Classifying Sentiment

CS135 - Intro to ML, Fall 2024, Tufts University

Mark Martirosian, Eddy Zhang

1A: Bag-of-Words Design Decision Description

In our model, the vocabulary consists of 18,107 words. We chose not to exclude rare words, as we observed that increasing the vocabulary size generally leads to better prediction results. Given that our dataset is not exceptionally large, the inclusion of all words did not significantly affect the model's efficiency. Similarly, we opted not to exclude common words, for the same reasons, as they can still contribute valuable information for prediction. We used the CountVectorizer function, which by default retains the count values of each word rather than using binary indicators for word presence. The CountVectorizer also automatically handles data cleaning, such as removing punctuation and converting text to lowercase using the lowercase=True parameter. This ensures that words like "Good" and "good" are treated the same during tokenization. Numbers, however, are treated like any other token and are included unless specific preprocessing steps are taken to remove them. The final vocabulary set is determined by the words present in the training data. We did not exclude any words, as our trials indicated that including all words improved the accuracy of the predictions. For handling out-of-vocabulary (OOV) words, which are words appearing in the test set but not in the training set, the CountVectorizer simply ignores them. These words do not contribute to the feature vector and have no impact on the model's predictions.

1B: Cross Validation Design Description

To train the classifier and tune hyperparameters in Problem 1, we employed 10-fold cross-validation with 20 different hyperparameter configurations. Cross-validation was chosen to ensure the model generalizes well and avoids overfitting. The data was split into 10 equal parts, with 8 folds used for training and 2 folds used for validation in each iteration. This approach rotated through all folds, ensuring that each part of the data served as the validation set once. By doing this, the model was trained and validated on various subsets of the data, reducing the risk of overfitting to any particular subset. The performance metric we optimized during the hyperparameter tuning process was the Area Under the Receiver Operating Characteristic Curve (AUROC). AUROC was chosen because it effectively measures the model's ability to distinguish between positive and negative classes across all classification thresholds. We used scikit-learn's GridSearchCV for both cross-validation and hyperparameter tuning. This off-the-shelf tool efficiently evaluated 20 different configurations of hyperparameters for the Logistic Regression model, specifically tuning the regularization strength (C) and the type of penalty (L2). The C-values tested ranged from 10^-6 to 10^6, which allowed us to explore a wide range of model complexities and prevent overfitting. After cross-validation identified the best hyperparameter configuration, we refitted the model on the entire training set using this optimal configuration. The final model was then applied to the test set, ensuring that all available data was used in training and that the model benefited from the most effective hyperparameters identified during the cross-validation process. This systematic approach ensured robustness and prevented overfitting, yielding reliable results on unseen test data.

1C: Hyperparameter Selection for Logistic Regression Classifier

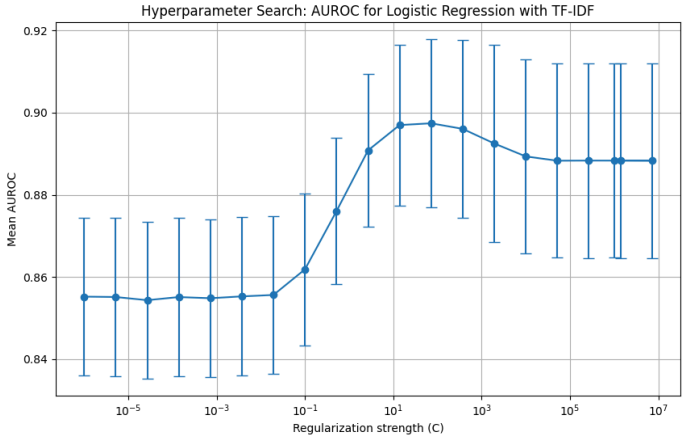

Figure 1C: Mean AUROC vs Regularization Strength

For hyperparameter tuning, we focused on three main parameters: regularization strength (C), regularization penalty, and the solver used for optimization. The regularization strength (C) controls the inverse of the regularization effect. Smaller values of C apply stronger regularization, helping prevent overfitting, while larger values allow the model more flexibility in fitting the training data. We tested 20 log-spaced values for C, ranging from 10^-6 to 10^6, ensuring thorough exploration of regularization levels and their impact on model complexity. For the regularization penalty, we used only L2 regularization, which penalizes large coefficient values, preventing overfitting. L2 is widely supported across solvers and is effective for our dataset's structure.

We also tuned the solver used for optimization. Two solvers, liblinear and l-bfgs, were tested to determine which worked better with our dataset. liblinear is efficient for small datasets, while l-bfgs handles larger datasets more efficiently. By searching across both solvers, we ensured the model could perform optimally based on the dataset's size and complexity. The hyperparameter search was executed using GridSearchCV, systematically testing the 20 C values across both solvers to identify the best combination. The results were evaluated using 10-fold cross-validation and ROC-AUC as the performance metric to assess how well the model discriminates between positive and negative classes. This approach ensured that we identified the best hyperparameter configuration, C = 2.683 and the L2 penalty, to prevent overfitting while maintaining optimal performance.

1D: Analysis of Predictions for the Best Classifier

Below are some representative examples of false positives and false negatives for our chosen best classifier from 1C (7 FP, 7 FN)

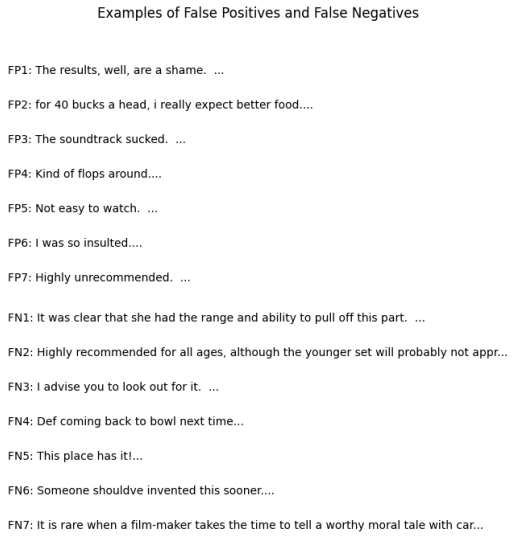

Figure 1D-1: examples of FP and FN

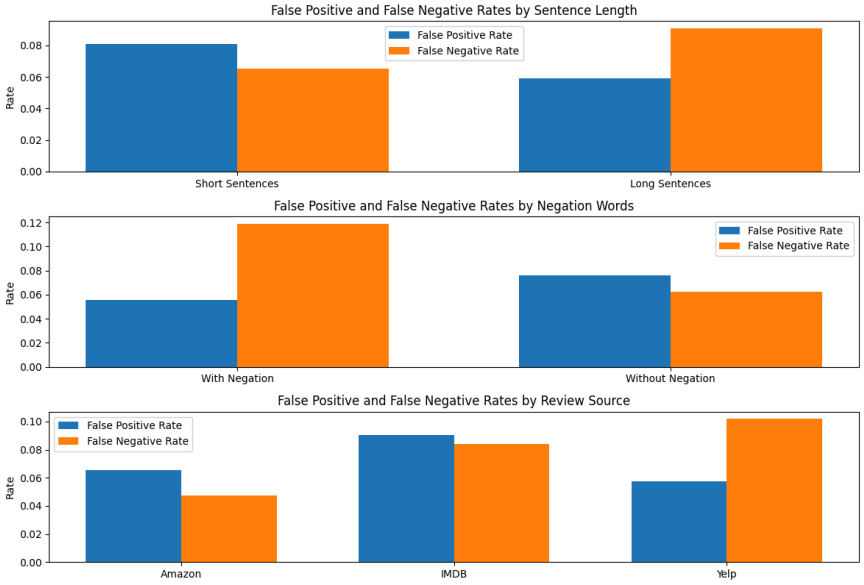

Figure 1D-2: FP & FN rates by different factors

After observing the performance and FP/FN samples of our model, we found these common patterns: In terms of review sources, the model performs best on Amazon reviews with a lower FP rate (6.55%) and FN rate (4.76%) compared to IMDB and Yelp.IMDB reviews have the highest FP (9.03%) and FN (8.39%) rates, suggesting the model struggles more with this source. Yelp reviews show a relatively low FP rate (5.73%) but the highest FN rate (10.19%), indicating it tends to miss positive examples more often. For sentence length, the model has a higher FP rate for shorter sentences (8.08%) compared to longer sentences (5.91%), meaning it misclassified more short negative sentences as positive. For FN rates, the model performs worse on long sentences (9.09%), suggesting difficulty in correctly identifying positive sentiment in longer text.

Moreover, sentences containing negation words have a lower FP rate (5.56%) but a much higher FN rate (11.90%). This implies that while the model is cautious about misclassifying negative sentences with negation as positive, it often fails to recognize positive sentiment in these sentences.

For sentences without negation, FP and FN rates are more balanced, but the FP rate is slightly higher (7.63%) while the FN rate is lower (6.21%).

1E: Report Performance on Test Set via Leaderboard

In the final test set evaluation, the model achieved an AUROC of 0.8848. This is slightly lower than the AUROC scores observed during cross-validation, where the mean AUROC was approximately 0.8987, with individual fold scores ranging from 0.87 to 0.93. Additionally, the validation set AUROC was higher, at 0.9195. The difference in performance between the test set and the cross-validation/validation sets can be attributed to several factors. One possible reason is that the test set data may be inherently different from the training and validation sets, reflecting a distribution shift or a wider variety of examples. Additionally, cross-validation performance tends to be an optimistic estimate because it averages across different folds of the training data, which may not always generalize to unseen test data. This discrepancy highlights the importance of having a distinct test set for a more realistic assessment of model generalization.

```
Validation AUROC: 0.9195186747929363
Cross-validation AUROC scores: [0.91134021 0.9032013  0.8712968  0.87760417 0.88932292 0.87706163
 0.91200087 0.89051649 0.93088108 0.92458767]
Mean AUROC: 0.8987813133496865
```

2A: Feature Representation description

For feature representation, we combined two methods: traditional TF-IDF vectorization for the text and BERT embeddings for capturing more nuanced contextual information. The text data was processed using TfidfVectorizer, which transforms the text into weighted feature vectors based on term frequency-inverse document frequency. We used an n-gram range of (1, 2) to capture both unigrams and bigrams in the text, providing a balance between simple word frequency and slightly more complex word pairings. In addition to this, we generated BERT embeddings to capture deep semantic relationships in the text. The embeddings were obtained using a pre-trained BERT model, and we extracted the CLS token to represent the entire input sequence. These embeddings were then scaled using StandardScaler to ensure that they were on the same scale as the TF-IDF features. By combining these two representations, we aimed to leverage both shallow, surface-level text features and deeper, context-aware information for improved classification performance.

2B: Cross Validation (or Equivalent) description

These parameters were mostly the same as 1b, with AUROC as the measure, 10 fold cross-validation split into 8 for training and 2 for validation, and GridSearchCV. The only difference was the parameters, since the classifier used was Random Forest instead of Logistic Regression. There were 99 different configurations that were analyzed, specifically looking at n_estimators and max_depth.

2C: Classifier description with Hyperparameter search

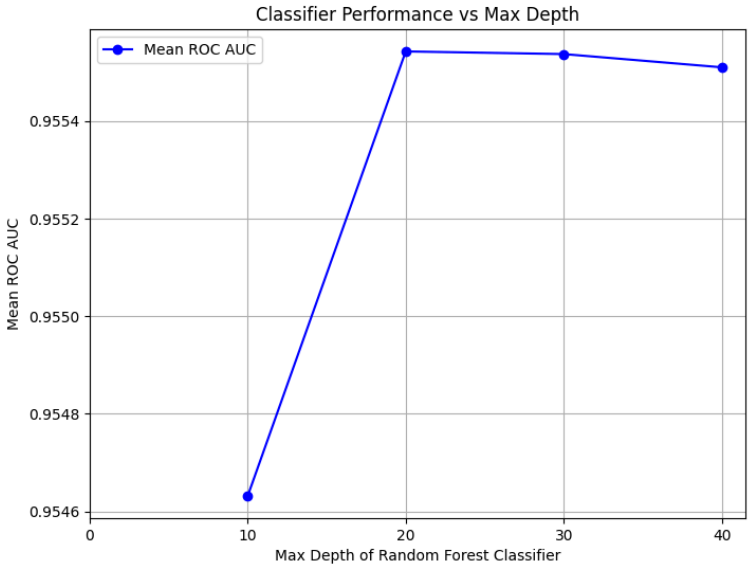

Figure 2C: Classifier Performance vs Max Depth

For classification, we selected the **Random Forest Classifier** due to its strong performance with mixed data types and its resistance to overfitting. Random Forest aggregates predictions from multiple decision trees, each trained on different data

subsets, which improves generalization and reduces variance. The grid search focused on two key hyperparameters: **n_estimators** (the number of trees in the forest) and **max_depth** (the maximum depth of each tree).

We tested **n_estimators** values from **300 to 800** in increments of 50. After tuning, we found that **700 trees** provided the best performance, offering a good balance between computational efficiency and predictive accuracy. For **max_depth**, we tested values from **0 to 40**, also in increments of 5. The model performed best with a **max_depth of 20**, balancing complexity and overfitting risks.

The hyperparameter tuning was performed using **GridSearchCV** with **10-fold cross-validation** to ensure the model generalized well across various data subsets. The best model was chosen based on the **ROC-AUC score**, which effectively captures the model's ability to distinguish between positive and negative classes across different classification thresholds. The selected hyperparameters resulted in a model that achieved a strong balance between overfitting and underfitting, ensuring high performance on unseen data. **The final model was evaluated using AUROC** on the validation set, confirming the model's robustness and discriminatory power.

2D: Error analysis

Below are some representative examples of false positives and false negatives for our model from 2C.

False Positives (Predicted Positive, Actually Negative)

Example 1: 'I got home to see the driest damn wings ever!'
Predicted Probability: 0.66 | True Label: 0

Example 2: 'The scallop dish is quite appalling for value as well.'
Predicted Probability: 0.53 | True Label: 0

Example 3: 'too bad cause I know it's family owned, I really wanted to like this place.'
Predicted Probability: 0.63 | True Label: 0

False Negatives (Predicted Negative, Actually Positive)

Example 1: 'So good I am going to have to review this place twice - once hereas a tribute to the place and once as a tribute to an event held here last night.'
Predicted Probability: 0.50 | True Label: 1

Example 2: '(The bathroom is just next door and very nice.)'
Predicted Probability: 0.48 | True Label: 1

Example 3: 'It was just not a fun experience.'
Predicted Probability: 0.29 | True Label: 1

In general, the false predictions are not extreme and their probabilities are mostly around (0.50 ± 0.15),

1. False Positives: Positive words or neutral phrases (e.g., “wanted to like”) within overall negative reviews confuse the model. Subtle negativity or mixed emotional tones are hard for the model to detect.
2. False Negatives: Descriptive or neutral-sounding statements (e.g., “the bathroom is nice”) lack strong positive emotional cues, leading the model to miss positive sentiment. Long or complex sentences also make it harder for the model to correctly identify sentiment.

Overall, these errors arise from limitations in detecting subtlety, sarcasm, and complex sentence structures. Fine-tuning the model and including more nuanced training examples could improve performance.

2E: Report Performance on Test Set via Leaderboard

In this second part, the model achieved a test AUROC of 0.95838, which represents a significant improvement over the test performance in Problem 1, where the test AUROC was 0.8848. The validation AUROC of 0.95379 and cross-validation mean AUROC of 0.95593 in Part 2 closely match the test performance, suggesting the model is well-generalized and robust. The improvement in test performance can be attributed to several factors. First, the use of BERT embeddings in combination with TF-IDF provided a richer and more informative feature set, capturing both semantic and lexical properties of the text data. This enhanced representation likely contributed to the model's ability to better differentiate between positive and negative sentiments, as BERT embeddings capture contextual nuances more effectively than the simple Bag-of-Words (BoW) model used in Part 1. Additionally, the hyperparameter tuning for the random forest classifier, with adjustments to the number of estimators and depth, helped optimize the classifier's performance. This careful tuning, along with the use of cross-validation, ensured that the model

was less prone to overfitting, which was possibly a limiting factor in the BoW-based model in Problem 1.

```
Validation AUROC: 0.9537948637807991
Cross-validation AUROC scores: [0.95713511 0.97438958 0.92718394 0.95616319 0.95106337 0.96191406
 0.94542101 0.96191406 0.96148003 0.96267361]
Mean AUROC: 0.9559337967515976
```