Wesley Xiao A14674605
Catherine Wang A14394510
LIGN 167, Bergen
December 14, 2018

Sentiment Analysis of IMDB Reviews

We are trying to achieve near accurate sentiment analysis. The basic task of sentiment analysis is to classify the polarity of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. Determining the sentiment of text is a relatively simple task for humans, but it is particularly challenging for machines to do. For our project, we attempt to design and implement model using various deep learning techniques to accomplish this task.

Movie reviews are an ideal dataset for sentiment analysis because they are a clear expresses on a variety of opinions. We used the Large Movie Review Dataset v1.0. It is a dataset of 50,000 highly polar movie reviews, evenly split between of positive and negative reviews. Positive reviews are labeled 1, while negative reviews are labeled 0. The completed dataset was split into the following sets: 40% training set, 10% validation set, and 50% testing set (17,500 reviews for training, 7,500 reviews for validation, and 25,000 reviews for testing). Raw text and already processed bag of words formats are provided. Our model is trained and tested using IMDB movie reviews. To load the dataset into our program, we used torchtext: a library that consists of many data processing tools and popular natural language datasets, including the IMDB dataset. Using torchtext made loading and processing the data relatively simple.

For our project, we implemented two different models.

Model 1

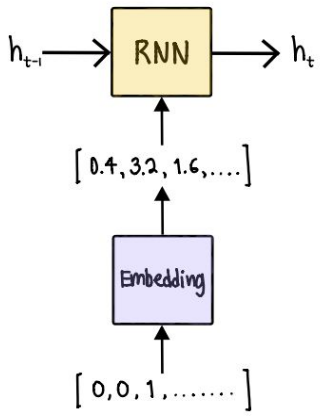


Figure 1

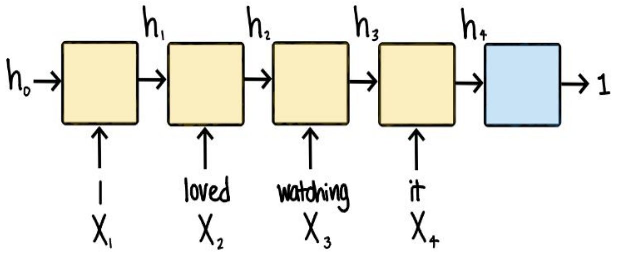


Figure 2

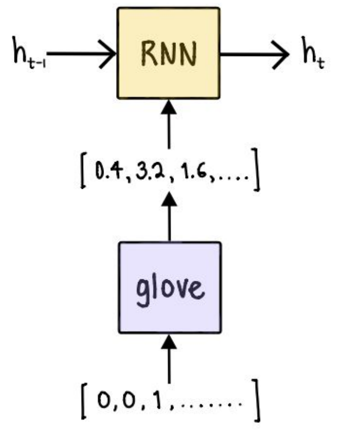
For our first approach, we use a basic recurrent neural network (RNN). The general outline of our model can be seen in figure 2. The RNN takes in a sequence of words, X *= {x1, x2, ... },* one at a time and produces a hidden state , *h*, for each word. The current word and the hidden state from the previous word is fed into the RNN to produce the next hidden state. The initial hidden state is a tensor initialize to all zeros. The final hidden state of the RNN is passed into a linear layer, where it is converted into a score between 0 and 1. That final score is the model's sentiment prediction: if it is closer to 0, then it is negative; if it is closer to 1, then it is positive. Before we put the word into the RNN, we convert it to a one-hot vector and then pass it through an embedding layer. This process is shown in figure 1. The one-hot vector is a sparse vector (most of the elements are 0) and is a unique representation for a word in our vocabulary. The embedding layer converts the one-hot vector to a dense vector (its dimensionality is smaller and it is all real numbers). Ideally, words that have similar meanings are mapped to similar dense vectors. That dense vector is what is actually passed into the RNN. The optimizer used is stochastic gradient descent with a learning rate of 0.001. For the loss function, we use binary cross-entropy loss. This function passes in the predicted score of the input into the sigmoid function. Then, it takes that result and passes it into cross-entropy loss. With s_1 as the predicted score (a number between 0 and 1 in this case) and t_1 as the true score for the input, the loss function does the following calculation.

$$f(s_i) = \frac{1}{1+e^{s_i}} \quad \Longrightarrow \quad CE = -t_1 log(f(s_1)) + (1-t_1) log(1-f(s_1))$$

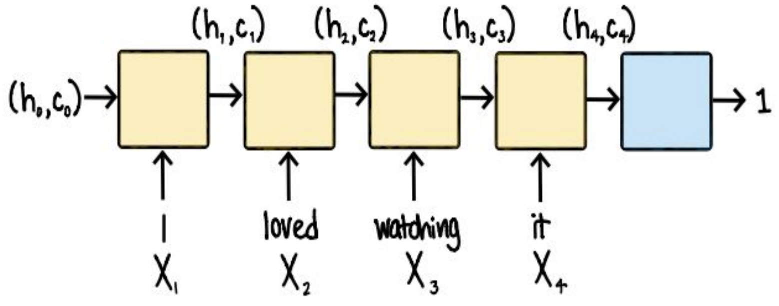
Sigmoid Function

Cros s-entropy loss

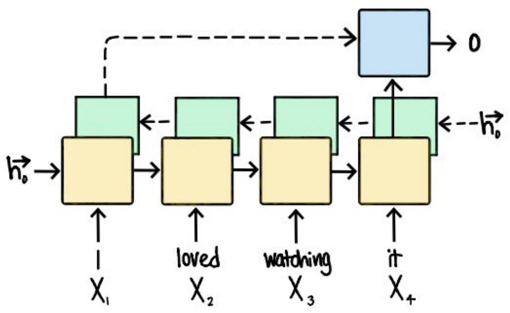
<u>Model 2</u>



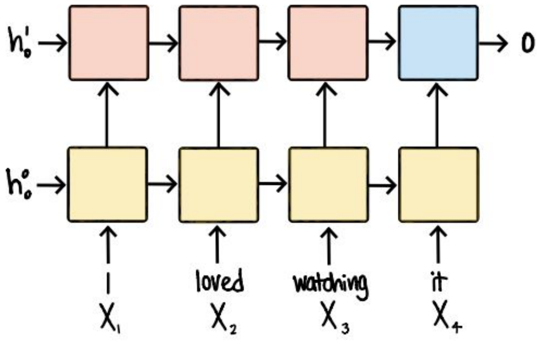
Embedding Layer



LSTM



Bidirectional RNN



Multi-layer RNN

However, we wanted to get more accurate with our results so we implemented a second model that is an upgraded sentiment analysis. For our second approach, we basically modify our first model to get a better accuracy. We used pre-trained word embeddings instead of randomized ones when we generate one-hot vectors, use a different RNN architecture that uses bidirectional RNNs, and multi-layer RNNs. We also bring in regularization method dropout, allowing us to not overfit (memorizing training data to cause a low training error but high validation and high testing error, making way for poor generalization to new and unseen examples).

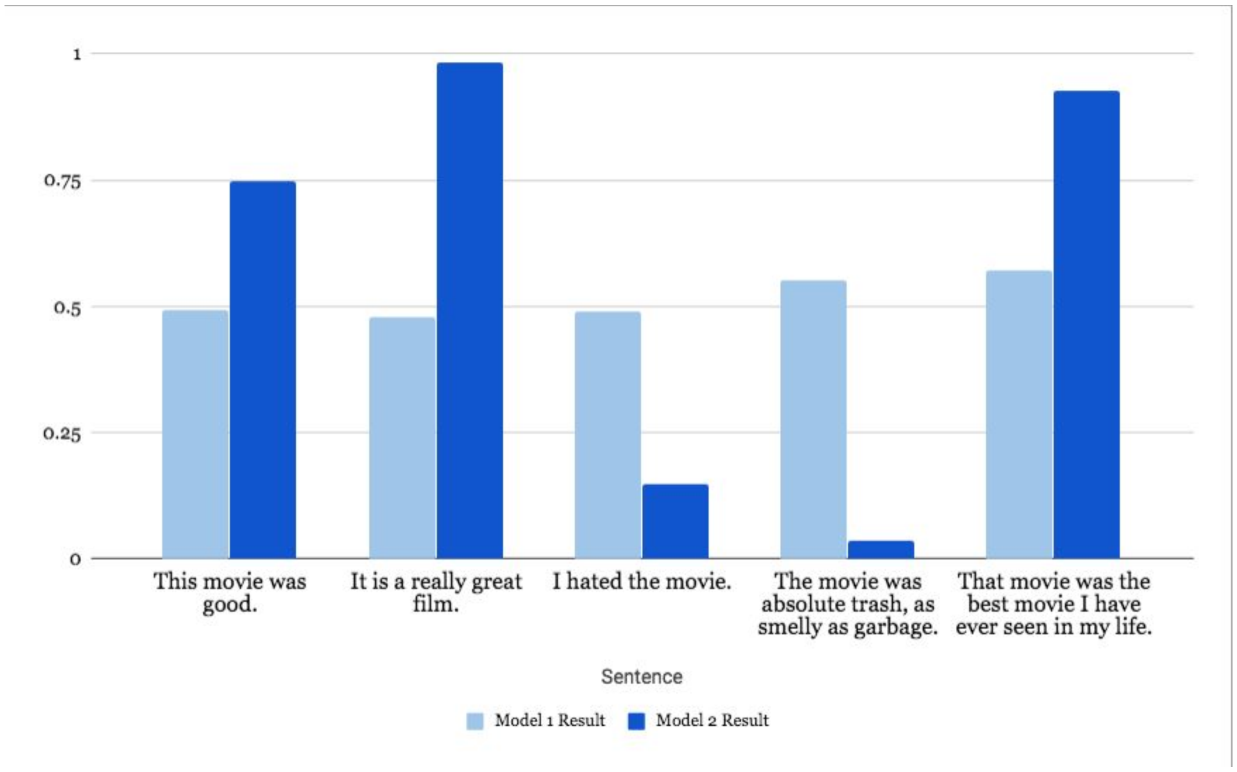
Lastly, we implemented a different optimizer of Adam as opposed to SGD before. While SGD (stochastic gradient descent) does update all the parameters with the same learning rate, Adam adapts to the learning rate for each individual parameter, giving parameters that are updated more often lower learning rates and parameters that are updated less often higher learning rates. vt is storing exponentially decaying of past squared gradients, mt is exponentially decaying average of past gradients, similar to momentum.

$$m_t=\beta_1 m_{t-1}+(1-\beta_1)g_t$$
$$v_t=\beta_2 v_{t-1}+(1-\beta_2)g_t^2$$

To counteract biases it computes bias-corrected first and second moment estimates and uses the third equation to update the parameters.

$$\hat{m}_t=m_t\div(1-\beta_1^t)$$
$$\hat{v}_t=v_t\div(1-\beta_2^t)$$
$$\theta_{t+1}=\theta_t-(\eta\div(\sqrt{\hat{v}_t}+\varepsilon))\hat{m}_t$$

What is great about using adam is that the parameters update are invariant to re-scaling of gradient, where if we have a function f(x), we change it to some constant k*f(x) and there won't be an effect on performance. While SGD decreased step size after some epochs, nothing like that is needed when we use adam.



Our second approach is a much more successful than our first approach. In our basic RNN model (model 1), the accuracy of the model on the testing set was 46.13%. Retraining and retesting our model multiple times, we saw that the accuracy of our first approach was always a little below 50%. Our second approach, the more sophisticated RNN architecture, was much more accurate. The accuracy of model 2 on the testing set was 86.50%. If we look at a few examples, the difference between the two models becomes very clear. Below, we have the results of our first approach alongside the results of our second approach on a few sample movie reviews we created.