# Parts of Speech Tagging using GRU

**A**
**Mini Project Report**

*by*

**Durgesh Kumar Singh(SC18M002)**


Department of Mathematics
Indian Institute of Space Science and Technology, Trivandrum
Semester-II

# Contents

# List of Figures



# List of Tables

# 1 Introduction

In Part of Speech (POS) tagging, an association of a word in a sentence is done to a corresponding part of a speech tag, based on its context and definition. POS Tags are useful for building parse trees, which are used in building NERs (most named entities are Nouns) and extracting relations between words. POS Tagging is also essential for building lemmatizers which are used to reduce a word to its root form.

# 2 Techniques for POS Tagging

The different POS tagging techniques are as follows-

1. Lexical Based Methods

   Lexical based methods assign POS tags by frequency of the word in the training corpus. The tag encountered in training set most frequently with the word is assigned a tag for the word. This approach has a problem that it may yield invalid sequences of tags.

2. Rule-Based Methods

   Assign POS tags based on rules. The rule-based technique can be used along with a lexical based approach to allow POS tagging. It is more useful when data is not present in training corpus but occurred in testing data.

3. Probabilistic Methods

   This method assigns the POS tags based on the probability of a particular tag sequence occurring. There are several approaches for probability based models like Conditional Random Fields (CRFs) and Hidden Markov Models (HMMs)

4. Deep Learning Methods—Recurrent Neural Networks can also be used for POS tagging.

# 3 Background

## 3.1 Recurrent Neural Network

Recurrent Neural Networks(RNN) are networks which persist information. RNN is able to remember the context from the previous input. RNN's are used in various time series analysis task i.e next word prediction, music composition, image captioning, speech recognition, time series anomaly detection, stock market prediction, etc.

However, RNN's suffer from short-term memory. If a sequence is long enough, it is difficult of RNN in carrying the information from the earlier time-steps to later ones. This is called the Vanishing Gradient Problem. Gated Recurrent Unit(GRU) and Long Short Term Memory(LSTM) were introduced to solve this issue.

## 3.2 Structure of GRU Cell

GRU's and LSTM's have repeating modules like the RNN, but the repeating modules have a different structure. The key idea of GRU is the cell state or memory cell. It allows it to retain any information without much loss. It also has gates, which help to regulate the flow of information to the cell state. These gates can learn which data in a sequence is important and which is not. By doing that, they pass information in long sequences.

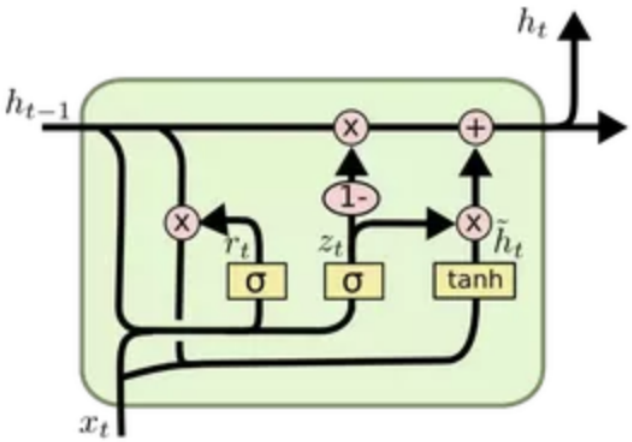


Figure 1: Structure of GRU cell

The architecture of a GRU cell is complex than a simple RNN Cell. The first thing we need to notice in a GRU cell is that the cell state $h_t$ is equal to the output at time $t$. We can see equations in GRU cell as follows-

$$\tilde{h}_t = tanh(W \cdot [r_t * h_{t-1}, x_t]$$

At each time step we have two options-

1. Retain the previous cell state.
2. Update its value.

The above equation shows the updated value or candidate which can replace the cell state at time $t$. It is dependent on the cell state at previous timestep $h_{t-1}$and a relevance gate called $r_t$, which calculates the relevance of previous cell state in the calculation of current cell state.

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

As we can see, the relevance gate $r_t$has a sigmoid activation, which has the value between 0 and 1, which decides how relevant the previous information is, and then is used in the candidate for the updated value.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t}$$

The current cell state $h_t$ is a filtered combination of the previous cell state $h_{t-1}$ and the updated candidate $\tilde{h_t}$. The update gate $z_t$ here decides the portion of updated candidate needed to calculate the current cell state, which in turn also decides the portion of the previous cell state retained.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

Like the relevance gate, the update gate is also a sigmoid function, which helps the GRU in retaining the cell state as long as it is needed.

# 4 Implementation details

## 4.1 Cell Structure

The purpose of this project is to learn how to implement GRU cell and compare different types of mutations of GRU cell on the task of POS Tagging using Brown and PennTreebank dataset. The implementation includes

1. An implementation of GRU cell
2. Mutations of original GRU cell by changing equation of relevance and update gate. The mutations can be defined as follows
   (a) MUT 1
   $z = \sigma(W_z \cdot x_t + b_z)$
   $r = \sigma(W_r \cdot x_t + W_r \cdot h_t + b_r)$
   $h_{t+1} = tanh(W_h[r \odot h_t] + tanh(x_t) + b_h) \cdot z + h_t \odot (1 - z)$
   (b) MUT 2
   $z = \sigma(W_z x_t + W_z h_t + b_z)$
   $r = \sigma(x_t + W_r h_t + b_r)$
   $h_{t+1} = tanh(W_h[r \odot h_t] + W_h \cdot x_t + b_h) \odot z + h_t \odot (1 - z)$
   (c) MUT 3
   $z = \sigma(W_z x_t + W_z \cdot tanh(h_t) + b_z)$
   $r = \sigma(W_r x_t + W_r h_t + b_r)$
   $h_{t+1} = tanh(W_h[r \odot h_t] + W_h x_t + b_h) \odot z + h_t \odot (1 - z)$
3. An POS Tagger which can be configured to use any of the above GRU implementations

## 4.2 Word Embedding

Word embedding is a representation of document vocabulary which is input to the training model. Each word is converted to a vector which is capable of capturing semantic and syntactic similarity, the context of a word and relationship of the word with other words. Word2Vec is one of the famous techniques to learn word embedding. Two methods for word2vec representation are: Skip Gram and Common Bag Of Words (CBOW)

In this project, the default word2vec model provided by the torch is used to obtain word embedding of input data.

## 4.3 Loss Function

The loss function is used to measure the error between predicted value $\hat{y}$ and actual label $y$. It is an important part of artificial neural netoworks. The loss function is non-negative value and as the loss function decreases the robustness of the model increase. The loss function is the core of empirical risk minimization.

In this project, Negative Log Likelihood loss(NLL) function is used which measures the accuracy of the classifier. The formula of the NLL loss function is as follows

$$L = -\frac{1}{n} \sum log(\hat{y}^{(i)})$$

NLL loss function is primarly used in multi class classification and returns probability of each class that a point might belong to.

## 4.4 Optimization algorithm

Optimization algorithms are used to find an optimal value of an objective function, which depends on the model's learnable parameters. These parameters are used to compute the target value from a given input. Optimization algorithm tries to find the parameters of the model iteratively to find the optimal value of the objective function.

In this project, SGD Optimizer is used to minimize the error function corresponding to each training example. This has high variance and causes the loss function to fluctuate to different intensities. Alternatively, we can use Adam optimizer as minimizer( or maximizer).

### 4.5 Dataset

In this project, POS tagger is trained using Brown and Penn Treebank corpus.

1. Brown Dataset

   Brown corpus was compiled in 1960 at Brown University. It is a general corpus in linguistics which contains 500 samples of English language text from various categories and containing 1 million words approximately.

2. PennTreebank Dataset

   It is widely used in machine learning of NLP (Natural Language Processing) research. The Penn Treebank (PTB) is a collection of articles from the Wall Street Journal (WSJ) over a three year of the period.

In this project, Brown and PennTreebank data set is used with universal tagset. The brown dataset contains approximately 46000 sentences and PennTreebank data set contains 11000 sentences. The universal tagset format is as follows

| Tagset | Code |
|---|---|
| 'VERB' | 0 |
| '.' | 1 |
| 'PRT' | 2 |
| 'NOUN' | 3 |
| 'DET' | 4 |
| 'ADJ' | 5 |
| 'ADP' | 6 |
| 'PRON' | 7 |
| 'NUM' | 8 |
| 'ADV' | 9 |
| 'CONJ' | 10 |
| 'X' | 11 |

Table 1: Universal Tagset format

## 5 Results

### 5.1 Accuracy per epoch

Accuracy is used as primary performance measure for POS Tagging problem. Following graphs show the performance of GRU POS tagger on PennTreebank and Brown Dataset

#### 5.1.1 PenTreebank dataset

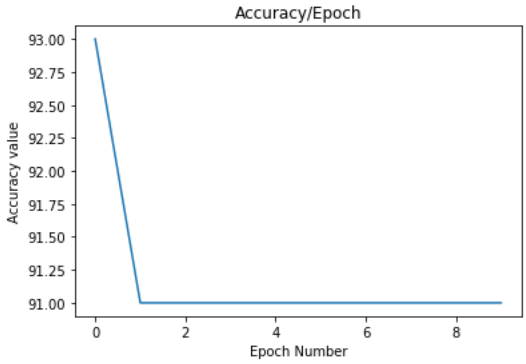


Figure 2: Accuracy/Epoch Graph for Standard GRU cell

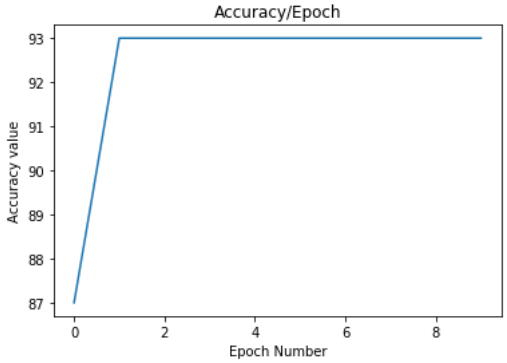


Figure 3: Accuracy/Epoch Graph for Mutation 1

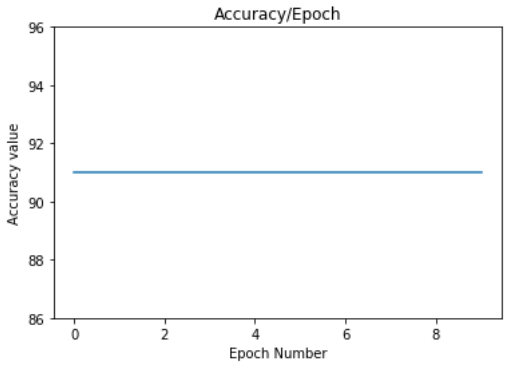


Figure 4: Accuracy/Epoch Graph for Mutation 2

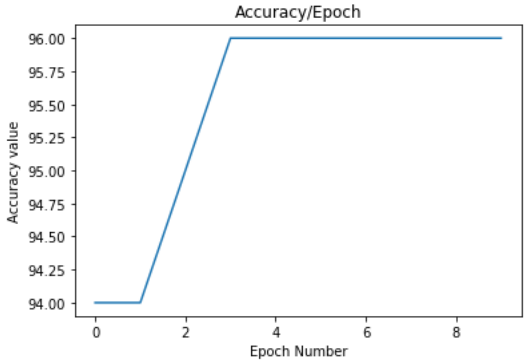


Figure 5: Accuracy/Epoch Graph for Mutation 3

### 5.1.2 Brown data set

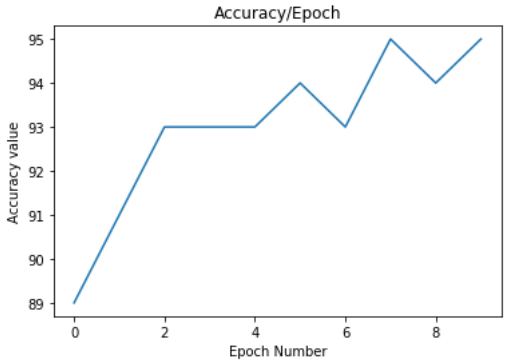


Figure 6: Accuracy/Epoch Graph for Standard GRU cell

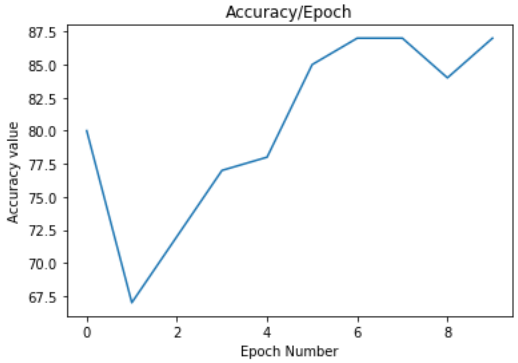


Figure 7: Accuracy/Epoch Graph for Mutation 1

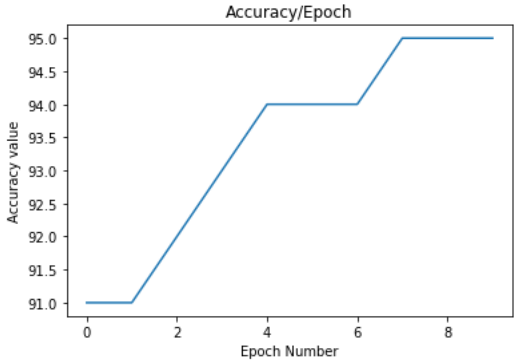


Figure 8: Accuracy/Epoch Graph for Mutation 2

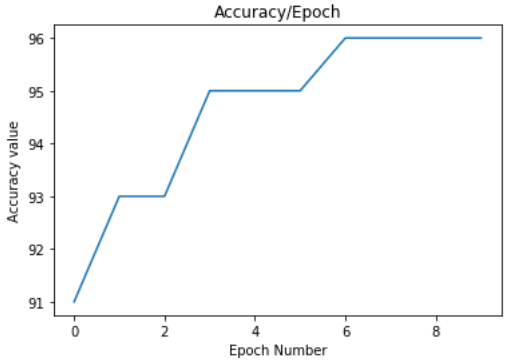


Figure 9: Accuracy/Epoch Graph for Mutation 3

### 5.2 Comparision Results with other POS Taggers

| | Performance in percentage | |
|---|---|---|
| Model | Brown dataset | TreeBank dataset |
| GRU-Standard | 95 | 91 |
| GRU-M1 | 88 | 93 |
| GRU-M2 | 95 | 95 |
| GRU-M3 | 96 | 96 |
| LSTM-Standard | 99 | 97 |
| LSTM-o | 99 | 97 |
| LSTM-f | 88 | 86 |
| LSTM-i | 99 | 96 |
| LSTM-b | 99 | 97 |
| RNN | 80 | 85 |

Table 2: Comparision Matrix of different POS Taggers

GRU-M1=Mutation 1,GRU-M2=Mutation 2,GRU-M3=Mutation 3,LSTM-o=LSTM without output gate,LSTM-f=LSTM without forget gate,LSTM-i=LSTM without input gate,LSTM-b=LSTM forget gate bias set to ones

## 6 Conclusion

We can say that GRU cell Mutation 3 performs better having 96 percent of accuracy in POS Tagging Task. The performance of GRU and LSTM cells are approximately equal and better as compared to plain RNN.

# 7 References


1. An Empirical Exploration of Recurrent Network Architectures : Rafal Jozefowicz, Wojciech Zaremba, Ilya Sutskever ; Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:2342-2350, 2015.

2. CS7015: Deep Learning Lecture by Mitesh M Khapra (IIT M) https://www.cse.iitm.ac.in/~miteshk/CS7015.html