Leo Kim and Sam Hecht
CS135: Machine Learning
October 19th, 2023

Project A - Classifying Sentiment from Text Reviews

Problem 1

1A : Bag-of-Words Design Decision Description:

The first step in our bag of words implementation was "cleaning" the data. We leveraged sklearn's CountVectorizer to perform most of the cleaning. CountVectorizer defaults to selecting "tokens of 2 or more alphanumeric characters" separated by punctuation and whitespace (i.e., it throws out punctuation and whitespace). We had success with this cleaning, but attempted to improve upon it. CountVectorizer mentions that since all punctuation is ignored, words such as "we've" would be split into "we" and "ve", which we found undesirable. Thus, we created our own regex to avoid this type of splitting but we found that this actually decreased our model's score on the validation set, so we did not keep this change.

We also tried excluding words from our vocabulary that appeared too frequently or infrequently in reviews. We used sklearn's GridSearchCV on several different min_df (i.e., the minimum number of times a word should appear in our dataset for it to be used as a feature) and max_df values (i.e., the maximum frequency of words to include in our dataset), and determined optimal values of 1 and of 0.333 respectively. A min_df of 1 corresponds to including all words which appear in our dataset as features and a max_df of 0.333 corresponds to excluding words which appear in more than 33.3% of data points (which was only the word 'the'). We settled on these values by running grid search on values of [1, 2, 3, 4, and 6] for min_df and 10 values linearly spaced in the range [0, 0.5] for max_df and seeing which logistic regression model produced the best mean_test_score optimizing for area under the AUROC curve.

In our feature vectors, we considered each word a feature, and used the counts of each word in a sentence to represent each review (so most vectors were quite sparse). In our final model, our vocabulary size was 4509 words, which does not vary across folds because our model refits on the entirety of the 2400 review training set once it finds the optimal parameters. Thus this vocabulary of size 4509 is all of the tokens considered words by the CounterVectorizer except for the word 'the'. Lastly, our model disregards any out-of-vocabulary words in the test set.

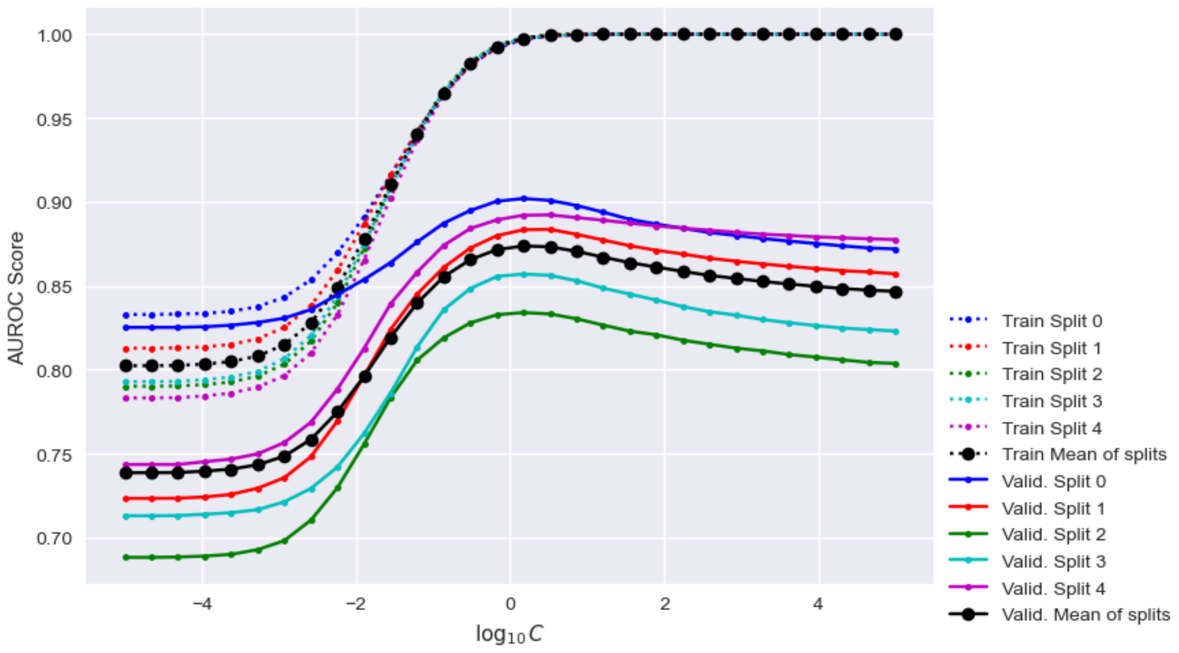
1B : Cross Validation Design Description:

The performance metric we chose to optimize for during our cross-validation (CV) was the area under the AUROC curve because we knew that the leaderboard would evaluate our predictions based on this value.

In order to execute CV, we used the sklearn.model_selection.GridSearchCV class. We used 5-fold validation, which is the default for the class. We decided that 5 folds would be enough to ensure that there were enough different fold combinations while not being too much so that the model took too long to train. Our 5 fold approach caused each fold to contain 480 reviews, meaning that in each iteration of CV our model would be trained on 1920 reviews and then validated on the heldout 480 reviews. We split the folds using the sklearn.model_selection. StratifiedKFold class, which randomly creates each fold split. We used the sklearn libraries whenever possible, assuming they would be much more thoroughly tested and optimized than any similar code we could produce. Before using any library, we thoroughly read the documentation and ensured that the libraries we were using made sense for our use case. Sklearn's GridSearchCV class has an option to refit the model on the entire training set once optimal hyperparameters have been found, which we used. Thus, after finding optimal hyperparameters and refitting, we accessed the best_estimator_ field of the GridSearchCV class to retrieve the optimal trained model.

1C: Hyperparameter Selection for Logistic Regression Classifier

We used a logistic regression classifier for this problem. This classifier is reasonable for this task because we are aiming to classify reviews as either positive or negative. Thus, since logistic regression produces numbers between 0 and 1, **the output of logistic regression represents probabilities that a data point is either positive or negative, which enables us to easily classify them as positive or negative reviews**. We used sklearn's GridSearchCV class to perform a grid search of C values (i.e., the inverse of the regularization strength or the coefficient on the area under the AUROC curve in our loss function) to identify the best value for the hyperparameter C for our model. **We trained models on 30 C values on a log scale from 10^{-5} to 10^{5}, performing 5 fold cross-validation for each C value and optimized for AUROC score**. We allowed our logistic regression a **maximum of 10,000 iterations**, and as a result our training took some time but **we did not run into convergence issues**.



This figure shows both underfitting for small values of C (low AUROC score) and overfitting for the larger values of C (nearly perfect AUROC scores on training but not on validation), and so we know that this log scale of C values is reasonable. We can clearly locate the **sweet spot in the middle of the graph where C is near 1**. We chose the classifier which optimized the validation's AUROC score (looking at the mean of all 5 splits) which was **C = 1.487**. **This classifier is clearly the best since it outperforms all models with other C values on every split it was validated on with regards to AUROC score**.

1D : Analysis of Predictions for the Best Classifier

False Positives	False Negatives
"I really wanted the Plantronics 510 to be the right one, but it has too many issues for me.The good"	"The soundtrack wasn't terrible, either"
"The service here is fair at best"	"They know how to make them here"
"The loudspeaker option is great, the bumpers with the lights is very ... appealing"	"Not much dialogue, not much music, the whole film was shot as elaborately and aesthetically like a sculpture"
"Not good enough for the price"	"You won't be disappointed"

One thing that is immediately apparent that our model struggles with is the presence of negations before words such as "not good" or "wasn't terrible". Since our BoW implementation is a count vector that does not convey word order, the model strictly reads the positive or negative words without recognizing the negation that comes before it. To account for this in the next model, we could try using bigrams and trigrams of words as features in addition to word counts so that the model can recognize these negations. In general the model seemed to perform better on Amazon and Yelp reviews than on the IMDB reviews, the latter of which generally contained more complicated and niche vocabulary. One solution to this problem would be to have a larger dataset to create our feature vectors from, to reduce the number of words absent from the vocabulary. We could explore looking at using feature representations trained on larger datasets in Part 2.

1E : Performance On Test Set

The AUROC score of our final model on the test set was 0.890. This value is slightly better than our mean AUROC for 5-fold cross validation which was 0.874. This makes sense because when performing 5-fold cross validation, the model is only trained on 80% of the training set at a time while the other 20% is held out for validation. However, after we select the correct hyperparameters we then refit the model using 100% of our training set, and this model is what we use to make predictions on the test set. Therefore, it makes sense that when trained with more data, the model performs slightly better.

Problem 2

2A : Feature Representation Description

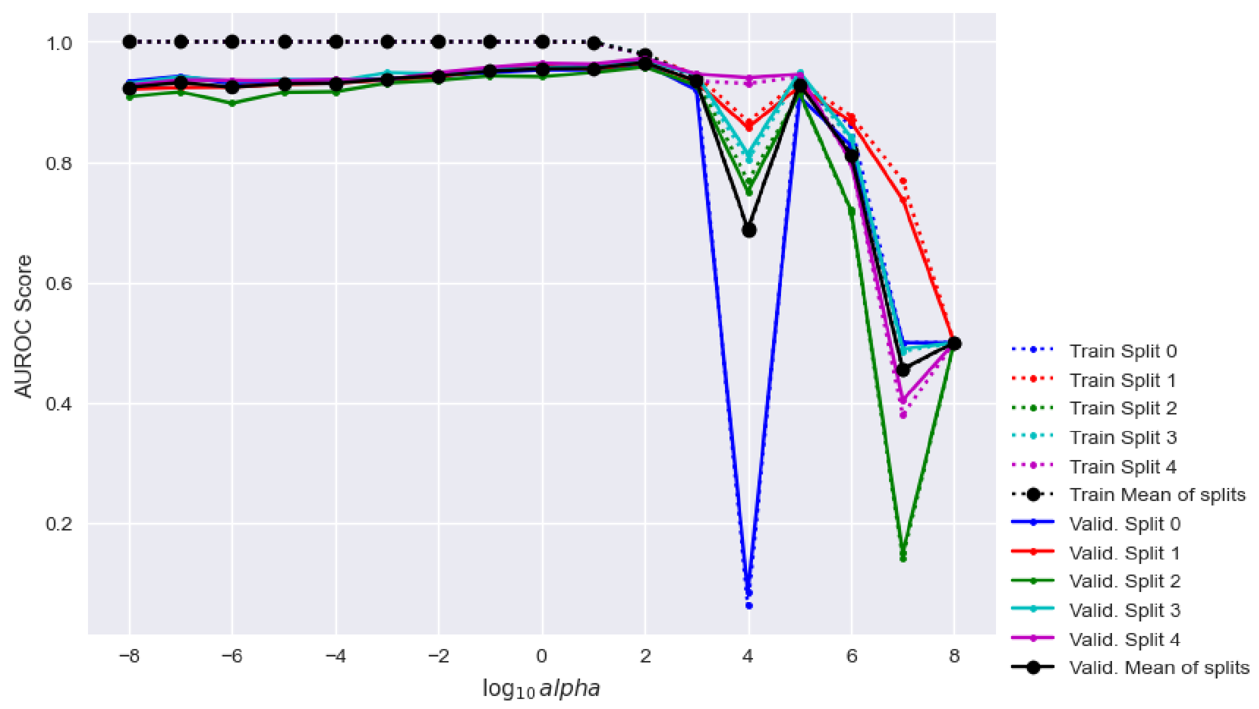
We decided to use the BERT embeddings of the reviews to represent the features for our model. BERT is a large neural network that has been trained on a dataset of many sentences that can be used to create fixed-length embeddings from a sentence. BERT does this by first tokenizing the given sentence and converting each token (which can be as short as one letter or as long as a whole word) to a corresponding ID in the BERT vocabulary. It also adds in certain special token IDs needed by the model, such as the padding token, which are added on to the end of all sentences that are shorter than the longest sentence provided. This ensures that the vector representations of all our reviews are all of uniform length for use in our classification model. After tokenizing the reviews, the BERT model then extracts contextual embedding values for each token and stores them in a list for each review. In practice, we did not create the review embeddings ourselves. Instead we used the review embeddings provided on Github/Piazza.

2B : Cross Validation (or Equivalent) description

Once again, the performance metric we chose to optimize during cross-validation was the AUROC score because this is the metric used in the leaderboard. We also once again used sklearn's GridSearchCV class in order to perform 5-fold cross validation. This meant that our fold split process was the same as in Problem 1. After performing cross validation on our chosen hyperparameters, we extracted the model with the best AUROC score to use on the test data.

2C : Hyperparameter Selection for Neural Network (MLP - Multilayer Perceptron)

We decided to use an MLP neural network from the sklearn library for our binary classifier. **We chose this classifier because neural networks allow for learning more flexible functions**, which is useful for building a complex model like this. We trained our model using **5 fold cross-validation on our training set with sklearn's GridSearchCV class, with a max_iter value of 10,000 to avoid convergence issues and early_stopping to True to prevent overfitting**. In doing so, we performed hyperparameter selection on 3 parameters: the activation function ('relu', 'sigmoid' or 'tanh') [**which differentiates how the model scales results between layers**], the solver ('adam' or 'lbfgs') [**which differentiates between a gradient-descent based algorithm which performs line search for step size and a quasi-Newton method of finding weights**], and the alpha value or L2 regularization penalty [**which discourages against large weights**] (17 samples on a logspace from 10^{-8} to 10^{8}). Below is a figure showing the AUROC scores for several values of the hyperparameter alpha (when activation is 'relu' and solver is 'lbfgs'):



This figure shows **underfitting to the data as alpha becomes too large** (train and validation AUROC score plummet) and **overfitting to the data when alpha is too small** (although the AUROC score for the train splits hovers around 1.0 for $\log_{10}(alpha) < 0$, the **smaller AUROC score on validation for extremely small alpha compared to less small alpha means we're overfitting on the extremely small values of alpha**). From this graph **the optimal value (sweet spot) for alpha is found where validation AUROC score is maximized at alpha = 100.0**. The results from **the other two hyperparameters are less decisive** although we found **marginal improvements when using 'relu' as the activation function and 'lbfgs' as our solver.**

2D : Error Analysis

False Positives	False Negatives
"The loudspeaker option is great, the bumpers with the lights is very ... appealing."	"Waste your money on this game."
"Excellent starter wireless headset"	"I struggle to find anything bad to say about it."
"Which has more depth and character than the man underneath it."	"I have yet to run this new battery below two bars and that's three days without charging."
"The chicken dishes are OK, the beef is like shoe leather."	"The Veggitarian platter is out of this world!"

This classifier does a much better job on double negatives than the first classifier did due to the word order inherently reflected in the BERT representation of the feature vectors. Whereas before the BoW representation threw away word order entirely, the BERT representations have features which correspond to the 1st word in the sentence, the 2nd word in the sentence, etc., and thus convey word order to the classifier. Additionally, it seems that when the sentiment of a sentence differs from its tone (i.e., “waste your money on this game.” and “I struggle to find anything bad to say about it” both have negative tones but a positive sentiment) the classifier struggles. Additionally, it struggles to identify sarcasm or nuance in reviews like “Excellent starter wireless headset” and “The loudspeaker option is great, the bumpers with the lights is very ... appealing.” in which the reviews may seem to have a positive sentiment perceived out of context but actually have negative sentiment.

2E - Report Parameters on Test Set via Leaderboard

Our best model used the BERT embeddings as feature vectors for the sentence and sklearn’s MLP classifier with hyperparameters ‘lbfgs’ solver, ‘relu’ activation, and alpha of 73.754. Our performance on the test set is better than problem 1, likely due to the better feature representation with BERT than BoW (BERT conveys word order and word meaning in its representation of text while BoW does not), and a more flexible model with an MLP classifier instead of a logistic regression classifier. Our best model achieved an AUROC score of 0.965 on the test set after achieving a mean test score of 0.966 across 5 folds.