SENTIMENT ANALYSIS ON IMDB MOVIE DATASET

Ruoxi Pan Zhaoxi Wang
Khoury College of Computer Sciences
Northeastern University
Boston, MA 02115

1 Introduction

In the digital age, with internet platforms such as Instagram, Facebook, and IMDB, consumers are passionate about sharing their pictures and opinions on these websites/APPs. These texts being uploaded carry great values for doing opinion mining. With the help of opinion mining systems, this unstructured information could be automatically transformed into structured data of public opinions about products, services, brands, politics, or any topic that people can express opinions about. This data can be very useful for commercial applications like marketing analysis, public relations, product reviews, net-promoter scoring, product feedback, and customer service. The aim of this project is to capitalize on this trend and analyze the sentiment behind the reviews provided by customers on IMDB. The goal of this project is to construct several sentiment analysis models that can accurately classify the binary-classes, and we will compare the results to see which model performs the best.

2 Method/Model

For the first step, to get an overview, we directly adopted the IMDB dataset of 50,000 movie reviews from Keras, labeled by sentiment (positive/negative). Reviews have been preprocessed already, and each review is encoded as a sequence of word indexes (integers). We used simple classification models including Logistic regression, LDA, decision tree and naive bayes as baseline models. The preliminary result given is less than promising. The accuracies given by these models are only roughly around 50%. We then started off by running the data on a simple RNN model, which contains an embedding layer, one LSTM layer and one dense layer with a total of 213301 parameters. The LSTM is short for long short term memory, and has the ability to model long term dependencies. It has three gates that help extending the RNN neuron: a forget gate, an input gate and an output gate. These gates enable the LSTM to learn long-term dependence and make the optimization easier. In our RNN model, we used cross-entropy as our loss function and adam as optimizer. The accuracy rate was 0.866, with batch size equal to 64 and number of epochs equal to 3.

3 Datasets and Preprocessing

The datasets used are taken from the Keras package and Kaggle. For milestone 1, we first explored several methods by using the Keras dataset, where the number of documents in the training data and testing data are both 25000. Since the dataset is already handled in Keras, we inspected the maximum and minimum review length and found them to be 2697 and 14 respectively. Then we decided to restrict the sequences by a max_words variable equal to 500 and pad the short ones with 0 in order to feed them into the RNN.

Because apparently the Keras dataset performs poorly on our binary classifiers, next we are in the process of doing our own preprocessing steps with the Kaggle dataset, which contains a labeledTrainData.tsv, testData.tsv and unlabeldTrainData.tsv. We will first use the labeled training Data and testing data, which contain 25,000 movie reviews each. We used the beautifulsoap, re and stopwords packages to remove the html tags, stopwords, non-alphabetic characters and converted upper cases to lower cases.Then we performed feature extractions by using CountVectorizer and TfidfVectorizer packages. We plan to do multi-model prediction by fitting both TF-IDF transformation and count vectorizer transformation on the following models: Multinomial Naive Bayes, Logistic Regression, RbfSVC. Ensemble learning: Random Forest, ExtraTree, AdaBoost, Gradient Boosting Classifier, XGboost, lightgbm and Voting Classifier.

4 Training Details and Evaluation Criteria

For the binary classifiers, regression was implemented. Results came close to the package ones. Then every classifiers were evaluated by the accuracy scores and the cross entropy loss. Confusion matrices were calculated as well.

Then for the RNN model, we trained only with a baseline model which contained one embedding layer, one LSTM layer and one dense layer. We performed a simple cross validation: Trained on 24936 samples and validate on 64 samples with 3 number of epochs. Accuracy and loss were calculated for both training and validation set. The RNN model outperforms the binary classifiers. We would like to investigate the different performance more by doing prep

5 Next Steps

We generally think it’s best to get baseline predictions with the simplest possible solution before spending time doing potentially unnecessary transformations. Consequently, after training the dataset on some models to get a first dive into the problem. We would next write and implement our own algorithms and compare the results we get from implementing python’s packages. Also for logistic regression, we

would like to discover the best decision threshold that gives the highest accuracy. For milestone 2, we plan to implement Logistic regression and SVM on our own. More options are considered depending on the progress. Then for the RNN model, we will add more layers such as dropout layer to make it more complicated and hopefully more comprehensive on doing prediction. We will keep doing feature engineering and model tuning to improve the classifiers accuracy. As our goal for this project is to learn various different algorithms in doing classification, by applying packages to implement and also write our own algorithm, we are getting more understanding of the classification algorithms.