Assignment-2 Report

Anshuman Mourya
Computer Science and Automation
Indian Institute of Science , Bangalore
anshumanm@iisc.ac.in

Abstract

To designed a language model on **Gutenberg**(D2) corpus using neural language models and compare LM performance from Assignment 1. Following are two tasks that we performed -

- **Task 1:** To Build the best token level LSTM-based language model
- **Task 2:** To Build the best character level LSTM-based language model.
- **Task 3:** Using best model from above,generate some sentences of 10 tokens

1 Dataset Description

All three models - trigram interpolation , words level LSTM and character level LSTM are trained on same dataset containing 5 files of the Gutenberg dataset namely austen-emma, austen-persuasion, austen-sense, bible-kjv, blake-poems. Train ,test and dev splits are then made in the ratio 70:15:15.

2 Preprocessing

- Replacing unigrams in train data with frequency less than two by UNK in order to handle out-of-vocabulary words in test data.
- Removing Punctuations and numeric characters from the data
- Converting every word to lowercase for language model task.
- Replacing out-of-vocabulary words in test data with UNK.

3 Literature Review

3.1 Long Short Term Memory

RNN may face problems like vanishing gradient. In order to eradicate this porblem, LSTMs were introduced. Following are the representation of a LSTM: The input gate equations :

$$g = tanh(b^g + x_t U^g + h_{t1} V^g)$$

$$i = \sigma(b^i + x_t U^i + h_{t1} V^i)$$

$$g \emptyset i$$

The forget gate equations :

$$f = \sigma(b^f + x_t U^f + h_{t1} V^f)$$

$$s_t = s_{t1} \emptyset f + g \emptyset i$$

The output gate equations :

$$O = \sigma(b^o + x_t U^o + h_{t1} V^o)$$

$$h_t = tanh(s_t) \emptyset O$$

4 Implementation

4.1 Language Model Used

- **Word level LSTM** : A 5-layer LSTM model is used . First layer is the Embedding layer whose input is a one hot vector of a 50 word sequence. This layer produces a condensed embedding of sixe 50x50. Second and Third layers are LSTM hidden layers.These layers have 100 memory cells each. A dense fully connected layer with 100 neurons connects to the LSTM hidden layers to interpret the features extracted from the sequence. The output layer predicts the next word as a single vector the size of the vocabulary with a probability for each word in the vocabulary. A softmax activation function is used to ensure the outputs have the characteristics of normalized probabilities. There are 100 neurons on both the hidden layers . Next two layers are fully connected dense layers, out of which first have 100 neurons while the second has

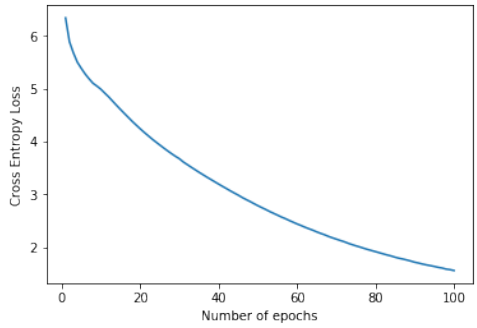

Figure 1: Word level LSTM with Stride = 3

neurons equal to the number of unique tokens. Hyperparameters in this model are - Stride and number of epochs. Tried varying each one of them. Stride is varied from 1 to 3 and maximum number of epochs used is 100.

- **Character level LSTM** : The model has an input layer that takes sequences that have 10 time steps and 38 features for the one hot encoded input sequences. I used the second and third dimensions on the X input data. This is so that if we change the length of the sequences or size of the vocabulary, we do not need to change the model definition. It has a single LSTM hidden layer with 75 memory cells. The model has a fully connected dense output layer that outputs one vector with a probability distribution across all characters in the vocabulary. A softmax activation function is used on the output layer.

4.2 Sentence Generation

Both the word level and character level LSTM models are tested for sentence generation . For word level LSTM, I passed a seed text of 50 tokens to let the model predict next 10 word sentence. I also applied padding of 0 to make exactly 50 token length. Similarly for character level LSTM, I passed a seed string of 5-10 characters to let it generate some next characters.

5 Link to github code

https://github.com/anshumanmourya/NLU-Assignment2.git

6 Result

6.1 Task 1 and Task 2

Interpolation n-gram model from Assignment1	
Perplexity on Test	75.43
Word Level LSTM	
Stride Used	3
Number of Epochs	100
Cross-entropy loss on Training data	1.5560
Perplexity on Training data	4.7398
Accuracy on Training data	0.6197
Perplexity on Test Data	97.56
Character Level LSTM	
Stride Used	1
Number of Epochs	20
Cross-entropy loss on Training data	1.265
Perplexity on Training data	3.54
Accuracy on Training data	0.712
Perplexity on Test Data	5.5

6.2 Task 3

Some examples of token generated from best Language Model (word level):

- that nothing can exceed the accommodations of a man of
- witness of edward i would have had the smallest advantage
- and i have not known thee and i will eat
- shall not be afraid of the land of israel and

7 Acuuracy/Measures

7.1 Task 1

Perplexity is used as the measure for this task. It is calculated as the exponent raised to the power cross entropy. Accuracy in the table denotes the correctly classified examples during training.

7.2 Task 2

Human Evaluation is required to measure the readability of sentence.