# Natural Language Understanding
# Assignment-2
# Neural Language Model Implementation

**Karan Malhotra**
M.Tech (Systems Engineering)
kmkaran212@gmail.com

## 1 Introduction

Language Models are the models that assigns probability to sequence of words by assigning probabilities to each possible next word.These models have variety of applications such as in speech recognition, Machine Translation , Handwriting recognition etc.

For a Given corpus with a number of sentences with vocabulary as **V**, the probability for sequence of words $(w_1, w_2, ..., w_n)$ can be estimated as

$$P(w_1, w_2, ..., w_n) = \prod_{k=1}^{n} P(w_k | w_1^{k-1})$$

where $w_1^{k-1} = (w_1, w_2, ..., w_{k-1})$ the last $K-1$ history of words.
Although Traditional **N-grams** models worked significantly well but **Neural models** have many advantages over them. Some of the advantages are that neural language models dont need smoothing, they can handle much longer histories, and they can generalize over contexts of similar words. Neural language modeling uses a network as a probabilistic classifier, to compute the probability of the next word given the previous N word and make use of embeddings, dense vectors of between 50 and 500 dimensions that represent words in the input vocabulary.
Despite of such advantages these models suffer from shortcomings like slow learning , poor performance on less data corpus and high computational resource needs.
The figure below describes a simple RNN based Language model which tries to predict the output word for a given input word with capturing the previous words context as the state vector.

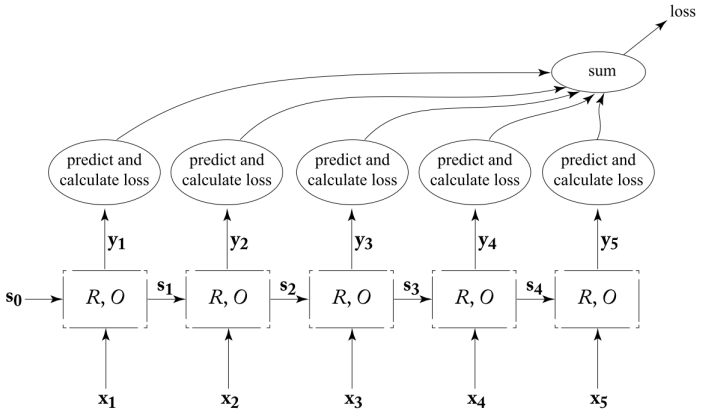


**Fig1: RNN approach for Language Model**
*Image taken from the book Neural Network Methods for Language Processing by Yoav Goldberg.
Based on this general approach the two models-character level and word level have been built with LSTM cells used for capturing the context in the given input sequence.

## 2 DataSets

The publically available Project Gutenberg corpus is used for this project. **Gutenberg Corpus** is a collection of 18 English books of several different authors containing around 49,000 unique words and 2,650,000 total words. Due to Computational limitations the Neural models were not trained on full data. Only certain files from gutenberg corpus were selected and then 80:20 train test split were performed.
The data was selected in such a quantity that the large number of model parameters does not cause overfitting problem.

## 3 Model Design

The generalized architecture of the neural model used in both character level and word level models is shown in figure below:

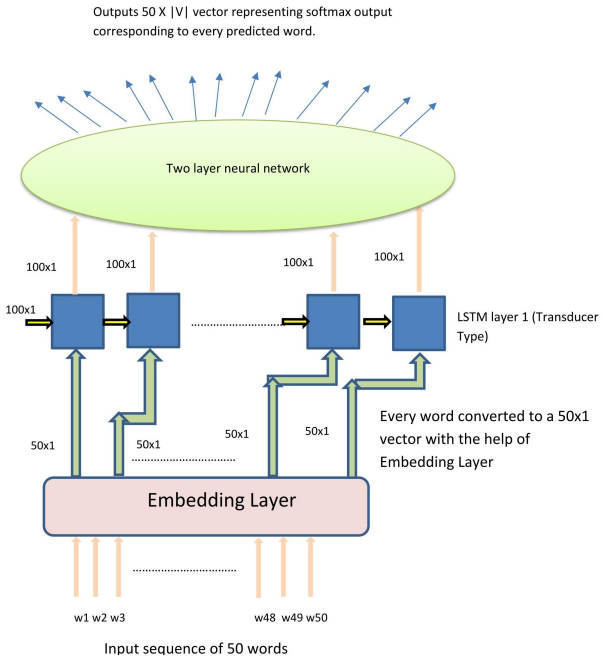


**Fig 1:Model Design Architecture**

The proposed model takes input of 50 word/char sequence at a particular instant and tries to predict the next word/char for every word/char in the sequence by utilizing the context of the previous words/chars in the sequence.
Following are the layers in the model implemented :
**1.Embedding Layer:** This layer is responsible for conversion of every word/char of input sequence into embedding with dimension passed as input parameter. The words/chars are passed by converting them into integers by replacing them with their index in vocabulary.
The Embedding layer is initialized with random weights and will learn an embedding for all of the words in the training dataset while model training.
**Parameters:** Vocabulary size ,Embedding length ,Input sequence length.

**2.LSTM layer1 :** This is a transducer type LSTM layer which takes the incoming sequence of 50 words/chars and tries to predict the next word/char for every word/char in the sequence.
The word/char prediction is done in such a way that, the context of previous words/chars get transferred to next predictions only in limited amount which is controlled through the different gates in LSTM cell.
**Parameters:** LSTM state dimension.

**3.Two layer neural network:** The single output(100X1) produced by an individual LSTM cell has to be converted into corresponding word/char it represent. So,for that purpose a 2 layer dense network is used which converts the incoming input into a vector of size as of vocabulary. The entries of output vector represents the probability of the corresponding word/char for coming out as predicted next word/char.
This is done by applying Softmax layer which is the second layer of this neural network.
**Parameters:** Weights and Activation function for both layers.

The following figures shows the parameters and layer description of the models built:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_2 (Embedding) | (None, None, 30) | 278970 |
| lstm_2 (LSTM) | (None, None, 50) | 16200 |
| dense_3 (Dense) | (None, None, 50) | 2550 |
| dense_4 (Dense) | (None, None, 9299) | 474249 |

Total params: 771,969
Trainable params: 771,969
Non-trainable params: 0

**Fig2: Build Word Level Model**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_2 (Embedding) | (None, 40, 50) | 2750 |
| lstm_3 (LSTM) | (None, 40, 50) | 20200 |
| lstm_4 (LSTM) | (None, 50) | 20200 |
| dense_3 (Dense) | (None, 50) | 2550 |
| dense_4 (Dense) | (None, 55) | 2805 |
| activation_2 (Activation) | (None, 55) | 0 |

Total params: 48,505
Trainable params: 48,505
Non-trainable params: 0

**Fig3: Build Char Level Model**

The two figures show the models built and the number of trainable and non trainable parameters.It can be observed that the char level model requires very less number of parameters to be trained compared to word level model.
Various parameters can be set as hyper-parameters and their optimal values are found as doing grid search over certain values and evaluating performance on validation data set. Some of hyper parameters tuned are:
The size of Embeddings .
The batch size during optimisation .
The state vector size of LSTM layers.

## 4 Random Sentence Generation

The **Word Level Model** is used to generate Random sentence although character model give better perplexity but comparison of perplexity at word level and character level does not seems to be valid as one wrongly predicted character can change meaning of the word generated and moreover better semantics are captured with word level models.0.The steps followed to generate random sentecne are:

**Step1.**Some words have been picked from Vocabulary and randomly one word is selected form these words to start the sentence.
This word becomes the initial input-sequence.

**Step2**.The input-sequence is then passed into model and the model outputs a probability vector of dimension as of vocabulary size for every word in input sequence and the output vector for last word in input-sequence is being selected

**Step3.**Now a word is selected using sampling from a Multinomial distribution using the selected o/p probability vector.This sampling ensures that words with higher probability for a given context history will be chosen with grater chances but aslo with some randomness .

**Step4.**. Now the selected word is merged into the generated sentence string and also in the end of input-sequence.

**Step5.**The above steps(2-4) are repeated till 10 token sentence is obtained.

Some of Random generated sentences are:-

1. "my intentions are gone back with some hour oh ! you did not come back "
2. "the disappointment i have received from it was confined him and not be renewed "
3. "my authority i will be too unhappy and be continually shaken off for ever had"
4. "her part of all an extremely and so romantic way with no one emma know"

## 5 Training and Evaluation of Language Model

**Training:** The technique used to train the model is in which the train data is divided into certain length sequence of words and these sequences are send into the model.The model predicts the next word for every word in the sequence and the loss is calculated through original next word also being provided.This same approach is also utilized to train the character level model.
The following hyperparameters values were decided through grid search on certain subsets of the options:

**Word Level Model**

- Embedding Size: 30
- Dimension of state vector in LSTM: 50

**Character Level Model**

- Embedding Size: 50
- Dimension of state vector in LSTM: 100

| **Model type **(Trained for 50 epochs)** | **No of Parameters trained** | **Accuracy** | **Cross-entropy Loss** |
|---|---|---|---|
| **Word Level Neural LM** | 771K | 35% | 3.21 |
| **Character Level Neural LM** | 48K | 54% | 1.51 |

T1: Table to show Training Results

The Accuracy in the above table describes by how much chance the model predicts the next word same as the given one for the incoming input word.

**Evaluation:** For this task an intrinsic measure called **Perplexity** (sometimes called PP for short) is chosen. Perplexity of a Language model on test set is the inverse probability of the test set, normalized by the number of words. For a given test set W perplexity is defined as:

PP(W)= P(w1,w2,w3....wN)$^{-1/N}$

where N is the number of tokens with including end of sentence marker but not beginning of sentence marker.

In the Neural languague models perplexity calculations were not so straightforward like that of in N-grams.

Now Evaluation on the test data also includes same type of sequence forming as in training from the test data and giving input to the model.The

model gives output a vector of probability with dimension equals to vocab size for every word in input sequence. The probability of true next word is taken and then the perplexity calculations are carried on by using this probability.

**Performance on Test Data :**

| **Model type **(Trained on 6 Files of Gutenberg)** | **Perplexity** |
|---|---|
| **Bigram (Kneyser Ney)** | 125 |
| **Word Level Neural LM** | 71 |
| **Character Level Neural LM** | 10 |

T1: Table to show Perplexity values

The above table represents the perplexity values calculated for the two neural language models and one traditional language model(from previous assignment) . These three models were trained on same data set with same train:test split configurations.

## 6 Observations and Conclusions

**1**. The Word level model has a significantly large number of parameters as compared to the character level model due to the fact that character vocabulary is much smaller than that of in word level.This causes faster training and faster convergence of character level model.(convergence in sense of lower cross entropy loss after same number of epochs)
**2**. The Accuracy of character level model is also coming high because for accuracy calculation the next character predicted is been looked upon and as the possibilities of next character predicted are very less in number as compared to that in word level case the chances of predicting true next character becomes comparatively high.
**3**. The perplexity of character level model was also very low compared to word level model and the reason was based on the same fact that low vocabulary size causes more chances of the next character predicted by model is same as the true next character.
**4**.But the Sentences when generated , it was observed that the word level model was giving more meaning full sentences and capturing better semantics compared to character level.

## 7 Github Link

The code for these models are uploaded in the github page:
https://github.com/karanMal/Neural_Language_Model