Sentiment Analysis of Movie Reviews on IMDb using Deep Learning and GloVe

CSE 572: Data Mining Final Project Report

Gowtham Narayan Gopinathan
Computer Science
Arizona State University
Tempe, Arizona, USA
ggopina2@asu.edu,
gowtham.narayan.8@gmail.com

ABSTRACT

The most frequently utilized technique for forecasting user assessments is sentiment analysis[1]. A plethora of machine-learning techniques have been applied to provide accurate predictions on the collected data. But, these classifiers do not consider maximal pooling or long-term dependency. In this project, we categorize reviews using Deep Learning technologies to improve predictions made using these parameters. In order to achieve greater accuracy with less loss and less time, this project uses both the Long Short Term Memory Recurrent Neural Network and the Convolution Neural Network, along with GloVe embeddings.

Additionally, multiple models using RandomForest and Regressors are used, along with FastText embeddings and various sequential models to compare accuracy and other evaluation metrics. Furthermore, BERT architecture is also implemented to check accuracy.

KEYWORDS

Sentiment Analysis, Natural Language Processing, Deep Learning, Long Short Term Memory Recurrent Neural Networks, GloVe Embeddings, Bag of Words, FastText embeddings, Sequential model, BERT

Arizona State University, Tempe, Arizona, USA

1 INTRODUCTION

Movie reviews are a valuable means to gauge the success of a movie. While assigning a movie with a number or star rating, it lets us know about the quantitative success or failure of a film; a collection of film reviews provides us a deeper qualitative perspective on many parts of the film. A textual movie review identifies the film's good and poor elements, and a more in-depth examination of the review might reveal if the film, as a whole, lives up to the reviewer's expectations.

One of the main areas of machine learning, sentiment analysis[1], tries to extract subjective information from textual evaluations. It may be used to assess the reviewer's perspective on particular subjects or the overall polarity of the review. By definition, sentiment analysis or opinion mining uses natural language processing (NLP) and text analysis to get semantic quantification of the statistics under investigation. Using sentiment analysis, certain textual content may be seen as a reaction to particular points of view (e.g. a tweet, or a product evaluation). These indicators are used by decision-makers as a result in planning and implementing suitable measures, as well as in advertising decisions, client searches, or business development in certain geographic locations. The need to comprehend, explore, and investigate this truth has greatly increased as a result of the vast records evolution and the volume of data being transmitted and generated every second.

For people to decide if a movie is worthwhile to see, movie reviews are crucial. A clear overview of all reviews for a film can help consumers make this decision by saving them time from having to read through all of them. Critics frequently post comments on blogs and websites that rate movies,

assisting viewers in deciding whether or not to see the film. Based on their criticisms, sentiment analysis can infer the mindset of the critics.

In this project, we utilize sentiment analysis to examine a collection of movie evaluations written by reviewers in order to determine how they felt about the film overall—that is, whether they enjoyed it or despised it. In order to forecast the review's general polarity, we plan to use the term relationships in the review. As part of this project, we seek to investigate the applicability of various feature extraction approaches used in text mining, such as keyword spotting, lexical affinity, and statistical methods. In addition to feature extraction, we also investigate several categorization methods and investigate how well they function for various feature representations. Ultimately, we determine which fusion of feature representations and classification methods is most effective for the current predicting goal. The project's conclusion includes various sentiment analysis comparisons for improved model prediction.

2 DATASET

The dataset utilized for this project was obtained from the Large Movie Review Dataset[2], which was compiled by Stanford University's AI department for their related paper [3]. It includes 25,000 training samples taken from IMDb[4], where each review is labeled with the movie's score on a scale of 1–10, and the sentiment as to whether the movie is good or bad. Positive reviews are those that have a rating of more than or equal to 7, and negative reviews are those that have a rating of less than or equal to 4, in order to prevent disagreement in training due to ambiguous reviews with ratings between 4 and 7. Additionally, it also includes a test set of 25,000 samples; both the training and test set contain 12,500 positive samples and 12,500 negative samples.

The dataset was first split into two groups for training and testing, each with 25,000 samples. We discovered that this divide was not ideal since there were insufficient training instances, which resulted in under-fitting. Then, we attempted to divide the cases into 40,000 for training and 10,000 for testing. This led to stronger models being developed, but it also resulted in over-fitting on training samples and lower performance on the test set. Finally, we made the choice to employ cross-validation, in which the entire dataset is divided into numerous folds with different samples for training and validation on each occasion, and the classifier's final performance metric is averaged over all outcomes. This increased the overall accuracy of our models.

The data was also the foundation for a Kaggle competition from late 2014 to early 2015 called "Bag of Words Meets Bags of Popcorn."[5] Above-average accuracy was attained, with champions obtaining 99%.

3 RELATED WORK

Researchers at Stanford University carried out the initial work(by Maas et al.)[3] on this dataset, using unsupervised learning to group words with similar semantic associations into word vectors. On these word vectors, different classification models were run in order to determine the polarity of the reviews. This method is especially helpful when the sentiment of the data is strong and subjectivity in the semantic affinity of the terms and their intended meanings is a factor.

In addition to the aforementioned, Bo Pang[6] and Peter Turnkey[7] have put a lot of research towards polarity detection of product and movie reviews. Additionally, they have focused on developing a multi-class classification of the review and forecasting the reviewer rating of the film or item. These studies included the use of SVMs and Random Forest classifiers for categorizing reviews, as well as the use of various feature extraction methods. One significant finding in these works was the absence of a neutral category in classification on the grounds that neutral texts are disproportionately challenging to categorize since they are near to the boundary of the binary classifiers.

3.1 Word Embeddings

A word embedding is a learnt representation of text in which words with the same meaning have a similar representation. Individual words are represented as real-valued vectors in a predetermined vector space in a process known as a word embedding[8]. The values of the vectors are learnt in a manner like a neural network, and each word is mapped to a single vector. Using a dense distributed representation for each word is one important strategy. A real-valued vector with frequently tens or hundreds of dimensions is used to represent each phrase. In contrast, sparse word representations, such a one-hot encoding, need hundreds or millions of dimensions. A real-valued vector representation for a predetermined set sized vocabulary is learned via word embedding techniques from a corpus of literature. On other tasks, like document classification, the learning process is either combined with the neural network model or it is an unsupervised process using document statistics.

3.1.1 GloVe Embeddings

The Global Vectors for Word Representation, or GloVe[9], technique is an addition to the word2vec approach for effectively learning word vectors created by Pennington et al. at Stanford. It combines the local context-based learning in word2vec with the global statistics of matrix factorization techniques like LSA. GloVe creates an explicit word-context or word co-occurrence matrix utilizing statistics throughout the whole text corpus rather than using a window to determine local context. The outcome is a learning model that could lead to more effective word embeddings overall.

3.1.2 FastText embeddings

A refinement of the word2vec model, fastText[10] is a word embedding technique. FastText renders each word as an n-gram of characters rather than immediately learning word vectors. This enables the embeddings to comprehend suffixes and prefixes and helps to grasp the meaning of shorter words. Once character n-grams have been used to represent the word, a skip-gram model is trained to learn the embeddings. Because the underlying structure of the word is not taken into consideration, this model is referred to as a bag of words model with a sliding window over a word. The sequence of the n-grams is irrelevant as long as the characters are contained inside this frame. Rare words perform nicely with fastText. Therefore, even if a word wasn't observed during training, its embeddings may still be obtained by splitting it up into n-grams. Words that are not in the model dictionary have no vector representation in Word2vec or GloVe, respectively. This is a significant benefit of this approach.

3.2 Neural Networks

Sentiment analysis employing neural network models[11] incorporating LSTMs, CNNs[12], and RNNs produced excellent accuracy results because the encodings and sequences are better learned by NN models. Hence, we shall use a combination of the above stated techniques to develop models that can accurately predict sentiments and return desirable evaluation metrics.

3.3 BERT

The cutting-edge machine learning model BERT[13], short for Bidirectional Encoder Representations from Transformers, is utilized for NLP applications. BERT was created by Jacob Devlin and his coworkers at Google in 2018. Devlin and his associates trained the BERT on the English Wikipedia (2,500M words) and BooksCorpus (800M words), and in 2018, they attained the highest accuracy rates for various NLP tasks. There are two general BERT variants that have been trained: The huge model has 24 layers, 1024 hidden layers, 16 heads, and 340 million parameters compared to the standard model's 12 layers, 768 hidden layers, 12 heads, and 110 million parameters. The usage of BERT architecture in this project might lead to better evaluation metrics than GloVe embeddings.

4 METHODS

4.1 Exploratory Data Analysis

First, we check if the dataset is balanced, and make sure that there are no missing values.

Fig 4.1.1: The dataset has an equal split of 25,000 positive and 25,000 negative reviews

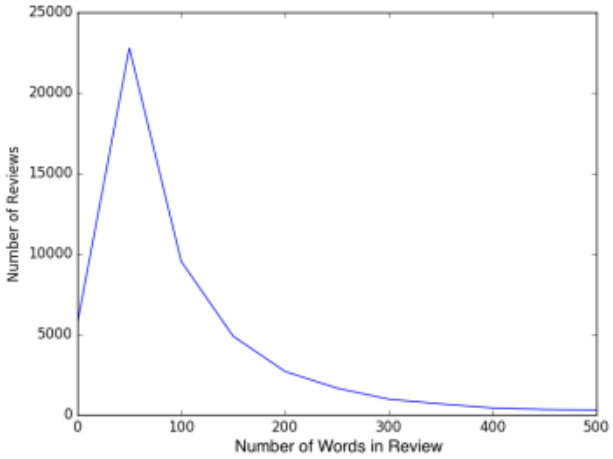

Fig 4.1.2: Plot of total number of words in each review

The implementation will be carried out utilizing embeddings and neural networks, both of which rely on pre-trained word vectors where each word in the corpus is given a probabilistic score. The "meaning" of the words is deduced from these vectors, which are shown in a low-dimensional plane along with these probabilities. Post this, the vocabulary is tokenized and the model is built.

4.2 Data Preprocessing and Visualization

Firstly, a model needs to be made to perform classification on the 50,000 samples present in the training set. To begin that process, the first step is to clean the data. The following should be performed so that the data is fit to be utilized:

- Remove any and all HTML tags present in the review text, using the BeautifulSoup[14] module
- Clear occurrences of punctuation, escape sequences, brackets, quotes, and any alphanumeric characters from the review text, using the regex module
- Convert all the letters in the review text to lowercase, since the model might consider the same words with different capitalizations as different ones
- Eliminate any stopwords(words that are used to convey proper grammar, like a, an, the) from the text by utilizing the NLTK stopWords corpus[15], because they have no bearing on the review's overall mood.
- Finally, lemmatize the text using the tokenizer module[16] to modify any words to their lemmatized form and ensure consistency throughout all reviews by utilizing the same versions of the terms.

After performing data preprocessing, we further analyze the reviews using word clouds and by checking the number of words and characters in the text.

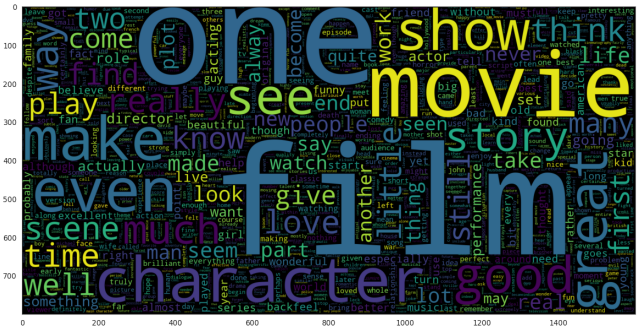

Fig 4.2.1: Word cloud for positive samples after preprocessing

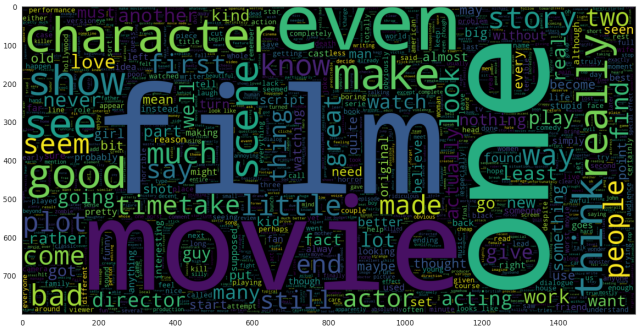

Fig 4.2.2: Word cloud for negative samples after preprocessing

Fig 4.2.3: Plot of number of characters in a review

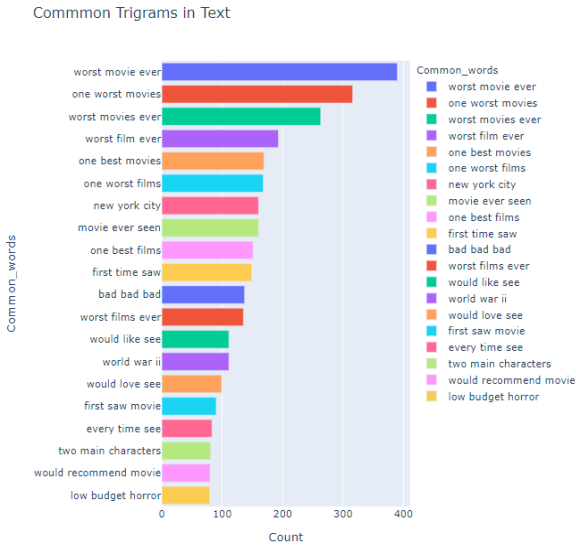

Fig 4.2.4 Trigram Analysis of words in review text

We also perform Unigram, Bigram and Trigram analysis to discover frequently used, and review defining words. Out of these, Trigram analysis is the most representative of the sentiment of the review.

4.3 GloVe Embedding

Then, we apply the GloVe embedding to the review text. The percentage of terms in our data that are covered by the vocabulary is shown by embedding coverage. After cleaning the dataset, it is observed that the vocabulary now covers over 87% of the training set and 95.5% of the testing set, which was earlier much less. Now, the data can be used in the model to develop classifications.

4.4 Neural Network approach

Firstly, a basic NN Sequential model with regularization is built and fit on the data, to see if the approach has any merit.

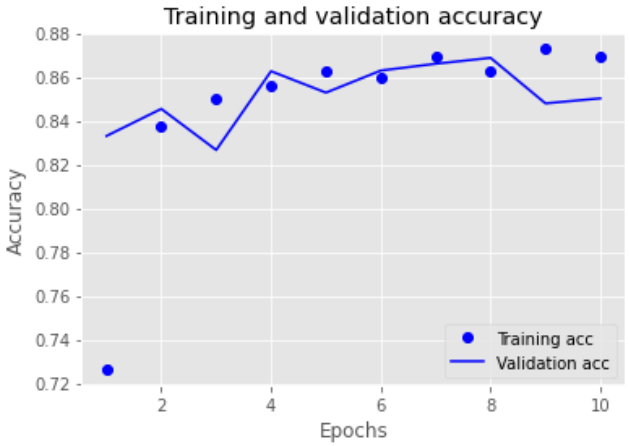

Fig 4.4.1: Accuracy vs Epochs plot for basic NN model

After this, Grid Search was applied to deduce the best parameters to fit the model with. Next, Keras architecture is used to build the model[17]. Two LSTM layers and a Conv1D layer are implemented to train the model. Since there is a lot of variance present, using dropout is essential since it lowers bias and hence decreases overfitting. Callbacks are very beneficial since they halt our model when its validation accuracy begins to decline for two consecutive epochs and they save the optimal weights that can provide our model the highest validation accuracy.

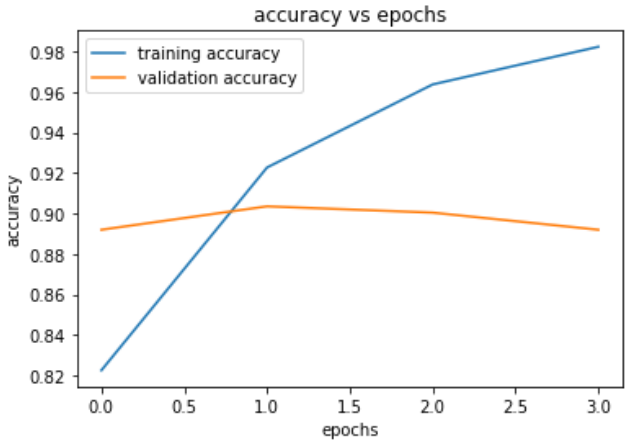

Fig 4.4.2 Accuracy vs Epochs plot for LSTM model

4.5 fastText Embedding

In place of GloVe embeddings, fastText embeddings are also applied and the above models are run again and outputs are recorded.

4.6 BERT approach

LSTMs are slow because they require a lot of computation. Since learning occurs independently in both directions, they are not really bidirectional. Thus, the full context is somewhat obscured. Since the data is read sequentially, GPUs that favor parallel processing struggle with them.

This is where the BERT architecture comes into play. BERT was developed using a huge text corpus for training, which allows the architecture or model to learn the heterogeneity in data patterns and perform effectively across a variety of NLP applications. Being bidirectional, BERT gathers knowledge from both the left and the right sides of the context of a token during training.

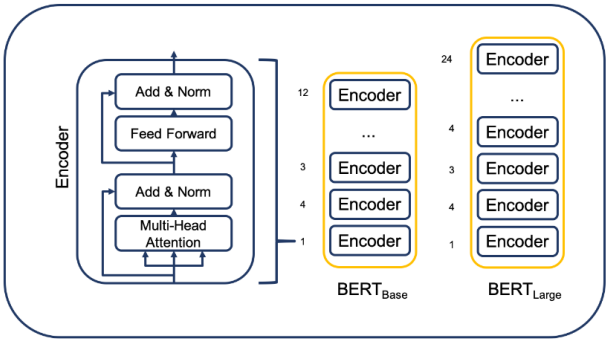

Fig 4.6.1 BERT Architecture

BERT is dependent on a Transformer (the attention mechanism that learns contextual relationships between words in a text). An encoder to read the text input and a decoder to provide a prediction for the job make up a simple Transformer. Since the objective of BERT is to produce a language representation model, just the encoder portion is required. A series of tokens that are first transformed into vectors and then processed by the neural network make up the input to the BERT encoder. However, BERT requires the input to be modified and embellished with additional metadata before processing can begin:

1. Token embeddings: At the start of the first sentence, a [CLS] token is added to the input word tokens, and at the conclusion of each sentence, a [SEP] token is added.

2. Segment embeddings: Each token has a marker that designates either Sentence A or Sentence B. Because of this, the encoder can tell which phrases are which.

3. Positional embeddings: Each token is given a positional embedding to show where it belongs in the sentence.

Since the BERT model is highly complex and due to computational shortcomings, only three successful runs were feasible, and their outputs were recorded.

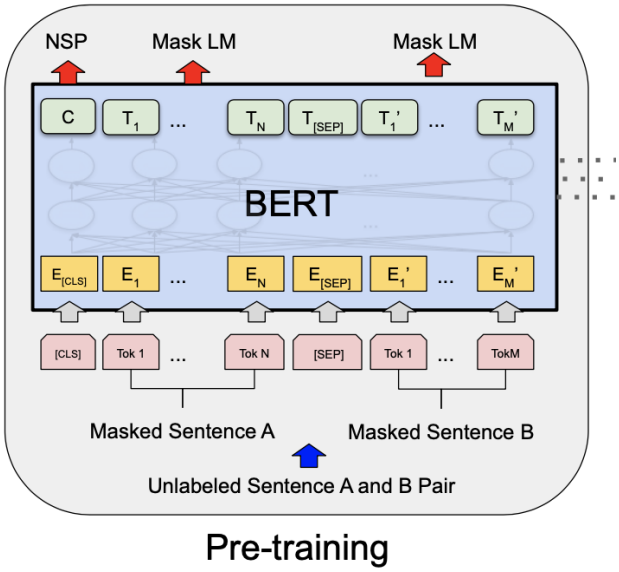

Fig. 4.6.2 BERT Architecture(ii)

5 RESULTS

After shuffling the training data, we train the models with distinct splits, leveraging Classification accuracy, Classification precision, Classification recall, and Classification F1 score as assessment measures.

The basic NN model resulted in an accuracy of 84.78%, and an F1 score of 0.85.

The model that makes use of LSTM artificial neural network is implemented, has a total trainable params of 20,673,954 and is first run over 3 epochs, with rmsprop optimizer, and fastText embedding. It resulted in an average accuracy of 0.9407 and a loss of 0.1675 on the test set and an accuracy of 0.8640 and a loss of 0.3752 on the test set.
Next, the adam optimizer is utilized along with GloVe embedding, and that resulted in an accuracy of 0.9621 on the training data and 0.8631 on the test set.
Then, we ran the LSTM network over 5 epochs and a batch size of 64 and using fastText embedding, resulting in an accuracy of 0.9708 on the training data and 0.8764 on the test set.
Similarly, running the LSTM network over 5 epochs and a batch size of 128 and GloVe embedding, resulted in an accuracy of 0.9843 on the training data and 0.8711 on the test set.
Taking the average of all these results, we arrive at an F1 score of 0.89.

Finally, the BERT model is implemented, using available pretrained resources and that gives out a model with a total trainable parameters of 110,074,370. Fitting this model to the dataset, we arrive at a train accuracy of 0.9093 and a test accuracy of 0.9378; and an F1 score of 0.91.

6 DISCUSSION

According to the results, the preprocessing methods were effective and may aid the model in learning the significance of words in the positive and negative spectrum when using word counts as weights.

The GloVe embeddings covered less than 40% of the vocabulary and text prior to preprocessing the data; this value jumped to more than 85% after performing the preprocessing techniques.

The GloVe embedding seems to work better than fastText, and ensures a better accuracy.

Finally, the BERT model seems to be powerful, especially when presented with huge amounts of data, and results in better accuracy than the computationally expensive (LSTM) neural networks.

7 FURTHER WORK

The accuracy of the neural network models may be compared to that of simple classification models that employ RandomForest and Logistic Classifiers[18] to predict on the same data.

A similar type of word embedding to GloVe and FastText called Word2Vec[19], can be incorporated into the project work, and the results from the three can be compared.

The model(s) created can be run on the movie polarity dataset[20] and the accuracies compared.

The same can be done on other social media datasets, such as the twitter tweet sentiment analysis.

8 CONCLUSION

The methods implemented and the general approach taken, with respect to data preprocessing, model creation and fitting and results evaluation prove that sentiment classification is successfully performed, with high accuracy. Furthermore, BERT architecture is also implemented, and the performance metrics from both approaches are compared, and the BERT model seems the best technique for sentiment analysis on the IMDb Dataset.

ACKNOWLEDGMENTS

This work was done as part of a requirement for Arizona State University's CSE 572: Data Mining course. I want to thank Dr. Hannah Kerner for her aid with the project work and advice during the course. She gave me a lot of insight into the inner workings of data mining techniques and model analysis, which I have utilized heavily in this project to help refine the models and obtain results.

REFERENCES

[1] Sentiment Analysis – Wikipedia – https://en.wikipedia.org/wiki/Sentiment_analysis

[2] Large Movie Review Dataset – http://ai.stanford.edu/~amaas/data/sentiment/

[3] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011, vol.1.

[4] Internet Movie Database – http://www.imdb.com/

[5] https://www.kaggle.com/c/word2vec-nlp-tutorial

[6]Pang, Bo; Lee, Lillian; Vaithyanathan, Shivakumar (2002). "Thumbsup? Sentiment Classification using Machine Learning Techniques". Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).

[7] Turney, Peter (2002). "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews". Proceedings of the Association for Computational Linguistics.

[8] https://www.morganclaypool.com/doi/10.2200/S00762ED1V01Y201703HLT037

[9] https://nlp.stanford.edu/projects/glove/

[10] https://fasttext.cc/

[11] https://link.springer.com/article/10.1007/s42452-019-1926-x#Sec11

[12] https://link.springer.com/article/10.1007/s11042-019-07788-7

[13] https://huggingface.co/blog/bert-101

[14] https://beautiful-soup-4.readthedocs.io/en/latest/

[15] NLTK Stopwords Corpus: http://www.nltk.org/book/ch02.html

[16]https://huggingface.co/docs/transformers/main_classes/tokenizer

[17] “Introduction to Keras for Researchers.” https://keras.io/getting_started/intro_to_keras_for_researchers/

[18] http://cs229.stanford.edu/proj2020spr/report/Wu_Shin.pdf

[19] https://www.tensorflow.org/tutorials/text/word2vec

[20] https://www.cs.cornell.edu/people/pabo/movie-review-data/