

# Natural Language Understanding with Distributed Representations - Assignment 3

Sean D Rosario - sdr 375

November 5, 2016

## 1 Bag of Words vs LSTM

With the Bag of words example, we would have all unique words in the corpus, along with their counts for each document. To perform classification, we would perform some computation on the learnt word embeddings (like averaging and softmax in Assignment 2 for CBOW). The shortcoming with the bag of words model is that position of the words in the sentence is not considered, which may affect performance.

For example, consider these two sentences : I hate this, I don't like it. I like this, I don't hate it. These sentences have the exact same words. However, the sentiment they express quite the polar opposite. The bag of words model would give the exact same prediction for the two sentences. LSTM on the other hand would look at a given words and the preceding words, thus encoding more context information. Hence the LSTM model would encode "this" after "don't like" differently from "this" after "don't hate"

The LSTM model for sentiment analysis would encode sequence dependent information. For example, the embedding for "like" followed by "do" would different from the embedding for "like" followed by "not". Hence, in order for the LSTM model to act as a bag of words model, each LSTM cell should not carry information about preceding words. This can be achieved by modifying the forget gate. As mentioned in Christopher Olah's blog post about LSTMs, a sigmoid layer called the "forget gate layer" looks at  $h_t$  and  $x_t$  and outputs a number between 0 and 1 for each number in the cell state  $C_{t-1}$ . A 1 represents "completely keep this" while a 0 represents "completely get rid of this."

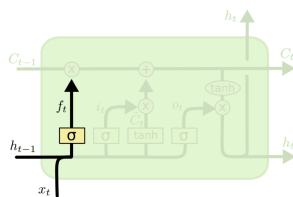


Figure 1: Illustration of the forget gate in the LSTM cell

Hence, to wipe out all the memory carried over from the previous cell, we want  $f_t$  to be completely zero. To implement this, we can hard code  $f_t$  as zero

## 2 RNN language Model

### 2.1 Train a language model

Using the starter code of the Tensorflow tutorial, I trained an RNN language model, on the Penn Tree Bank dataset. I used the LSTM variant of the RNN cell, which is better at encapsulating important long term dependencies.

Since training on my local machine would take hours, I used NYU's mercur to train the RNN on a GPU.

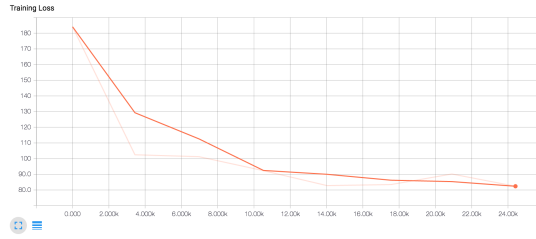


Figure 2: Training loss for the default LSTM model

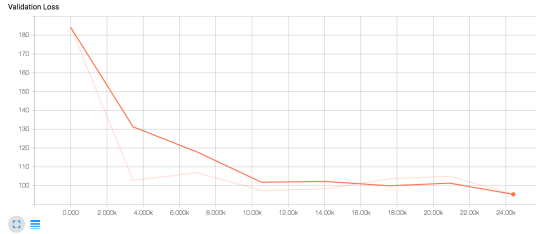


Figure 3: Validation loss for the default LSTM model

## 2.2 Removing gates

I look at the performance of the LSTM model by removing component of the cell architecture one at a time

### 2.2.1 Removing input gate

Without any input gate, the model achieved the following performance:

Train Perplexity: 54.751, Valid Perplexity: 124.707, Test Perplexity: 118.952,

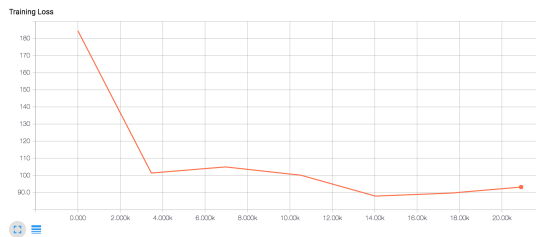


Figure 4: Training loss over time for LSTM without input gate

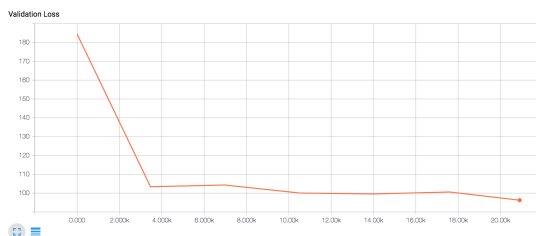


Figure 5: Validation loss over time for LSTM without input gate

### 2.2.2 Removing forget gate

Without any forget gate, the model achieved the following performance:

Train Perplexity: 89.062, Valid Perplexity: 138.464, Test Perplexity: 131.596,

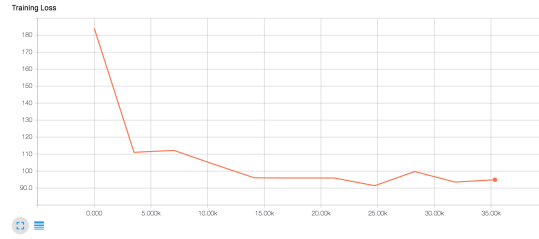


Figure 6: Training loss over time for LSTM without forget gate

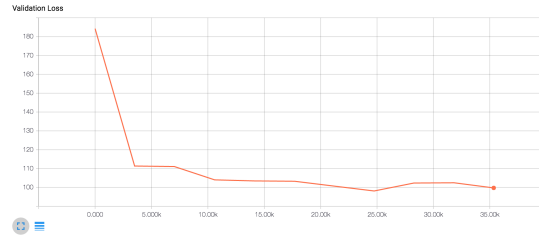


Figure 7: Validation loss over time for LSTM without forget gate

### 2.2.3 Removing output gate

Without any output gate, the model achieved the following performance:

Train Perplexity: 650.302, Valid Perplexity: 650.733, Test Perplexity: 610.948,

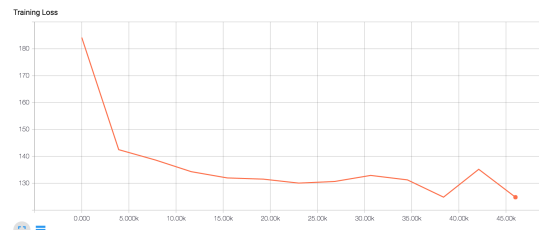


Figure 8: Training loss over time for LSTM without output gate

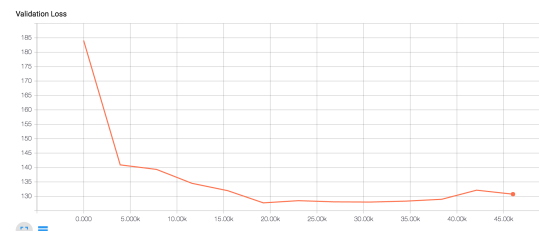


Figure 9: Validation loss over time for LSTM without output gate

## 2.3 Gated Recurrent Unit

I replaced the LSTM architecture with the Gated Recurrent Unit.

While training I noticed that the training perplexity did not converge, and resulted in a final test perplexity of 274.928. The model trained for only 6 epochs, due to early stopping with a patience of 1. Hence I decreased the learning rate from the default 1.0 to 0.6. This lower learning rate solved the problem.

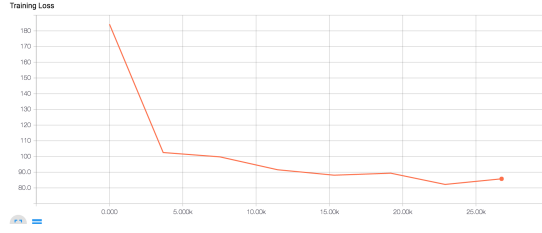


Figure 10: Training loss over time for GRU

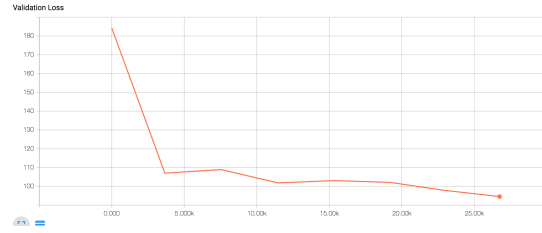


Figure 11: Validation loss over time for GRU

## 2.4 Setup evaluation

I used the given `score()` function to calculate the score of the the default LSTM model (with all gates intact). I use the sci-kit learn cosine similarity function. I get a model score of 9 out of 10 I use the sci kit learn TSNE function to do visualize some of the word vectors in 2 dimensions

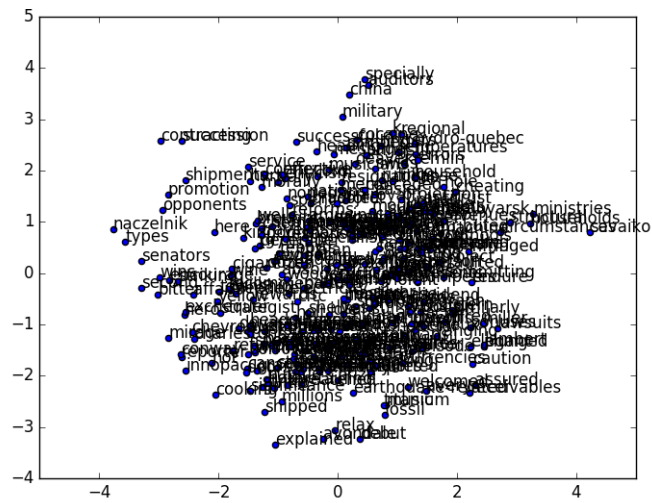


Figure 12: t-SNE visualization of embeddings using sklearn