Embedding Approaches for PoS Tagging Indian Languages

Anirban Majumder

under the supervision of
Dr. Amitabha Mukherjee    Dr. Harish Karnick

Department of Computer Science
IIT Kanpur

June 2016
1 Introduction
   - Introduction
   - Embedding for Indian Languages
   - Morphologically Rich Embedding

2 Proposed Models
   - Proposed Embedding Approaches
   - Bi-LSTM Recurrent Neural Network

3 Experimental Setup and Results
   - Dataset
   - Results

4 Conclusions
   - Conclusion and Future Work
1 Introduction
   • Introduction
   • Embedding for Indian Languages
   • Morphologically Rich Embedding

2 Proposed Models
   • Proposed Embedding Approaches
   • Bi-LSTM Recurrent Neural Network

3 Experimental Setup and Results
   • Dataset
   • Results

4 Conclusions
   • Conclusion and Future Work
Embedding Techniques

Brief History

- **Idea of Word embedding**: contextual information alone constitutes a viable representation of linguistic items.
- **Theoretical roots**: in the works of Zellig Harris, John Firth, and Ludwig Wittgenstein in 1950’s.
- **Methods for automatically generate contextual features**: such as LSA[3], PLSA etc. were also used in the 90’s for semantic word representation.
Introduction of Neural Networks

- First proposed by Bengio et al in the year 2003 as neural probabilistic language model.

- A standard neural network approach was proposed by Collobert and Weston et al. in the year 2011[1].

- In 2013, the model Word2Vec[7] by Mikolov et al. have been proved to be really carry the semantics better than the previous works.

- In 2014, Socher et al. proposes a model called GloVe[8], to merge the context information with corpus statistics.
Word2Vec

- Word2Vec maps words in the Vocabulary to a vector space, for each word we get an embedding[7]

- Two approaches: CBOW and Skip-gram
  - CBOW used for predicting the next word given the context
    \[
    \max \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_t | w_{t+j}) \tag{1}
    \]
  - Skip-gram used for predicting the context given the center word
    \[
    \max \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t) \tag{2}
    \]
  - Uses Softmax to predict the output word(s).
CWE and Skip-gram

**Figure:** CWE and Skip-Gram Network Structure[7]
GloVe

- Glove measures the word-word co-occurrence probabilities[8]
- Training objective of GloVe: to learn word vectors so that dot product equals the log of words probability distribution
- Associates logarithm of these probability ratios with vector differences
### GloVe Word Co-Occurrence Statistics

<table>
<thead>
<tr>
<th>Probability and Ratio</th>
<th>$k = solid$</th>
<th>$k = gas$</th>
<th>$k = water$</th>
<th>$k = fashion$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(k</td>
<td>ice)$</td>
<td>$1.9 \times 10^{-4}$</td>
<td>$6.6 \times 10^{-5}$</td>
<td>$3.0 \times 10^{-3}$</td>
</tr>
<tr>
<td>$P(k</td>
<td>steam)$</td>
<td>$2.2 \times 10^{-5}$</td>
<td>$7.8 \times 10^{-4}$</td>
<td>$2.2 \times 10^{-3}$</td>
</tr>
<tr>
<td>$P(k</td>
<td>ice)/P(k</td>
<td>steam)$</td>
<td>$8.9$</td>
<td>$8.5 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

**Figure:** Co-occurrence Probability Ratios[8]
Drawbacks
Less Morphological Information

- traditional models ends up capturing the contextual information more rather than syntactical information
- morphologically rich languages like Hindi, Bengali contains more information in the syntax
- concentration on the intra-word information
Outline

1. Introduction
   - Introduction
   - Embedding for Indian Languages
     - Morphologically Rich Embedding

2. Proposed Models
   - Proposed Embedding Approaches
   - Bi-LSTM Recurrent Neural Network

3. Experimental Setup and Results
   - Dataset
   - Results

4. Conclusions
   - Conclusion and Future Work
Embedding for Indian Languages

Analysis of Word2Vec Representation

- Evaluating embedding techniques: nearest neighbours, visual representation, analogy tasks
Embedding for Indian Languages
Analysis of Word2Vec Representation

- Evaluating embedding techniques: nearest neighbours, visual representation, analogy tasks
- nearest neighbours $\rightarrow$ similar context words
Evaluating embedding techniques: nearest neighbours, visual representation, analogy tasks

- nearest neighbours $\rightarrow$ similar context words
- visual representation: t-SNE plot
Embedding for Indian Languages
Analysis of Word2Vec Representation

- Evaluating embedding techniques: nearest neighbours, visual representation, analogy tasks
- nearest neighbours $\rightarrow$ similar context words
- visual representation: t-SNE plot
- analogy results: evaluating linear analogy to check the properties we use here the analogy task created for Word2Vec for evaluation purposes
Nearest Neighbours

- The nearest is calculated using k-NN algorithm.

<table>
<thead>
<tr>
<th>किलोमीटर (km)</th>
<th>किलोमीटर (km)</th>
<th>नीला (Blue)</th>
<th>नीला (Blue)</th>
</tr>
</thead>
<tbody>
<tr>
<td>माइल (Mile)</td>
<td>मील (Mile)</td>
<td>सादा (white)</td>
<td>बैरनी (Violet)</td>
</tr>
<tr>
<td>वर्ग किलोमीटर (Sq.km.)</td>
<td>मीटर (Distance)</td>
<td>काला (black)</td>
<td>आसमानी (Sky Color)</td>
</tr>
<tr>
<td>वर्ग (Square)</td>
<td>किमी (km)</td>
<td>गहराव (depth)</td>
<td>पिगमेंट (Pigment)</td>
</tr>
<tr>
<td>दूरी (Distance)</td>
<td>गज (yard)</td>
<td>रंग (color)</td>
<td>पीला (Yellow)</td>
</tr>
</tbody>
</table>

**Figure:** Nearest neighbours of the words (Hindi and Bengali)
t-SNE Plot

- proposed by Maaten et al in 2008[6]
- dimensionality reduction technique for visualizing high dimensional data in 2 or 3 dimension
- main intuition to model similar objects with high probability for pair similarity by nearby points and dissimilar objects by distant points
t-SNE plot

Figure: plot for 2-D t-SNE data
t-SNE plot

Figure: zoomed version of t-SNE plot
t-SNE plot

Figure: zoomed version of t-SNE plot
insufficient work in Indian languages on word embeddings

standard task for evaluating Word2Vec: Capital-Country Analogies, Country-Currency analogies, Grammatical Analogies

Using machine translation to create similarity tasks and also manual creation
### Analogy tasks for Evaluation

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>एथेंस</td>
<td>ग्रीस</td>
<td>बगदाद</td>
<td>इराक</td>
<td>इराक़ (17)</td>
</tr>
<tr>
<td>एथेंस</td>
<td>ग्रीस</td>
<td>बॉर्लिन</td>
<td>जर्मनी</td>
<td>जर्मनी (6)</td>
</tr>
<tr>
<td>एथेंस</td>
<td>ग्रीस</td>
<td>रोम</td>
<td>इटली</td>
<td>इटली (5)</td>
</tr>
<tr>
<td>शुद्ध</td>
<td>अशुद्ध</td>
<td>चल</td>
<td>अचल</td>
<td>उचल (21)</td>
</tr>
<tr>
<td>उपमहाद्वीप</td>
<td>महाद्वीप</td>
<td>उपकप्तान</td>
<td>कप्तान</td>
<td>No Match</td>
</tr>
</tbody>
</table>

**Figure:** Analogy Results for Hindi
Problems with this Embeddings

- semantically rich but morphologically not
- more focus needed on the word structure i.e intra-word information
- Indian languages have more grammatical inflections compared to English, i.e morphological information required to give a better embedding
- we propose two model which focuses to capture this characteristic and from that create a word representation
Outline

1. Introduction
   - Introduction
   - Embedding for Indian Languages
   - Morphologically Rich Embedding

2. Proposed Models
   - Proposed Embedding Approaches
   - Bi-LSTM Recurrent Neural Network

3. Experimental Setup and Results
   - Dataset
   - Results

4. Conclusions
   - Conclusion and Future Work
Morphologically Rich Embedding

Character Embedding

- Very few works exist on these domain
- char-WNN[11] by Santos et al. which obtains character embedding and using CNN merge them to obtain word embeddings
- trained for PoS tagging task for English and Portuguese, it shows good results
Leonard et al. [12] proposes a model which creates a character embeddings along with word embeddings, implementing the skip-gram model of Word2Vec.

- main difference with Word2Vec: maintains a character matrix along with the word matrix.
- 3 types of character representation for each character based on its position in the word: begin, middle and end.
the objective function is same as CBOW in Word2Vec, just the word vectors have been replaced by the composition of the word and character vectors based on their positions

\[
\max \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_t' | w_{t+j}')
\] (3)

\[
w_t' = \frac{1}{2} (w_t + \frac{1}{N_k} (c_1^B + \sum_{i=2}^{N_k-1} c_i^M + c_{N_k}^E))
\] (4)
Figure: position based character enhanced word embedding for Chinese[12]
Morphologically Rich Embedding
Morpheme Embedding

- another proposed model is to create representation of morphemes and from that word embedding
- Socher et al. [5] in 2013 proposed a model which learns word representation from morphemes using Recursive Neural Network
- other method by Qiu et al. [9] shows a model similar to character enhanced embedding, just instead of character, it uses morphemes
Outline

1. Introduction
   - Introduction
   - Embedding for Indian Languages
   - Morphologically Rich Embedding

2. Proposed Models
   - Proposed Embedding Approaches
   - Bi-LSTM Recurrent Neural Network

3. Experimental Setup and Results
   - Dataset
   - Results

4. Conclusions
   - Conclusion and Future Work
we implement the Leonard’s algorithm\cite{12} to obtain the character embedding

obtaining the character embedding we merge them by following rules:
- simple addition
- weighted addition

For the weighted addition, we need morpheme segmentation of words.

the morpheme vectors are obtained from the character vectors and the morpheme representation will be combined to obtain a word by weighting the morphemes by a simple rule.
Character Embedding: CWE

**Simple Addition**

Adding the character vectors based on their positions and normalizing the resultant vector to obtain the word representation.

**Weighted Addition**

Morpheme segmentation is done by Dasgupta’s algo[2] which segments words into morphemes using word-root ratio.

\[
w_k = \frac{1}{n} \times \sum_{i=1}^{n} [(|\text{pos}(\text{morph}) - \text{pos}(\text{root})| + 1) \times m_i]
\]  \hspace{1cm} (5)

\[
m_i = \sum_{j=1}^{m} c_j
\]  \hspace{1cm} (6)
### Figure: Similar Word (5-NN) Results for Hindi and Bengali

<table>
<thead>
<tr>
<th>भारतीय</th>
<th>तकनीक</th>
<th>विभाजित</th>
<th>खननकाल</th>
<th>सूपारस्टारेज</th>
<th>सवसमस्य</th>
</tr>
</thead>
<tbody>
<tr>
<td>भारतीय</td>
<td>सतनीक</td>
<td>अविभाजित</td>
<td>आलोचनाकाल</td>
<td>स्टाइकारेर</td>
<td>सवरकस</td>
</tr>
<tr>
<td>जातीय</td>
<td>यूनीक</td>
<td>प्रभाजित</td>
<td>शुनामिकाल</td>
<td>फाइनलिस्टेर</td>
<td>सवकिफ़ौ</td>
</tr>
<tr>
<td>उत्तरीय</td>
<td>टेकनीक</td>
<td>विभाजित</td>
<td>वर्तमानकाल</td>
<td>डिफेन्डारेर</td>
<td>सवजायगायई</td>
</tr>
<tr>
<td>काकतीय</td>
<td>तहरीक</td>
<td>द्विभाजित</td>
<td>अनूकुल</td>
<td>सेन्टिमिटारेर</td>
<td>सवकिफ़ौज़े</td>
</tr>
<tr>
<td>सजातीय</td>
<td>प्रतीक</td>
<td>समविभाजित</td>
<td>अपहरणकाल</td>
<td>चेयरपरसरेर</td>
<td>सवजनम्बिकृत</td>
</tr>
</tbody>
</table>

**CWE Nearest Neighbours**
basic unit to learn representation is morpheme

using Dasgupta’s algo[2], the word segmentation is obtained for the word in vocabulary

modified the corpus by replacing the words with its morphemes

using skip-gram of Word2Vec, obtained the morpheme representations of morphemes

using simple and weighted addition, we obtain the word representation from the morpheme embeddings
MorphVec

Simple Addition
Adding the different morpheme vectors and normalizing the resultant vector to obtain the word representation

Weighted Addition
Adding the different morpheme vectors using the following weighting rule:

\[ w_k = \frac{1}{n} \sum_{i=1}^{n} [(|pos(morph) - pos(root)| + 1) \times m_i] \] (7)
Morpheme Segmentation: Dasgupta’s Algorithm

- the algo was proposed by Sajib Dasgupta and Vincent Ng[2]
- for the morpheme segmentation, it follows the rules below.
  - Detecting incorrect attachments using relative frequency
  - applying suffix level similarity i.e similar suffixes can go with a particular word
  - inducing orthographic rules and allomorphs, i.e learn the character-change rules by a single character replacement, addition and deletion at the segment boundary(eg: denial = deny + al)
  - handle small roots i.e small roots remains unsegmented if not applied
we now compare our embedding approaches with the existing embeddings like Word2Vec and GloVe for a specific task: PoS tagging for Indian Languages.

no external word features other than the embeddings would be used for prediction, i.e. no hand-crafted rules.

color the proposed model is language independent and unsupervised.
Introduction

1. Introduction
   - Introduction
   - Embedding for Indian Languages
   - Morphologically Rich Embedding

2. Proposed Models
   - Proposed Embedding Approaches
   - Bi-LSTM Recurrent Neural Network

3. Experimental Setup and Results
   - Dataset
   - Results

4. Conclusions
   - Conclusion and Future Work
Due to its temporal behaviour, RNNs can learn to model various kinds of sequence learning tasks.

Speech Recognition, Word Prediction, Spelling Correction, sentiment analysis, machine translations are some examples of the tasks.

PoS tagging can also be modelled as a sequence tagging task by a recurrent neural network model.

**Basic RNN**

\[ h(t) = f(Ux(t) + Wh(t - 1)) \]  \( (8) \)

\[ y(t) = g(Vh(t)) \]  \( (9) \)
Simple RNN Structure

Figure: Simple RNN
LSTM

- proposed by Sepp Hochreiter and Jürgen Schmidhuber in 1997[4]
- removing the vanishing gradient problem of the vanilla recurrent neural network, keeping the required information after many time-steps
- the internal functions of the LSTMs can be divided into following four sub-functions:
LSTM Updating Rules

**Figure:** LSTM Network Structure

- Information to remember/forget

\[ f(t) = \sigma(W - f[h(t-1), x(t)] + b_f) \]  

\[ i(t) = \sigma(w_i[h(t-1), x(t)] + b_i) \]  

\[ c'(t) = \tanh(W_c[h(t-1), x(t)] + b_c) \]
LSTM Updating Rules

- Updating the candidate values
  \[
  c(t) = f(t) \ast c(t - 1) + i(t) \ast c'(t)
  \]  
  (13)

- Updating the Output and Hidden State
  \[
  o(t) = \sigma(W_o[h(t - 1), x(t)] + b_o)
  \]  
  (14)
  \[
  h(t) = \tanh(c(t)) \ast o(t)
  \]  
  (15)

Figure: LSTM Network Structure
Proposed Model: Bi-LSTM

- Bi-LSTM processes the input from both directions and effectively captures the forward and backward word contexts and tags as information.

- Our LSTM layer has 3 layers, one input layer, one hidden layer and one output layer.

- The input layer will contain the word represented as a vector. The output layer will be the probability distribution over the tag set.

- Dimension of the output layer will be equal to the number of tags and dimension of the input will depend on the size of the vector embedding and sentence length.
Bi-LSTM PoS Model Structure

**Figure**: 3 layers of the model

**Figure**: Bi-LSTM network structure
Bi-LSTM Model for PoS Tagging

- the output layer is a Softmax layer which tries to predict the probability distribution of the tags
- the input has to be of fixed length for the input to the LSTM, for that we have to maintain the length of the sentences
- the sentence with length more than max length is divided into multiple sentences
- the sentence with length less than max length is padded with random vectors between -0.1 and 0.1
Outline

1. Introduction
   - Introduction
   - Embedding for Indian Languages
   - Morphologically Rich Embedding

2. Proposed Models
   - Proposed Embedding Approaches
   - Bi-LSTM Recurrent Neural Network

3. Experimental Setup and Results
   - Dataset
   - Results

4. Conclusions
   - Conclusion and Future Work
Dataset for Experiment

- Dataset for PoS tagging in Hindi is the HindiENCorp collected by Charles University, Prague.
- It was mainly collected for a machine translation task but it is annotated with POS for both English and Hindi.
- Consists of several sources including Wikipedia, Launchpad.net, Indic corpus 2012.

<table>
<thead>
<tr>
<th>Language</th>
<th>Sentences</th>
<th>Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>273K</td>
<td>3.60 M</td>
</tr>
<tr>
<td>Hindi</td>
<td>273K</td>
<td>3.97 M</td>
</tr>
</tbody>
</table>

Table: Statistics for the Hindi Dataset
Pre-processing, Training and Testing Data

- cleaning the data using Hindi Unicode range, to remove any other language present in the corpus

- the data contains 33 PoS tags including the 'UNK' tag

- we use the last 25K to test our model, in group of 5K in one test set, the rest is use for training data
Outline

1. Introduction
   - Introduction
   - Embedding for Indian Languages
   - Morphologically Rich Embedding

2. Proposed Models
   - Proposed Embedding Approaches
   - Bi-LSTM Recurrent Neural Network

3. Experimental Setup and Results
   - Dataset
   - Results

4. Conclusions
   - Conclusion and Future Work
LSTM and Bi-LSTM Model

- Table shows that the Bi-LSTM model properly considers both forward and backward context and its performance is better than the unidirectional LSTM model.

- Bi-LSTM model is a better sequence learner than the LSTM model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>88.82</td>
<td>89.05</td>
</tr>
<tr>
<td>BI-LSTM</td>
<td>91.15</td>
<td>91.25</td>
</tr>
</tbody>
</table>

Table: Comparing results for LSTM & Bi-LSTM
exploring the data statistics for the training dataset by checking its mean and standard deviation, we found that mean is 13.23 and the standard deviation is 17.32.

<table>
<thead>
<tr>
<th>Length</th>
<th>Accuracy</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>89.30</td>
<td>89.29</td>
</tr>
<tr>
<td>20</td>
<td>90.32</td>
<td>90.31</td>
</tr>
<tr>
<td>30</td>
<td>91.03</td>
<td>91.04</td>
</tr>
<tr>
<td>50</td>
<td>90.86</td>
<td>90.83</td>
</tr>
<tr>
<td>100</td>
<td>91.15</td>
<td>91.25</td>
</tr>
</tbody>
</table>

Table: Results for different sentence length as input
Figure: Results for various sentence lengths
increasing the number of hidden layers in a model can increase the accuracy of the model, after also increase the no of parameters to train

We found that the results seem to saturate after the second hidden layer, so in further experiments we use 2 hidden layers

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Precision</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-LSTM(1)</td>
<td>89.73</td>
<td>90.16</td>
<td>89.95</td>
</tr>
<tr>
<td>BI-LSTM(2)</td>
<td>91.15</td>
<td>91.25</td>
<td>91.35</td>
</tr>
</tbody>
</table>

Table: Results for Bi-LSTM with 1 and 2 hidden layers
 sized hidden layer corresponds to the amount of information the model can hold.

<table>
<thead>
<tr>
<th>Hidden Layer Size</th>
<th>Accuracy</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>90.84</td>
<td>91.03</td>
</tr>
<tr>
<td>50</td>
<td>91.15</td>
<td>91.25</td>
</tr>
<tr>
<td>100</td>
<td>91.13</td>
<td>91.10</td>
</tr>
<tr>
<td>200</td>
<td>91.42</td>
<td>91.41</td>
</tr>
</tbody>
</table>

Table: Results for different hidden layer sizes
Varying Hidden Layer Size

Figure: Comparison of different hidden layer sizes
Comparing the model with traditional word embedding like Word2Vec and GloVe, we found our model has performing better than the other embeddings.

<table>
<thead>
<tr>
<th>Embeddings</th>
<th>Accuracy</th>
<th>Precision</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word2Vec</td>
<td>90.69</td>
<td>90.72</td>
<td>90.70</td>
</tr>
<tr>
<td>Glove</td>
<td>90.89</td>
<td>91.12</td>
<td>91.01</td>
</tr>
<tr>
<td>CWE</td>
<td>91.05</td>
<td>91.19</td>
<td>91.12</td>
</tr>
<tr>
<td>MorphVec</td>
<td>91.15</td>
<td>91.35</td>
<td>91.25</td>
</tr>
</tbody>
</table>

**Table:** Results for different embedding approaches
Comparing with the Existing Model

- to the best of our knowledge, there has not been any other work on PoS tagging using word embeddings for Indian Languages.
- results of the previous system which has achieved good results is a method proposed by Reddy et al. [10].
- we can see that our proposed model does reasonably well compared to the existing supervised tagger.

<table>
<thead>
<tr>
<th>Models</th>
<th>Reddy et al.</th>
<th>MorphVec</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoS</td>
<td>91.31</td>
<td>91.15</td>
</tr>
</tbody>
</table>

Table: Results for the two models in Indian language.
we have recently tried our model on a larger dataset and shows that the accuracy increases with the increase of the training data.

<table>
<thead>
<tr>
<th>Data Points</th>
<th>Precision</th>
<th>Accuracy</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>20K</td>
<td>91.35</td>
<td>91.15</td>
<td>91.25</td>
</tr>
<tr>
<td>1L</td>
<td>93.01</td>
<td>92.73</td>
<td>92.87</td>
</tr>
<tr>
<td>1.5L</td>
<td>93.32</td>
<td>93.13</td>
<td>93.23</td>
</tr>
</tbody>
</table>

Table: Results with different size of dataset
Introduction

Introduction
Embedding for Indian Languages
Morphologically Rich Embedding

Proposed Models

Proposed Embedding Approaches
Bi-LSTM Recurrent Neural Network

Experimental Setup and Results

Dataset
Results

Conclusions

Conclusion and Future Work
observed that the traditional embeddings fail to capture the syntactic information which is essential for morphologically rich languages like Hindi or Bengali

discuss alternative embeddings which focus on word-structure and give importance to morpho-syntactic information

also compare our results for PoS tagging the Indian Language Hindi using LSTM network

The technique can be considered superior to previous ones as in our model there are no hand-crafted rules and it is completely language independent
Future Works

- To learn the weights for the morpheme representations to obtain the word vectors from them.
- Test the results on other NLP tasks. Here we focus our attention to PoS tagging task. It can also be used for NER and chunking where morphology plays an important role.
- Reverse Engineering: Whether we can find a better algorithm for morpheme segmentation for a language using the combined word vectors and the annotated POS corpus.
- Applying convolution to remove the drawback of the fixed length of the sentences
Thank You.
Questions ?


