Project 2 - Team New Line

Steven Bradley (stbradle@calpoly.edu)
Andrew Engel (awengel@calpoly.edu)
Markelle Kelly (mkelly23@calpoly.edu)

Abstract. Our team analyzed a dataset of movie reviews from Stanford with the goal of predicting whether a review was tagged as positive or negative. To do this, we extracted features from the raw text, then implemented the following three classifiers: Logistic Regression, Linear Discriminant Analysis, and Support Vector Machine. In the end, we were able to improve upon the original features of the Stanford team, achieving better predictive accuracy on all three classifiers. We observed few notable differences in the accuracy of the classifiers for the old and new features.

1. INTRODUCTION

The Large Movie Review dataset, published by Stanford, contains hundreds of movie reviews tagged by hand as positive or negative. A basic feature set (bag of words) is provided alongside this dataset − this simply records the count of every possible word contained in each review. This feature set was used as our baseline to compare our new features to.

Objectives

For this project, our goal was to answer the following questions:

1. Can we extract a better feature set from the text than the creators of the dataset?
2. How do the three classification techniques we implement compare on the Large Movie Reviews dataset?

To this end, we decided which features we could create to capture certain pieces of information about the review, then trained and tested three different classifiers on the original Stanford features and our new features: Logistic Regression, Linear Discriminant Analysis, and Support Vector Machine.

2. FEATURE CREATION PROCESS

Motivating Ideas

For predicting whether a review was tagged as positive or negative, we expected writing qualities such as word choice, sentiment of the words used, and level of animation to be useful. Positive reviews might contain different common words or word pairs than negative reviews. Even when different words are used, it makes sense that more positive reviews would use words with more positive sentiments overall. For this reason, we wanted to incorporate both the words used and their sentiments as features. We also thought other writing qualities, such as punctuation and capitalization, might be related to whether a review was positive or negative. If a review was written with lots of exclamation points or completely in capital letters, that may give us a clue about the writer's opinion of the movie.

Implementing Ideas

To capture word choice, we started by extracting unigrams and bigrams from the reviews with nltk's TF-IDF vectorizer. Words were stemmed with a Porter stemmer and English stop words were removed. Due to the size of the resulting feature set, we chose to keep only the top 500,000 unigrams and bigrams. Furthermore, we performed dimension reduction with singular value decomposition, so ultimately, only 250 columns of features were added.

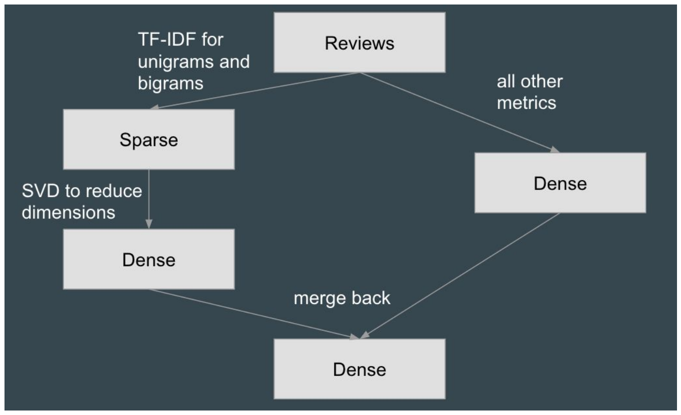
We represented the sentiment of the words themselves with an average score of sentiment over all nouns, adjectives, and adverbs in the review. We chose these parts of speech because we expected their word choice to be most subjective and thus most important in determining average sentiment of the reviewer. Sentiment for each word was estimated with nltk's senti_synset function, which returns positive and negative sentiment scores for a given synset. We found this set of synonyms for each relevant word and computed its overall score as its negative score subtracted from its positive score. The average of these scores became the sentiment feature for each review.

We also investigated swear words to help capture the tone of a review. Using a list of profanity from bannedwordlist.com, we counted the number of swear words in each review. This was divided by the total number of words in the review to give a "density" of profanity.

The other writing qualities we measured were number of exclamation points and question marks, number of words in all capital letters, and number of uses of the word "I". These were also standardized by the number of words in the review. Since the length of the review could also be related to how positive it is, we added the review length in words to the feature set as well.

Final Dataset

After performing singular value decomposition on the unigrams and bigrams, we had a dense matrix that could be combined with the other features we extracted. Merging these two feature sets gave us our final dataset, which has 257 columns − 250 from TF-IDF and our seven other variables (average sentiment, swear word density, exclamation point density, question mark density, capitalized word density, "I" statement density, and number of words). The following diagrams illustrates how the data was broken up and recombined:



Modifications to Original Dataset

Since the data for the original Stanford features − a sparse matrix constructed from the bag of words counts − was so large, it was difficult to run our classifiers on it. To resolve this, we also performed singular value decomposition on the given dataset, reducing it to 250 columns.

Standardization

For the models that used gradient descent, we found that standardizing the input features with a z-score helped them to converge faster. We believe that this is because all variables were placed on the same scale. Because of this, we chose to standardize the original Stanford features and our new features before passing them in to every model.

3. CLASSIFICATION MODELS

To answer our questions, we implemented three different linear models: Logistic Regression, Linear Discriminant Analysis, and Support Vector Machine.

Logistic Regression

For our implementation of this model, we used Stochastic Gradient Descent with a batch size of 1. In class, we discussed the following gradient descent update rule:

$$\beta_{n+1} \leftarrow \beta_n + \eta(Y - p)x$$

where $p = \frac{1}{1 + e^{-x^T \beta}}$

However, we expanded on this a little, including L2 regularization, to prevent overfitting. Thus, our gradient descent update rule became [1]:

$$\beta_{n+1} \leftarrow \beta_n + \eta((Y - p)x - \frac{\alpha\beta_n}{d})$$

where α is a new parameter of the model and d is the number of predictor variables in the model.

Linear Discriminant Analysis

For our Linear Discriminant Analysis classifier, we used the Fisher LDA objective to find the best separating linear discriminant vector between good and bad movie reviews. That is, there was a closed form solution to fitting this model:

$$(S^{-1}B)w = J(w)w$$

where S is the sum of the scatter matrices for both of the classes and B is the between class scatter matrix. Solving this equation, the linear discriminant vector is the eigenvector associated with the largest eigenvalue of $S^{-1}B$. We used numpy's built-in linear algebra package to perform the eigenvector decomposition.

Support Vector Machine

In our implementation of a Support Vector Machine classifier, we used Stochastic Gradient Descent with a batch size of 1. We decided to solve the Primal Problem when fitting our model instead of the Dual Problem. This choice was made prior to us covering the Dual Problem in class. Thus, our gradient descent update rule was:

$$\beta_{n+1} \leftarrow \beta_n - \eta(\beta_n - Cy_i x_i) \text{ if } y_i x_i^T \beta_n < 1$$
$$\beta_{n+1} \leftarrow \beta_n - \eta\beta_n \text{ otherwise}$$

4. EVALUATION PROCESS

Each model underwent the same evaluation process. For each model, we took the original feature set and trained the model on the train portion, then classified and assessed accuracy on the test portion. Then, we repeated this process for our feature set.

Model Tuning

It made sense to hypertune models with outside parameters using grid search. We did not consider learning rate a parameter for gradient descent, so the only parameter we had to hypertune was the slack weighting C for our Support Vector Machine Classifier.

C	Accuracy
1e-5	85.1%
1e-4	85.2%
1e-3	85.1%
1e-2	85.1%
1e-1	85.1%
1e0	86.2%
1e1	87.6%
1e2	87.4%
1e3	85%

The above chart shows that we obtained the best results using $C = 10$. Note that this grid search optimized for our feature set's test accuracy.

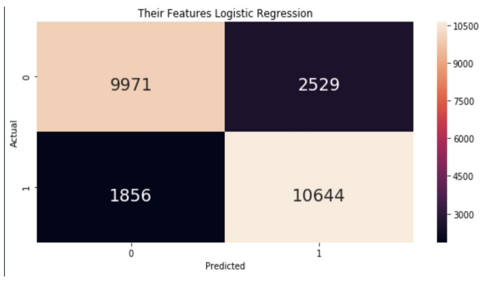
For the models that used gradient descent, we found that standardizing the input features with a z-score helped them to converge faster. We believe that this is because all variables were placed on the same scale. Because of this, we chose to standardize the original Stanford features and our new features before passing them in to every model.

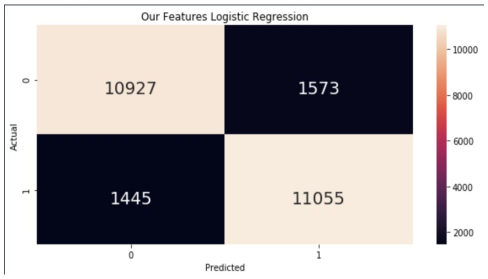
5. RESULTS

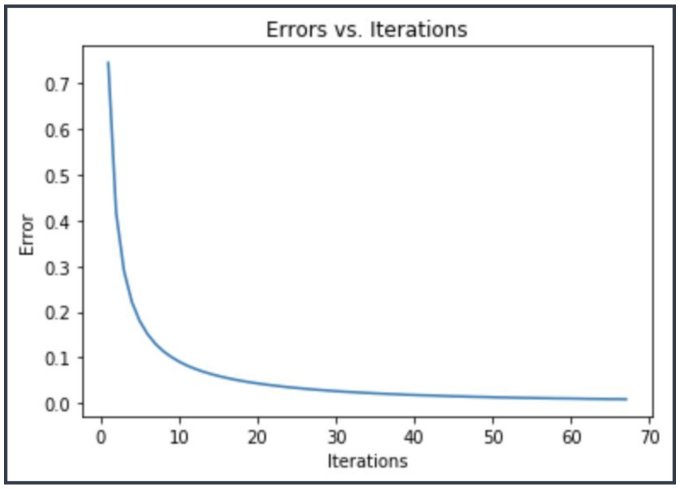
Below is a summary of the performance of each model. This includes the test accuracy of the model on the original features and our features, as well as a test confusion matrix for the original features and our features. For Logistic Regression and Support Vector Machines, which used gradient descent, graphs are shown of the error convergence across iterations.

Logistic Regression

	Accuracy
Original Features	82.4%
Our Features	87.9%



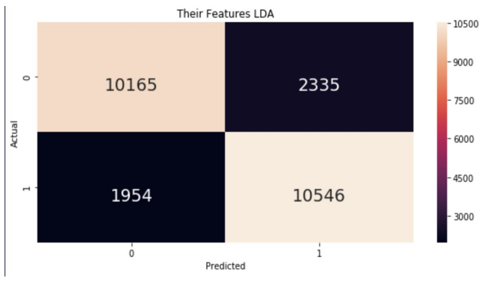


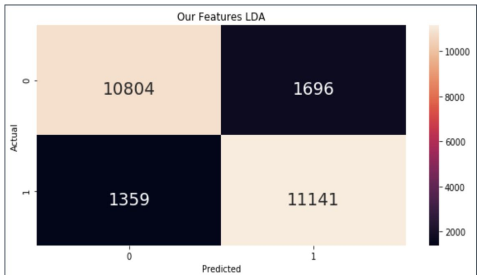


For the Logistic Regression model, the use of our features improves the predictive accuracy by 5.5%. On Stanford's features, logistic regression prediction is slightly imbalanced − predicting more false positives than false negatives, but this is fixed with the addition of our features. The model converged in just under 70 iterations using a learning rate of 1e-4.

Linear Discriminant Analysis

	Accuracy
Original Features	82.8%
Our Features	87.7%

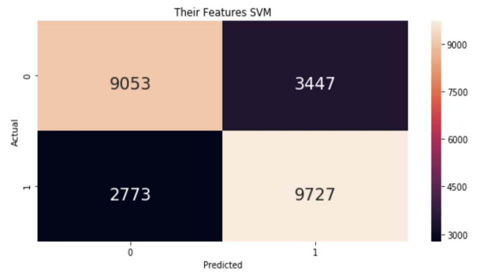


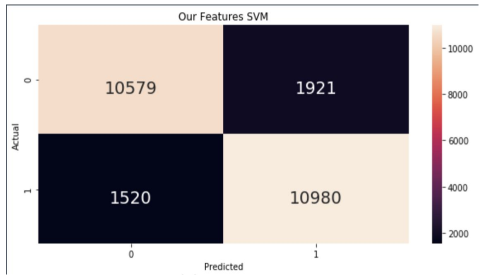


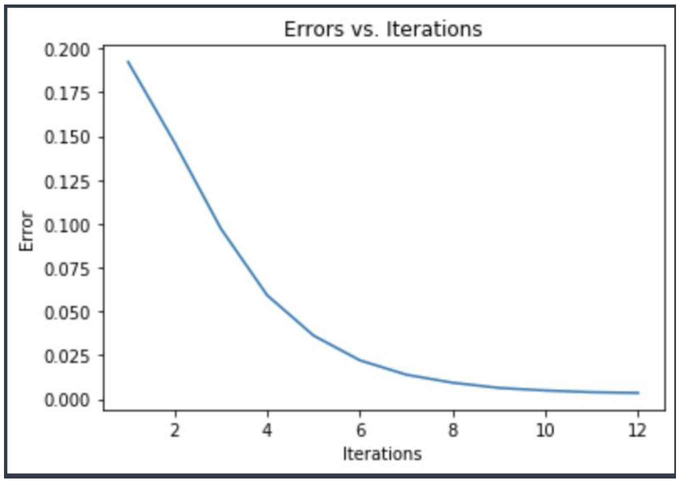
The predictive accuracy of the Linear Discriminant Analysis model improved by 4.9% with the addition of our features. The balance of the confusion matrices stays relatively constant, with slightly more false positives than false negatives both on our features and Stanford's.

Support Vector Machine

	Accuracy
Original Features	75.1%
Our Features	87.6%







The predictive accuracy of our Support Vector Machine model showed the most dramatic improvement − 12.5 − between Stanford's features and ours. As shown in the confusion matrices, our support vector machine classifier again leaned slightly towards more false positives. The support vector machine converged after only 12 iterations using a learning rate of 1e-5.

Notes

The confusion matrices for each classifier and input feature set all appear to be fairly balanced, only leaning towards false positives to a small degree. We don't think that any of the classifiers were particularly biased towards predicting any class, which was expected given the training data contained an equal number of good and bad reviews.

Additionally, the error convergence graphs show nothing out of the ordinary - our models look like they worked as intended.

6. CONCLUSION

Comparing the Classifiers

Below is a table that aggregates our results across all three classifiers:

	LR	LDA	SVM
Original Features	82.4%	82.8%	75.1%
Our Features	87.9%	87.7%	87.6%

There are a few things to note here. First, there is improvement in the classification accuracy from the original Stanford features to our features for all three models. This indicates that our features were able to capture more information for discriminating between reviews than the original ones. That said, the original Stanford features did quite well given their simplicity - it appears that just using bag of words is very impactful for movie review sentiment analysis.

Second, the Logistic Regression (LR) and Linear Discriminant (LDA) classifiers performed much better than the Support Vector Machine (SVM) classifier on the original Stanford features, but the SVM classifier was able to catch up to the others on our new features. We hypothesize that either a new feature we added made the data more linearly separable, or that SVMs simply perform better on higher dimensional data.

Third, despite using different classification techniques, all three models arrived at about the same accuracy on our features, which is quite interesting. There could be multiple reasons for this, but perhaps our new features have a limit as to how well they can separate the data, and all three classifiers hit that limit via different methods.

For the original Stanford features, it is obvious that LR and LDA performed better than SVM. Using our features, LR has a slightly higher test accuracy. That said, we found it difficult to decide on a "best" model for our features because the test accuracies of the models are so close. For a tiebreaker, the SVM was by far the fastest model to converge of the three, and since it was the underdog, we'll cast our vote on it.

Objectives

Let's revisit our motivating questions to evaluate our answers.

1. Can we extract a better feature set from the text than the creators of the dataset?

Since we significantly improved the accuracy of all three predictors, we can confidently say that our feature set is better than the original one.

2. How do the three classification techniques we implement compare on the Large Movie Reviews dataset?

On the original Stanford features, LR and LDA clearly outperformed SVM. However, on our new features, all three models reached around the same accuracy, with LR barely taking the lead.

Based on these answers, we feel confident that we were able to adequately address the requirements of this project.

Difficulties/Improvements

The biggest roadblock is this project was something we did not anticipate at all. When we coded our pipelines and classifiers, we expected the data to follow a certain shape. We used our classifiers and didn't get any errors, so we assumed things were working, but we sometimes got terrible results. The tools we used for feature engineering and pipelining converted between various shapes of data, but all of our calls to numpy happily accepted any data shape (even if it was not what we expected) and somehow applied functions to them. We spent a significant amount of time debugging the data shapes at various stages of our code.

Another difficulty that we had was testing if our classifiers were correctly implemented or not. In the end, we gained confidence in our classifiers by comparing our outputs to the scikit-learn implementations, ensuring they matched.

REFERENCES

[1] S. S, "Regularized Logistic Regression," *Machine Learning Medium*, 15-Sep-2017. [Online]. Available: https://machinelearningmedium.com/2017/09/15/regularized-logistic-regression/. [Accessed: 02-Nov-2019].