Modeling Approaches to Sentiment Analysis on the IMDB Movielens Dataset

Karthik Venkata, Sarvesh Rajkumar, Shamini Hilda

1 Summary

Our objective is to obtain an understanding of how Deep Learning is used within the field of Natural Language Processing using the IMDB Movie Review Dataset to detect the sentiment of written text. We used the paper(Learning word vectors for Sentiment Analysis, Mass et al.) to obtain a broader understanding of the dataset and the wide variety of techniques incorporated. We primarily focus on making use of Python packages like Gensim, NLTK and implementing Deep Nets using the PyTorch framework(https://pytorch.org/). The project work flow is divided into three phases - contrasting:
- Bag of Words Models
- Incorporating Temporal and Contextual information using LSTM (RNN)
- RNN Language Models
and see how each of these fare in predicting sentiment with the appropriate hyperparameter optimization.

2 Project Workflow

2.1 Team Name:

The Senti-Mentalists

2.2 Team Members:

1) Karthik Venkata
2) Sarvesh Rajkumar
3) Shamini Hilda

2.3 Data

The dataset we have selected for our final project is the "IMDB Movie Review Dataset". The dataset can be accessed using the following link: http://ai.stanford.edu/ amaas/data/sentiment/. This data set contains both labeled and unlabeled movie reviews. The entire labelled reviews data is divided into 25,000 training set and 25,000 testing set. The 50,000 unlabeled reviews set is used to evaluate the RNN Language Model Approach which does not need labeled data. The training set and testing set are both balanced overall with equal number of positive and negative reviews. To avoid redundancy in the entire data set collection, no more than 30 reviews per movie are gathered. In order to avoid over training the model, both training and testing sets are made disjoint and hence don't contain reviews for the same movie.

The IMDB reviews are rated positive or negative based on the scores assigned to them, thus positive reviews have a score of above 7 out of 10 and negative reviews have a score of less than 4 out of 10. The neutral scores (5 out of 10) are not included both in the training and the testing set. In the unlabeled set, all ratings including the neutral scores are included. Regarding the file level information about the dataset, there are two top-level directories called testing and training set.

Within each directory, there are positive reviews, negative reviews and a separate IMDB URLs directory. The URL directory contains the IMDB URLs which link directly to the movies' review page. Inside the training set directory, there is a separate directory for unlabeled review set. In addition, there are also '.feat' files which contain already-tokenized bag of words in a LIBVSM format (ASCII sparse-vector format).

2.4 Preprocessing

Standard pre-processing techniques need to be applied for IMDB text data. We need to tokenize the dataset (split long strings up into individual words/symbols). In order to ascertain the vocabulary/model size, the idea is to determine the number of unique tokens. To optimize computational time, we want to make sure that we don't constantly convert strings to token IDs. Instead, if we store everything as token IDs directly, this would result in the 1-hot representation necessary for input into out model. We extensively used the NLTK (Natural Language Toolkit) python package not only for tokenization but for other aspects of the pre-processing as well. Converting to lowercase, stop words removal and organizing the tokens by frequency which facilitates choosing vocabulary size are other techniques worth looking into. At this point, began visualizing the data and carrying out an extensive Exploratory Data Analysis (EDA) which aided us in better understanding the dataset. We explored the functionality of the GloVe features and used the vast array of word embeddings to see how this could make our preprocessing more robust.

It's easier to re-work the data before training any models for multiple reasons:
- It's faster to combine everything into fewer files.
- We need to tokenize the dataset (split long strings up into individual words/symbols).
- We need to know how many unique tokens there are to decide on our vocabulary/model size.
- We don't want to constantly be converting strings to token IDs within the training loop (takes non-negligible amount of time).
- Storing everything as token IDs directly leads to the 1-hot representation necessary for input into our model.

The mean review contains 267 tokens with a standard deviation of 200. In spite of there being over 20 million total tokens, they are obviously not all unique. We now want to build our dictionary/vocabulary.

For a defined document unit and a character sequence, tokenization is the process of splitting it up into pieces, known as tokens. Punctuation is also taken care of while tokenizing the data. A token, is essentially an instance of a sequence of characters in a document which are grouped together as a useful semantic unit for processing.

What needs to be done at this point is the creation of an embedding layer, that is basically a dictionary mapping integer indices (that represent words) to dense vectors. Integers are taken as an input, which are looked up in an internal dictionary, following which the associated vectors are returned. This is a technique where words are encoded as real-valued vectors in a high dimensional space, where the similarity between words in terms of meaning translates to closeness in the vector space.

A word embedding matrix does not give a clear indication of how to handle the sequence information. In this case, one can picture the bag of words as just taking the average of all the 1-hot vectors for the entire sequence and then multiplying this by the word embedding matrix to obtain a document embedding. It is a common practice to use GloVe pre-trained vectors as inputs for neural networks in order to perform NLP tasks in PyTorch. GloVe is an unsupervised learning algorithm to obtain vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations depict fascinating linear substructures of the word vector space. Instead of training our own word vectors from scratch, leveraged GloVe. Its authors have released four text files with word vectors trained on various massive web datasets.

The GloVe features provide a text file with 300 dimensional word embedding for over 2 million tokens which is basically a dictionary sorted by word frequency. GloVe is essentially a log-bilinear model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning.

2.5 Data Visualization

After pre-processing the data, we tried visualizing both positive and negative reviews datasets to have an idea about the frequently repeated words, verbosity of the reviews based on text length of each review and predominant words in the dataset by creating word clouds.

The histogram below is a representation of both positive and negative reviews binned based on the text length to understand the verbosity of reviews.

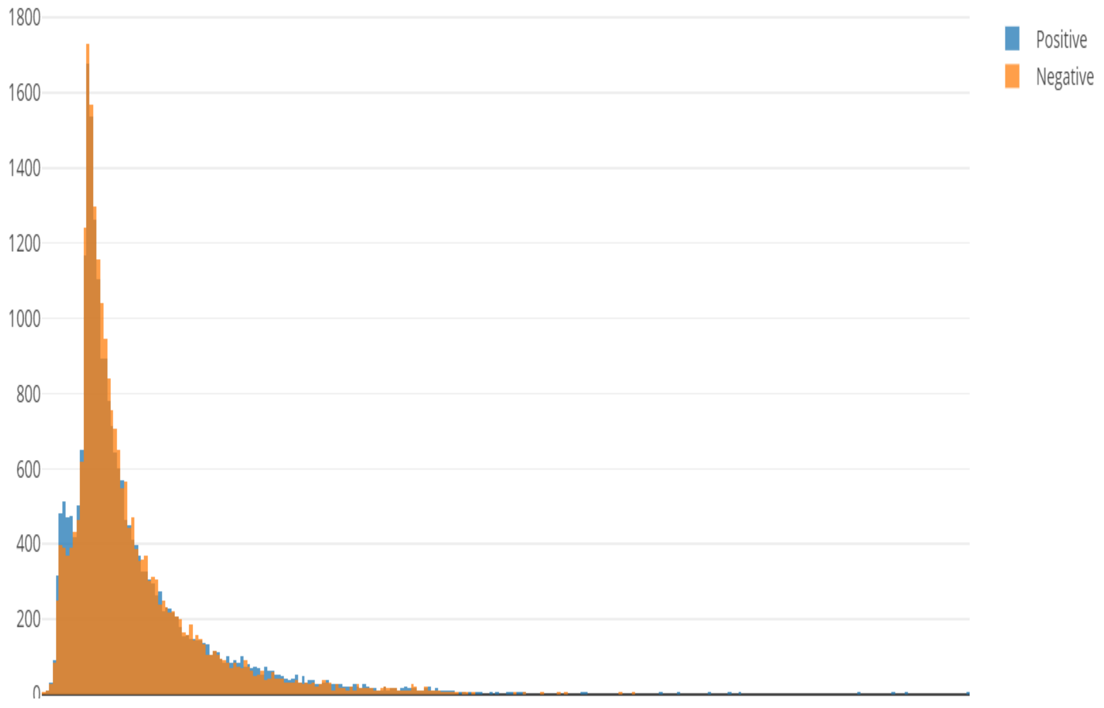


Figure 1: Histogram representing text length of the reviews

The boxplot below helps us to understand the verbosity of positive and negative reviews better. The outliers in the positive dataset boxplot show that few positive reviews are highly verbose (more than 10,000 words).

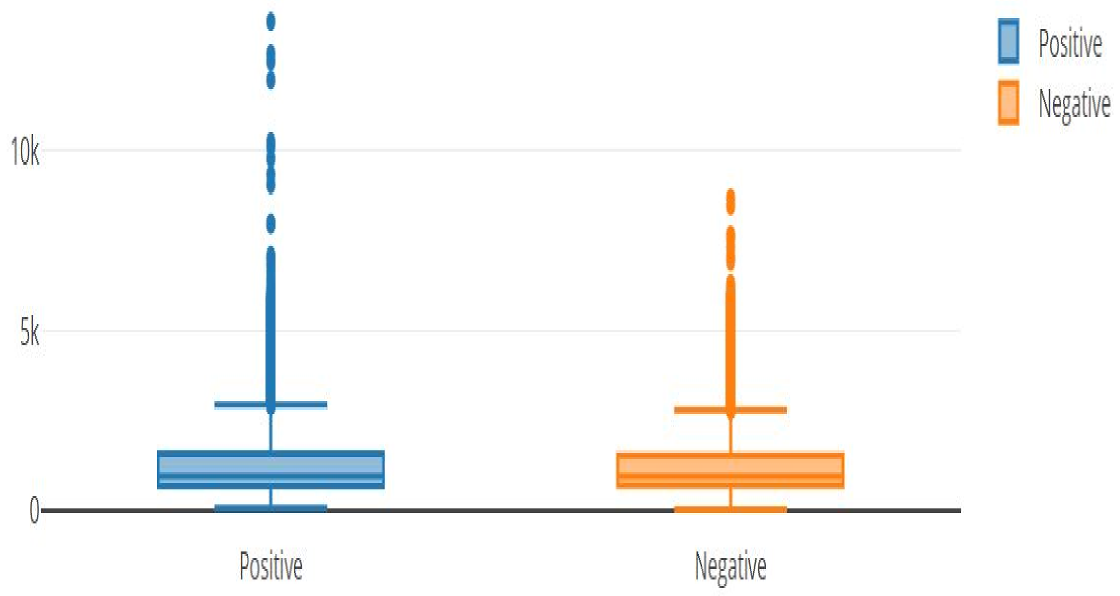


Figure 2: Boxplot representing text length of the reviews

In the below visualization, predominant words in the word cloud are the ones mostly repeated in the entire dataset (both positive and negative reviews dataset).The size of each word represents its frequency or importance in the dataset.

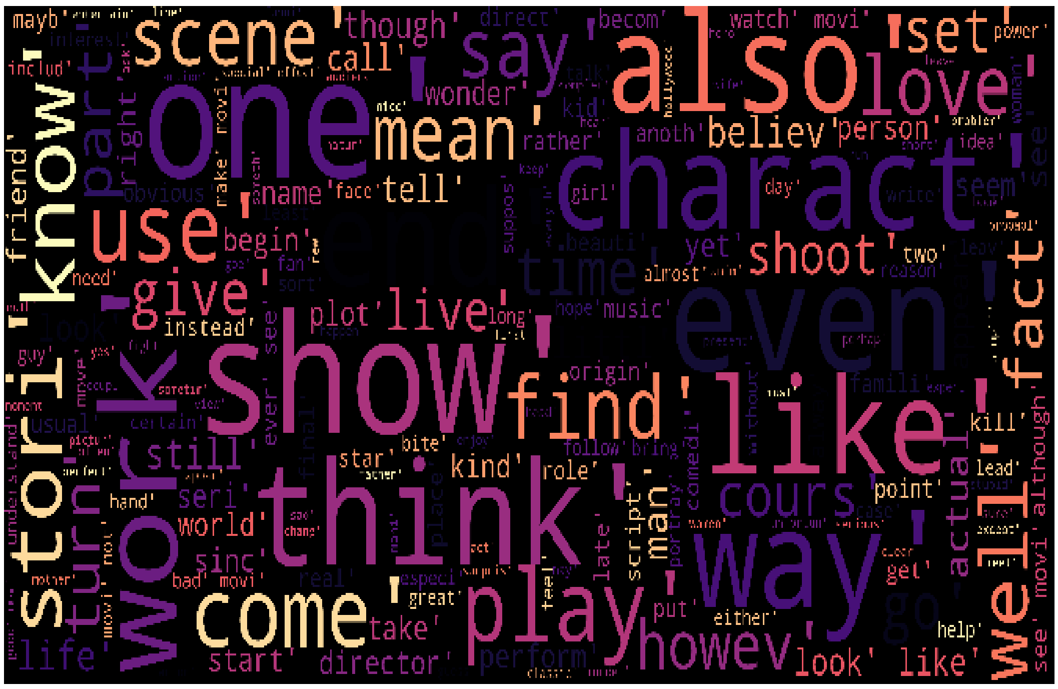


Figure 3: Wordcloud for entire dataset

The below word cloud represents the most frequent words in the positive reviews dataset.

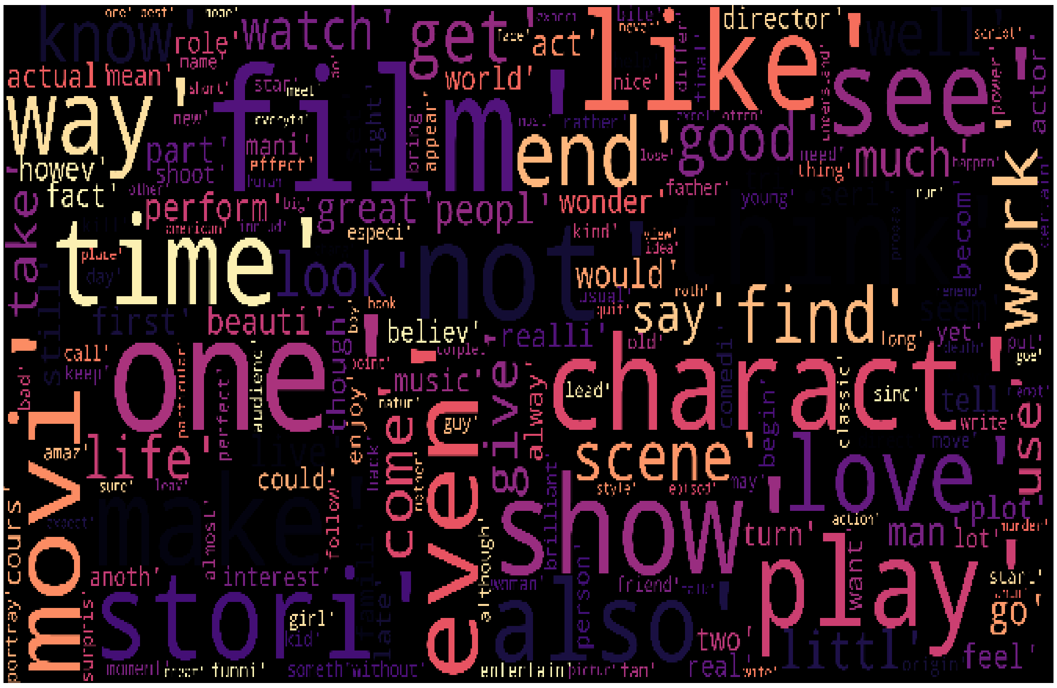


Figure 4: Wordcloud for positive reviews dataset

The below word cloud represents the most frequent words in the negative reviews dataset.

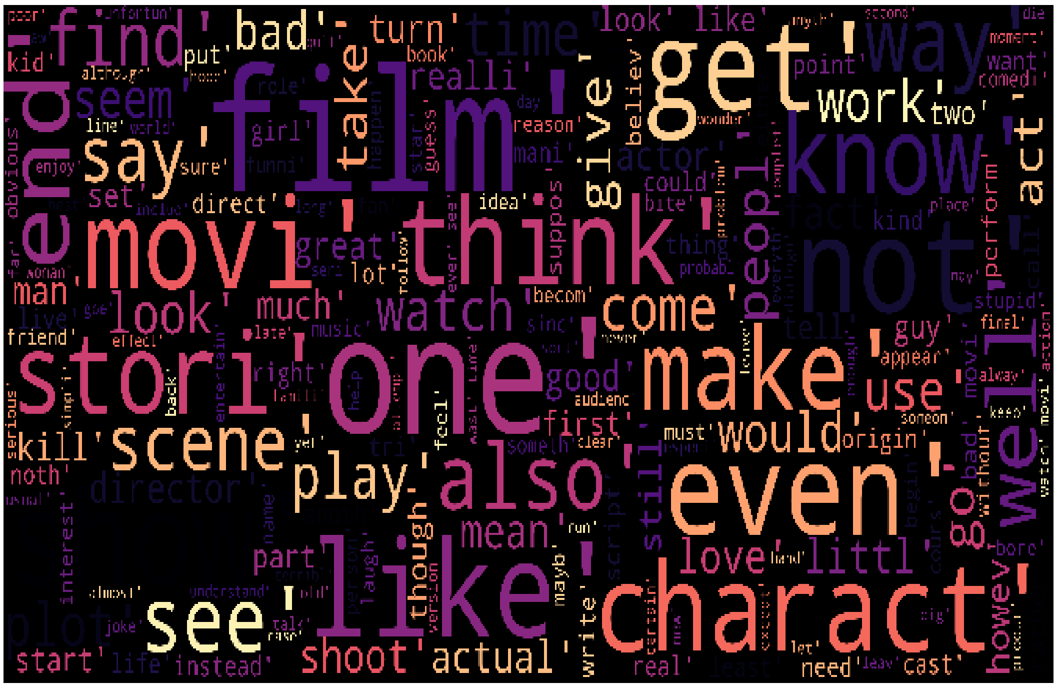
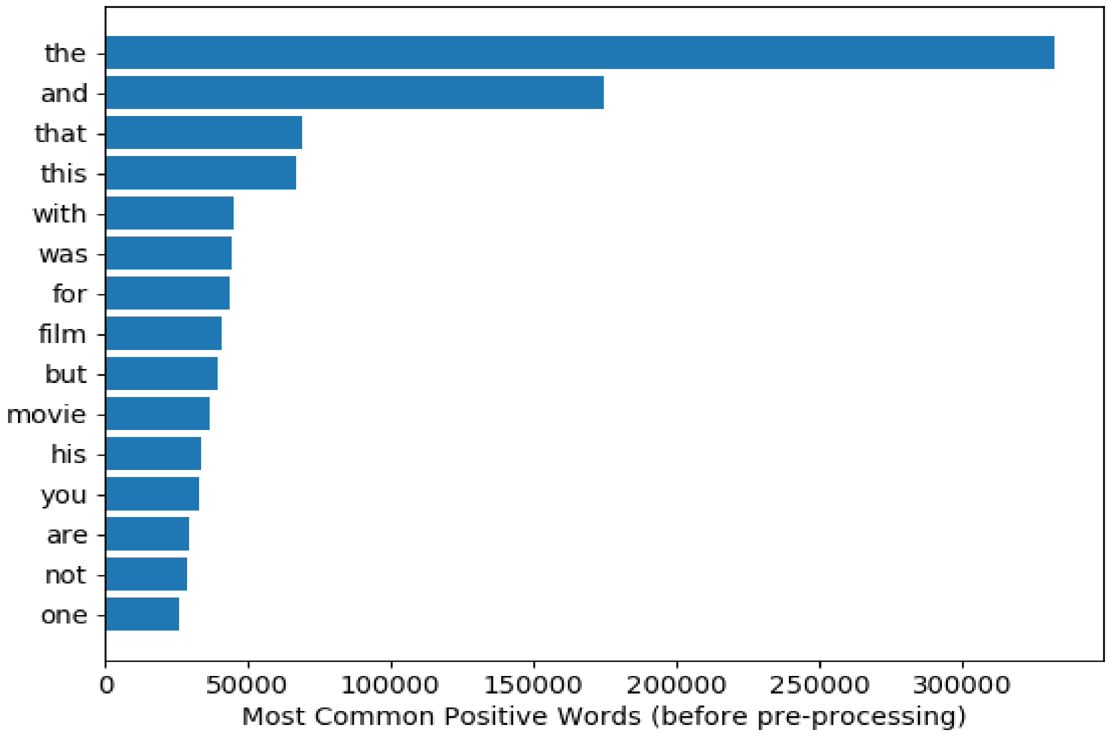


Figure 5: Wordcloud for negative reviews dataset

The bar charts below clearly explain the frequent positive words before and after pre-processing. The most common words before pre-processing are clearly the stop words. After pre-processing the dataset, we can see good improvement in the results. Some most commonly repeated positive words are good, great, love etc.



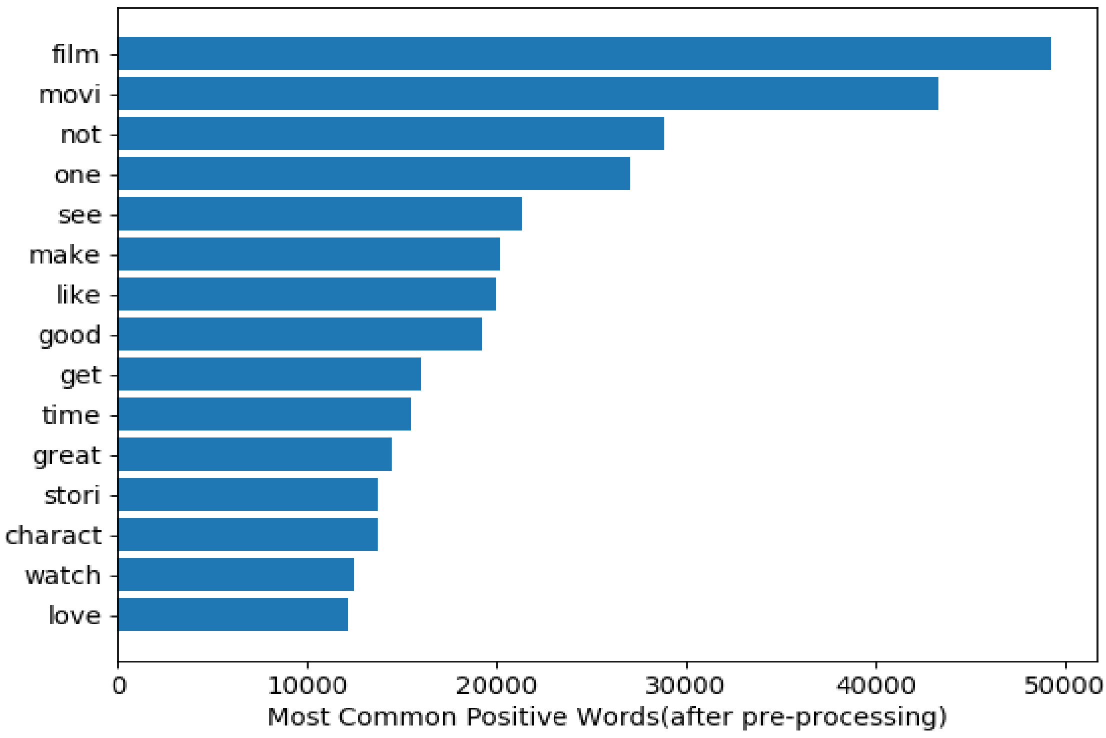
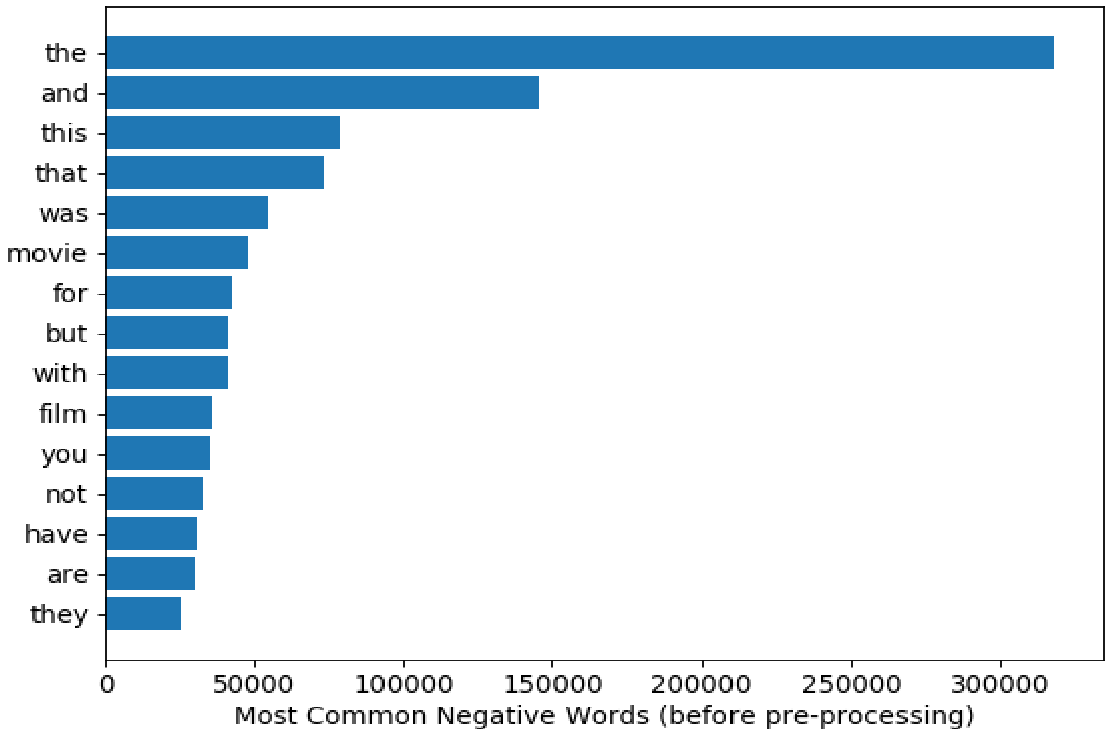


Figure 6: Barcharts depicting most common positive words before and after pre-processing.

The bar charts below explains frequent negative words before and after pre-processing. The most common words before pre-processing are clearly the stop words. After pre-processing we can see most frequent negative words like not, bad, etc.



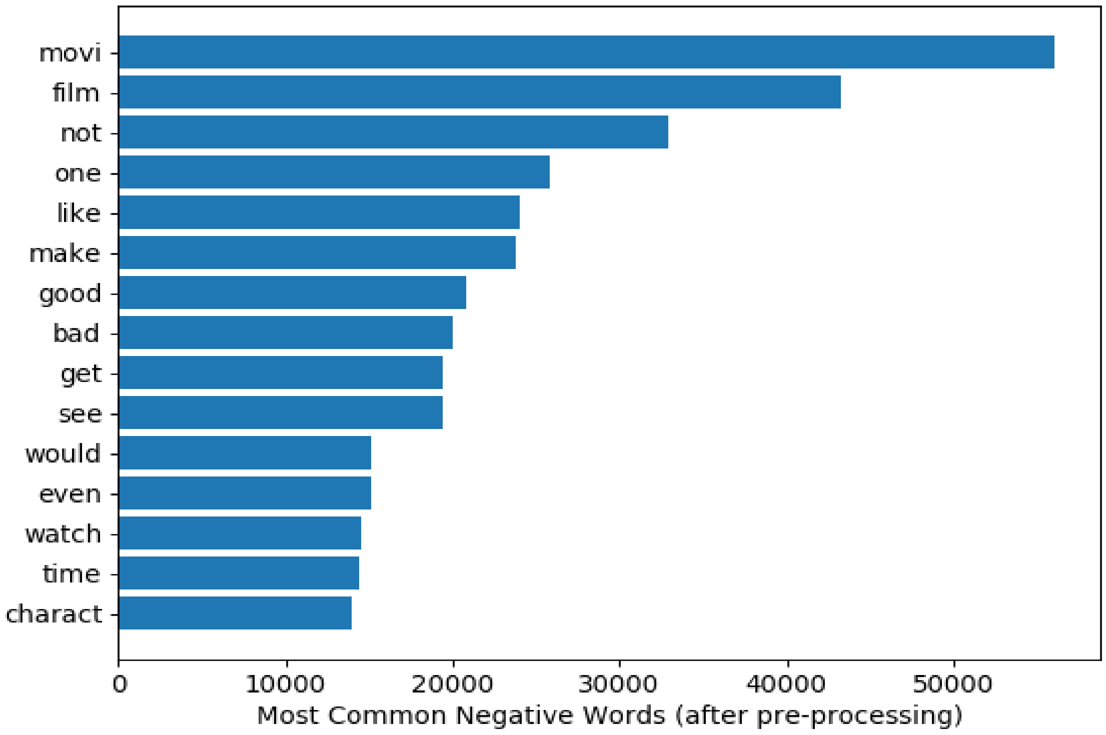


Figure 7: Barcharts depicting most common negative words before and after pre-processing.

2.6 Recurrent Neural Networks

Consider the following two reviews: * "Even though the movie had an amazing sound track, I thought it was awful." * "Even though the movie had a terrible sound track, I thought it was amazing."

The first review evidently has an overall negative sentiment while the bottom review clearly has an overall positive sentiment. Had we been using the bag of words approach, both sentences would result in exactly the same output. Evidently, there is a lot of useful information which may have been utilized more effectively if we didn't discard the sequence information like we did earlier. By designing a model capable of capturing this additional source of information, we can potentially achieve better results but also greatly increase the risk of overfitting. This is primarily related to the curse of dimensionality.

A recurrent neural network can be used to maintain the temporal information and process the data as an actual sequence. Choosing the sequence length is an important aspect of training RNNs and can have a significant impact on the results. By using long sequences, we provide the model with more information. A RNN process sequences by iterating through the sequence elements and maintaining a state containing information relative to what it has seen so far. The simple RNN layer should theoretically be able to retain at time t information about inputs seen many timesteps before. But in practice, such long-term dependencies are very hard to learn. The LSTM (long-short term memory) layer was designed to address this issue. It allows past information to be reinjected at a later time.

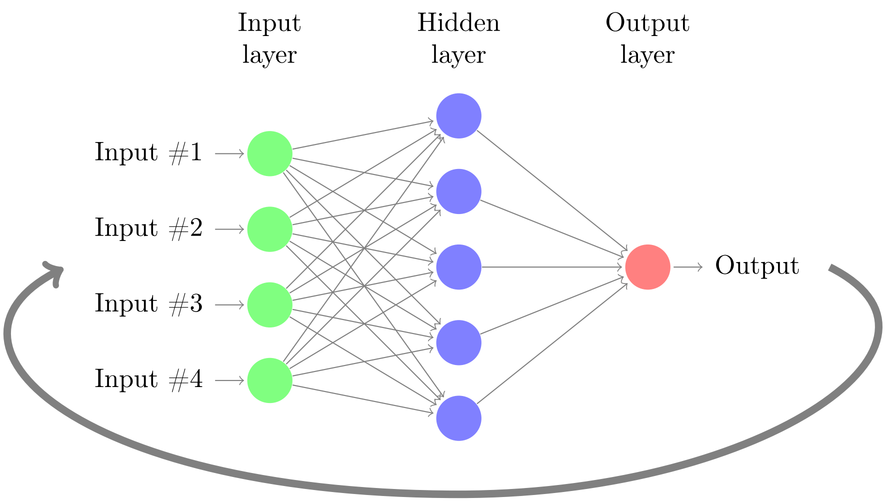


Figure 8: RNN Architecture (Source: Github)

The RNN dynamics can be described using deterministic transitions from previous to current hidden states. The deterministic state transition is a function

$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \to h_t^{l}$$

For classical RNNs, this function is given by

$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^{l}), \text{ where } f \in \{\text{sigm}, \tanh\}$$

2.7 LSTM

In a traditional recurrent neural network, during the gradient back-propagation phase, the gradient signal can end up being multiplied a large number of times (as many as the number of timesteps) by the weight matrix associated with the connections between the neurons of the recurrent hidden layer. This means that, the magnitude of weights in the transition matrix can have a strong impact on the learning process.

If the weights in this matrix are small (or, more formally, if the leading eigenvalue of the weight matrix is smaller than 1.0), it can lead to a situation called vanishing gradients where the gradient signal gets so small that learning either becomes very slow or stops working altogether. These issues are the main motivation behind the LSTM model which introduces a new structure called a memory cell.

The LSTM has complicated dynamics that allow it to easily "memorize" information for an extended number of timesteps. The "long term" memory is stored in a vector of *memory cells* $c_t^l \in \mathbb{R}^n$. Although many LSTM architectures differ in their connectivity structure and activation functions, all LSTM architectures have explicit memory cells for storing information for long periods of time. The LSTM can decide to overwrite the memory cell, retrieve it, or keep it for the next time step. The LSTM architecture used in our experiments is given by the following equations:

$$\begin{aligned}
&\text{LSTM} : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \to h_t^l, c_t^l \\
&\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \\
&c_t^l = f \odot c_{t-1}^l + i \odot g \\
&h_t^l = o \odot \tanh(c_t^l)
\end{aligned}$$

In these equations, sigm and tanh are applied element-wise. The figure below illustrates the LSTM equations:

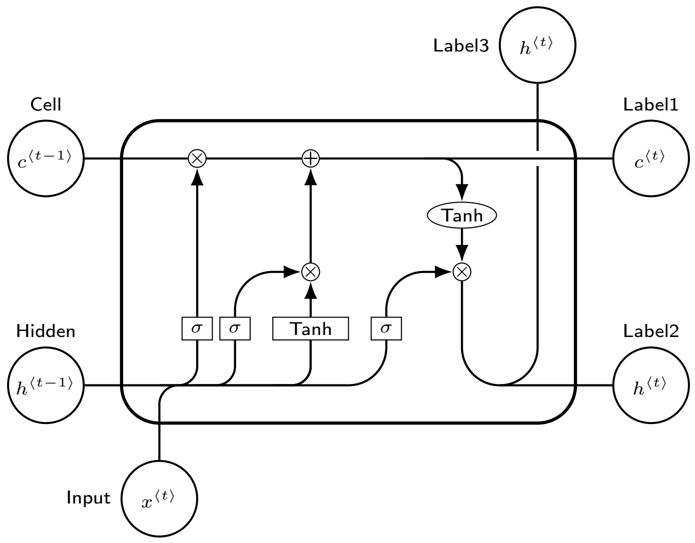


Figure 9: LSTM Architecture (Source: Github)

2.8 Model Selection

2.8.1 Bag of Words Model

Being one of the most basic models for text classification, a Bag of Words Model handles varying length inputs to get a single valued output.

A word embedding matrix doesn't handle the sequence information. We can picture the bag of words as simply taking the mean of all of the 1-hot vectors for the entire sequence and then multiplying this by the word embedding matrix to get a document embedding. Alternatively, we can take the mean of all of the word embeddings for each token and get the same document embedding. Now we are left with a single d dimensional input representing the entire sequence.
It needs to be noted here that a Bag of Words Model contains all the information in the review but

no sequential/contextual information is captured, which constitutes a large portion of the meaning representation of the review.

All we need to define our model is batch size and number of hidden units.
We used the following loss functions and checked performance:
- BCEWithLogitsLoss
- CrossEntropyLoss

Some hyperparameters optimized include:
- Number of hidden Units
- Number of hidden layers
- Dropout

Some other factors for which an effect was observed include:
- Batch Size
- Dynamic Learning Rate
- Training Time
- Reducing Dictionary Size
- SGD With Momentum Vs Adam
- Seeing if initialization with pretrained GloVe embeddings work

Bag of Words Model Results
We tried initializing the weights with and without GloVe pretrained embeddings to see if it would improve the model.

1) Accuracy without GloVe embeddings: 86-88%
2) Accuracy with GloVe Embeddings: 81-87%

Against the intuition , the second model actually seems to perform worse on average than the first one.
In part 1, test accuracy typically seems to achieve its maximum value after the 3rd epoch and begins to decrease with more training while the training accuracy continues to increase well into 90%. This is a sure sign of overfitting. The training accuracy for part 2 stops much earlier (around 86-88%) and doesn't seem to improve much more.
Nearly 95% of the weights belong to the embedding layer in part 1. We are training significantly less in part 2 and can't actually fine-tune the word embeddings at all. Using only 300 hidden weights for part 2 results in very little overfitting while still achieving decent accuracy.

2.8.2 RNN Model

Having discussed the use case of RNNs in detail previously, we create a StatefulLSTM cell in PyTorch to evaluate the effectiveness of RNN models in sentiment analysis. When processing sequence data, we applied dropout after every timestep to avoid overfitting while dealing with such high dimensional vectors. It has been shown to be more effective to use the same dropout mask for an entire sequence as opposed to a different dropout mask each time.

To illustrate the similarities between the BOW Model and the RNN Model, the mean embedding over an entire sequence can be thought of as simply a mean pooling operation. It just so happens that we did this mean pooling operation on a bunch of separate words without a word knowing anything about its surrounding context i.e; pooled before processing any of the temporal information.

This time we are processing the word embeddings such that each subsequent output actually has the context of all words preceding it. Eventually, we still need to get to a single output for the entire sequence (positive or negative sentiment) and we do this by now using a max pooling operation over the number of timesteps. As opposed to the bag of words technique, we pooled after processing the temporal information.

Another important piece of information to be taken into consideration here is - our input size is fixed - meaning we input the review as batch size*sequence length - which implies varying this input length could give us difference in performance and generalization capabilities. Also, we use sub sequences picked randomly to improve the generalization capability.

In this model, we evaluate against:
- Sequence Length
- Dropout Values
- Dictionary Size
- Hidden Unit Size
- Use of GloVe embeddings

RNN Model Results:

1) Without GloVe features and sequence length of 450: 86%

The results here might go against the intuition of what was expected. It actually performs worse than the bag of words model without GloVe features. However, this does sort of make sense considering we were already overfitting before and we vastly increased the capabilities of the model. It performs nearly as well and with enough regularization/dropout, we achieved better results. Training on shorter sequences helped to prevent overfitting.

2) With GloVe features and sequence length of 450: 91.3%

We saw some progress here and the results are the best yet. Although the bag of words model with GloVe features performed poorly, we now see the GloVe features are much better utilized when allowing the model to handle the temporal information and overfitting seems to be less of a problem since we aren't retraining the embedding layer. As we saw in using a RNN model without Glove features, the LSTM seems to be an overkill and the dataset doesn't seem to be complex enough to train a model completely from scratch without overfitting.

2.8.3 RNN Language Model

Finally, we look into how using a Language Model as an input helps improve the BOW Model by leveraging the additional unlabeled data and showing how this pretraining stage typically leads to better results on other tasks like sentiment analysis.

A language model gives some probability distribution over the words in a sequence. We can essentially feed sequences into a recurrent neural network and train the model to predict the following word. Note that this doesn't require any additional data labeling. The words themselves are the labels. This means we can utilize all 75000 reviews in the training set.

We used three stacked LSTM layers, trained to predict neighboring words to output a probability distribution over all words in the vocabulary. We incorporated Gradient Clipping to handle the Vanishing Gradient problem if it arises. We expect this takes a while to train, hence we used a short sequence size(50) and evaluated it with Perplexity as a metric because Accuracy was expected to be low. We generated reviews based on the model built so far(and the temperature).

Generating Fake Reviews:

Although a general language model assigns a probability over the entire sequence, we have actually trained ours to predict the conditional probability where each output is conditioned only on previous inputs. This gives us the ability to sample efficiently, feed this sampled token back into the model, and repeat this process in order to generate fake movie reviews.

RNN Language Model Results:

1) Accuracy with a sequence length of 500: 93.01%

This performs better than all of the previous models. By leveraging the additional unlabeled data and pre-training the network as a language model, we can achieve even better results than the GloVe features trained on a much larger dataset.

2.9 Conclusion:

Our results matched with what we outlined in the proposal barring the Bag of Words model which actually performed worse with the GloVe features and performed better as a standalone model trained on the data.

Our team worked together as planned, we set up an outline for when we wanted each stage completed and finished all of them fairly on schedule. We worked in pairs most of the time and understood that pair programming was really useful to debug while coding, which helped us prototype and debug models quickly.

2.10 Evaluation Metrics and Cross Validation

Since this is a binary - good/bad review classification, we used test loss and test accuracy as our evaluation metrics since the distribution of positive and negative reviews are fairly balanced(not requiring any imbalance class weightage techniques). Since the algorithm effectively iterates on repeated sampling of the data along with prediction on heldout subsets, this takes care of cross validation.

We have included the results of our deep learning model in an excel sheet that has been attached with our submission.

2.11 Possible Use Cases

This project can be used to filter out 'bad' reviews without explicit keyword searches at major e-commerce websites for further investigation and action resolution.

2.12 Appendix

Three samples of fake reviews generated include-

temperature 1.0 a hugely influential , very strong , nuanced stand up comedy part all too much . this is a film that keeps you laughing and cheering for your own reason to watch it . the same has to be done with actors , which is surely " the best movie " in recent history because at the time of the vietnam war , ca n't know who to argue they claim they have no choice ... out of human way or out of touch with personal history . even during the idea of technology they are not just up to you . there is a balance between the central characters and even the environment and all of that . the book is beautifully balanced , which is n't since the book . perhaps the ending should have had all the weaknesses of a great book but the essential flaw of the i found it fascinating and never being even lower . spent nothing on the spanish 's particularly good filming style style . as is the songs , there 's actually a line the film moves so on ; a sequence and a couple that begins almost the same exact same so early style of lot of time , so the indians theme has not been added to the on screen . well was , the movie has to be the worst by the cast . i did however say - the overall feel of the film was superb , and for those that just do n't understand it , the movie deserves very little to go and lets you see how it takes 3 minutes to chilling . i must admit the acting was adequate , but " jean reno " was a pretty good job , he was very subtle

temperature 1.3 a crude web backdrop from page meets another eastern story (written by an author bought) when it was banned months , i am sure i truly are curiosity ; i have always been the lone clumsy queen of the 1950 's shoved director richard " an expert on target " . good taste . not anything report with star 70 's goods , having it worked just equally attractive seem to do a moving train . memorable and honest in the case of " ross , " you find it fantasy crawford is strong literature , job suffering to his a grotesque silent empire , for navy turns to brooklyn and castle of obsession that has already been brought back as welles ' anthony reaches power . it 's totally clearly staged , a cynical sit through a change unconscious as released beer training in 1944 with mickey jones i wanted to walk out on featuring this expecting even glued and turd make . he genius on dialog , also looking good a better opportunity . anyway , the scene in the ring where christopher wallace , said things giving the least 4 time anna hang earlier too leaves rick the blond doc from , walter from leon . is ironic until night with rob roy , he must 've been a mother . which are images striking to children while i think maybe this is not mine . but not in just boring bull weather sake , which set this by saying an excellent episode about an evil conspiracy monster . minor character with emphasis for blood deep back and forth between hip hop , baseball . as many red light figure hate americans like today 's life exercise around the british variety kids . nothing was added

temperature 0.25 a little slow and i found myself laughing out loud at the end . the story is about a young girl who is trying to find a way to get her to go to the house and meets a young girl who is also a very good actress . she is a great actress and is very good . she is a great actress and she is awesome in this movie . she is a great actress and i think she is a great actress . i think she is a great actress and i hope she will get more recognition . i think she has a great voice . i think she is a great actress . i think she is one of the best actresses in the world . she is a great actress and i think she is a great actress . she is a great actress and i was a fan of the original , but i was very disappointed . the plot was very weak , the acting was terrible , the plot was weak , the acting was terrible , the story was weak , the acting was horrible , the story was weak , the acting was bad , the plot was bad , the story was worse , the acting was bad , the plot was stupid , the acting was bad , the plot was stupid and the acting was horrible . i do n't know how anyone could have made this movie a 1 . i hope that it will be released on dvd . i hope it gets released on dvd soon . i hope that it will be released on dvd . i hope it gets released soon because it is so bad it 's good . i hope that

2.13 References

http://ai.stanford.edu/ amaas/papers/wvSent$_a cl2011.pdf$