Assignment 2: LSTM based Language Models

Anand Mohan - SR No: 14417 - M Tech(SE)
moghan.anand@gmail.com

1 Theory

1.1 Language Models

Models that assign probabilities to sequences of words are called language models. LSTM based language models at both character level and word levels can be implemented. Basically, LSTMs accept an input vector x and give you an output vector y. However, crucially this output vectors contents are influenced not only by the current input you just fed in, but also on the entire history of inputs fed into it in the past.

1.2 Model Implemented

Both character and token level models are implemented with similar structures. All unique characters or tokens are encoded into integers and are fed into an embedding layer which learns embedding for each of the character and tokens present in the training set. The next two layers are LSTM layers of size 128 each. The output layer is a dense softmax layer having nodes which gives probabilities of each character or token in the vocabulary. Output is decided on basis of the probability. The loss metric used is cross entropy loss. The code is written using tensorflow library.

1.3 Handling Unknowns

Character Level LSTM: A fixed character set is used for integer encoding. All other characters present in the training set is replaced by an unknown token and which is also present in the character set. This ensures that even the characters not present in the vocabulary are assigned some probability in the output layer.

Token Level LSTM: Counts of all unique tokens in the training set are taken and a percentage(say 20%) of words having unit counts in the training set are replaced by the unknown token. The vocabulary is then made and all words not in it will be treated as unknown. This ensures that even the words not present in the vocabulary but present in the test data will be assigned some non-zero probability while testing.

1.4 Sentence Generation

Some random seed can be given for both character and token level models. The seeds are then encoded as integers and given to the network. The network will change its state and give probabilities as output which can be sampled to get the next character or token.

1.5 Metric

Perplexity: The perplexity (PP) of a language model on a test set is the inverse probability of the test set, normalized by the number of words.
For a test set $W = w_1 w_2 ... w_N$:

$$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}}$$

For character level perplexity, probability of characters are used instead of probability of words.

2 Methodology

Whole of Gutenberg corpus is used. Data is split into training, dev and test sets.

2.1 Character Level LSTM Model

Each character in the train, dev and test are encoded to an integer value based on a dictionary made from a character vocabulary of size 46. All characters not present in the vocabulary are replaced by an unknown token present in the character vocabulary itself. The encoded data is then

split into batches of size 50 sequences. Each sequence again is a set of 50 consecutive characters. Each output corresponding to an input sequence is a one character down sequence. The states of LSTMs are reset every batch. The model is trained for 30 epochs minimizing cross entropy loss using an Adam optimizer starting with a learning rate of 0.002 and a decay rate of 0.97 every epoch. Model checkpoints are saved every 1000 batches.
Testing: A set of 100 length sequence is made from the test data and given as input to the model, which outputs 100 sets of probabilities. Log probability corresponding to each output character can be found out and be added together for the whole test data to get the log perplexity. Anti-log is taken to get normal perplexity.

2.2 Token Level LSTM Model

Each character in the train, dev and test are encoded to an integer value based on a dictionary made from a vocabulary after removing some of the unit count words for including unknown token. The encoded data is then split into batches of size 30 sequences. Each sequence again is a set of 30 consecutive words. Each output corresponding to an input sequence is a one word down sequence. The states of LSTMs are reset every batch. The model is trained for 30 epochs minimizing cross entropy loss using an Adam optimizer starting with a learning rate of 0.002 and a decay rate of 0.97 every epoch. Model checkpoints are saved every 1000 batches.
Testing: A set of 100 length sequence is made from the test data and given as input to the model, which outputs 100 sets of probabilities. Log probability corresponding to each output token can be found out and be added together for the whole test data to get the log perplexity. Anti-log is taken to get normal perplexity.

3 Results

All models were trained and tested on whole Gutenberg corpus.

- Character Level LSTM: Perplexity = 3.67
- Token Level LSTM: Perplexity = 59.9
- Assignment 1 Model: Perplexity = 152.3

Sample Generated Sentence: thy lord for the house of the ark of every.

4 Observations

- Even though Character Level LSTM perplexity value cannot be compared with other perplexities, the low value indicates a good model.
- Token Level LSTM when trained on the whole corpus for 30 epochs, perplexity has reduced to almost one-third of that of previous model, which shows the effectiveness of LSTM models.
- Also, when trained over the whole corpus, the sentences generated by the Character Level LSTM model were biased to bible parts of the Gutenberg corpus.