Natural Language Understanding Assignment 2

Udit Gupta
M.tech CDS
14454
udit2008.gupta@gmail.com

Dataset: Gutenberg Corpus
Libraries used: Keras(Tensorflow backend), Nltk, Numpy
Preprocessing:The data is split into 60-20-20 train, validation, test. These splits are consistent with the ones used in the previous assignment to ensure proper comparison between the Markovian language model and the Neural language model. As in the previous assignment the punctuations were removed, text converted to lowercase and each sentence was prepended with <s> and appended with </s> tokens.

Task 1

Metric used for comparison: Perplexity of the next word in the sequence on test set.
No. of unique tokens/Vocab size: 14625
Methodology:

1. Each word in the vocab is assigned a unique integer.
2. The nested lists of sentences are flattened. In the flattened list each sentence is discriminated by <s> and </s> tokens.
3. The 100 dimensional embeddings were initialized using pre-trained glove embeddings **'glove.6B.100D'**. Making them the model parameters and allowing them to adapt according to the context gave better results on validation data.
4. In each step, 9 consecutive elements from the sequence were fed as input to the LSTM working in 'Acceptor' mode with 50 hidden units. The 10th element, the target, is one hot encoded. In each successive step the window is shifted by one, so in the next step, the 11th element is the target and its preceding 9 elements its input. For the task of language modeling, the Acceptor mode gave better results than Transducer mode on the validation data. This is because the true input word in the next state is known (Acceptor) and using the predicted word (Transducer) will be less beneficial.
5. After experimenting with several combinations and checking accuracy on validation data, the best model for this task was a Bidirectional LSTM with 50 hidden units and a 'tanh' activation function using a dropout of 20 % to prevent overfitting. It is connected to a dense layer with 'vocab-size' number of outputs and a 'softmax' activation function.
6. In case of Bi-LSTMs, the forward and backward outputs are concatenated and then fed to the next layer.
7. The objective function is the categorical cross-entropy. The minimizer used is 'adam' because it gives fast convergence.
8. The model was trained for 500 epochs (batch size:128).

Baseline: Markovian language model implemented in Assignment 1. **Perplexity**: 144, with a vocab-size of 14625

Perplexity for the current model

Embedding size	Model	Train-perplexity	Test-perplexity
50D	LSTM	1.52	105
100D	LSTM	1.41	98
50D	Bi-LSTM	1.28	89
100D	Bi-LSTM	1.15	78

Text generation:The trained model is fed with a seed text. If the length of the seed text is shorter than 9 tokens, then it is padded with zeros to make

the input sequence of length 9. This sequence is used as input to the trained model to predict probabilities for each word. The word with the highest probability is chosen as the next word. Then, the predicted word along with 8 words preceding it is used to generate the next word and so on.
Observations:The Neural language model using LSTMs has much lower perplexity than the Markovian language model. Improving the model using Bidirectional LSTMs and increasing the dimension of word vector embedding further reduces the perplexity on the test set.
Conclusion: Neural language model using LSTMs makes better predictions for the next word in the sequence. This is because it can keep track of long range dependencies which was a serious drawback in the Markovian language model. Bidirectional LSTMs perform even better than the vanilla LSTMs as they also encode the future context along with the past.

Task 2

Metric used for comparison: Perplexity of the next character in the sequence on test set.
No. of Unique characters: 43
Methodology:

1. Create a mapping of unique characters to integers.

2. In each step, 50 consecutive characters from the preprocessed text are fed as input to the LSTM working in 'Acceptor' mode. The 51st character, the target, is one hot encoded. In each successive step the window is shifted by one, so in the next step, the 52nd character is the target and its preceding 50 characters its input. For the task of language modeling, the Acceptor mode gave better results than Transducer mode on the validation data.

3. After experimenting with several combinations and checking accuracy on validation data, the best model for this task was a Bidirectional LSTM with 256 hidden units and a 'tanh' activation function using a dropout of 20 % to prevent overfitting. It is connected to a dense layer with 43 ('# unique characters') number of outputs and a 'softmax' activation function.

4. The objective function is the categorical cross-entropy. The minimizer used is 'adam' because it gives fast convergence.

5. The model was trained for 500 epochs (batch size: 8192).

Perplexity for the current model

#Hidden Units	Model	Train-perplexity	Test-perplexity
128	LSTM	5.13	6.58
256	LSTM	4.94	5.42
128	Bi-LSTM	4.85	5.04
256	Bi-LSTM	4.47	4.55

Text generation:The trained model is fed with random seed text which the model interprets as a sequence of characters. If the length of the sequence is shorter than 50 characters then it is padded with zeros to make the input sequence of length 50. This sequence is used as input to the trained model to predict probabilities for each character. The character with the highest probability is chosen as the next character in the sequence. Then, the predicted character along with 49 characters preceding it is used to generate the next character and so on.
Observations:The Neural language model using LSTMs has much lower perplexity than the Markovian language model. Improving the model using Bidirectional LSTMs and increasing the number of hidden units further reduces the perplexity on the test set.
Conclusion: Even for character level encoding, Bidirectional LSTMs perform better than the vanilla LSTMs as they also encode the future context along with the past.

Task 3

Sentence Generation: For sentence generation, the word (token) level model has been used, since some of the words generated using character level model are incorrectly spelled. The method of text generation has been explained in the respective tasks above.
Generated sentence: **"buster bear yawned as he lay on his comfortable bed of leaves and watched the first early morning sunbeams"**