Assignment2 Report

Praveen Gupta
Computer Science and Automation
Indian Institute Of Science
praveengupta@iisc.ac.in

Abstract

Build a neural language model on **Gutenberg**(D2) corpus. And then evaluate the perplexity

- **S2**: Train: D2-Train, Test: D2-Test
- **Task 1:** Build the best **token** level LSTM-based language model and evaluate the perplexity for the setting above.
- **Task 2:** Build the best **character** level LSTM-based language model and evaluate the perplexity for the setting above. Also compare and contrast performances of these two models with the best classical LM for this setting from Assignment 1
- **Task 3:** Using best model from above, generate a sentence of 10 tokens.

1 Preprocessing

I divide the dataset D2 in test, dev and train in the ratio of 8:1:1. Then i remove the punctuations except for full stop and comma and convert the data to lowercase.

- **Token Level LSTM-based LM:** For word level LM i replace all the single appeared word with UNK symbol. This i done to handle out of vocabulary words in test data. Then i use the 30-gram to feed to my network, so i convert train data into the lines of length 31. But words cannot be given to network, so now i use tokenizer to encode these lines. It gives each unique word a unique number. So now i have lines of 31 numbers each. Next part is to separate the 30 words as input and 1 word to be predicted word from each line. So i separate the input data as X and do one-hot encoding the word to be predicted as Y.
- **Character Level LSTM-based LM:** Here i use the 10-character to feed to my network, so i convert train data into the list of length 11 characters. But character cannot be given to network, so now i use dictionary to encode these characters. I find all the unique character and give them a number. It gives each unique character a unique number. So now i have lines of 11 numbers each. Next part is to separate the 10 characters as input and 1 character to be predicted for each 10 input character as output. So i separate the input data as X and do one-hot encoding the character to be predicted as Y.

2 Implementation

I have implemented following language models for given datasets.

2.1 Token Level LSTM-based language model

2.1.1 Word Model 1:

I used 50-dimensions to represent each word. I used a two LSTM hidden layers with 70 memory cells each. A dense fully connected layer with 70 neurons connects to the LSTM hidden layers to interpret the features extracted from the sequence. Then comes the output layer which predicts the next word as a single vector equals to the size of the vocabulary with a probability for each word in the vocabulary. Then i used a softmax activation function to normalize the probabilities. I used the cross entropy loss to calculate the error and to update the weights accordingly. I run it for 40 epochs with the batch size of 400.

2.1.2 Word Model 2:

This time i used 100-dimensions to represent each word. This is the size of the embedding. I used a two LSTM hidden layers with 125 memory cells each. A dense fully connected layer with 125 neurons connects to the LSTM hidden layers to interpret the features extracted from the sequence. Then comes the output layer which predicts the next word as a single vector equals to the size of the vocabulary with a probability for each word in the vocabulary. Then i used a softmax activation function to normalize the probabilities. I used the cross entropy loss to calculate the error and to update the weights accordingly. I run it for 45 epochs with the batch size of 400.

2.2 character level LSTM-based language model

I used a single LSTM hidden layer with 100 memory cells. As i am giving the 10 character as input so my input layer have size 10 times the vocabulary size. A dense fully connected output layer that outputs one vector with a probability distribution of all characters in the vocabulary. Again i used a softmax activation function normalize the probabilities.

2.3 Sentence Generation

As i used 30 words to predict the next word, i have to provide 30 words to start with then the model will generate the next word. For each 30 word input, model will give the vector with probability of each word. Then i took the maximum probability and pick the corresponding word using this index. Then next time this word is appended to the input text and the first word from the input is removed.

3 Result

3.1 Task 1 and 2

Train:Dev:Test = 8:1:1

S2 : train = D2 train and test = D2 test	
Stupid back off using trigram	90.41
Word level LSTM 1	263.87
Word level LSTM 2	89.63
Character Level LSTM	6.38

Table contain perplexity for Interpolation method from last assignment(this time tested on subset of dataset), Word Level LSTM and Character Level LSTM model.

3.2 Task 3

Some example of sentences generated from Token level LSTM based Language Model:

- **Input:** ....behold the man hath appeared unto me that came
  Generated: to the lord and said unto him i will not be ashamed of the lord
- **Input:** ....then shall he offer it before the lord but
  Generated: the lord hath given thee and the lord said unto moses behold i will make
- **Input:** ....and such as had ability in them to stand in the
  Generated: day of the lord and the king s chamberlain and

4 Acuuracy/Measures

4.1 Task 1

Perplexity is used as the measure for this task.
(All values in Result section table contain perplexity value)

4.2 Task 2

Human Evaluation.

5 Code Link on Github:

https://github.com/praveengupta18/NLU-Assignment2