Sentiment Analysis for Text Documents

ECE 6254 Project Proposal

Charles Topliff, Elvan Uğurlu, Jinbang Fu, Huy Thong Nguyen, Sanghoon Lee

Georgia Institute of Technology

October 18, 2019

Project Summary

In recent years, the information explosion has led to an overabundance of online data. Text classification has become a key technique to handle such data, where sentiment analysis is one such task studied by many machine learning researchers. The objective of our project is to investigate different classification methods for sentiment analysis of IMDb movie reviews. Specifically, this is a text classification task to determine whether or not a review is considered positive (the reviewer enjoyed the movie) or negative (the reviewer did not enjoy the movie). There exist many semantic details in the english language that could make this a difficult classification problem, like word connotations or capturing the context of a certain word(s).

To establish a baseline, first a vectorizer found all of the words in the training corpus, then a vocabulary was constructed with an ordering from most popular to least popular. A movie review was then represented as a count of each word in the vocabulary. We then trained a Multinomial Naïve Bayes (MNB) classifier with and without parameter smoothing. To improve the accuracy of the classification algorithms and avoid overfitting, we applied pre-processing to the text reviews by removing unrelated characters and stop words, transforming the remaining words back to their root form, considering word sequences consisting of 2 words in addition to individual words, and representing every feature by the count of tokens (1- or 2-word sequences) etc. We then applied our classification algorithms to the pre-processed data. For Support Vector Machines (SVMs), we trained 5 different classifiers such as a linear kernel function, radial basis kernel function, as well as 2nd, 3rd, and 4th degree polynomial kernel functions. We also trained a logistic classifier (LC) with a specific choice of the regularization strength, a convolutional neural net (CNN), and a recurrent neural network (RNN). The testing accuracy achieved by these classifiers is summarized in the table below.

Classifier	SVM(LIN)	SVM(RBF)	SVM(2)	SVM(3)	SVM(4)	RNN	CNN	LC	Baseline (MNB)
Testing Accuracy (%)	85.48	85.04	84.45	72.66	70.46	87.4	88.5	89	82.03

Our results demonstrate an improvement over the baseline MNB approach. We also remark that there exists many potential future directions for this work. One such direction is to consider ensemble methods for SVMs to combat dimensionality issues encountered in training SVMs. Others include further exploring how pre-processing can help capture intangible aspects like context and connotation and provide a better representation of the data for classifiers, while reducing inherent overfitting that could occur in subsequent training session.

Project Description

The primary objective of our project is to investigate different sentiment classifiers. However, due to the plethora of relevant features that cannot be neglected without loss of information, this may lead to a very high dimensional input space. We want classifiers that are robust to overfitting, but do not neglect the potential dependence between features in the data. Thus, we looked into different pre-processing schemes to prepare the dataset for classification.

Previously, we looked at the author attribution problem using a MNB classifier. While the Naïve Bayes (NB) classifier does perform well, we expect that other classifiers will perform better since NB assumes conditional independence between features. In the problem, the data is represented as a vector of word counts in a bag of words model. One might expect NB to perform well because different authors use different and unique language. In the sentiment analysis problem, movie reviews come from a plethora of different authors who may not use words unique to a particular class. Consider the following phrases taken from the test data:

"Let's face it: kids tend to be horrible actors, which is understandable and we can't blame them for that."

"As for the heist, the action, the drama....it was put into the last 8-10 minutes of the movie and was pathetic and anticlimactic. Horrible!"

Although *horrible* appears in both, the first is excusing the performance of the actors, while the second is describing the movie as a whole. Such neighboring context is critical to determine sentiment, as *horrible* may exhibit different intentions despite being the same word.

One other issue is the problem of overfitting. Because there exist some words in the data that only appear in a very small number of documents, it is possible to overfit to these features. These kinds of issues appear often in our data, and we want to investigate the use of classifiers that do not assume conditional independence when applied to the sentiment analysis problem. To overcome these challenges, we plan to explore different classifiers such as LC, SVMs, and deep neural nets (CNN/RNN) and compare their performances on how well they can distinguish between positive and negative IMDb reviews.

Data Characterization & Pre-Processing

The IMDb movie review dataset contains 50,000 reviews for movies as text files. Each text file contains a reviewer's rating on a scale from 1 to 10, excluding 5 and 6. For sentiment classification, movies rated greater than 6 or less than 5 are labelled numerical 1 or 0 to denote if the reviewer thought the movie was good or bad, respectively, resulting in perfectly balanced classes. The data is split into 12,500 reviews each for training and testing. Thus, a bag of words model results in 85,000 unique words, where they are ordered in frequency of appearance in all reviews (i.e. common words like 'the', and 'and' are higher in the ordering, whereas uncommon words like 'cliche-ridden' and 'overly-critical' are lower in the ordering).

The main goal of pre-processing is to obtain *feature vector representations* for the text files to use for machine-learning techniques. To reduce the vocabulary size and minimize overfitting, some words and characters that do not necessarily carry useful information like punctuation or conjunctions are removed. Although some words have the same root, they appear in different forms (e.g. "enjoy" and "enjoyable" have the same root "enjoy"). Considering all different forms as different features not only increases the vocabulary size but also leads to overfitting. Thus, we applied different implementations of *stemming* and *lemmatization*, two text normalization techniques to identify the root of words. Since *stemming* is a heuristic technique that chops off the ends of the words and *lemmatization* is a more complex technique that finds the base dictionary form of the given words, *lemmatization* performed better as *stemming* very aggressively truncated the words. Finally, the cleaned and normalized texts are split into tokens, which are meaningful parts of the sentence by utilizing N-grams scheme. N-grams considers contiguous sequence of N words that helps to preserve the context the words appear in [1]. In particular, we used 1-gram and 2-grams. To vectorize the tokens, we used Count Vectorizer scheme [2], where the number of appearance of each token represents the value of that specific feature. The total number of features is kept as a parameter to determine the complexity of the models in the classification stage.

Multinomial Naïve Bayes Classification & Preliminary Results

We applied MNB classifier to the un-processed IMDb movie review dataset to initially investigate model complexity & overfitting. We trained on the first N most popular words for values of N ranging from 1 to 40,000. Figure 1 shows classification accuracy for predictions made on training and testing data for both MNB with smoothing and MNB without smoothing.

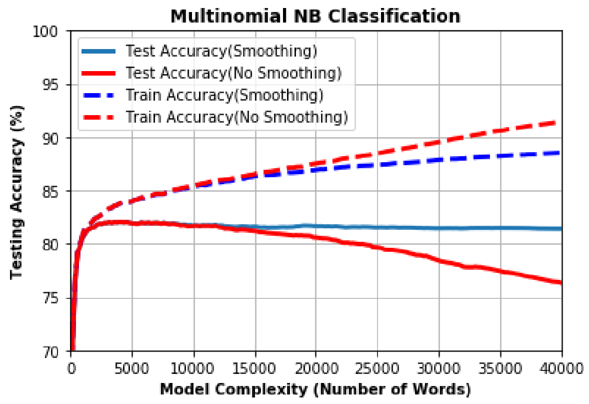


Figure 1: MNB Classification accuracy for Testing & Training Sets

The MNB classifier clearly demonstrates overfitting. As the model complexity grows, it becomes more dependent on less frequent words, learning features that do not occur out-of-sample, increasing the training accuracy especially in the case with no smoothing. With smoothing, the test accuracy approaches 82%, but the training accuracy still increases with N. These results provide a good baseline to compare our other classifiers. Given the diversity of the language and the number of unique authors, the Naïve Bayes (NB) approach performs surprisingly well. Yet, the problem of overfitting in NB and the unexplored pre-processing techniques for training data are avenues to further explore in this project. We also ran the MNB classifier using the pre-processed data to fairly compare the effects of pre-processing. We do not include a figure for space limitations, but our experiments show that the pre-processed data not only reduces the maximum accuracy attained by the classifier, but also appears to be more robust to overfitting. This is likely because combinations of words appear in more documents than niche words in the case where we are just doing unique word counts.

SVM Classification

The problem of training an SVM classifier can be stated as finding the maximum-margin hyperplane that separates two variables. For some dataset X is not separable, we introduce slack variables and the Lagrange dual of this program as

$$\min_{\omega,b,\xi} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{N}\xi_i \quad \text{s.t.} \quad y_i(\omega^T\phi(x_i)+b) \geq 1-\xi_i, \quad \xi_i \geq 0, \quad \forall i \in \{1\ldots N\}$$

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \quad \text{s.t.} \quad y^T\alpha = 0, \quad 0 \leq \alpha_i \leq C, \quad \forall i \in \{1\ldots N\}$$

where e is a vector of all ones, $Q_{ij} = y_i y_j K(x_i, x_j)$, and K is the kernel matrix that defines an inner product between two datapoints. We investigated different kernel matrices given below.

Kernel	Linear	Polynomial (Degree n)	Radial Basis Function
Inner Product Rule $f(x_i, x_j)$	$x_i^T x_j$	$(\gamma x_i^T x_j + r)^n$	$exp(-\gamma\|\|x_i - x_j\|\|_2)$

We employed Linear (n=1), Radial Basis Function (RBF), and Polynomial (Poly) of degrees 2, 3, and 4 in training our SVMs. The γ parameter is scaled to the variance of the data. Due to the data size, the kernel matrix can be large especially when considering polynomial kernel functions because the dimensionality of the data increases more than quadratically. So, we trained on a random subsample of the training data (8000 reviews) using a range of 1 to 40,001 features and computed the test accuracy. Figure 2 shows test results for the SVM classifier.

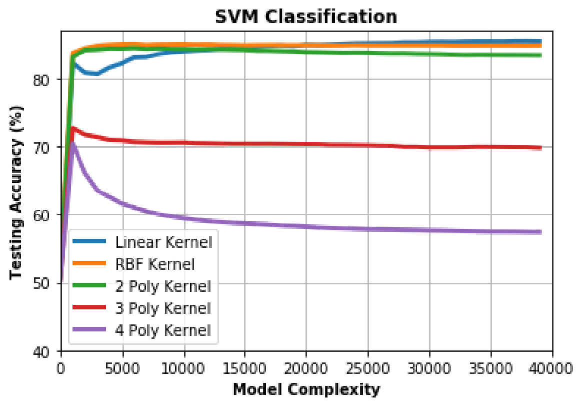


Figure 2: SVM Classification accuracy for Testing Set

Clearly, Linear, RBF, and 2 Poly kernel SVMs achieve greater performance than the MNB classifier. For degree 3 and 4 Poly kernel functions, the classifier appears to experience overfitting, likely due to the fact that the SVM has more degrees of freedom and can thus fit to more noisy data. The maximum test accuracy is achieved with Linear, although the 2 Poly and RBF saturate with a lower number of features. Since the SVMs are trained only on a subset of the data, they cannot access all of the information in the training dataset. In future work, ensemble methods can be investigated where the ensemble of SVMs covers the entire dataset.

Logistic Classification

Logistic regression model is a powerful and widely used choice to model a binary dependent variable, particularly to determine whether the reviews are positive or negative. The issue of overfitting was addressed by introducing bias to generalize the problem via regularization. In sklearn library, the function LogisticRegression uses the variable “C” to represent the inverse of regularization strength. To determine "C", grid search combined with 5-fold cross validation is utilized. By doing grid search, the value is tested from 0 to 10 in steps of 0.001, where each value is evaluated via the 5-fold cross validation. As shown in Figure 3a, first, it splits the data set into 5 consecutive folds, and then uses each fold once as the validation set while the rest four are used as the training set. With pre-processing, the best estimator is achieved with "C=0.021" with 89% test accuracy, which has 1% improvement over without pre-processing. Figure 3c shows the estimated coefficients for the 10 best/worst features with the best estimator, making intuitive sense as respective words were correctly identified to have either strong positive or negative connotations, implicitly proving the validity of our estimator.

To check for overfitting, we tested with a range of 200 to 10000 most frequently used words. Grid search combined with 5-fold cross validation was used to determine the values of "C", but the step size is reduced to 0.01 to save time. Figure 3b shows the test accuracy across the number of features, demonstrating that regularization greatly minimizes overfitting. Also, test accuracy saturates 0.88 when the number of features >7000, which means that LC can provide a relatively high test accuracy with less features.

Recurrent/Convolutional Neural Network Classification

Deep neural networks (NN) have been a major driving force for the recent developments of machine learning techniques, so we will apply NN to build up a model for the IMDB sentiment classification. NN has an embedding layer that converts word indices to multi-dimensional

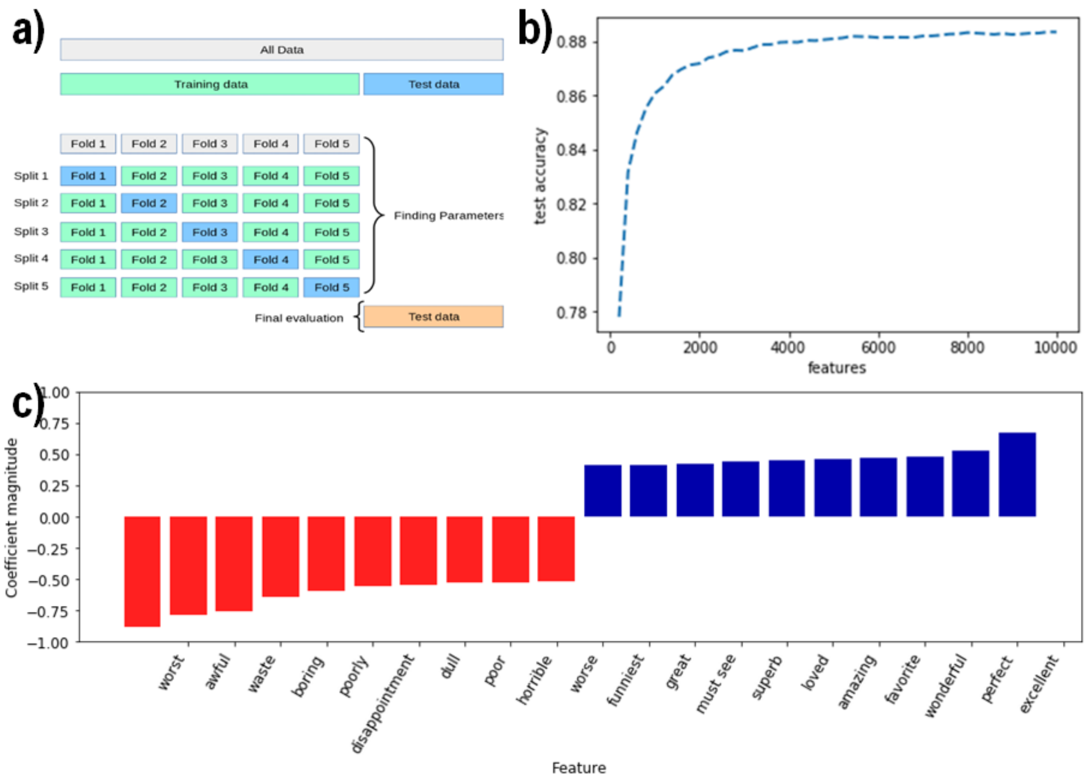


Figure 3: (a) 5-fold cross validation (b) test accuracy (c) top 10 best/worst features

vectors, where words with similar meanings will be in close proximity in this vector space. Consequently, we do not need to pre-vectorize the word indexes. Instead, we only feed the pre-processed text with a fixed sequence length (600 by padding sequences) to the neural net. The weight matrix of the embedding layer is automatically calculated via back propagation.

We use RNN or CNN for classification since we only require a single output. We use Embedding, LSTM, Dense, Dropout layers for RNN, and utilize Embedding, Conv1D, Global Maxpooling Dense, Dropout layers for CNN, as depicted in Figure 4a. The loss is Binary Cross Entropy and the optimizer is Adam. We develop our RNN/CNN model based on the Keras library [3] and they demonstrate similar results with >98% training accuracy and 87.5-88.5% testing accuracy, respectively (Fig. 4b). After each epoch, the best model with lowest test loss is saved as our final RNN/CNN model for subsequent analysis.

Some interesting "intuition" can be learned from the weight matrices. We extracted similar words by looking at the distance between word vectors in the embedding layers. In Figure 5, the network recognized words with similar connotations and thus made their distances close. Furthermore, when we fed a single word to the network and recorded the score at the sigmoid function, the network differentiated between positive and negative words, appropriately scoring its weight. Overall, the NN captured contextual information up to a certain extent.

Conclusions & Future Work

The results achieved by the classifiers exceed the baseline MNB classifer. In particular, careful pre-processing of the data and *dimensionality promotion* improve classifier performance in general. Increasing the number of features through word combinations to preserve

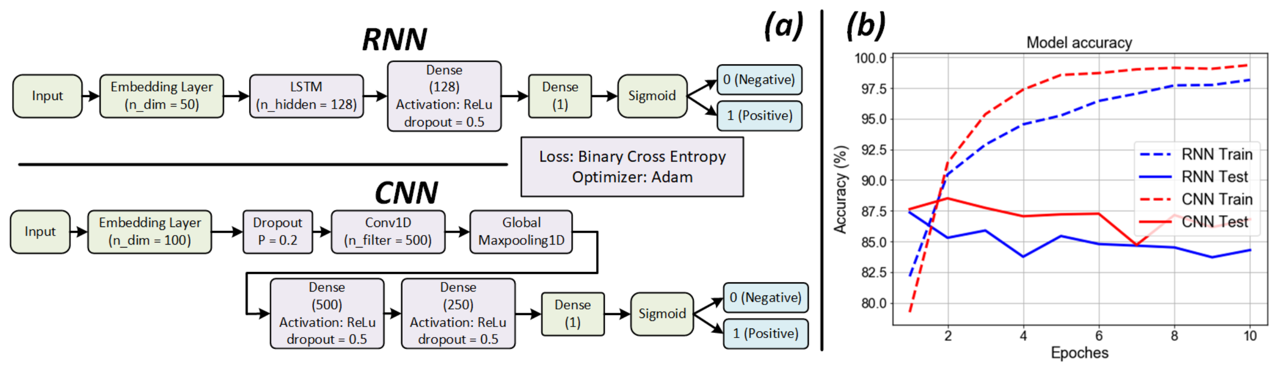
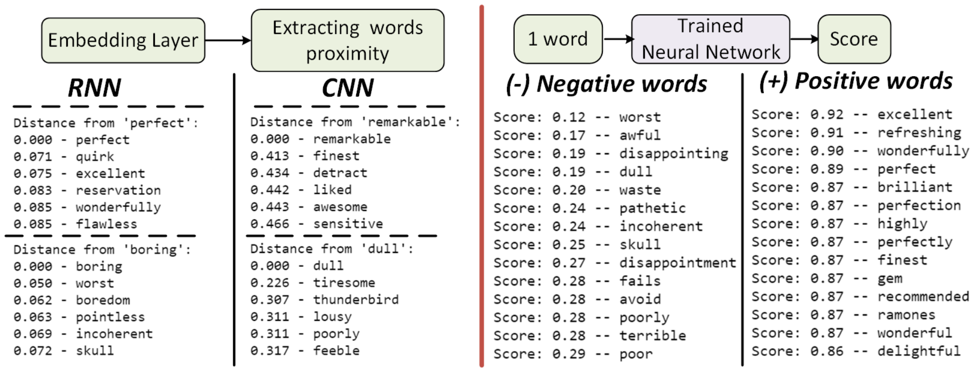


Figure 4: (a) RNN and CNN models (b) Training and Testing Accuracy of RNN/CNN model



```
Distance from 'perfect':
0.000 - perfect
0.071 - quirk
0.075 - excellent
0.083 - reservation
0.085 - wonderfully
0.085 - flawless
```

```
Distance from 'boring':
0.000 - boring
0.050 - worst
0.062 - boredom
0.063 - pointless
0.069 - incoherent
0.072 - skull
```

```
Distance from 'remarkable':
0.000 - remarkable
0.413 - finest
0.434 - detract
0.442 - liked
0.443 - awesome
0.466 - sensitive
```

```
Distance from 'dull':
0.000 - dull
0.226 - tiresome
0.307 - thunderbird
0.311 - lousy
0.311 - poorly
0.317 - feeble
```

```
Score: 0.12 -- worst
Score: 0.17 -- awful
Score: 0.19 -- disappointing
Score: 0.19 -- dull
Score: 0.20 -- waste
Score: 0.24 -- pathetic
Score: 0.24 -- incoherent
Score: 0.25 -- skull
Score: 0.27 -- disappointment
Score: 0.28 -- fails
Score: 0.28 -- avoid
Score: 0.28 -- poorly
Score: 0.28 -- terrible
Score: 0.29 -- poor
```

```
Score: 0.92 -- excellent
Score: 0.91 -- refreshing
Score: 0.90 -- wonderfully
Score: 0.89 -- perfect
Score: 0.87 -- brilliant
Score: 0.87 -- perfection
Score: 0.87 -- highly
Score: 0.87 -- perfectly
Score: 0.87 -- finest
Score: 0.87 -- gem
Score: 0.87 -- recommended
Score: 0.87 -- ramones
Score: 0.87 -- wonderful
Score: 0.86 -- delightful
```

Figure 5: Some results learned from the trained neural net

contextual information achieves some level of information gain. There were many challenges encountered in this project, namely the dimensionality in the SVM training and overfitting of the classifiers. Finally, we remark that there exists much future work, such as looking at more advanced metrics for data pre-processing or ensemble learning methods that will further address the rapidly increasing need to analyze the abundance of online data beyond movie reviews.

The responsibilities of this project were divided in the following manner: data pre-processing (Elvan Uğurlu), SVM classification (Charles Topliff), Logistic classification (Jinbang Fu), RNN/CNN classification (Huy Thong Nguyen), and complete report/poster organization (Sanghoon Lee).

References

[1] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175. Citeseer, 1994.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[3] François Chollet et al. Keras. https://keras.io, 2015.