Sentiment Classification on Amazon Fine Food Reviews and Beyond

David Larance and Matthew Prout
W266.6 Summer 2018

Instructor's Note: The implementation of our project can be found at our project repo. The presentation can be found here.

Abstract

Recurrent Neural Networks (RNN), specifically Long Short Term Memory (LSTM) models are a proven leading edge classification tool shown to work on an array of sentiment tasks. While previous research has been performed on applying LSTMs to common sentiment classification tasks, little has been done to optimize and measure an LSTM classifier across multiple content domain. In this paper we train and optimize a custom built LSTM classification model designed to predict the sentiment of a Amazon food reviews and apply this model across various sourced review sources (beer, restaurant, and movies). The purpose of this is to measure the success of the model against these various data sets while analyzing the error reasons led to their success or failure. The team came away with three main understandings. First, an LSTM sentiment model can be successfully reused on text with a similar profile. Second, a variety of review lengths do not translate well across models. Third, data domain relevance is an important component to sentiment model success.

1 Introduction

Sentiment analysis is used by organizations to market good and services, and manage their reputations. This area of research has steadily grown as more organizations pursue sentiment analysis on social media and other sites where review and rating information is available. Sentiment analysis is inexpensive, quick to perform, and in many ways a preferred alternative to traditional product research that focus on qualitative findings (i.e. focus groups, interviews).

However sentiment analysis is still an evolving field because it is processing natural language which may be beset with incorrect grammar and misspellings. This natural language text may contain idioms and sarcasm which conveys meaning that cannot be directly derived from the words. In addition, there are cases where even humans disagree on the sentiment of a review, which makes it unlikely that an algorithm can come to the correct conclusion for these cases.

In this study, we are investigating how well a sentiment classification model trained on one review data set can be applied to other types of review data sets. While significant research has been performed on model generalization, little has been done to investigate how a model generalizes across datasets types. In short, we hypothesize that performing sentiment classification error analysis across multiple types of dataset domains should yield findings to help future classification models generalize across multiple types of datasets.

2 Background

Early work on using machine learning techniques for sentiment analysis were performed by Pang and Lee[6] for IMDb movie reviews. In their study, they applied naive Bayes, maximum entropy, and SVM models in combination with unigram, bigram, and POS features.

Tang et al[10] survey state of the art techniques in sentiment analysis. They describe sentiment analysis as being dominated by two mainstream directions: lexicon-based approach and corpus-based approach. The direction of the field is moving away from feature engineering towards approaches that use word embeddings which store semantic information. Neural networking techniques such as convolutional neural (CNN) networks and recurrent neural networks (RNN) using LSTMs provide powerful semantic compositions for learning sentiment.

Li et al[4] discuss using LSTMs for performing sentiment analysis, and provide an explanation

on how these networks learn sentiment using visual analysis.

Finally, the work by Hong and Fang[2] has proved to be insightful for our project. In their study they perform sentiment analysis on several large corpus using state-of-the-art techniques including LSTM based RNNs.

2.1 Word Embeddings

Word embeddings are vector representations of words that preserve syntactic and semantic information about the words. Work by Joseph Turian, et al (2010) showed that they can be used as features in NLP models

The most popular word embeddings are word2vec and GloVe. word2vec embeddings are derived from running a simplified feed forward neural network on a large corpus of documents. It is a predictive model that can be used to predict a word based on context (CBOW) or contex words given a word (Skip-gram). GloVe embeddings are generated from the probabilities of the co-occurrences of words in a corpus. Jeffrey Pennington, et al (2014) demonstrate that GloVe embeddings have state-of-the-art performance on various word analogy tasks. GloVe embeddings are trained over large corpuses of data and have different dimensions.

For this project, we will use GloVe embeddings trained on Wikipedia and Gigaword. Pennington[8] states that there are diminishing returns using vectors larger than dimension 200, and Hong ang Fang[2] saw no improvement in sentiment analysis when using vectors larger than 100 dimensions.

2.2 Recurrent Neural Network (RNN)

RNNs are neural networks that consist of a chain of single update functions that feed their context into the following update function. They are naturally suited to language modelling where each word needs to be processed sequentially.

Theoretically they can support long range dependencies, but due to the 'vanishing gradient' problem, basic RNNs have length limitations. Long Short Term Memory (LSTM) cells have been developed which do not suffer from the 'vanishing gradient' problem, and are now widely used with RNNs.

For our project, we will be performing sentiment analysis over reviews that consist of multiple sentences. The sentences may be a mixture of different sentiments, and sentences themselves may express dichotomous sentiment.

Sentiment analysis techniques that just use local word context such as bag-of-words (BOW) models often suffer from the "thwarted expectations" effect that Pang and Lee[7] have discussed, where the sentence begins to provide a positive review, but ends up providing a negative review.

To avoid this behavior, we need to use techniques that can weigh the sentiment from the entire review. RNNs maintain state over a series of tokens, which can be used by our project to solve this problem.

3 Methods & Approach

3.1 Datasets

The training dataset we are using to build our classification model from is the Amazon Fine Food Reviews data set from the Kaggle website. The dataset was prepared by the Amazon-Kaggle team to support a larger sentiment classification contest. Each entry in the dataset contains a long form text review and a rating that ranges from 1-5, however as we'll detail below the model was eventually constructed to expect a binary input (positive / negative).

The datasets that we have compiled to test our classifier against include the following review domains:

1. *Beer:* Compiled from the RateBeer website. We scraped reviews from the provided text file and prepared a small (700 reviews) and large (70,000 reviews) test data sets. Scores range from 0-20 and were transformed to fit our models expected binary input. Reviews are long, description focused (color, viscosity, etc.), and fact based.

2. *Movies:* Compiled from the IMDB website. Data was provided in individual .txt files, requiring significant processing. Reviews are extremely long, with complicated language and phrasing, and contain no food, taste, or smell descriptors.

3. *Restaurants:* Compiled from Yelp website. Vast amounts of review data is made

available by Yelp however since we are just using the entries for testing purposes a small set was compiled and prepared (1,500 reviews). Reviews are short, mix description and experience, and focus on food related descriptors.

Our hypothesis is that the accuracies on all the data sets should be above 50%. In addition, the accuracies for the the Yelp and RateBeer data sets should be better than the IMDB data set because the language used is food centric.

3.2 Binary Sentiment Classification Model Design

Initial data preparation focused on cleansing, tokenizing, and embedding each review prior to running through the LSTM model. The tokenization was managed by the NLTK word_tokenize function with special attention paid to strip special characters, switch out upper case, and handle special punctuation situations. Following the tokenization, embedding ids were converted using a standard GloVe word embedding matrix. For the Amazon reviews, we partitioned the data into 60% training, 10% validation, and 30% test.

The neural network was built using TensorFlow. It consists of an embedding layer that converts the word IDs into word vectors, a hidden layer consisting of the LSTM cells, an output layer that converts the output of the last LSTM cell to a logit, and a softmax activation function.

The architecture of the sentiment analysis model is shown by figure 1.

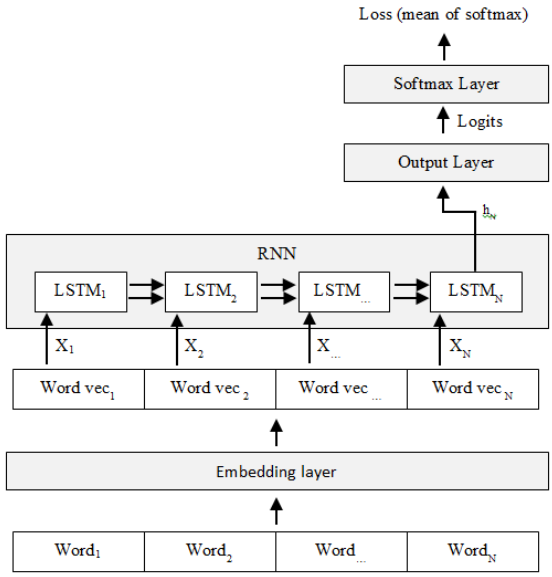

Figure 1: Layers in RNN Model

The plan was to just use a 1 layer LSTM and add additional layers based on the performance of the model. Hong and Fang[2] stated that they achieved better results when averaging the hidden states over the entire sentence, rather than just using the result of the last cell, however we opted for the standard LSTM model. The model was optimized using the Adam gradient descent optimization algorithm.

3.3 Exploratory Data Analysis - Amazon Reviews

The Amazon reviews data set is good for sentiment analysis for several reasons. First, the user has given their review a rating along with a free form text description of their review. Second, there are over 568,454 reviews, and neural network models perform better when they train with more data.

A distribution of the number of reviews for each rating shows that there is an imbalance in the number of reviews that are given a score of 5.

During training an equal number of reviews of each score will be used. For sentiment analysis, we are also interested in knowing about the length of the reviews because this dictates the length of the recurrent neural network (RNN).

The average review length is 96, but there is one outlier review with a length of 4141. Because longer RNNs suffer from the vanishing gradient problem, it is desired to keep the length of the RNNs as compact as possible. The length

of the review that captures 95% of all the reviews is 267.

3.4 Test Data Transformation

The IMDB and Yelp data were both labelled with binary sentiment labels (positive and negative) requiring no score transformation to fit our model. This was one of several reasons we chose to build the classifier as a binary model (other reasons being classifier complexity and dataset distribution are discussed later). However, the RateBeer dataset required transformation from its score range of 1-20 into a binary score. The team assumed that any beer rating over or under 10 was positive or negative and transformed the dataset into a binary model respectfully.

3.5 Naive Bayes Baseline Model

Our initial baseline model for predicting an Amazon food review was a naive Bayes Bernoulli model. The data was prepared by loading it into a dataframe and tokenizing the review text using the word_tokenize() function from the NLTK library. The data set was then split into a training and test set. Next, the data was converted to a sparse matrix of token counts using the CountVectorizer class, which feeds the BernoulliNB class from scikit-learn. For binary classification, we discarded reviews that were rated a 3, and with this data set the test accuracy reported by the model is 87.6%.

We also ran the test cases through the Naive Bayes baseline model. Results shown below in Figure 2.

RateBeer	IMDB	Yelp
59.0	50.4	57.8

Figure 2: Naive Bayes Accuracy Performance

3.6 Neural Bag of Words (NBOW) Baseline Model

A second baseline model was created in order to better compare our final model against a neural networking mode; a NBOW model was used for this purpose. This model converted words to embedding vectors, and those vectors are then added together creating a new vector that stores the representation of the sentence, which is then fed to a feed-forward NN model. This baseline model is using pretrained GloVe word vectors rather than training the embedding vectors.

For binary classification, we discarded reviews that were rated a 3, and the test set accuracy is 93.7%. We also ran the test cases through the NBOW baseline model. Results shown below in Figure 3.

RateBeer	IMDB	Yelp
73.8	69.2	76.5

Figure 3: Naive Bayes Accuracy Performance

3.7 Implementation

For our first iteration of the 1-layer LSTM model, we let it run on 500K batches on the training data. It had a training accuracy of 98% and a test accuracy of 87%, which indicates a degree of overfitting due to the difference between the train and test accuracies.

3.8 Optimizations

We noticed that the original 1-layer LSTM model was unstable and the test accuracy was below the NBOW baseline model. We wanted to investigate optimizations what we could try to improve stability and performance. We decided to try adjusting the hyper parameters such as the batch size, learning rate, and dropout. The various optimizations are displayed below in Figure 4.

Adjusting the batch size to 50 improved stability, but training was slow. Training 260K batches took 11 hours on a 2-CPU instance, and the test accuracy was 81%. Changing the number of LSTM layers from 1 to 2 improved the learning speed and accuracy. After 250K batches the training accuracy was above 90%. We therefore incorporated the 2-layer LSTM into our model. Changing the ‘dropout keep probability’ from 0.75 to 0.9 had disappointing results. The training accuracy seemed to bounce around 50%, and no learning seemed to occur.

The final model we used was a 2-layer LSTM model using a batch size of 24, learning rate of 0.001, and dropout keep probability of 0.75. The test accuracy with this model is 90.8%, which is better than the Naive Bayes baseline model, but worse than the NBOW model.

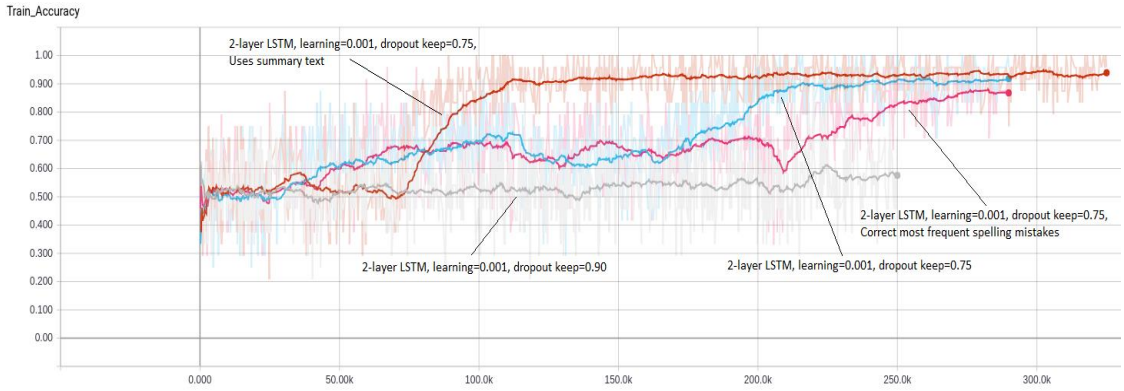

Figure 4: LSTM Model Optimization

3.9 Categories of Errors

The validation test code generates a .csv file with the IDs of the reviews that the model failed to predict the sentiment correctly. When we analyzed the misprediction file for the Amazon reviews, we noticed that the errors seemed to fall into common categories:

Language issues: Some reviews had misspelled words, used idioms not in the dictionary, and used made-up words which prevented our model from detecting the sentiment. Other reviews were written in a sentiment neutral manner.

Thwarted expectations: We saw many reviews use the "thwarted expectations" technique as mentioned by Pang. A similar phenomenon we saw was when reviews had both positive and negative sentiment, but the true sentiment was obscured by more complex language, causing the review to be rated by the more direct sentiment.

Comparison to external knowledge: Some reviews relied on comparing the product to an abstract concept such as "tastes like plastic", or comparison to another product the reader is assumed to know about.

Reviews mixed with the review of another product: A common error we encountered was due to reviews mixing the review with another product. Sentiment was then associated with the other product, which was opposite the sentiment of the product the review was about.

Domain Knowledge: Some reviews used details that required domain knowledge to know whether the description was positive or negative for that product.

3.10 Improvements To Our Model

While analyzing the errors, we noticed that some of the failures were due to reviews that had neutral text, or not much expressed sentiment, but we noticed that the review summary had strong sentiment words.

This gave us the idea to include the review summary in our model. The word ID matrix was expanded to include room for the word summaries, and the max sequence length was lengthened. With the change, the model trained much faster, but the test accuracy was still only 91%, which did not improve over the existing model. In addition, the other test sets that we are comparing the Amazon data to do not have a summary field, so this idea could not be used.

Another feature of the reviews we noticed while analyzing the data set was the number of misspellings and slang in the reviews. These words would not be found by the word embedding, and therefore be marked as an unknown word.

One thing we tried was to determine the list of top misspellings in the corpus. Product names and food centric words were eliminated, creating a short list of commonly misspelled words, such as "*recomend*", "*reccomend*", and "*reccommend*". This list was used by a dictionary to replace these words when the word ID matrix was generated. The model was re-trained with the updated matrix, but the test accuracy fell to 86%, which was a surprising result, since we expected the accuracy to be at least as good as the existing model.

4 Results

Test results are separated below by dataset and include an accuracy forecast as well as a brief error analysis synopsis.

RateBeer: We expected good performance on the ratebeer dataset since the same food related descriptions were expected to apply to beer. Taste related descriptions (ie sour, rich, sweet) were common in both datasets. The accuracy and loss test values are in Figure 5.

Accuracy	0.66
Loss	0.72

Figure 5: RateBeer Test Data Performance

The model showed moderate performance relative to the other test data sets, although lower than the baseline NBOW model. Most errors were discovered to be caused by the "domain knowledge" error where the sentiment was expressed via beer specific language (head, color, beer specific taste, etc.).

IMDB: We expected moderate performance on the IMDB dataset with an assumption that movies and food have a lower vocabulary overlap. Furthermore the IMDB reviews contained long, plot driven descriptions with little sentiment based expression. The accuracy and loss test values are in Figure 6.

Accuracy	0.44
Loss	1.38

Figure 6: IMDB Test Data Performance

The model tested lower than any other sentiment test performed during the study, including the baseline Naive Bayes and NBOW models. A sampling of errors confirmed the hypothesis that movie vocabulary provided less sentiment expression phrasing, but more notably was the impact that the review length made.

Yelp: The team expected good performance as the typical Yelp restaurant review had a similar profile to the training data set. Specifically both datasets contained expressive reviews using similar food related vocabularies. Also, the text was similar in length. The accuracy and loss test values are in Figure 7.

Accuracy	0.72
Loss	0.59

Figure 7: Yelp Test Data Performance

Yelp test data performed well relative to the other two test sets. A sampling of errors showed that common failures were due to more complicated reasons such as other restaurant comparisons (restuarant B was better than A), situational descriptions, and implicit price/quality descriptions. While there were certainly plenty of sentiment failures on the Yelp dataset, it proved that an LSTM can absolutely have success across domain datasets.

5 Conclusion

The purpose of this project was to see how well a sentiment model trained on one review data set would translate to other review data sets. There were three primary lessons from this research. First, that an LSTM sentiment model can be successfully reused on text with a similar profile, such as the Yelp dataset, which had an accuracy of 72%. Second, a variety of review lengths do not translate well across models, such as the IMDB reviews that had a long description of the movie preceded the reviewer's opinion. Third, data domain relevance is an important component to sentiment model success. This is apparent with reviews that are specific to a domain such as beer. For example:

> *Deep reddish colour with a small tan head. Malty, light roasted aroma with notes of coffee and overripe fruit. Malty, prune sweet flavor with a light roasted coffee bitter finish.*

"Overripe fruit" and "bitter finish" sound negative, but this review was rated a 5 (very positive).

References

1. Hochreiter, S. and Schmidhuber, J.. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.
2. Hong, J., Fang, M., "Sentiment analysis with deeply learned distributed representations of variable length texts", 2015.
3. Leskovec, J.. *RateBeer reviews.* Retrieved July05, 2018, from https://snap.stanford.edu/data/web-RateBeer.html
4. Li J, Chen X, Luong MT, Jurafsky D. Visualizing and understanding neural models in NLP. 2015, arXiv preprint arXiv:1506.01066.
5. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In ICLR.
6. Pang, B., Lee, L., and Vaithyanathan, S., 2002. *Thumbs up? Sentiment Classification Using Machine Learning Techniques.* In Proc. of EMNLP 2002

7. Pang, B., & Lee, L. (n.d.). Movie Review Data. Retrieved July 06, 2018, from http://www.cs.cornell.edu/people/pabo/movie-review-data/
8. Pennington, J., Manning, C. Glove: Global vectors for word representation. 2014.
9. Tang, D., Qin, B.,, and Deep, T. Learning for sentiment analysis: successful approaches and future challenges. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 5(6):292–303, 2015.
10. Turian, J., Ratinov, L., and Bengio, Y. 2010. Word representations: a simple and general method for semi-supervised learning. In Proceedings of ACL, pages 384–394.
11. Yelp, Inc. (n.d.). Yelp Open Dataset. Retrieved from https://www.yelp.com/dataset