AML Midterm Project (CS5785/ORIE5750/ECE5414)

Alex Bean (atb95) Donovan McManus (dpm262) Kamron Vaswani (kv277)

In this project, we contributed equally across all areas. We all participated in brainstorming to drive methodologies and decipher which is best for our data-augmentation processes to develop a robust solution strategy and shared the responsibility of coding. There was no structured way of assigning things, just people picking up and building whenever their schedule permitted. For the write-up, we collectively crafted our insights that allowed us to leverage everyone's strengths and point of view to ensure a cohesive representation of our work.

Abstract

We developed a semi-supervised and supervised learning pipeline for classification using a partially labeled training dataset. In the unsupervised phase, k-means clustering helped analyze sentiment distribution and impute missing sentiment labels, enhancing the dataset's completeness. For supervised learning, we evaluated several models, ultimately selecting multinomial Naive Bayes and softmax regression due to their probabilistic strengths and high F1 scores. Notably, multinomial Naive Bayes performed best when trained on the entire dataset (after imputing labels), achieving an optimal weighted F1 score of 0.8936. Conversely, softmax regression proved most accurate when restricted to only the labeled data, which accounted for 40% of the training set, demonstrating its robustness with high-quality, labeled information. It is important to note that we used Naive Bayes for all of our Kaggle submissions. These findings underscore the effectiveness of probabilistic models like multinomial Naive Bayes and softmax regression in handling complex, semi-supervised datasets, aligning well with our understanding of class material.

Methodologies

Pre-Processing Methods

Aiming to build a semi-supervised learning model for sentiment classification, this project challenged us to creatively leverage both labeled and unlabeled data. Given the inherent ambiguity of semi-supervised learning, our goal was to approach the problem in a unified manner that integrated unsupervised and supervised learning techniques rather than treating them as separate processes. This motivated us as a team to explore the data in depth, especially because our training data was 60% unlabeled. We needed a thorough understanding of the distributions within each segment and how data manipulation would impact results.

Diving into the variety of methods for preprocessing and feature engineering, we began by handling missing values, removing 5 null entries from the training set, 6 from the validation set, and 6 from the test set. Next, we implemented a custom preprocessing function that converted movie review text to lowercase, removed non-ASCII characters, URLs, mentions, punctuation, and stop words, and performed lemmatization. This function was applied to all three datasets (training, validation, and testing) to prepare them for vectorization. Rather than using 'CountVectorizer' with a binary count-based representation, we opted for TF-IDF, which considers term frequency across documents. While the relatively short length of the phrases limited the boost in accuracy, TF-IDF still offered better performance than simple counts and was the optimal choice for our vocabulary, with 'max_features=10000' set to control vocabulary size (explored and determined from our multithreaded function for exploratory purposes — this will be discussed further later).

This naturally led us to utilize PCA to reduce the dimensionality of the training set. However, besides its functionality as an unsupervised learning technique, we tapped into its versatility to help learn the unlabeled data. Furthermore, we tested both GMM and k-means where our F1 scores, which had been promising with labeled training data with our multithread function (discussed later), dropped significantly when we applied these clustering techniques to the unlabeled data. For each cluster, we identified the most common label (the mode of the cluster: 0, 1, 2, 3, 4) and assigned this label to the unlabeled data. We recognized that as we increased our number of clusters, our accuracy improved. However, we often ended up with clusters that only contained unlabeled data (-100). Because we couldn't assign them a label, we deleted these data points. Similarly, GMM struggled to map sentiment values accurately due given the overlapping nature of these classes and GMM's assumption of Gaussian-distributed clusters, leading to misclassifications. Despite its struggles, GMM allowed us to understand the data set even further. For one, the limitations of clustering sentiments labeled as 0 through 4 pushed us to expand our approach by increasing the number of clusters, setting $k = 50$ to capture a broader range of sentiment distributions. The existing five clusters were not effectively capturing the full spectrum of sentiment in our data, as they often forced diverse sentiments into the same category, limiting the granularity of the learning method. By increasing k to 50, we were able to create more detailed clusters, which allowed us to parse the training data at a much finer level. Encouraged by the fact that we could uncover the distributions of these clusters, we further increased k to 1900 (another arbitrary value that is explained in our unsupervised section) to explore even finer distinctions among sentiment clusters.

In the final stages of our preprocessing pipeline, we then applied PCA with the number of principal components of $n = 300$. This value was provided as the optimal balance between dimensionality reduction and retention of significant variance from our multithreaded PCA and k-means function. Furthermore, tuning the data to 300 components allowed us to maintain the integrity of the data while eliminating noise to enhance our compatibility.

Unsupervised Learning

Our unsupervised method began by using a combination of PCA and k-means clustering to classify our text data based on sentiment. We first vectorized the text using TF-IDF instead of 'CountVectorizer,' as TF-IDF better considers term frequency across documents, which proved beneficial for capturing more meaningful patterns in our data. The 'pca_and_kmeans' function applies PCA and iteratively tests different numbers of clusters with k-means, selecting the number that yields the highest accuracy in predicting sentiments. Coming together, performing k-means on the entire training data so that each cluster was composed of both labeled and unlabeled data, we found that the larger our k value, the more clusters there were with only unlabeled data and the more data we had to delete. This clustering-based accuracy is calculated by mapping each k-means cluster to the most common sentiment within that cluster and comparing predicted and actual sentiments. To improve efficiency, 'test_pca_and_kmeans_multithreaded' leverages multithreading to test various PCA component counts in parallel, identifying the best combination of PCA components and clusters. The optimal configuration found used 1900 clusters, balancing dimensional reduction and clustering accuracy effectively. This increased granularity allowed us to impute unlabeled sentiment values (initially marked as -100) by assigning them the most frequent sentiment within their respective clusters, significantly enhancing our F1 score. During this process, we encountered clusters that consisted solely of unlabeled sentiments, leading us to disregard these clusters. Only 1652 data points (or 4.73% of the unlabeled data) were excluded as a result. To avoid deleting too much data, we decided to set an arbitrary limit of how much data we could delete. We decided that we would not delete more than 5% of the unlabled data. Utilizing this constraint, we ended with k = 1900.

Supervised Learning

In our supervised step, we implemented a setup to evaluate the best model and optimal hyperparameters for classification. We structured the workflow so that each model (denoted as M) and each feature per model was assigned to an individual thread or "job," which was then dispatched for parallel processing. To ensure accuracy, we required that all threads and jobs were completed before aggregating results. This setup avoided duplication of data points in arrays, which was essential to maintain the integrity of our data mappings and enhance the reliability of our model evaluations.

We defined a helper function to train each model with specific parameters, iterating through configurations to maximize the weighted F1 score. Our experiments included models such as softmax regression and multinomial Naive Bayes, tested with parameters like 'n_values,' 'max_iterations', and 'max_features.' We explored, but excluded, configurations like L1 regularization, logistic regression, and no regularization due to consistently low F1 scores. Naive Bayes emerged as particularly effective, leveraging Bayes' theorem, which was well-suited to our discrete, labeled dataset and efficiently handled missing data. Given its robust performance, Naive Bayes achieved a high F1 score of 0.8898. We concluded that adding text-based embedding methods would not contribute additional predictive value, as our discrete features were well represented through vectorized transformations. Based on these results, we trained our final model using the entire training set, vectorized the 'Phrase' column, and saved the test predictions in 'test_predictions.csv' for easy export and further analysis.

Ultimately, the F1-scores favored probabilistic approaches like multinomial Naive Bayes and softmax regression, which offered superior performance. Their probabilistic foundations provided better generalization and more reliable predictions for our data structure. KNN's distance-based approach, lacking a learning phase and probabilistic interpretation, was less effective for our dataset's complexity. While multinomial Naive Bayes and softmax Regression were chosen as the primary models due to their strengths, we figured that KNN could be considered for smaller, more balanced datasets where its proximity-based classification would be more applicable.

Results

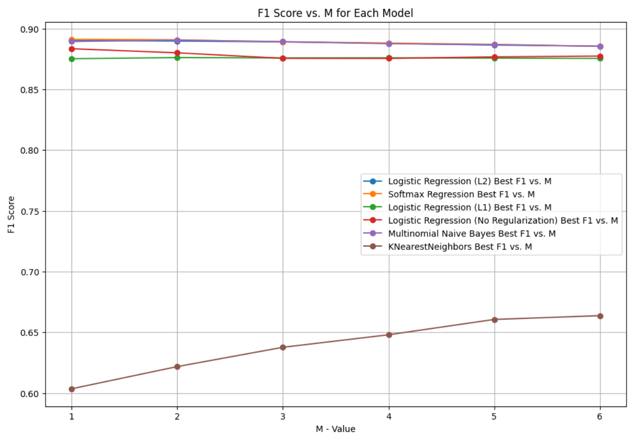

Figure 1. Performance with only labeled data.

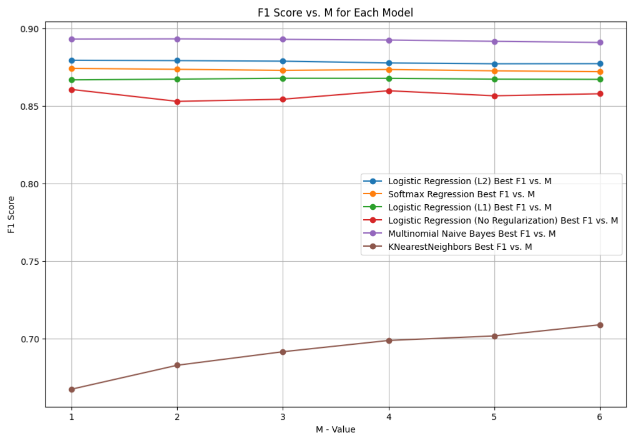

Figure 2. Final performance with all data.

Model (With Labeled & Unlabeled Data)	F1 Score
Multinomial Naive Bayes	0.893099
Logistic Regression (L2)	0.879311
Softmax Regression	0.874064
Logistic Regression (L1)	0.867717
Logistic Regression (No Regularization)	0.860497
K Nearest-Neighbors	0.708885

Note: All models were trained after TF-IDF vectorization, PCA, and k-means clustering.

Discussion

Only Labeled Data

We experimented with several machine learning models on labeled data, including L2-regularized logistic regression, softmax regression, and multinomial Naive Bayes. Each model's performance was assessed at progressively higher levels of max_features, starting from 1000 up to 20000, to evaluate scalability and feature coverage. Softmax regression achieved the highest F1 score of 0.8912 at M=1, demonstrating compatibility with a lower complexity level. Multinomial Naive Bayes followed closely, reaching a peak F1 score of 0.8906 at M=2, showing its effectiveness with a moderate feature count. Logistic regression with L2 regularization obtained a maximum F1 score of 0.8793, slightly trailing behind Naive Bayes and softmax regression. Logistic regression with no regularization and L1 regularization achieved lower F1 scores, peaking at 0.8835 and 0.8762, respectively. K-Nearest Neighbors (KNN) performed the lowest, with an F1 score of 0.6637 at M=6, indicating its limitations in handling high-dimensional data effectively. While softmax regression performed best at lower complexity (M=1), multinomial Naive Bayes demonstrated robust performance across configurations, making it an effective choice for balancing model complexity and feature utilization in labeled data analysis.

Data Including Unlabeled

We experimented with various machine learning models on labeled data, including L2-regularized logistic regression, softmax regression, and multinomial Naive Bayes, adjusting the max_features parameter incrementally from 1000 to 20000 to assess scalability and feature coverage. Multinomial Naive Bayes consistently achieved the highest F1 scores, reaching a peak of 0.8931, which confirmed its robustness across configurations. Logistic regression with L2 regularization followed with an optimal F1 score of 0.8793, and softmax regression achieved a maximum F1 score of 0.8741. Logistic regression with L1 regularization reached an F1 score of 0.8677, while the unregularized logistic regression model attained an F1 score of 0.8605. K-Nearest Neighbors (KNN) performed the worst, with a peak F1 score of 0.7089, reflecting its limitations with high-dimensional data in this context. Modifications in the feature count had minimal impact on the performance of multinomial Naive Bayes, highlighting its ability to perform efficiently with fewer features. Ultimately, multinomial Naive Bayes emerged as the most effective model, balancing feature utilization and model complexity, making it the optimal choice for labeled data analysis.