-noun in the training corpus etc.

It is very difficult to add corpus based features in HMM based tagger. However we can add these features in CRF based tagger but it increases training time. The encoding scheme chosen here captures corpus based features implicitly, which reduces the training time. This shows that the encoding scheme used for the input is elegant.

**E1: Contextual features**

Words are often ambiguous in their part of speech. In most of the cases this ambiguity is resolved by the context of usage. Initially we trained a single-neuro tagger over local neighborhood of the current word with a window size of 3, 4, 5 and 6. It was observed that a single-neuro tagger gave better results with a window size of 4 and the context includes 1 previous word, current word and 2 next words. In this experiment for unseen words equal probabilities are assigned to each tag. Using these basic features the accuracy was 93.19%.

**Word Features**

**E2: Root of the word**

To reduce errors occurred in disambiguating NNPs from NNs, some more information was required. In English, proper nouns are capitalized which is a very useful information. In Hindi no such help is available. It was observed that for all NNPs the root of the word is same as the word and for most of the NNs the root of the word is different from the word. Hence we added root of the word information. Unseen words are handled as discussed in E1. After this experiment the observed accuracy was 93.38%.

**E3: Length of the word**

First we included word length as a feature. But the performance of the system was decreased. Later we used length indicator as a feature because in Indian languages the average length of non-functional word is around 3. Value of length indicator is set to 0 for words having length less than 3 and 1 for words having length more than or equal to 3. Unseen words are handled as discussed in E1. After this experiment the observed accuracy was 94.04%.

**E4: Strategy used to handle seen words**

First the root of the unseen word is searched in the root lexicon. If it is found then prior tag probabilities of the root is passed as an input. If it is not the case then unseen word is searched first in Universal Word-Hindi Dictionary then in Wordnet and finally in the morph-analyzer. If it is found then a list of possible POS for that word is prepared and N(w) which is the number of possible POS tags that can be assigned to the word w is calculated. Then as discussed in section 5.2 prior tag probabilities are prepared and passed as an input. If it is not the case then unseen word is passed to the model of morfessor which was built after performing unsupervised training on the training corpus. The output of a model is either suffix s or prefix p or both. Using these information INPUT = (t₁, t₂, , tₙ) is prepared as follows.

\[ t_j = \frac{c(pos_j, s) + c(pos_j, p)}{c(s) + c(p)} \]

Where c(pos, s) is the number of occurrences of word w which contains suffix s tagged as pos_j in the training data. c(s) is the number of occurrences of word w which contains suffix s in the training data. Same for prefix. If it is not the case then equal probabilities are used and INPUT = (t₁, t₂, , tₙ) is prepared as follows.

\[ t_j = \frac{1}{n}, \text{ where } n \text{ is the number of tags in the tag set.} \]

The accuracy observed after this experiment was 95.62%. Figure 2 shows the development of the system.

**Handling of unseen words**

Tagging of unseen words is one of the classical problems in the computational linguistics. The strategy used to handle unseen words uses resources like Universal Word - Hindi Dictionary, Wordnet and tools like Morfessor.

![Development of the System](image-url)
6 Multi-neuro tagger

6.1 Structure

Figure 3 shows the multi-neuro tagger composed of m single-neuro taggers with fixed but different lengths of contexts. The individual $SNT_i$ has input $INPUT_i$ which consists of contextual features, corpus-based features and word features and for which the following relation holds. Context length($INPUT_i$) is less than or equal to context length($INPUT_j$) for i less than j. Encoding and decoding scheme discussed in section 5.2 is used. When all $OUTPUT_i$ are decoded $RESULT_i$ are obtained which are nothing but tags chosen by different $SNT_i$. Then, all $RESULT_i$ are inputed into the voting based selector to obtain final result.

6.2 Training

If all single-neuro taggers of the multi-neuro tagger are trained independently then the multi-neuro tagger requires more training time than that of a single-neuro tagger with fixed length of context. To reduce the training cost of the multi-neuro tagger up to a single-neuro tagger new learning method is used. First a single-neuro tagger with small input is trained independently and then its trained weights are used as initial weights for a single-neuro tagger with large input. Figure 4 shows an example of training a tagger which contains four words, root and length of the word as inputs. The trained weights, w1 and w2, of the tagger which contains three words, root and length of the word as inputs are copied to the corresponding part of the tagger and used as initial values for its training. Weights w3 are initialized with random values. For every odd context (context of size 3, 5, 7, etc.) 24 neurons in the hidden layer are added to increase discriminative power of the Neural network. Weights from the input layer to these added neurons and weights from these added neurons to the output layer are initialized with random values.

6.3 Voting based Selector

If the multi-neuro tagger is used for tagging then we will have multiple outputs. To obtain the final output some selection rule is required. First we assigned priority to every single-neuro tagger which are the part of the multi-neuro tagger, based on its precision. Then voting scheme is used as follows. The tag given by any single-neuro tagger is considered as a vote for that tag. The tag which get highest vote is selected as a winner. In the case of tie priority of the tagger is considered to select the winner.

7 Results

The performance of a single-neuro tagger which is trained independently is shown in Table-1. The interpretation of notation x-y-z used in structure field is the number of neurons in input-hidden-output layer. The interpretation of notation x_next used in context field is that there are more next words in the context of length x. Same interpretation for x_prev. Table-2 shows the performances of all single-neuro taggers which are part of the multi-neuro tagger and trained using previous results. After applying voting based selection rule the accuracy achieved was 95.78% for develop-
ment data and 92.19% for testing data. This result clearly shows that the precision of the multi-neuro tagger is slightly better than any of the single-neuro tagger with fixed length context. Table 3 shows comparison of HMM, CRF and Neural network based tagger. From table-3 we can conclude that Neural network based tagger performs as well as the HMM based tagger and the CRF based tagger.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Context</th>
<th>Development</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>121-48-24</td>
<td>4_next</td>
<td>95.62%</td>
<td>91.89%</td>
</tr>
</tbody>
</table>

Table 1: Performance of a single-neuro tagger trained independently

<table>
<thead>
<tr>
<th>Structure</th>
<th>Context</th>
<th>Development</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>97-48-24</td>
<td>3</td>
<td>95.44%</td>
<td>91.87%</td>
</tr>
<tr>
<td>121-48-24</td>
<td>4_prev</td>
<td>95.64%</td>
<td>92.05%</td>
</tr>
<tr>
<td>121-48-24</td>
<td>4_next</td>
<td>95.66%</td>
<td>91.95%</td>
</tr>
<tr>
<td>145-72-24</td>
<td>5</td>
<td>95.55%</td>
<td>92.15%</td>
</tr>
<tr>
<td>169-72-24</td>
<td>6_prev</td>
<td>95.56%</td>
<td>92.14%</td>
</tr>
<tr>
<td>169-72-24</td>
<td>6_next</td>
<td>95.54%</td>
<td>92.14%</td>
</tr>
<tr>
<td>193-96-24</td>
<td>7</td>
<td>95.46%</td>
<td>92.07%</td>
</tr>
</tbody>
</table>

Table 2: Performances of all single-neuro taggers which are the part of multi-neuro tagger and trained using previous results.

<table>
<thead>
<tr>
<th>Tagger</th>
<th>Development</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
<td>95.18%</td>
<td>91.58%</td>
</tr>
<tr>
<td>Multi-Neuro</td>
<td>95.78%</td>
<td>92.19%</td>
</tr>
<tr>
<td>CRF</td>
<td>96.05%</td>
<td>92.92%</td>
</tr>
</tbody>
</table>

Table 3: Comparison of HMM, CRF and Neural network based tagger

8 Error Analysis

Error Analysis for the multi-neuro tagger is shown in table 4.

<table>
<thead>
<tr>
<th>Actual Tag</th>
<th>Assigned Tag</th>
<th>Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNP</td>
<td>NN</td>
<td>624</td>
</tr>
<tr>
<td>NN</td>
<td>NNP</td>
<td>133</td>
</tr>
<tr>
<td>NN</td>
<td>JJ</td>
<td>103</td>
</tr>
<tr>
<td>JJ</td>
<td>NN</td>
<td>90</td>
</tr>
<tr>
<td>NNP</td>
<td>JJ</td>
<td>65</td>
</tr>
<tr>
<td>VAUX</td>
<td>VM</td>
<td>62</td>
</tr>
<tr>
<td>NN</td>
<td>VM</td>
<td>54</td>
</tr>
<tr>
<td>NNP</td>
<td>VM</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 4: Error Analysis for the multi-neuro tagger.

Figure 5: Learning curves of a single-neuro tagger with context of 4_next

Epochs required to train first a single-neuro tagger of context 3 and then, epochs required to train a single-neuro tagger of context 4_next which uses trained weights of a single-neuro tagger of context 3 as initial weights. From the figure we can conclude that the training time can be reduced greatly by using previous results.

As we developed our model using development data, the best context for a single-neuro tagger in both cases, trained independently and trained using previous results was 4_next. Using this context the maximum precision achieved on test-data was 91.95% which was 0.24% less than that of the multi-neuro tagger. It shows that the multi-neuro tagger performs slightly better than a single-neuro tagger with best context, which was decided from development data. From the results it is also clear that it performs slightly better than any single-neuro taggers of fixed but different lengths of contexts. The main drawback of the multi-neuro tagger was that it requires more memory than a single-neuro tagger.

9 Discussion

9.1 Single-neuro tagger and Multi-neuro tagger

Figure-5. shows the learning curves of single-neuro taggers with context of 4_next. Dotted line shows the case in which a single-neuro tagger is learned independently and solid line shows the
9.2 Multi-neuro tagger and other taggers

If the context of five words is chosen for tagging and there are 24 tags in the tagset, then the n-gram models must estimate $24^5 = 79,626,244$ n-grams, where $n=5$. However a Neural network in which the context of five words, root of the current word and word-length indicator are inputs and hidden layer is half of the input layer requires estimation of 250,560 weights. As it requires few parameters its performance is less affected by a small amount of training data than that of the statistical methods and offers faster tagging. (Q. Ma and Isahara, 1998). This encourages the use of Neural network in POS tagging for Indian languages where the corpora, especially annotated ones, are still considerably in small size.

It requires very less training time than CRF based tagger because its input can capture corpus-based features implicitly and there is no need to include features for unseen words because its input can also capture dictionary-based features implicitly.

10 Conclusion

In this paper, the multi-neuro tagger was presented, a part-of-speech tagger which is based on a MLP-network and uses multiple lengths of contexts and voting based selection rule to obtain final output. It also uses clever encoding scheme in which input to the tagger captures corpus-based and dictionary-based features implicitly. Its performance is also compared with other approaches such as HMM and CRF. It was also shown that Neural network based tagger performs as well as the HMM based tagger and the CRF based tagger. It was also shown that it requires less training time than the CRF based tagger. Its ability to deal with sparse data, its capability to manage multiple contexts, its compatibility with the encoding scheme presented here, its less training time and good accuracy suggest that the multi-neuro tagger could be a better substitute for traditional tagging approaches for Indian languages.

11 Acknowledgement

I would like to express my gratitude to Dr. Rajeev Sangal, Dr. Dipti Misra Sharma, Kishore S. Prahrllad for their support. I would also like to thank Avinesh PVS, Uttam Tiwari, Yogesh Bathina, Karthik Gali, Kamal Devangan, Sunil Verma, Rajesh Harsiyani for their valuable suggestions.

References


