Liam Strand and Charlotte Versavel
CS135: Machine Learning
19 October, 2023

Project A Report

Problem 1:

1A : Bag-of-Words Design Decision Description

We chose to use SciKit-Learn's CountVectorizer to represent the features in our dataset. We cleaned our data by **removing all punctuation** before passing it into the CountVector and also used the built-in **"strip_accents**" and "lower" parameters to convert accented characters into their similar ascii character components and **make all letters lowercase**.

To exclude some words, we used the CountVectorizer's "stop_words" parameter with the setting "english", but found that our model performed better when stop_words was set to none, so we choose not to include any stop words. We did play with excluding words that appeared too commonly and also very infrequently using the CountVectorizer's "**min_df**" and "max_df" parameters. We found that excluding uncommon words had a detrimental effect on model performance. We performed a GridSearch using "**max_df**" as a hyperparameter and found that **excluding words** that appeared in **more than 19.8 of reviews** resulted in the best performance. After excluding those most common words, **our vocabulary contained 4677 words.** It's worth noting here that this means the optimal vocabulary only **excludes the 3 most common words**. Our approach counts the occurrences of words in the vocabulary set to form the feature vector, and ignores words not in the vocabulary.

1B : Cross Validation Design Description

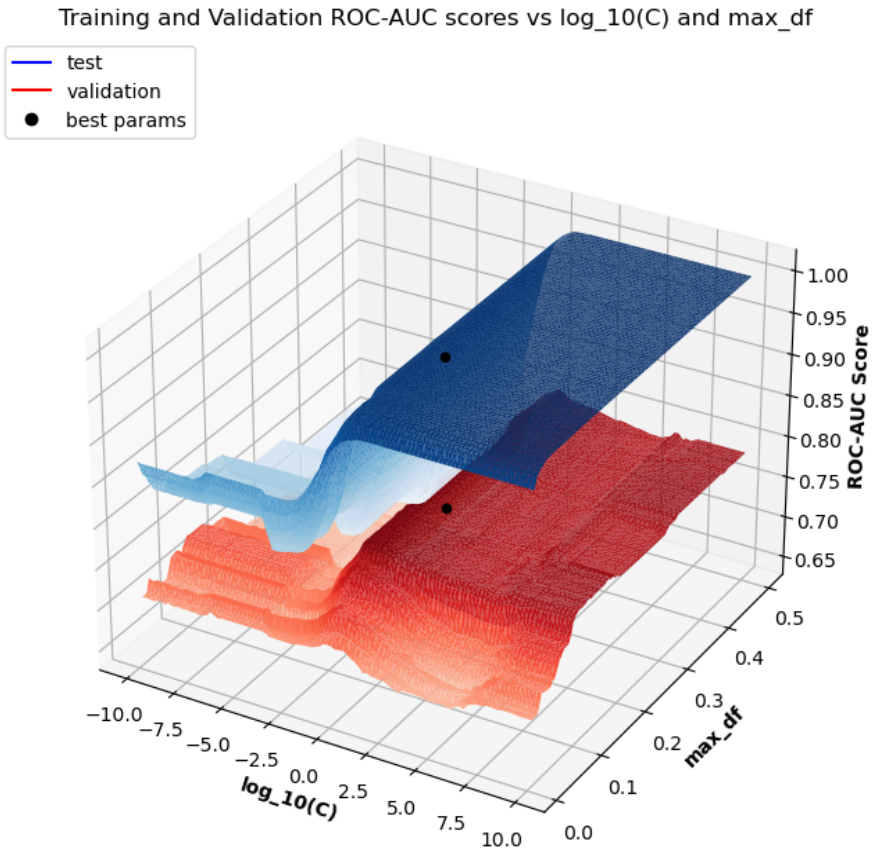
We used the SKLearn **GridSearchCV** to perform our cross validation and hyperparameter searches, **optimizing for the best ROC-AUC score**; meaning that we wanted to select the hyperparameters that maximized the area under a ROC curve. We chose to use GridSearchCV, even though the course favors RandomSearchCV, because we weren't favoring efficiency, and wanted to have full control and the ability to check every combination of parameters that we had specified. SKLearn's GridSearchCV tools automatically perform **5-fold cross validation**, where each fold is ⅕ the size of the training set, so **each fold contains approximately 480 reviews**. The folds are split by SKLearn's StratifiedKFold. Reviews are **shuffled** before hyperparameter selection starts, but not during selection or between hyperparameter trials. We do this so that each hyperparameter candidate operates on the same splits, but all splits have a balanced mixture of positive and negative documents. Both searching strategies eventually return values for the hyperparameters that resulted in the highest validation score. Those values are then plugged into the final model that is trained on the entire training dataset.

1C : Hyperparameter Selection for Logistic Regression Classifier

The advantage of using a LogisticRegression classifier is that it trains extremely quickly, and performs well for how simple it is. We needed to increase the number of iterations to allow the

model to converge using the default step size in some cases. We did not stop the model early and instead selected LogisticRegression's L2 penalty to avoid overfitting.

We chose two hyperparameters to investigate: the regularization strength C of the LogisticRegression classifier, and the upper cutoff max_df of the CountVectorizer vocabulary. A lower value of C penalizes large weights in the classifier, which are an indicator of overfitting. We tested **100 values of C distributed logarithmically between 10^-10 and 10^10**. A logarithmic distribution is a good choice here because the barrier between over and underfitting is likely to appear near C=1, so we want lots of density there, but we also want to test very high and very low values of C. Lower values of max_df indicate that only less frequent words will be considered in the bag-of-words. We tested **100 values of max_df distributed linearly between 0.01 and 0.50**. This distribution makes sense because we wanted to span a wide range of values uniformly. We tested every combination of these hyperparameters in a GridSearch, resulting in 10,000 candidate hyperparameters. The results of this search are summarized in the figure below. The best hyperparameters found were **C=1.262 and max_df=0.198**. Based on the figure, it appears that there is a clear overfitting that occurs after C gets greater than approximately 10, and an underfitting when C is less than 0.01. There are many regions of hyperparameter values that result in a reasonable ROC-AUC score on the validation data, so the evidence for the selected parameters is **not decisive**.



1D : Analysis of Predictions for the Best Classifier

Assuming a threshold of 0.5, the following are representative examples of false positives and false negatives of held out examples.

Representative examples of False Positives:	
Note:	**Count: 43**
Negation	I was not happy with this item. (**Amazon)**
Long Review	My phone sounded OK (not great - OK), but my wife's phone was almost totally unintelligible, she couldn't understand a word being said on it. (**Amazon)**
Short Review, Negation	The service here is fair at best. (**Yelp)**
Representative examples of False Negatives:	
Note:	**Count: 55**
Very short review	Magical Help (**Amazon**)
Very long review	It's as continuously beautiful to look at as a Bertolucci, but the relationships here are more convincing and the narrative more engaging than some of that master's work. (**IMDB**)
Negation	I did not have any problem with this item, and would order it again if needed (**Amazon**)

When looking at predictions for a single fold of testing data, the model makes many mistakes, but tends to perform **best** on sentences of **medium length**, and really seems to struggle with (give **incorrect** results for) **long reviews**, and **very short reviews** (less than 8 words). Additionally, the model does **best on IMDB reviews**, and **worst on Amazon reviews**. This is likely because the Amazon reviews are most varied in their content and word choice. It scored **worse than average on sentences with negation words**, as shown in Figure 1D. When the model encountered reviews that contained words that were generally positive, it usually classified them positively, even if the words were being negated (ie “The service here is fair at best”). Similarly, if a review used words that were generally negative, but negated them (ie “I did not have any problem…”) it would classify a positive review as negative.

1E : Report Performance on Test Set via Leaderboard

Our ultimate test performance was an AUROC score of **86.5%** on the official test dataset. This result surprised us, because based on our scores in cross validation, the model consistently maxed out at around an average validation score of **80.02%** across all folds, so seeing a value much greater than this was unexpected.

Problem 2:

2A : Feature Representation description

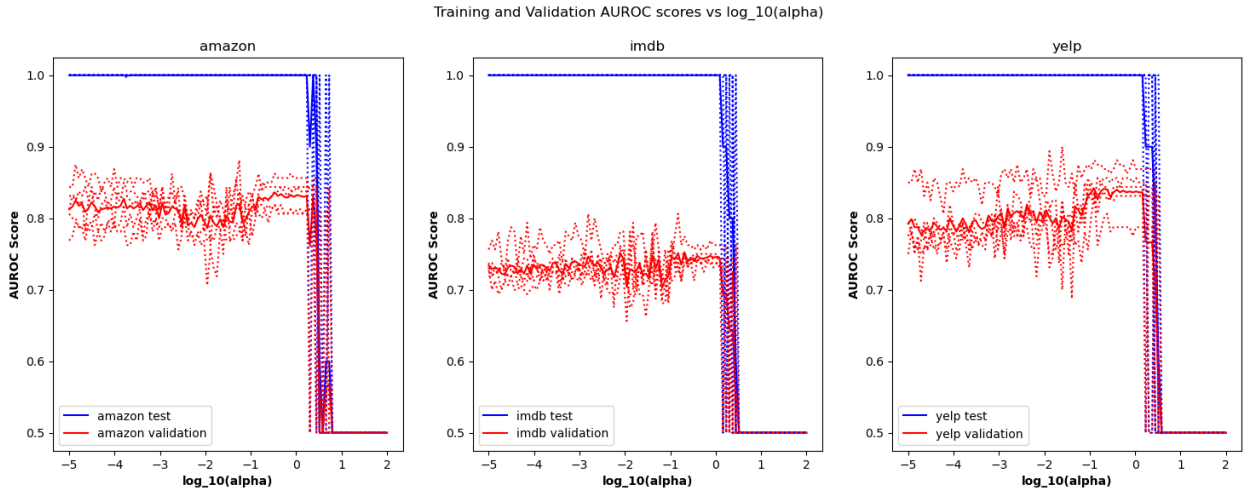
Our strategy expands on the basic “bag-of-words” in a few ways. First, we upgrade from a CountVectorizer to a TfidfVectorizer, which essentially decreases the weight of common tokens. We also use 2 word bigrams in addition to 1 word monograms to form the vocabulary. In theory we would expect that this would help with negations and similar tricky word formations. This made the vocabulary much larger than with the “bag-of-words”. Lastly, we trained (and did hyperparameter searches for) three different models, one on each data set. This allowed the models to be tailored more specifically to the classification task.

2B : Cross Validation (or Equivalent) description

For the three MLPClassifier models we trained, we again used SKLearn **GridSearchCV** to perform our cross validation and hyperparameter searches for the same reason as above, and **optimized for the best ROC-AUC score.** For each of the three models, we performed **5-fold cross validation**, where each fold was ⅕ the size of the corresponding training set and **contained approximately 480 reviews.** Reviews were **shuffled** in the same way as above. The hyperparameter values that resulted in the highest validation score were again used to create a final model that is trained on the entire training dataset.

2C : Classifier description with Hyperparameter search

We chose two hyperparameters to search. First we examined **the shape of the MLPClassifier**. We tried models of various shapes to see which designs resulted in better performance, and which only increased training time without improving prediction accuracy. For example, we tried tall and narrow models, with a single layer of 1000 perceptrons, and short and long models, with 10 layers of 20 perceptrons. We found that generally, cone-shaped models performed well for us, so **we selected a cone of 100 nodes, 50 nodes, and then 25 nodes**, because it generally yielded predictions that did not overfit and trained reasonably quickly. Our selection process involved a few uses of GridSearchCV, first testing **the number of nodes in a single layer in a logarithmic scale ([10], [100], [1000], [10000])**, then testing **the number of layers on a linear scale ([20], [20, 20], [20, 20, 20])**. Once we selected the shape of our model [100, 50, 25], we performed a GridSearchCV on **the regularization parameter alpha at 100 points along a logarithmic scale from 10^-5 to 10^2**. These searches were performed on all three models, one for each of the data sources (amazon, imdb, and yelp). The model performed best on the holdout validation data at **amazon_alpha=0.335, imdb_alpha=0.206, and yelp_alpha=0.126**. Based on the figures, it is clear that overfitting occurs when alpha is less than 1, and underfitting occurs when alpha is greater than 10 across all three datasets. Based on the figure, the evidence for this hyperparameter is **not decisive**. There is a variance on the performance of the model on the validation set between folds, and that variance is greater than the variance in the performance across candidate hyperparameter values.



2D : Error analysis

Representative examples of False Positives:		
<u>Amazon Model:</u>	**Count: 11**	**Review**
Short negative review		Very Displeased.
Blatantly Negative		I find this inexcusable and so will probably be returning this phone and perhaps changing carriers.
Negative review with positive words		The loudspeaker option is great, the bumpers with the lights are very ... appealing.
<u>IMDb Model:</u>	**Count: 17**	**Review**
Creative/Unique Language		In fact, this stinker smells like a direct-to-video release.
Short review		It was so BORING!
Negation		The casting is also horrible, cause all you see is a really really BAD Actors, period.
<u>Yelp Model:</u>	**Count: 17**	**Review**
Blatantly Negative		The server was very negligent of our needs and made us feel very unwelcome... I would not suggest this place!
Blatantly Negative		Unfortunately, it only set us up for disapppointment with our entrees.
Has some positive words		The service here is fair at best.
Representative examples of False Negatives:		
<u>Amazon Model:</u>	**Count: 16**	**Review**

Positive Review with negative language		It quit working after I'd used it for about 18 months, so I just purchased another one because this is the best headset I've ever owned.
Blatantly positive review		I was very excited to get this headset because I thought it was really cute.
Blatantly positive review		They keep getting better and better (this is my third one and I've had numerous Palms too).
IMDb Model:	**Count: 27**	**Review**
Blatantly positive		All things considered, a job very well done.
Positive review with negative words		I struggle to find anything bad to say about it.
Blatantly positive		This movie is so awesome!
Yelp Model:	**Count: 23**	**Review**
Blatantly positive		I didn't know pulled pork could be soooo delicious.
Blatantly positive, mentions seafood		I got to enjoy the seafood salad, with a fabulous vinegrette.
Blatantly positive, mentions seafood		Best fish I've ever had in my life!

The trends for neural networks don't seem to be as clear. The models generally didn't work better on reviews of a specific length. They generally did better on sentences with negation. Confusingly, they often incorrectly classified reviews that were blatantly positive and negative. This was true of all three neural networks. Overall, the Amazon model performed the best, though it struggled with longer reviews. Unlike the BoW representation, this model usually succeeded on shorter reviews. The IMDb model mainly struggled to classify reviews that had very uncommon or flowerful language, and had the most false negatives out of all three models. The Yelp model also had many false negatives, and the main theme for these were that they were on reviews that to an observer, were blatantly positive. Many of these reviews involved mentions of seafood, when compared to reviews that were correctly classified. Overall, the neural network results are less straightforward, and more difficult to simply explain.

2E : Report Performance on Test Set via Leaderboard

Our ultimate test performance was an AUROC score of **89.8%** on the official test dataset. This result is better than the result obtained by the BoW representation. This is because of the variety of changes we made to the model. First, the increase of word combinations created by including 2-word combos effectively creates more features in the feature vector of words. Next, the neural network is more powerful and able to better pull apart small distinctions. Finally, we trained each neural network on a more specific dataset that would be more relevant to the training examples that could be provided.