CS 5785 - Applied Machine Learning
Midterm Project Writeup

Matthew Maitland
Cornell Tech
New York, NY
mjm638@cornell.edu

Shreeya Indap
Cornell Tech
New York, NY
si223@cornell.edu

Sofia Beyerlein
Cornell Tech
New York, NY
sb2669@cornell.edu

Abstract

In this project, we explore both supervised- and unsupervised-learning techniques for multi class sentiment classification by augmenting a dataset with some labeled samples, and some unlabeled samples. Our training data consists of two columns and roughly 60,000 rows. The first column, 'Phrase', contains English texts that are written reviews of movies. The other column is 'Sentiment', which is the label for the text data. This column takes on integer values between 0 and 4, where zero is a bad sentiment and four is a good sentiment. This mimics a 5-Star movie rating system. Any entries that are missing their labels were initialized with a -100 rating.

To address the challenge of incomplete labels, we implement and explore methods that assign labels to the unlabeled data, then use this augmented dataset to train a supervised model. We explored both a Gaussian Mixture Model and an iterative application of Logistic Regression in order to impute the missing labels, and we used a multi-class logistic regression for the data classification. Our final approach improves classification performance, which we evaluate using both accuracy and weighted F1-score on a validation set. The results indicate that our techniques can effectively support sentiment analysis tasks with incomplete data.

We will see that our preprocessing techniques allowed us to perform fairly well without using any additional data. Unsupervised methods were unsuccessful in improving our results, but using an iterative approach and probabilistic aspects of supervised methods were able to improve our model.

1. Methods

1.1. Preprocessing

In our preprocessing steps, we were careful to ensure that the training, validation, and testing data all were processed through the same pipeline in order to ensure consistency in our results. After reading all of the data into individual Pandas DataFrames, we first had to ensure that the data did not contain any `NaN`. values. In the training and development set, there were less than 10 `NaN` values. We were confident that due to the size of our datasets, these values would not be of significant importance to our final outputs, so we elected to simply drop these rows. We developed an initial preprocessing pipeline that standardizes the text data by converting all text to lowercase, lemmatizing each word to its root form, and removing punctuation. Additionally, the pipeline expands contractions, removes non-essential ASCII symbols (such as "@"), strips out links (URLs beginning with "http," "www.," etc.), and filters out stopwords to retain only the most relevant words. After testing the initial preprocessing pipeline, we decided to remove additional text elements that could introduce noise, including mathematical symbols and special characters. We also removed any extraneous whitespace, single-character words, and numbers, leaving only alphanumeric words. Each text entry was further standardized by reducing repeated punctuation, such as excessive exclamation marks or question marks, to a single instance. We also expanded some abbreviations, like "u", to "you". These steps were selected in order to capture the information within a text while eliminating unnecessary noise.

Finally, once our text was cleaned, we split our training data into two sets. The first set contained roughly 35,000 unlabeled rows, and the other contained about 25,000 labeled rows.

1.2. Vectorization and Unsupervised Learning for Term Selection:

To enable mathematical processing of text data, we utilized a TF-IDF vectorizer. This transforms the text into numerical vectors by weighing the importance of each term relative to its frequency within a given text and across the entire dataset. This way, common words that might not be as salient are given lower weights, while terms that are more unique to certain texts are given higher weights.

In our implementation, the TF-IDF vectorizer is given `ngram_range=(1, 3)`, so in addition to vectorizing each individual word, it also vectorizes sequences of two and three words. This helps capture some additional context surrounding a single word.

The TF-IDF vectorizer is an unsupervised method, as it identifies and selects the most salient terms in the dataset without using labels.

We fit and vectorize on the labeled portion of our training data, and then only vectorize (without refitting) the unlabeled training data. We also only vectorize the validation and testing data when we use each set in later predictions.

In this step, there are two parameters that need to be tuned to maximize the performance of our models. As mentioned above, our final model vectorizes single words, bi-grams, and tri-grams. After testing our final models, we found the most success with this range, as opposed to using only individual words or individual words and bi-grams.

The other parameter that requires tuning in this step is `max_features`. We initially found success in this model when `max_features=5000`. However, as we increased this value, the accuracy and F1 scores in each of our classifiers improved, and we settled on a final `max_features=50,000`. Increasing this value further could possibly improve our results slightly, but we expect these improvements to be minimal since they would only involve including words, bi-grams, or tri-grams with very low weights. We chose to stop at 50,000 features because the minor improvements beyond this point do not justify the increased computational power required.

Once the vectorizer is built, it selects the top 50,000 terms based on their TF-IDF scores. Any terms not included in this are discarded and not considered by the model when the vectorizer is applied.

1.3. Initial Model - Before Augmentation

Before taking any steps to augment our data, we evaluated the performance of our multi-class logistic regression model using only the provided labeled training data. This step was crucial to validate our later imputation of labels for the unlabeled training data.

Our first step was to initialize the Logistic Regression classifier with `max_iter` set to a sufficiently large enough value to ensure convergence. Setting `max_iter=3000` prevented any convergence issues. We specified a `random_state` to enable reproducibility of our experiments and ensured that `class_weight` was set to `'balanced'`. We chose balanced class weights since the assignment template indicated that all five sentiment class labels appeared with approximately equal frequency.

We trained this initial model on the labeled portion of the training data, using the provided sentiment class labels as the target values.

To evaluate the initial model, we used it to predict sentiment on the development set, which we transformed using the TF-IDF vectorizer at this step. This initial model yielded an accuracy of 0.9366 and an F1 score of 0.9365 on the validation set.

1.4. GMM:

Although we did not use the GMM in our final model, it is still important to note how we tested this unsupervised model for data augmentation. To implement this, we fit a Gaussian Mixture Model on only our vectorized labeled data, using 5 components to represent each of the 5 classes. After predicting on the unlabeled data, we had to map the outputs of the model to actual labels, since unsupervised models do not do this automatically. Once the GMM initialized labels for all of the unlabeled data, we combined the datasets, and predicted on the development set. The best development training scores that we achieved were 0.8426 accuracy, and 0.8438 F1.

1.5. Iterative Confidence-Based Data Augmentation with Logistic Regression

Our initial model performed fairly well without the unlabeled portion of the dataset. However, we were interested in determining if we could impute labels for the unlabeled portion to increase the sample size for the logistic classifier and in turn, bolster its performance.

To achieve this, we implemented an iterative procedure that uses the logistic regressor trained on the labeled portion to classify the unlabeled rows. In each iteration, we added the most confidently classified rows (using `predict_proba` from `sklearn`) to the training set, retrained the model, and repeated the process with a slightly lower confidence threshold.

In each iteration, we decreased the threshold required for an entry to be added to the training set. Through trial and error, we tested a variety of threshold values and quantity of thresholds, ultimately finding the most success using `thresholds = [0.99, 0.98, 0.97, 0.96, 0.95, 0.9, 0.85, 0.8]`.

After all iterations were complete, our final training set consisted of 53374 complete rows. This indicates that we added 28621 new entries of 34948 possible unlabeled entries. Thus we were able to confidently include about 82% of the unlabeled data.

We validated these optimization steps by training a model on a vectorized version of this augmented dataset, which consisted of all original labeled entries and the newly created high-confidence entries, and then predicting on the validation set.

This model slightly improved over the initial model, resulting in an accuracy of 0.9383 and an F1 score of 0.9384.

1.6. Final Model:

Our final step was to predict on the test set. Since we no longer needed to hold out the validation set while optimizing our model, we were able to combine the augmented dataset with the validation set. Once combined, we re-fit our vectorizer on all of the data. Using the new vectorizer, we transformed both the final training data and the test data.

We then retrained the logistic regression classifier on the complete, vectorized training data and made predictions on the vectorized test data. On Kaggle, our highest score was 0.95184, which is approximately a 1.5% improvement over our initial model that did not include any augmented data.

2. Results

Model	Val. Acc.	Val. F1	Kaggle
Initial	0.9366	0.9365	-
GMM	0.8426	0.8438	-
Iterative	0.9383	0.9384	-
Final	-	-	0.95184

Table 1. Summary of Performance Metrics Across Models

2.1. Model Performance

In the above table, we can see that the initial model achieved strong baseline performance, with a validation accuracy of 0.9366 and a validation F1 score of 0.9365. This was achieved using only the labeled portion of the training data, without any additional augmentation.

The GMM-based augmentation, which assigns labels to the unlabeled data using a Gaussian Mixture Model, performed somewhat lower, with scores of around 0.84 for both validation accuracy and F1. The decision to not include the GMM in our final model will be discussed the next section.

Our iterative approach to augment the data yielded a slight improvement over the initial model, reaching a validation accuracy of 0.9383 and a validation F1 score of 0.9384. This approach uses a threshold-based strategy to add high-confidence predictions from the logistic regression model into the training data. The incremental addition of these high-confidence examples helped the classifier generalize slightly better on the validation set, demonstrating the effectiveness of this approach in improving model performance.

Finally, our last model—trained on both the augmented data and the validation set—achieved a Kaggle test accuracy of 0.95184. This model's performance highlights the impact of including more data for training, as well as optimizing the iterative augmentation. This final result marks an improvement of roughly 1.5% over the initial model, indicating the benefits of using the full dataset effectively.

2.2. Unsupervised Learning Ablation

In the results shown in Table 1, we observe that the GMM-based augmentation approach achieved validation scores significantly lower than our baseline logistic regression model, with approximately 0.84 for both accuracy and F1 score. Given that these results were notably worse than the baseline, we chose not to include GMM-augmented data in the final model.

As part of our experimentation with unsupervised methods, we also tried K-means clustering. However, K-means performed even worse than GMM, likely due to its sensitivity to the initial cluster centers and the difficulty in clustering classes with overlapping features. Since K-means clusters data points into hard, non-overlapping groups, it struggled with the subtle distinctions between sentiment classes in this dataset. This challenge in accurately separating fine-grained classes might explain its underperformance.

The lower performance of the GMM approach also likely stems from the difficulty of clustering nuanced, overlapping sentiment classes in an unsupervised manner. Since our sentiment labels range from 0 to 4, with subtle differences between adjacent classes, GMM struggled to accurately distinguish and assign meaningful labels to the unlabeled data points. It is important to note, however, that we had to use a different vectorizer for the GMM due to its significant runtime requirements. As a result, our TF-IDF vectorizer had less features, which likely capped GMM's performance. To validate that the supervised methods always outperformed GMM, we tested the iterative approach using the smaller vectorizer, and confirmed that it still yielded higher accuracy and F1 scores on the validation set.

One of the main reasons the supervised logistic regression model performed so effectively was our optimization of the TF-IDF vectorizer. By including not only single words but also bi-grams and tri-grams, the vectorizer created a much richer representation of the text, capturing additional context and nuances in phrases. This approach significantly increased the amount of informative data fed into the logistic regression model compared to using only single words or simpler vectorization methods. Consequently, the labeled portion of the dataset—approximately 40% of the original training set—provided more than enough data to train a robust model.

Given the effectiveness of our vectorized data, the main challenge we faced was not a lack of data but rather the need for highly accurate labels. Without this, it simply would not make sense to include any additional data. In this context, unsupervised methods like GMM and K-means would only be beneficial if we had extremely high confidence in their label predictions. However, our experiments with GMM and K-means showed that their predicted labels lacked the necessary accuracy. As a result, it made more sense to follow our iterative approach to only augment the training dataset with points that we were highly confident were correct.

This decision to exclude the unsupervised methods reflects the importance of evaluating decisions critically, rather than defaulting to modeling techniques that initially seem well suited without considering other approaches.

3. Discussion

Our findings highlight the strengths and limitations of some different approaches to handling incomplete labels in datasets. The iterative augmentation was our most effective method, as it selectively grows the labeled training set with minimal noise. This outperformed the GMM-based augmentation, which struggled with the nuances of overlapping sentiment classes.

As mentioned above, our success likely stems from the additional use of bi- and tri-grams, allowing the labeled portion of the dataset to be sufficient for a quality model, and therefore reducing the need to rely heavily on imputed labels. This gave us freedom to be ”picky” about which labels to include.

The unsupervised methods we tested revealed challenges in clustering for sentiment analysis tasks. Future work could explore more sophisticated techniques for label augmentation, such as using ensemble models to improve label imputation accuracy. Perhaps other unsupervised methods could be suitable for this task, but considering the initial success we had with supervised methods, as well as the drop in accuracy when we tested some unsupervised methods, we found the supervised methods more valuable. A key takeaway from this project is that sometimes, slight modifications to otherwise simple approaches can perform just as well, if not better than fancier solutions.

We are particularly proud of the creative way in which we identified this as our most promising and effective model. Initially, we built the Logistic Regressor in order to set our benchmark. Based on the accuracy requirement for the assignment, we did not need to change much, but we wanted to include more data in the spirit of the assignment. However, when the GMM and KMeans failed, we elected to test what would happen if we only included values from the Logistic Regressor that were at least 75% likely, and saw some success with this. As we slowly increased this threshold until we hit 99%, we saw increased success, but a decrease in the amount of data used. The iterative version of the Logistic Regression meets these issues in the middle, as it increases performance and uses most of the unlabeled data.

It was noted that a 2-4% increase from initial to final models are expected. However, due to the our initial model's success, this was extremely difficult to achieve. Despite this, we believe that our exploration of unsupervised methods, as well as our creative and successful final model, meet all other grading criteria.