Sentiment Classification of *Animal Crossing: New Horizons* Reviews

1. Introduction

Animal Crossing: New Horizons is a game that was extremely popular during the pandemic (and continues to have active players) and seemed to have received high praises from online communities. What exactly were people saying about the game, though? Since this project focuses on a specific game, there are many unique features that the players of *Animal Crossing: New Horizons* have encountered–some that were received positively and some received negatively. Many people expressed positive sentiments towards features of the game that brought calmness amid the chaos during the global pandemic, but there were likely equally as many features of the game–such as needing to pay in order for the game to function as multiplayer–that may have caused people to view the game in a negative light. The goal of this project is to discover whether an NLP model can identify the sentiment of reviews of the game.

2. Data

The dataset was retrieved from Kaggle[1] and consists of Metacritic reviews that were collected in 2020. The original dataset consists of the raw score from 0-10 (where a higher score equates to a better score), the username of the reviewer, the raw text of the review, and the date that the review was published. The dataset was cleaned to include only the raw score and the raw text of the review. The raw scores were binned into the three sentiment labels for the three-way classification task: 0-3 for Negative, 4-6 for Neutral, and 7-10 for Positive. There were a total of 2999 instances: 1130 Positive, 227 Neutral, and 1642 Negative.

Label	Text
Negative	This is so annoying. Only one player has the ability to play this game because only one island. Nintendo doesn't deserve my money anymore.
Neutral	The game is fun to play, but once again Nintendo fumbles online which is why my review is a 5.
Positive	This game is absolutely amazing as a constant side game for one's day. It a great game to relax with.

Figure 1. Sample text from each label in the dataset.

The dataset was split 80% in the train set, 10% in the development set, and 10% in the test set using the train_test_split function from scikit-learn and a random state of 77. The train set contains a total of 2399 instances: 889 Positive, 185 Neutral, and 1235 Negative. The development set contains a total of 300 instances: 115 Positive, 20 Neutral, and 165 Negative. The test set contains a total of 300 instances: 125 Positive, 22 Neutral, and 152 Negative.

[1] 3

3. Development set results

The models ran were Naive Bayes, Logistic Regression, and Random Forest Classifier. Two feature sets were used: unigrams (lowercased, excluding stop words, punctuation, and spaces) and bigrams (lowercased, excluding punctuation and spaces). The hyperparameters that were experimented with were *alpha* (α) for Naive Bayes; *C*, *max_iter*, and *class_weight* for Logistic Regression; and *max_depth*, *n_estimators*, and *max_features* for Random Forest. The hyperparameters for each model were tuned on accuracy. The baseline accuracy for both feature sets was a score of 55.00, where the predicted class was defaulted to the most frequent class in the dataset. Later, the baseline accuracy was adjusted to be a score of 0.84, which came from running Logistic Regression on the unigram feature set with default hyperparameters.

Figure 2 contains a condensed list of the hyperparameters ran on the unigram feature set and the resulting accuracy. Many more tests were run on the unigram feature set, but for the purpose of avoiding redundancy, the hyperparameters shown contain the minimum, maximum, and average accuracy scores per model, as well as the hyperparameters that were used to run on the bigram feature set. Figure 3 contains the same information as Figure 2 but for the bigram feature set.

Model	Hyperparameters	Accuracy
Naive Bayes	$\alpha = 0.5$	82.00
Naive Bayes	$\alpha = 1.35$	83.67
Naive Bayes	$\alpha = 1.4$	84.00
Logistic Regression	C = 0.01 max_iter = 4000 class_weight = balanced	79.33
Logistic Regression	C = 0.1	86.00
Logistic Regression	C = 0.1 max_iter = 4000	**86.33**
Logistic Regression	C = 0.1 max_iter = 4000 class_weight = balanced	84.00
Logistic Regression	C = 0.5 max_iter = 4000	85.00
Random Forest Class	n_estimators = 100	82.67
Random Forest Class	n_estimators = 150	82.33
Random Forest Class	max_depth = 1	55.00
Random Forest Class	max_depth = 9	75.33

Figure 2. Unigram feature set development set results.

Model	Hyperparameters	Accuracy
Naive Bayes	α = 1.0	81.33
Naive Bayes	α = 1.35	81.33
Logistic Regression	C = 1.0	**82.67**
Logistic Regression	C = 0.1 max_iter = 4000	**83.00**
Logistic Regression	C = 0.1 max_iter = 4000 class_weight = balanced	81.00
Logistic Regression	C = 0.5 max_iter = 4000	**82.67**
Random Forest Class	n_estimators = 150	81.33
Random Forest Class	n_estimators = 400	81.67

Figure 3. Bigram feature set development set results.

4. Test set results

Following the findings from running various tests on the development set, the model to best run on the test set turned out to be Logistic Regression using the unigram feature set. The three hyperparameter configurations chosen were the ones that resulted in the highest accuracy when the model was run on the development set. In order to ensure no convergence errors would occur for the test set, all three configurations contain the *max_iter* hyperparameter set to 4000, rather than its default value of 100.

Model	Hyperparameters	Accuracy
Logistic Regression	C = 0.1 max_iter = 4000	**84.33**
Logistic Regression	C = 0.1 max_iter = 4000 class_weight = balanced	81.33
Logistic Regression	C = 0.5 max_iter = 4000	83.00

Figure 4. Unigram feature set test set results.

	Precision	Recall	F1-Score	Support
Negative	84.05	90.13	86.98	152
Neutral	23.08	13.64	17.14	22
Positive	91.13	89.68	90.40	126

	Precision	Recall	F1-Score	Support
Accuracy			84.33	300
Macro Avg	66.09	64.48	64.84	300
Weighted Avg	82.55	84.33	83.30	300

Figure 5. Classification report for the configuration resulting in the best score on the test set.

5. Discussion

Naive Bayes and Random Forest struggled immensely with identifying text as Neutral, while Logistic Regression was able to do so but not much better. The precision for the Neutral label was always zero for Naive Bayes and Random Forest while it was always non-zero for Logistic Regression. This raises concern regarding whether the issue is a result of the Neutral label having much less support in the overall dataset or if there is something to be said about the performance of Naive Bayes and Random Forest for multiclass classification.

Naive Bayes assumes that the features operate independently of each other, which is especially not true in this case, as there are features that appear in text with all three labels. It is crucial in this case that the relationship between unigrams within each text are elevated to reveal the true label. For example, Naive Bayes failed to classify any of the Neutral texts as Neutral. When observing the dataset, it is apparent that many of the texts labeled as Neutral tend to have elements that occur in Positive texts and Negative texts. This is reasonable, as the Neutral reviews most likely both praise and criticize. Naive Bayes struggles to recognize that there are both praises and criticisms within the texts, as there is no way to correlate features with one another, thus ignoring the Neutral label altogether.

The Random Forest classifier is said to contain numerous decision trees that are used in combination with mathematical tools to identify the best results. It is known to decrease chances of overfitting data due to its complexity. While it seems that there are many reasons why this model is great, it failed to classify any of the Neutral texts as Neutral. Perhaps it struggled due to a similar reason as Naive Bayes, where the fact that Neutral texts contain elements that occur in texts with the other two labels may be confusing the model. While the model did not perform

well enough to be included in the set of models and configurations to be run on the test set, there is potential that was left hidden due to unfamiliarity with the model.

Logistic Regression utilizes statistics to predict the probability for classes to occur. It seemingly explores the relationship between dependent and independent features, which is the issue that was raised previously about the other models. If that is the case, this may be the reason why Logistic Regression is able to classify some Neutral texts as Neutral. Perhaps it detects the notion that there are elements from both Positive and Negative texts that occur in Neutral texts. In either case, this model worked well overall, despite there being less support in the dataset for the Neutral class.

Unigrams were expected to perform better than bigrams because of the lack of stop words in the former and the fact that the latter contains many more unique features that could be difficult for models to use to make generalizations. Bigrams still performed fairly well, especially considering the Neutral class precision scores. In terms of accuracy, the tests run show that it performs just slightly worse than unigrams, but it seems that there is potential for better results.

6. Conclusion

Logistic Regression performed the best for the three-way sentiment analysis task. While the accuracy scores for all models were similar during the experimental phase with the development set, it was clear that this model is the only one that has support for the Neutral class. It may be the case that Logistic Regression is able to associate the idea that Neutral texts have elements that can occur in both Positive and Negative texts, which is something that the other two models seemed to struggle with.

Throughout this experience, it was harder to determine which hyperparameters were appropriate to tune than expected. Given more time, there may be other models that are better to use for this particular task and other hyperparameters that could be explored with the current models. Since the hyperparameters to run for the bigram feature set were determined based on the performance of the configurations run on the unigram feature set, it would be interesting to branch out and explore different hyperparameters that may work better for bigrams than unigrams.