



Introduction to NLP

Neural Language Modelling

Assignment 2

Anishka Sachdeva (2018101112)

Overview

Implement LSTM Language Model or GRU Language Model and compare its performance with the statistical N-gram Language Model.

Goals

1. Implementing LSTM Language Model or GRU Language Model trained on the Brown Corpus. Data cleaning should be performed to remove any special characters and use it for training the Neural LM. Inbuilt libraries for tokenization can be used. Either PyTorch or TensorFlow should be used as the deep learning frameworks.
2. Calculate the perplexity scores for each sentence of train, validation and test corpus.

3. Calculate average perplexity score on the train, validation and test corpus.
 4. Compare the perplexity scores of the statistical n-gram language model and neural language model trained on the Brown Corpus.
-

Implementation Specs

Libraries Used

1. Tokenization : Did not use any particular library. Tokenized the dataset manually.
2. Deep Learning Framework : **Tensorflow**

Code Implementation Details

Task 1 : Preprocessing the Dataset

1. Firstly the dataset i.e. Brown Corpus is loaded. Then the text preprocessed in the following manner :
 - All alphabets are converted to lowercase .
 - All the tokens containing purely alphabets are kept as it is.
 - Tokens containing characters other than alphabets are handled as follows (based on the trends observed in data) :
 - '-' - If it occurs at the beginning/end of the token, it is removed and if the resulting token contains only alphabetic characters, the token is retained.
 - '^' character is removed if it occurs in a token and if the resulting token contains only alphabetic characters, the token is retained.
 - If a token contains a single non-alphabetic character, and that too at the last index , we remove the non-alphabetic character and retain the token if the number of characters is greater than one or the token results in 'i' or 'a'.
 - Rest all the tokens containing non alphabetic characters have been discarded.
2. Further after breaking the dataset into tokens, vocabulary is created (a set of unique words in the corpus).

3. Now we create 2 dictionaries representing word-to-id and id-to-word which will be used when the neural network predicts the next word. We require an id-to-word dictionary because the neural network predicts the id of the next word in a 4 gram model instead of the word itself. And thus the mapping of id to word is required.
4. For word embeddings : **One Hot Encoding** is used.

Task 2 : Splitting the Dataset

1. The dataset has been split in the order of 7:2:1 for Train : Test : Validation. It is done in the following manner :
 - First the dataset is split as 8 : 2 for Train : Test.
 - Then the Train set is further split as 7 : 1 for Final Train : Validation.

Task 3 : Building the Neural Model

1. The following are used :
Loss = 'categorical_crossentropy'
Optimizer = 'adam'
Activation = 'softmax'
Metrics = ['accuracy']
Epochs = 300 - 400 gave better results.

Task 4 : Calculating the Perplexities

1. Perplexity is calculated using the following formula :
Perplexity = float(1)/float(math.exp(float(probability)/float(n)))
Here probability = probability of each sentence in the test set.
Probability of each sentence is calculated by the formula
 $\exp(\text{math.log}(p_1) + \text{math.log}(p_2) + \text{math.log}(p_3) + \dots + \text{math.log}(p_N))$
Here n = number of tokens in sentence after preprocessing after preprocessing .
-

Statistical Language Modelling VS Neural Language Modelling

The Neural net-based Language Models turn out to be advantageous over the N-gram Language Models.

N-gram Language Models

Merits :

- Really easy to build, can train on billions and billions of words.
- Smoothing helps generalize to new data.

Demerits :

- Only work well for word prediction if the test corpus looks like the training corpus.
- Only captures short distance context.
- Statistical N-gram Language Models suffer from **Data Sparsity**.

Neural Language Models overcame some of the issues addressed above.

Neural Language Models

Merits :

- They do not require smoothing techniques.
- They can handle longer histories.
- They can generalize over contexts of similar words.
- For a training set of a given size, a Neural Language Model has much higher predictive accuracy than an N-gram Language Model.

Demerits :

- There is a cost involved for this improved performance of Neural Language Models. These are slower to train the traditional language models.
-

Observations and Inferences on running Brown Corpus on both Language Models

Average Perplexity in Neural Language Modelling

Train Set = 47.03318707455234

Validation Set = 49.01847669221968

Test Set = 47.852277217687

Average Perplexity in Statistical Language Modelling

Kneser Ney :

Train Set = 18.148999210964906

Test Set = 775647.1252473283

Validation Set = 848820.8206521824

Witten Bell :

Train Set = 19.31509174277874


Test Set = 391399.7706115251

Validation Set = 483337.09168899996

Observation : Neural net based language model showed better results as compared to the statistical n-gram language model.

Reasons :

1. Neural Language Model generalises the context much better for similar words.
2. Also, Neural language Model helps in keeping a track of longer histories as well.
3. The perplexity scores for training set in Statistical n-gram Language Model is much better as compared to that in validation and test set. Thus, we can conclude that the statistical model works well for training data but as soon as



the model comes across some unseen data/sentences, the perplexity for such sentences increases, thereby increasing the average perplexity score.

Whereas, we can notice that for Neural Language Model, average perplexity scores for train, test and validation sets are almost equivalent.

Improvements :

1. Using Word2Vec for word embeddings instead of One Hot Vector Embedding because One Hot Vector Embedding for words does not generalise the context of similar words well.
-