DePaul University – College of Computing and Digital Media

CSC 578: Machine Learning and Neural Networks-Fall 2016

Final Project Report

Project Title: Sentiment Analysis from Movie Reviews using Deep Learning

Student: Naga Venkateshwarlu Yadav Dokku,

Abstract

The aim of the project is to experiment with different neural networks to predict the sentiment and Sequence Classification of movie reviews and explore natural language processing (NLP) methods to perform sentiment analysis and discover how we can predict the sentiment of movie reviews as either positive or negative and predict a category for the sequence in Python using the Keras deep learning library. IMDB sentiment analysis problem for natural language processing and how to load it in Keras and use word embedding's in keras for NLP. Develop and evaluate a multi-layer perception convolutional neural network models for sentiment prediction and LSTM recurrent neural network model for sequence classification model for the IMDB problem. Finally, I compared and analyzed each model's performance on both binary classification and sequence classification of sentiment analysis tasks.

Introduction

Sentiment analysis is a well-known task in the realm of natural language processing. Given a set of texts, the objective is to determine the polarity of that text. [4] Provides a comprehensive survey of various methods, benchmarks, and resources of sentiment analysis and opinion mining. The sentiments can consist of different classes. For this project, I have consider two cases: 1) A movie review is positive (+) or negative (-). This is similar to [2], where they also employ a novel similarity measure.

For this case, I picked Large Movie Review Dataset. The objective of this part is to train a binary classifier for movie reviews (i.e., output classes are positive/negative). As in many natural language tasks, the first task here is to clean up, and convert the input texts (movie reviews) into numbers but for this project I have used keras datasets. Keras provides access to the IMDB dataset built-in.

Datasets

Binary classification dataset

I have used the keras built-in data provided in [1]. Here is a description of the data, provided by keras:

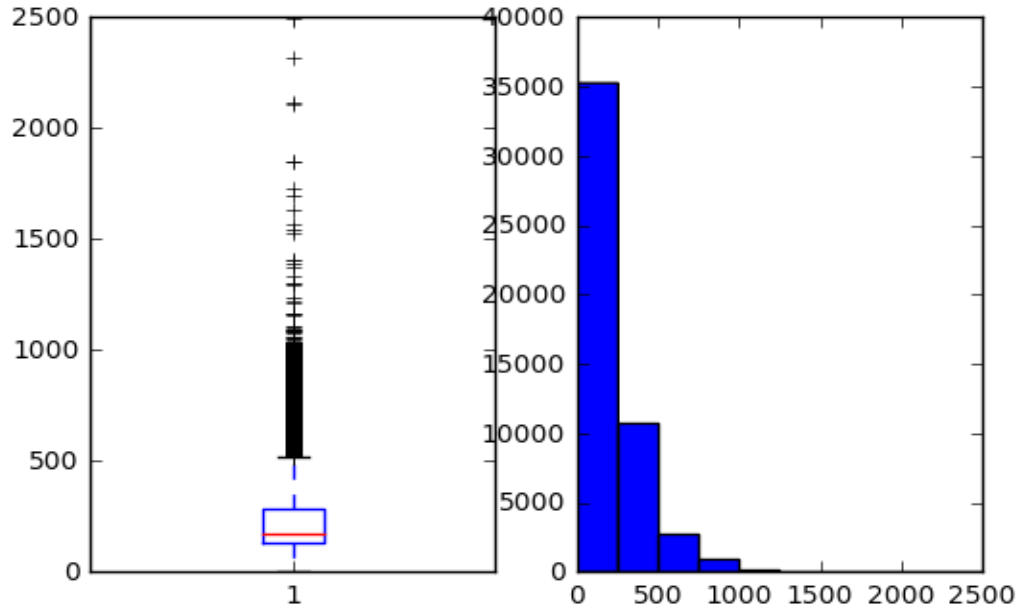
The Stanford Large Movie Review (IMDB) Dataset [5] consists of 50,000 "highly polar", binary labeled reviews from IMDB. These reviews are split 50:50 into training and testing sets. The distribution of labels within each subset of data is balanced. The dataset also includes a further 50,000 unlabeled reviews which may be used for unsupervised training. We found that reviews averaged 267.9 tokens in length with a standard deviation of 198.8 tokens; the precise distribution of review lengths is shown below.

The keras.datasets.imdb.load_data () allows you to load the dataset in a format that is ready for use in neural network and deep learning models. The words have been replaced by integers that indicate the absolute popularity of the word in the dataset. The sentences in each review are therefore comprised of a sequence of integers.

The imdb.load_data () provides additional arguments including the number of top words to load (where words with a lower integer are marked as zero in the returned data), the number of top words to skip (to avoid the "the"'s) and the maximum length of reviews to support.

We can see that it is a binary classification problem for good and bad sentiment in the review.

```
Classes:
[0 1]
```
Interestingly, we can see that there are just under 100,000 words across the entire dataset.
```
Number of words: 88585
```
The average review has just under 300 words with a standard deviation of just over 200 words.
```
Review length: Mean 234.76 words (172.911495)
```



Distribution of review lengths in the IMDB dataset.

For sequence classification dataset

Large Movie Review Dataset

This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. There is additional unlabeled data for use as well. Raw text and already processed bag of words formats are provided

```
Link for dataset: http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
```

Introduction to Deep Learning Paradigm

Neural networks recently have become a very popular topic of research in the field for natural language processing, including sentiment analysis. Neural networks are proving useful in solving almost any machine learning classification problem. The only adjustment required is defining its architecture — number of hidden layers to be used, number of hidden units to be present in each layer, activation function for each node, error threshold for the data, the type of inter-connections, etc.

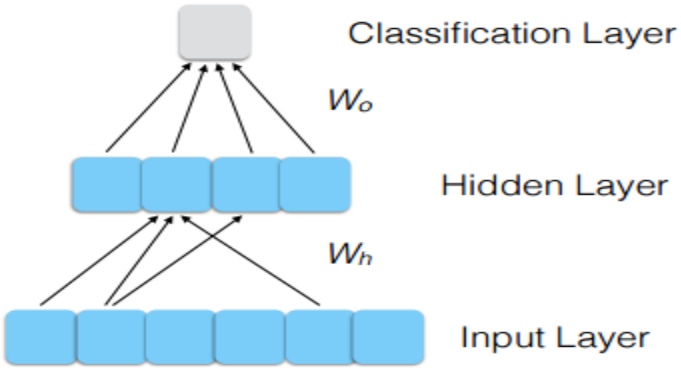
Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using model architectures, with complex structures or otherwise, composed of multiple nonlinear transformations. – defined by Deng and Yu (Deng and Yu, 2014)

Clearly, a traditional machine learning algorithm can be designed using deep learning but not necessarily vice-versa. This is because neural networks are capable of capturing very complex characteristics of data without any significant involvement of manual labor as opposed to the machine learning systems. Deep learning uses deep neural networks to learn good representations of the input data, which can then be used to perform specific tasks.

MULTILAYER PERCEPTRON

The architecture I am going to present using Python's Theano as backend for keras deep learning libraries is the single-hidden-layer Multi-Layer Perceptron (MLP). The most basic form of such a network is the multilayer perceptron ("MLP"), which consists of one or more fully-connected hidden layers. An example of how such networks are often visualized can be seen in Figure. In below figure, each of the "cells" (no pun intended) represents a single numerical value (though often, as here, the size of these vectors shown in figures is significantly reduced from the size actually used, for illustrative purposes). Here the input dimensionality is six and the size of the single hidden layer (i.e., the output dimensionality of Equation $y = f(Wx + b)$) is four. The arrows represent "connections" between cells realized by the weight matrix multiplication. Sometimes, for clarity of expression, not all of the connections are shown, as is the case between the input and hidden layer here. Unless otherwise specified, all connections between layers should be assumed to be full connections, meaning the weight matrix is not restricted to be sparse and each input value affects each output value. In more complex diagrams, such full connections are often represented by a single arrow.

An MLP (or Artificial Neural Network - ANN) with a single hidden layer can be represented graphically as follows:



A basic representation of a multilayer perceptron classifier with one hidden layer.

CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNN) are biologically-inspired variants of MLPs. From Hubel and Wiesel's early work on the cat's visual cortex [Hubel68], we know the visual cortex contains a complex arrangement of cells. These cells are sensitive to small sub-regions of the visual field, called a receptive field. The sub-regions are tiled to cover the entire visual field. Convolutional neural networks were designed to honor the spatial structure in image data whilst being robust to the position and orientation of learned objects in the scene. This same principle can be used on sequences, such as the one-dimensional sequence of words in a movie review.

There are three types of layers in a Convolutional Neural Network:

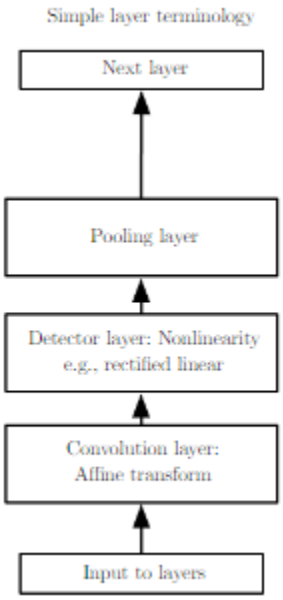
1. Convolutional Layers.

2. Pooling Layers.

3. Fully-Connected Layers.

Convolutional layers are comprised of filters and feature maps. The filters are essentially the neurons of the layer. They have both weighted inputs and generate an output value like a neuron. The feature map is the output of one filter applied to the previous layer. The pooling layers down-sample the previous layers feature map. Pooling layers follow a sequence of one or more convolutional layers and are intended to consolidate the features learned and expressed in the previous layers feature map. Fully connected layers are the normal at feedforward neural network layer. These layers may have a nonlinear activation function or a softmax activation in order to output probabilities of class predictions.

The components of a typical convolutional neural network layer



Word Embeddings

Problems in natural language processing often lend themselves naturally to using discrete features, such as words or n-grams of words. In a machine learning context, such features are often represented by binary-valued (or sometimes count-valued) vectors of very high dimensionality. Features of this type are

generally unsuitable as inputs to neural networks, however, because their sparsity makes learning intractable. Moreover, a well-chosen relatively low-dimensional continuous representation of such discrete units could have the additional advantage of encoding all types on interesting relationships between (such as different notions of similarity). This could be useful in particular as neural network inputs, since the highly non-linear function represented by the network could learn to exploit that geometry in the way best suited for the task at hand. Because of these advantages, such representations for words have been explored for years and are known colloquially as “word embeddings.

Keras provides a convenient way to convert positive integer representations of words into a word embedding by an Embedding layer and Keras utility to truncate or pad the dataset for each observation using the sequence.pad sequences() function.

What is an LSTM neural network?

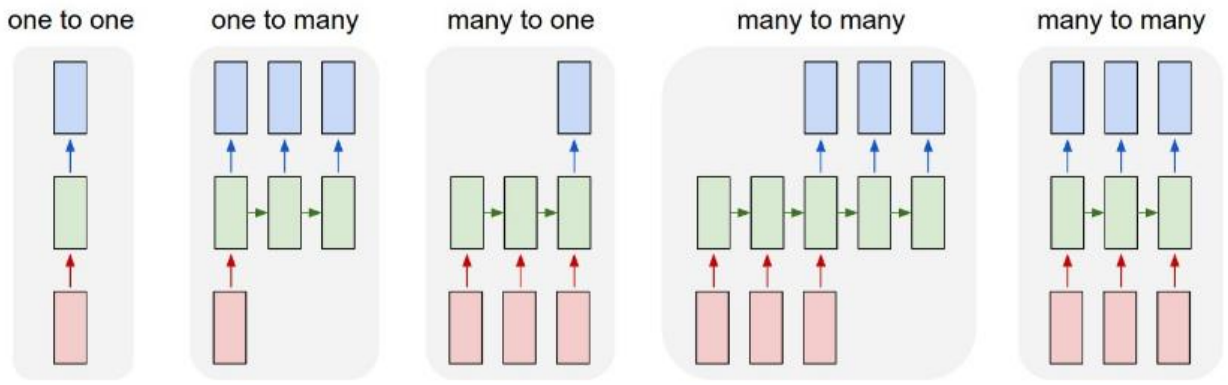
A Long Term Short Term Memory Network is a type of Recurrent Neural Network. RNN's have multiple time steps with a feature vector input at each time step and the prior layers output/hidden state as an input vector. RNN's come in many varieties, for example, an RNN can take one input and predict multiple outputs, multiple inputs and multiple outputs, multiple inputs to a single output dimension, ext.

Long Short Term Memory networks (Hochreiter and Schmidhuber, 1997) are a modified version of the recurrent neural networks but with much more complicated activation units. The key element of an LSTM model is a memory cell. Here, information is stored in two ways (hence the name Long Short Term Memory):

Short-term Memory as activations of the neurons which capture the recent history,

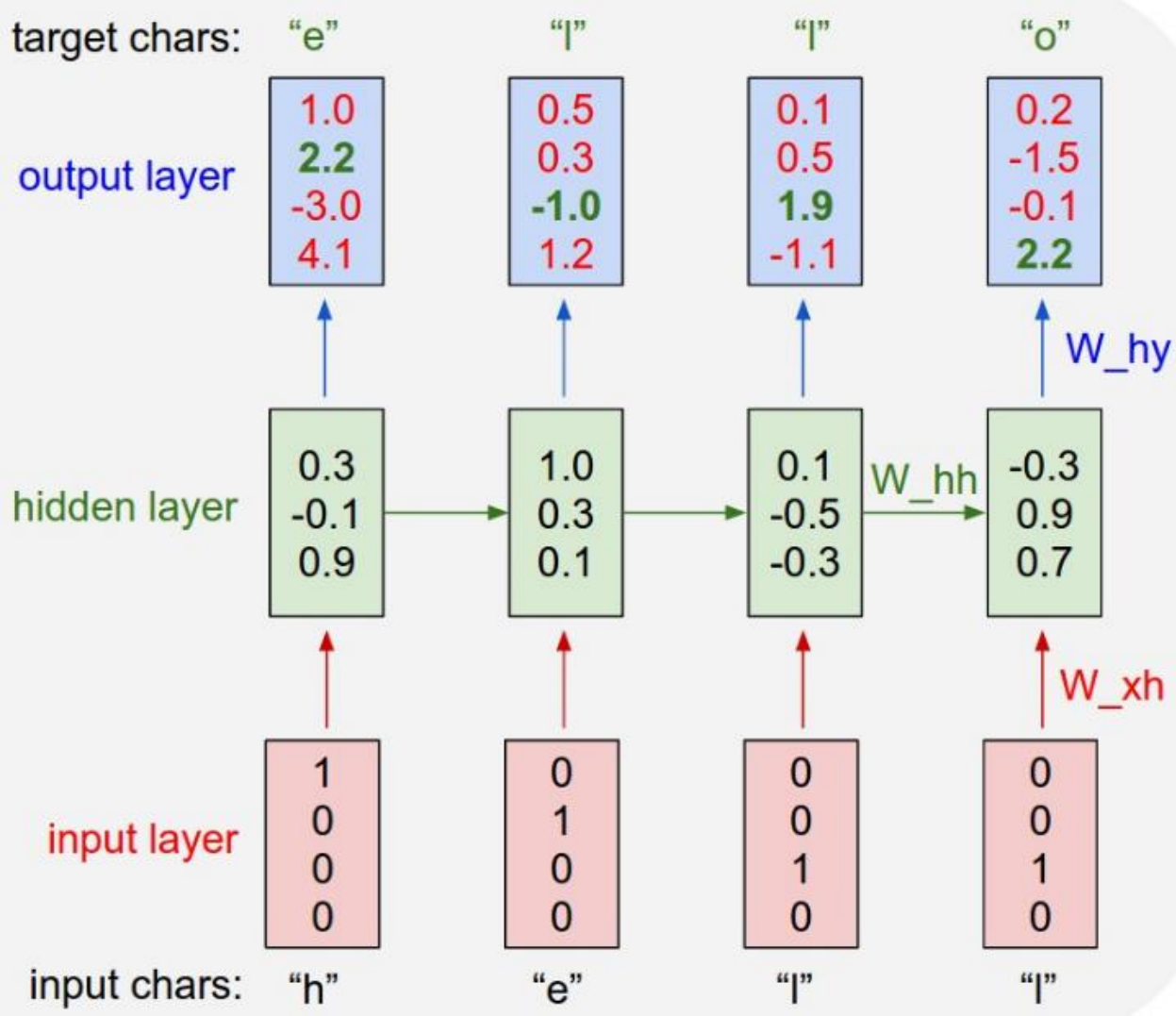
Long-term Memory as weights which are modified based on back propagation

This model allows retention of information over a much longer period (more than the usual 10-12 steps as in case of RNNs) through the use of the memory cell and hence produces appreciable results when applied to NLP tasks.



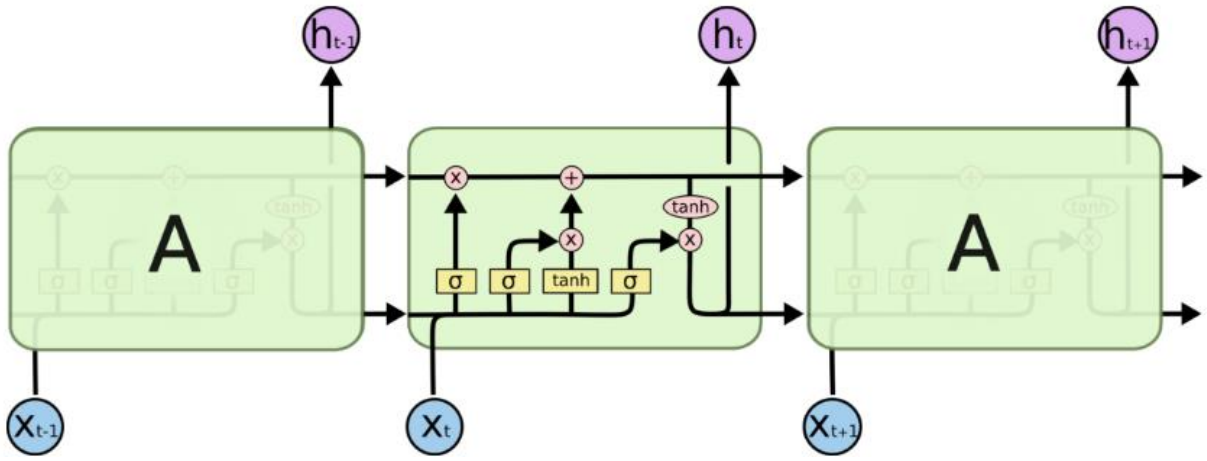
Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

A more specific RNN where one-hot vector of English characters is the input and at character space in the text an input is provided and an output is generated for the predicted next character in the text.



So, how is an RNN different from an LSTM? A normal RNN has the prior output concatenated with the current input to form the feature vector for the current layer. In a basic RNN structure long term dependencies (i.e.: the subject of a sentence) can be hard for the network to remember over many time steps. This is referred to as the vanishing/exploding gradient problem. More reading is suggested on this, but essentially the non-linearity's effect will compound over time step's causing the prior (historic) gradient to approach either zero or infinity.

We prevent this problem through the LSTM architecture. LSTM's have an input gate, output gate and forget gate. As simply as possible, this gates control what the output is for the current layer, what part of the past hidden state is forgotten based on the current input, and what part of the current input should be added to the networks long term memory vector which is passed from layer to layer along with the hidden state.



Source http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Related work

Sentiment analysis is a long standing research topic. Referring to [Liu, 2012] for a recent survey. Sentiment classification is one of the key tasks in sentiment analysis and can be roughly categorized as document level, sentence level and aspect level. This project work falls into the sentence level. In the next we review topics closely related to this project work

Deep Learning for Sentiment Classification In recent years, deep learning has received more and more attention in the sentiment analysis community. Researchers have explored different deep models for sentiment classification. Glorot et al. used stacked denoising auto-encoder to train review representation in an unsupervised fashion, in order to address the domain adaptation problem of sentiment classification [Glorot et al., 2011]. Socher et al. [Socher et al., 2011; 2012; 2013] proposed a series of Recursive Neural Network (RecNN) models for sentiment classification. These methods learn vector representations of variable-length sentences through compositional computation recursively. Kim investigated using CNN for sentence sentiment classification and found it outperformed RecNN [Kim, 2014]. A variant CNN with dynamic k-max pooling and multiple convolutional layers was proposed in [Kalchbrenner et al., 2014]. Researchers have also investigated using sequential models such as Recurrent Neural Network (RNN) and Long ShortTerm Memory (LSTM) for sentiment classification [Tang et al., 2015].

Results

Performance of Models on Stanford Sentiment Large Movie Reviews (IMDB) Dataset

The sentiment movie review dataset introduced by Stanford (ACL paper et al., 2012) contains 25,000 highly polar movie reviews for training, and 25,000 for testing. These are annotated with sentiment scores ranging between 0 and 1 where 0 means most negative and 1 means very positive. Experiments are performed on this dataset (using the standard train/development/test splits) for both binary classification and sequence classification:

Prediction of sentiment for movie reviews using **Multi-Layer Perceptron** and **Convolutional Neural Networks** results

Model Network	Optimizer	batch_size	Input_length	nb_epoch	Accuracy
Multi-Layer Perceptron	adam	128	500	10	86.84%
Multi-Layer Perceptron	adam	128	500	5	86.69%
Multi-Layer Perceptron	adam	128	1000	5	86.78%
Multi-Layer Perceptron	rmsprop	128	500	5	84.26%
Multi-Layer Perceptron	rmsprop	128	1000	5	85.46%
Convolutional Neural Networks	adam	128	500	5	87.49%
Convolutional Neural Networks	adam	128	1000	5	87.07%
Convolutional Neural Networks	rmsprop	128	500	5	88.52%
Convolutional Neural Networks	rmsprop	128	1000	5	88.40%

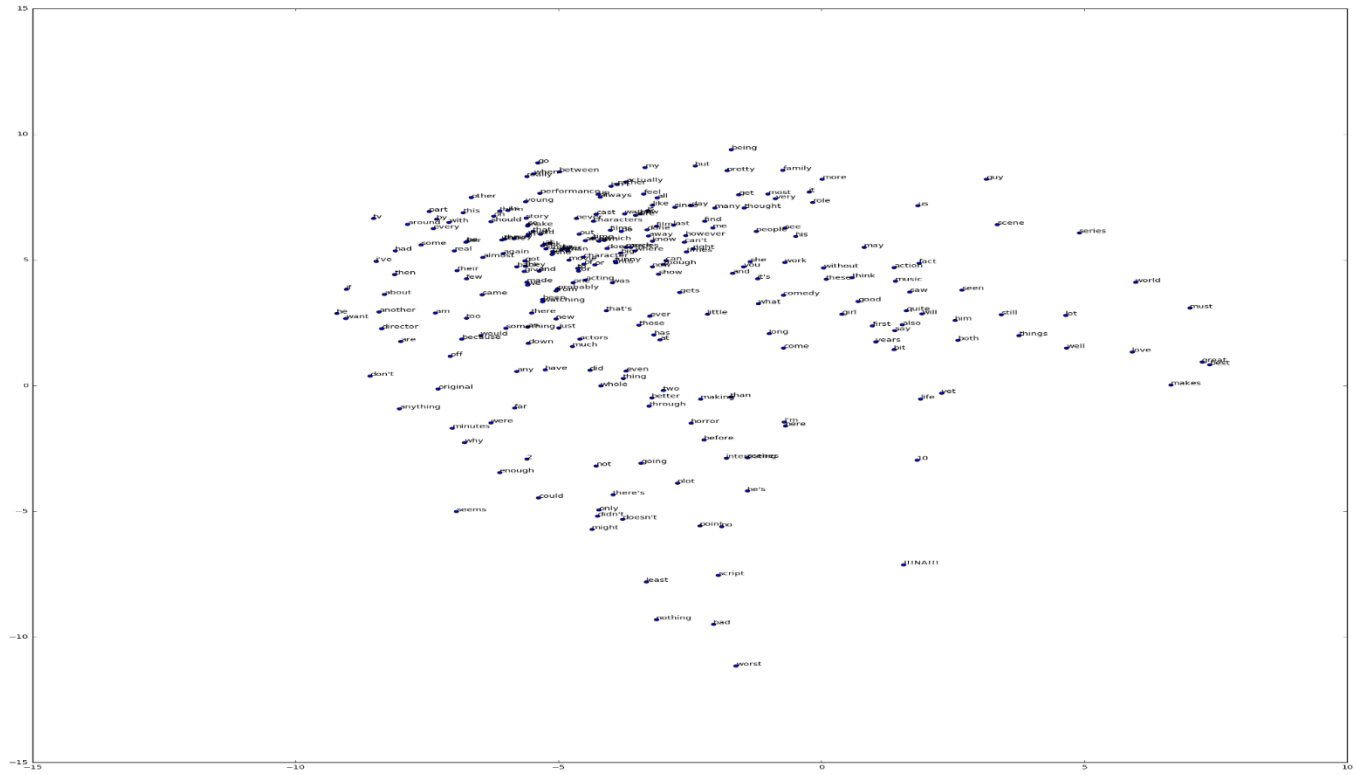
Table 1: Results (Accuracy values in %) of different neural network deep learning approaches for sentiment classification on Stanford Sentiment movie review dataset

Sequence Classification of movie reviews using **LSTM-RNN Network** results

LSTM-RNN Network	Optimizer	nb_epoch	embedding_neurons	lstm_neurons	batch_size	Avg_sec_per_epoch	Accuracy
Forward-Pass	adam	6	128	64	32	368.793916345	86%
Bi-directional	adam	6	128	64	32	644.418573181	86.40 %
Bi-directional	rmsprop	6	128	64	32	620.017516812	87.70 %

Table 2: Results (Accuracy values in %) of different LSTM recurrent neural networks approaches for sentiment classification on Stanford Sentiment movie review dataset

We can notice that great, most, well are clustered, bad don't even are clustered. We've learned structure in our sentiment embedding and neural networks give us this and other useful features for free



Weights from embedding layer and visualization

Note in the image above that 'script' is close to 'bad', apparently more people use that in the negative context. Also, 'performance' is close to 'him', it appears that the embedding is learning a reference back to the pronoun. Note, how far apart the clusters of positive and negative words are, negative words at the top of the TSNE embedding. Another cool observation, 'music' is close to 'feel', again think that aspect of this embedding is cinema specific.

We can notice that great, most, well are clustered, bad don't even are clustered. We've learned structure in our sentiment embedding and neural networks give us this and other useful features for free

For Multilayer Perceptron Model with a single hidden layer. The word embedding representation is a true innovation. Selecting only the top 5,000 words and 50%/50% split of the dataset into training and test. Embedding layer as the input layer, setting the vocabulary to 5,000, the word vector size to 32 dimensions and the input length to 500.The output of this first layer will be a 32 x 500 sized matrix flatten the Embedding layers output to one dimension, then use one dense hidden layer of 250 units with a rectifier activation function. The output layer has one neuron and will use a sigmoid activation to output values of 0 and 1 as predictions. The model uses logarithmic loss and is optimized using the efficient ADAM and rmsprop optimization procedures. This model overfits very quickly so I have used very few training epochs, these models for this particular dataset easily attains high accuracy on the training set while failing to generalize to the test set. Optimal performance was obtained after 5 training epochs for Multi-Layer Perceptron and Convolutional Neural Networks whereas for LSTM-RNN Network epochs are just 6. There is a lot of data so I used a batch size of 128.

For Convolutional Neural Network Keras supports one dimensional convolutions and pooling by the Convolution1D and MaxPooling1D classes respectively. Here after the Embedding input layer we insert a Convolution1D layer. This convolutional layer has 32 feature maps and reads embedded word

representations 3 vector elements of the word embedding at a time. The convolutional layer is followed by a MaxPooling1D layer with a length and stride of 2 that halves the size of the feature maps from the convolutional layer

Conclusion

The classification performance of the different models on the IMDB movie review dataset gives a rough idea of the utility of the different models in different scenarios for sentiment classification.

The main contributions of the project are: predict the sentiment of movie reviews as either positive or negative and predict a category for the sequence in Python using the Keras deep learning library. For this dataset, the performance of the deep learning models, especially CNN and LSTM variants, is interesting. Convolution Neural Networks give the best accuracy for binary sentiment classification on IMDB movie review dataset. I have this question after seeing the LSTM results, what went wrong for LSTM RNN? Why does this complex neural net not do leaps and bounds better than CNN? Neural networks tend to excel with lots of data. If I had around a million movie reviews the LSTM would done better than Multi-Layer Perceptron and Convolutional Neural Networks

Future Work

In the future, since LSTM generally have the problem of overfitting, adding dropout applied between layers using the Dropout Keras layer we can see desired impact on training with a slightly slower trend in convergence and probably use a few more epochs of training. I would expect that even better results could be achieved if this problem is further extended to use dropout. I am hope to implement a recursive neural network on top of the CNN and Forward Pass LSTM Network multi step prediction for classification of movie reviews, which I am anticipating that will improve the performance of the overall model.

References

[1] https://keras.io/ https://keras.io/datasets/

[2] Pang, Bo, and Lillian Lee. ”Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales.” Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2005.

[3] Pang, Bo, and Lillian Lee. ”A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts.” In Proceedings of the 42nd annual meeting on Association for Computational Linguistics, p. 271. Association for Computational Linguistics, 2004.

[4] Pang, Bo, and Lillian Lee. ”Opinion mining and sentiment analysis.” Foundations and trends in information retrieval 2, no. 1-2 (2008): 1-135.

[5] Learning Word Vectors for Sentiment Analysis. Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher

[6] http://deeplearning.net/tutorial/deeplearning.pdf

[7] https://machinelearningmastery.com/deep-learning-with-python/

[8] http://ai.stanford.edu/~amaas/data/sentiment/