ENSF 612

Engineering Large Scale Data Analytics Systems

Fall 2022

Online Game Reviews

STEAM™

Group 5:

Justin Nguyen - 30042258

Garnet Crookes - 30173480

Ben Kaminski - 00301335

Contributions:

All	Preplanning, Data-collection, Labeling, Preprocess adjustment, Model Discussion, Presentations, Report, Implementing Professor Feedback
Justin	Labeling: FIFA, New World, For Honor, Beat-Saber Coding: Preprocessing (1-18), Modeling (19-24) and Hyper-Parameter Tuning (25-49) Report: Abstract, Results, Conclusion, Discussion
Garnet	Labeling: Harvest Moon, Call of Duty, Total War, Beat-Saber Coding: Preprocessing (0-11), Modeling (19-24, 50-75) and Hyper-Parameter Tuning (31-46) Report: Introduction, Results, Discussion
Ben	Labeling: PUBG, Project Cars 3, Path of Exile, Beat-Saber Coding: Preprocessing (1-18), TF-IDF (4-18) Report: Abstract, Results, Discussion, Conclusion

Notebooks: Project - Databricks PreProcessing - GitHub

Submission Date: December 15, 2022

Table of Contents

1. Abstract

1.1 Context

Online game reviews directly influence the game's public perception, negatively or positively. Through filtering out irrelevant or low quality reviews, game developers and potential buyers gain a better understanding of the game and thus make better informed decisions based on high quality reviews.

1.2 Objective

The objective is to create a machine learning model that accurately predicts relevancy in game reviews through context analysis and other features available from Steamworks API.

1.3 Method

This report will outline the steps used to collect, label, and process Steam game reviews as input into machine learning models. Multiple machine learning models will be used to evaluate the best performer in predicting relevant or irrelevant game reviews. The model's hyper-parameters were tuned to optimize performance.

1.4 Results

SVM and Logistic Regression models performed better compared to Naïve Bayes and Random Forest, with accuracy scores of 86% and 84% respectively after tuning. Random Forest performed slightly worse with an accuracy score of 80%, however, Naïve Bayes performed poorly in comparison with a score of 76%. This is due to the nonlinear nature of the project, as many words could be relevant or irrelevant based on the context.

1.5 Conclusion

Through this work, results were provided indicating effectiveness of the data collection, preprocessing, and model training done, allowing for real-world use. This concept can not only be applied to filter game reviews, but in other applications, such as a discussion forum to moderate postings.

2. Introduction

2.1 Motivation

Many gamers read reviews on major game platforms like Steam and Unity before purchasing a game and would prefer to only see reviews relevant to them. A majority of the reviews left are not useful to other consumers, due to being poorly thought out or unrelated to the game. With this project we aim to train a model that can remove irrelevant reviews to help a user in making a purchasing decision.

2.2 Background

On Steam there are thousands of games with hundreds of reviews each. These reviews are left by players, and are often used by potential buyers to decide if they want to try the game. However a lot of the reviews are irrelevant for this process due to not being well thought out, or not being about the game. This type of review is not relevant for the potential buyer trying to decide if they want to try the game. Potential buyers would prefer to only read the reviews that are well thought out and discuss the pros and cons of the game. In this project we use a handful of games on steam to train a model aimed at classifying reviews into being relevant or irrelevant.

2.3 Data Collection

The Steamworks API was used to collect reviews. The data received from the API had the text of the review and other data including factors, such as hours played, review rating, and funny tags. In total 200 reviews for 10 different games were collected resulting in 2000 rows of data. The games were chosen to span across many styles to prevent overfitting to one particular community of users.

2.4 Data Labeling

The data was labeled into a binary classification of relevant and irrelevant. Relevant indicates the review is related to the game and constructed in a way that would be useful for other customers. Key aspects of a relevant review are defining aspects of the product in a genuine way, and focusing on generalizable aspects of the experience.

Our project is not following an existing report and doesn't have any original data to compare the labeling distribution against. The distribution of labels in our dataset is 70% of reviews being irrelevant and 30% being relevant. This distribution among games varies slightly, this is likely due to different communities of users playing different categories of games.

2.5 Data Preprocessing

The data was then preprocessed to prepare it for machine learning. From the reviews: stopwords, punctuation, numbers, and words shorter than three letters were all removed. Additionally, words identified a lot in the reviews that did not add value to the model were removed as stopwords. Then the remaining text was lemmatized using a wordnet lemmatizer with the default PoS tagging in NLTK.

The additional data that was not textual was normalized using min—max normalization in Spark. The game names were indexed and split out into separate columns using one hot encoding to improve performance.

2.6 Model Performance

Four models were trained on the labeled data and all performed well with accuracies ranging from 76% to 86%. The best performing models were SVM at 86% and Logistic Regression at 83% when using the default parameters. For a filtering algorithm based on word processing this is a reasonable accuracy that could be used in an actual application.

2.7 Hyper-parameter Tuning

The hyper parameters were tuned using a grid search to find the optimal parameters. The parameters were only tuned for the better models noted above. For the SVM model the grid search did not result in improved performance.. For the Logistic Regression the performance was improved from 83% to 84% which is a modest improvement. For this model a bad choice of hyper parameters dropped the accuracy to 71% which is much less accurate.

2.8 Misclassifications

For the SVM model 9% of the test set was predicted to be irrelevant when it was labeled relevant and 6% was due to relevant predictions actually being irrelevant. The bias towards labeling reviews as irrelevant is likely due to that case being overrepresented in our training set. Since we want to filter our data that is irrelevant this is the preferred bias of the model.

3. Results

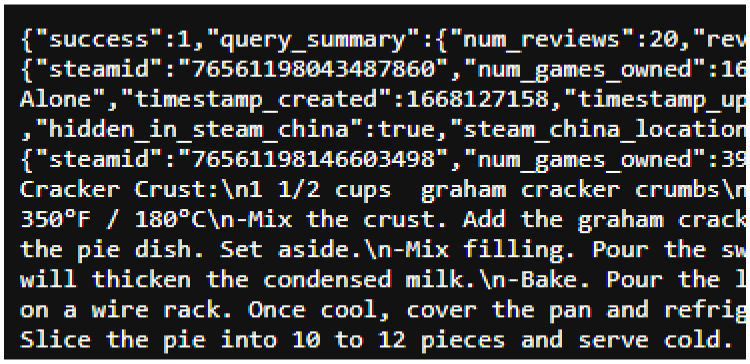
3.1 Data Collection

Data collection was done with the use of Steamworks API. Accessing the reviews from one game can be achieved through the link shown in Figure 1, where the app id is the game’s identifier:

$$https://store.steampowered.com/appreviews/ < appid >?json = 1$$

Figure 1 [1]: App reviews from one game.

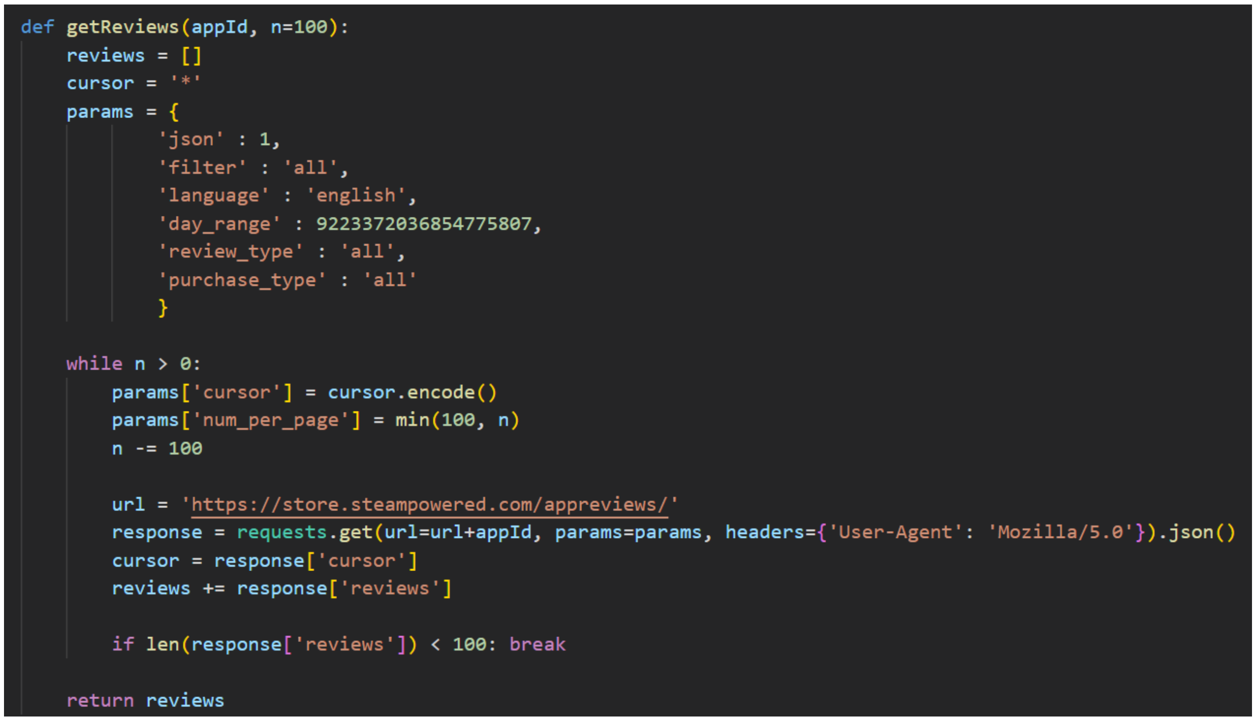
The responses received from visiting the link are reviews and user information in JSON format, as shown in Figure 2, which can then be parsed and loaded into a pandas dataframe for data analysis.



```
{"success":1,"query_summary":{"num_reviews":20,"rev
{"steamid":"76561198043487860","num_games_owned":16
Alone","timestamp_created":1668127158,"timestamp_up
,"hidden_in_steam_china":true,"steam_china_location
{"steamid":"76561198146603498","num_games_owned":39
Cracker Crust:\n1 1/2 cups  graham cracker crumbs\n
350°F / 180°C\n-Mix the crust. Add the graham crack
the pie dish. Set aside.\n-Mix filling. Pour the sw
will thicken the condensed milk.\n-Bake. Pour the l
on a wire rack. Once cool, cover the pan and refrig
Slice the pie into 10 to 12 pieces and serve cold.
```

Figure 2: Steamworks API JSON response.

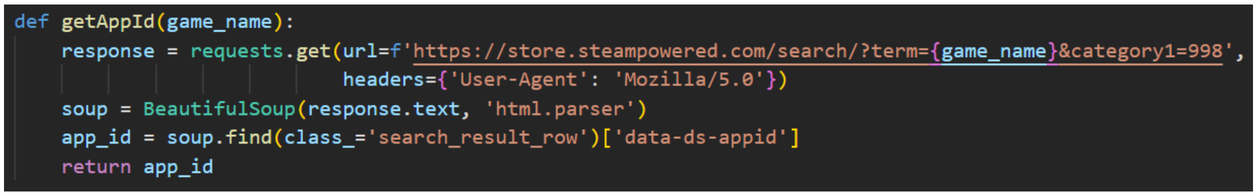
Data collection was increased to 200 game reviews for 10 games in order to increase machine learning model performance, resulting in a total of 2000 reviews. However, the number of game reviews received through the link used in Figure 1 are limited to 100. In order to receive 200 game reviews, a python function was created with the use of the ‘cursor’ attribute from the API, which marks the review the request completed on. By including the same cursor on the next request, the cursor will start on the next 100 game reviews, as seen in Figure 3.



```python
def getReviews(appId, n=100):
    reviews = []
    cursor = '*'
    params = {
            'json' : 1,
            'filter' : 'all',
            'language' : 'english',
            'day_range' : 9223372036854775807,
            'review_type' : 'all',
            'purchase_type' : 'all'
            }

    while n > 0:
        params['cursor'] = cursor.encode()
        params['num_per_page'] = min(100, n)
        n -= 100

        url = 'https://store.steampowered.com/appreviews/'
        response = requests.get(url=url+appId, params=params, headers={'User-Agent': 'Mozilla/5.0'}).json()
        cursor = response['cursor']
        reviews += response['reviews']

        if len(response['reviews']) < 100: break

    return reviews
```

Figure 3 [1]: Obtaining 'n' number of game reviews.

App id's of multiple games were retrieved with the help of BeautifulSoup. Through searching the Steam store within the 'Games' category, BeautifulSoup uses the 'find()' method to grab the first result that meets the required characteristics, as seen in Figure 4.



```python
def getAppId(game_name):
    response = requests.get(url=f'https://store.steampowered.com/search/?term={game_name}&category1=998',
                            headers={'User-Agent': 'Mozilla/5.0'})
    soup = BeautifulSoup(response.text, 'html.parser')
    app_id = soup.find(class_='search_result_row')['data-ds-appid']
    return app_id
```

Figure 4 [1]: Obtaining the app id of the game.

The required characteristics used were 'search_result_row', which is what Steam uses for each search result, and 'data-ds-appid', which stores the app id from the search result. By default, this query grabs the current 'top sellers'. Lastly, the retrieved game reviews were loaded into a pandas dataframe and exported as excel files for data labeling, as shown in Figure 5.

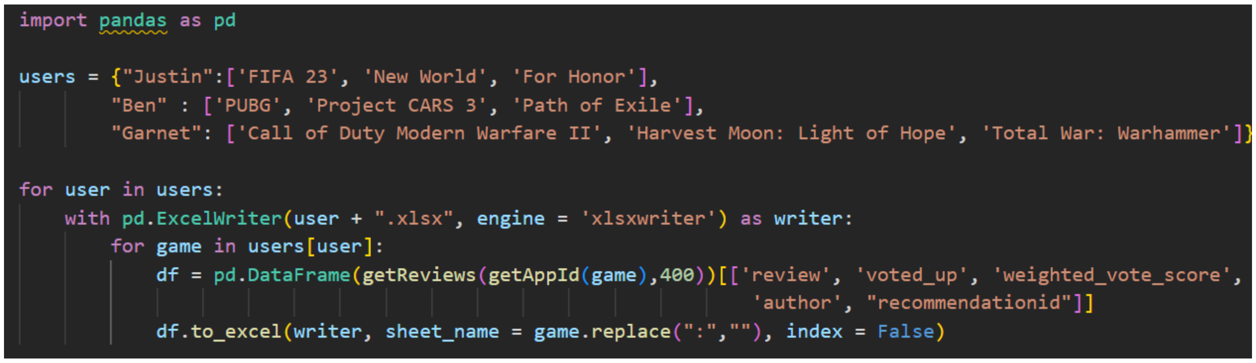
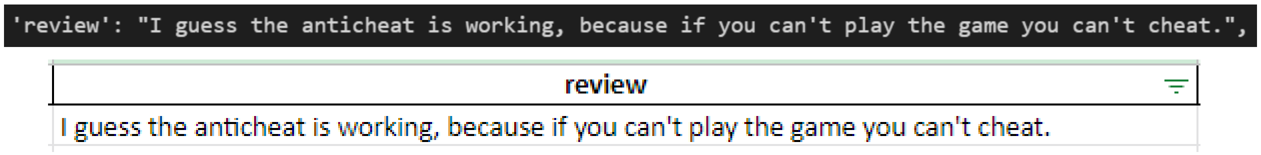

```
import pandas as pd

users = {"Justin":['FIFA 23', 'New World', 'For Honor'],
         "Ben" : ['PUBG', 'Project CARS 3', 'Path of Exile'],
         "Garnet": ['Call of Duty Modern Warfare II', 'Harvest Moon: Light of Hope', 'Total War: Warhammer']}

for user in users:
    with pd.ExcelWriter(user + ".xlsx", engine = 'xlsxwriter') as writer:
        for game in users[user]:
            df = pd.DataFrame(getReviews(getAppId(game),400))[['review', 'voted_up', 'weighted_vote_score',
                                                                'author', "recommendationid"]]
            df.to_excel(writer, sheet_name = game.replace(":",""), index = False)
```

Figure 5: Exporting game reviews for labeling.

From the excel files obtained, the review was successfully exported with one-to-one translation, as shown in Figure 6.


```
'review': "I guess the anticheat is working, because if you can't play the game you can't cheat.",
```

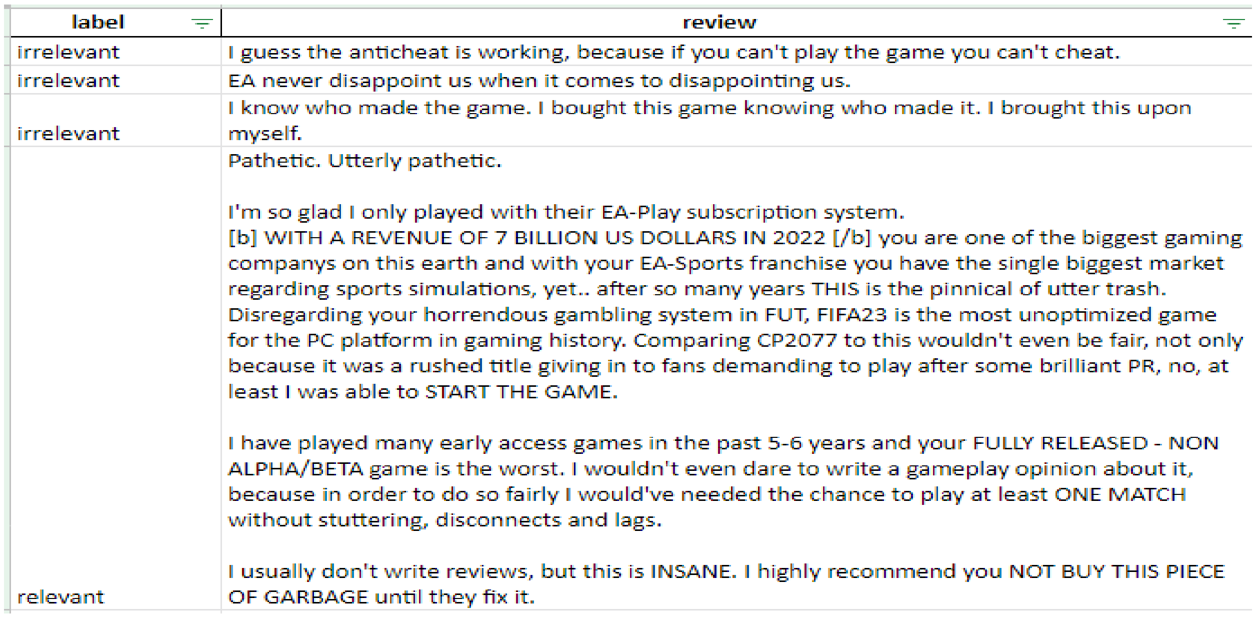
review
I guess the anticheat is working, because if you can't play the game you can't cheat.

Figure 6: Top review directly received from API, while bottom review is from exported excel file.

Along with the review, other features were exported, such as the user's playtime, number of votes up, and the overall score of the review. With the quality of the data collection ensured, the next step in the process was data labeling of 2000 game reviews.

3.2 Data Labeling

Labeling was done with the concept of percent agreement. One game, with 200 reviews, was chosen to be labeled together as a team. An individual on the team would label the first 50 reviews with the other two observing. After labeling was completed, deliberation was made on each review to agree upon a consensus. This process was done between each team member until all 200 reviews were successfully labeled. Subjective bias was minimized with a strategy of key labeling targets, such as word uniqueness, organization of thought, game detail, and detailed language.



label	review
irrelevant	I guess the anticheat is working, because if you can't play the game you can't cheat.
irrelevant	EA never disappoint us when it comes to disappointing us.
irrelevant	I know who made the game. I bought this game knowing who made it. I brought this upon myself.
relevant	Pathetic. Utterly pathetic. I'm so glad I only played with their EA-Play subscription system. [b] WITH A REVENUE OF 7 BILLION US DOLLARS IN 2022 [/b] you are one of the biggest gaming companys on this earth and with your EA-Sports franchise you have the single biggest market regarding sports simulations, yet.. after so many years THIS is the pinnical of utter trash. Disregarding your horrendous gambling system in FUT, FIFA23 is the most unoptimized game for the PC platform in gaming history. Comparing CP2077 to this wouldn't even be fair, not only because it was a rushed title giving in to fans demanding to play after some brilliant PR, no, at least I was able to START THE GAME. I have played many early access games in the past 5-6 years and your FULLY RELEASED - NON ALPHA/BETA game is the worst. I wouldn't even dare to write a gameplay opinion about it, because in order to do so fairly I would've needed the chance to play at least ONE MATCH without stuttering, disconnects and lags. I usually don't write reviews, but this is INSANE. I highly recommend you NOT BUY THIS PIECE OF GARBAGE until they fix it.

Figure 7: Data labeling.

Each review was labeled with either 'irrelevant' or 'relevant', as shown in Figure 7, indicating the review's helpfulness to the user. In the case of a review that borders between the two labels, a consensus would be reached between all team members to decide on the label. The remaining 1800 reviews were done individually with 600 reviews from differing games assigned to each team member.

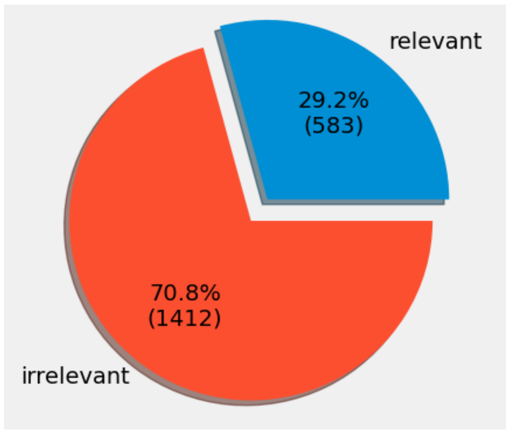


Figure 8A: Pie chart of label distribution.

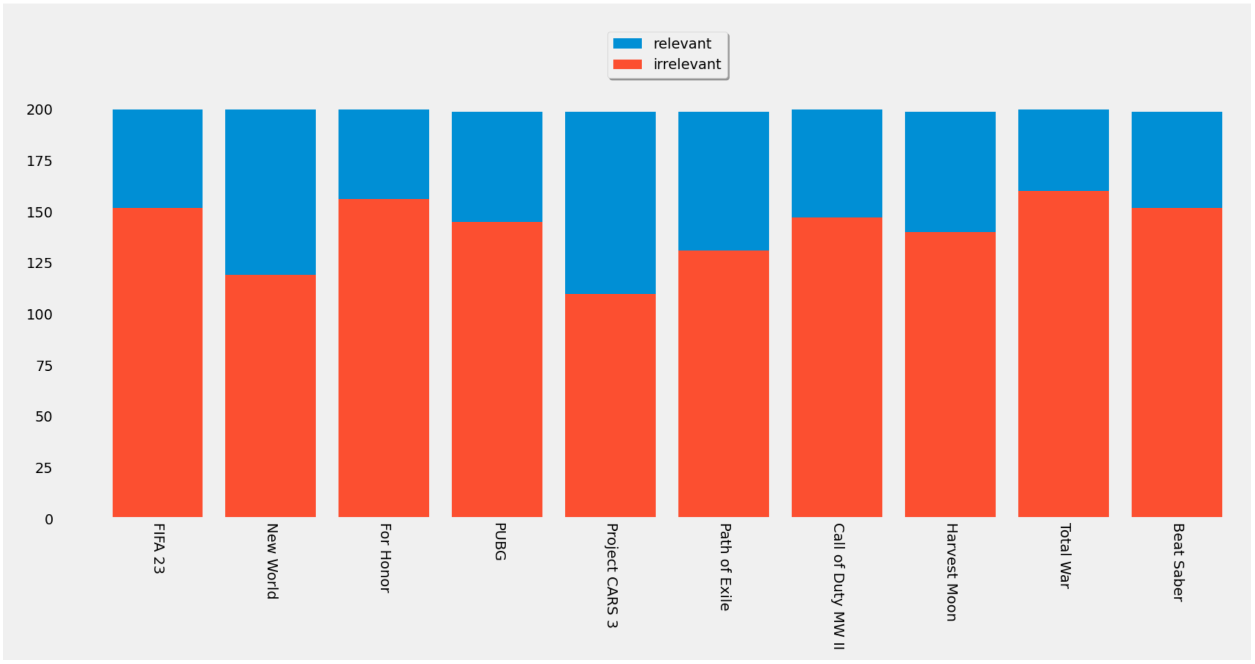


Figure 8B: Bar chart of label distribution.

Figures 8A and 8B indicate the consistency of labeling between team members, despite the variance between games and the distribution of reviews. This consistency was desired in order to increase performance of the machine learning model. Additionally, there is small variance between games as a result of different communities of users. There was no existing data to compare our labeling distribution to.

3.3 Data Preprocessing

The labeled excel files were imported into a pandas dataframe, where each row was fed into a preprocessing function. Preprocessing was done in order to filter out noise in the review body that was deemed not useful to the machine learning model.

The data was preprocessed through the following stages:

1. Removing stopwords, including additional project specific words
2. Removing punctuation
3. Removing numbers
4. Removing small or incomplete words
5. Lemmatization

Initially, snowball stemming was utilized in order to obtain the root word for a bag-of-words. However, due to suggestion, lemmatization was used as it takes into consideration the context of the word to determine the intended meaning of the word. As a result, the processed review was left with important keywords intended for context analysis and a 3% accuracy improvement over Stemming.

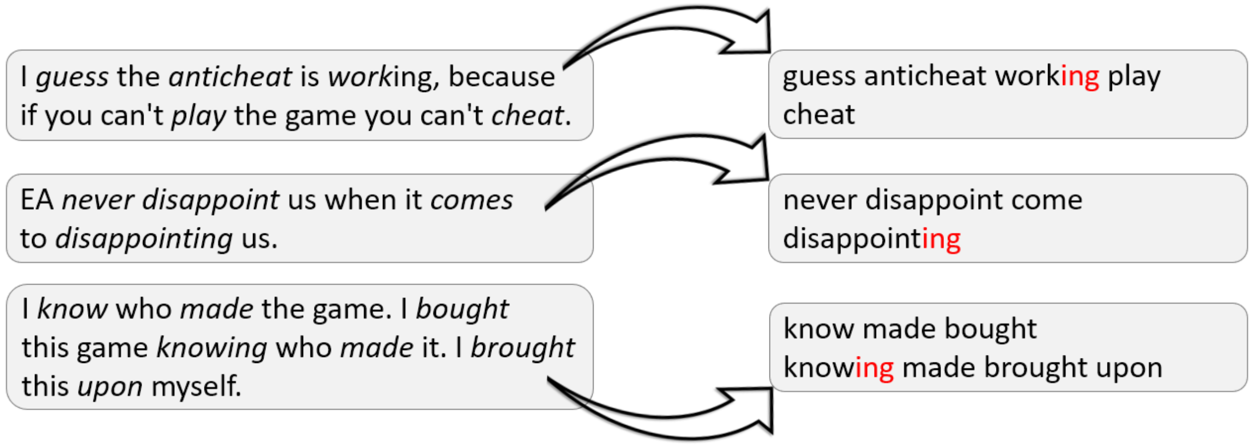
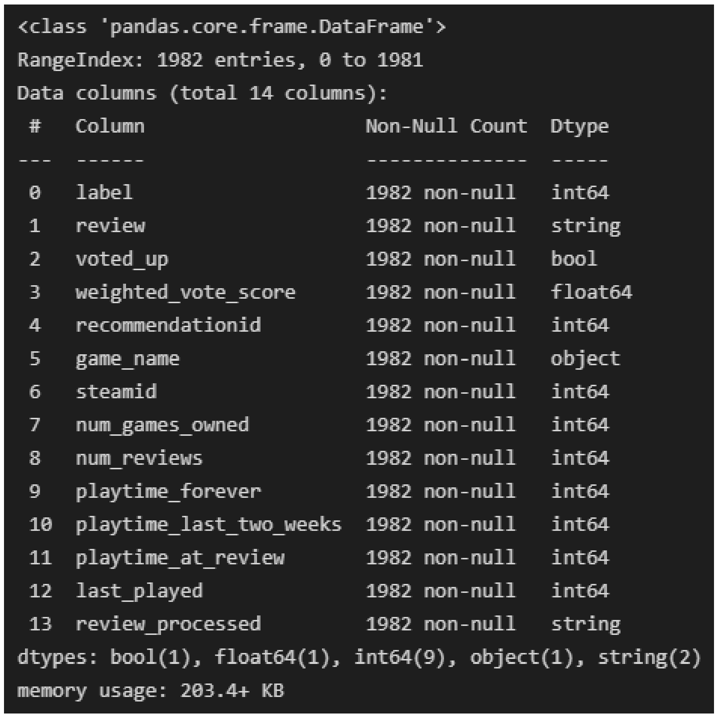

I *guess* the *anticheat* is *working*, because if you can't *play* the game you can't *cheat*.
guess anticheat working play cheat
EA *never disappoint* us when it *comes* to *disappointing* us.
never disappoint come disappointing
I *know* who *made* the game. I *bought* this game *knowing* who *made* it. I *brought* this *upon* myself.
know made bought knowing made brought upon

Figure 9: Processed reviews.

The additional data that was not game reviews was normalized using min-max normalization in Pyspark. The game names were indexed and split out into separate columns using one hot encoding for model training. As seen in Figure 9, stopwords, punctuations, and small words were successfully filtered out, with lemmatization keeping the intended meaning of the words. With the processed reviews optimized for context analysis, it is then ready to be used in the machine learning model.

Below are summary statistics of the processed dataframe.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1982 entries, 0 to 1981
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   label                    1982 non-null   int64
 1   review                   1982 non-null   string
 2   voted_up                 1982 non-null   bool
 3   weighted_vote_score      1982 non-null   float64
 4   recommendationid         1982 non-null   int64
 5   game_name                1982 non-null   object
 6   steamid                  1982 non-null   int64
 7   num_games_owned          1982 non-null   int64
 8   num_reviews              1982 non-null   int64
 9   playtime_forever         1982 non-null   int64
 10  playtime_last_two_weeks  1982 non-null   int64
 11  playtime_at_review       1982 non-null   int64
 12  last_played              1982 non-null   int64
 13  review_processed         1982 non-null   string
dtypes: bool(1), float64(1), int64(9), object(1), string(2)
memory usage: 203.4+ KB
```

Figure 10: Processed dataframe summary statistics.

A small number of short and illegible reviews were entirely filtered out after preprocessing, resulting in a total of 1982 entries.

To enhance the model, TF-IDF was used on the processed reviews. The reviews were tokenized and collected into a bag-of-words. Then the inverse frequency of all words in the set of reviews was found using Spark's IDF function to obtain the TF-IDF frequency of each word in the reviews. This enhanced feature set was used for model training.

3.4 Model Performance

Four classification models were trained on the preprocessed data using the default parameters provided by Spark. The methodology for training these was taken from [2]. The result of evaluating the models against the test set are shown in Figure 11 below.

Legend

False Positive	True Negative
True Positive	False Negative

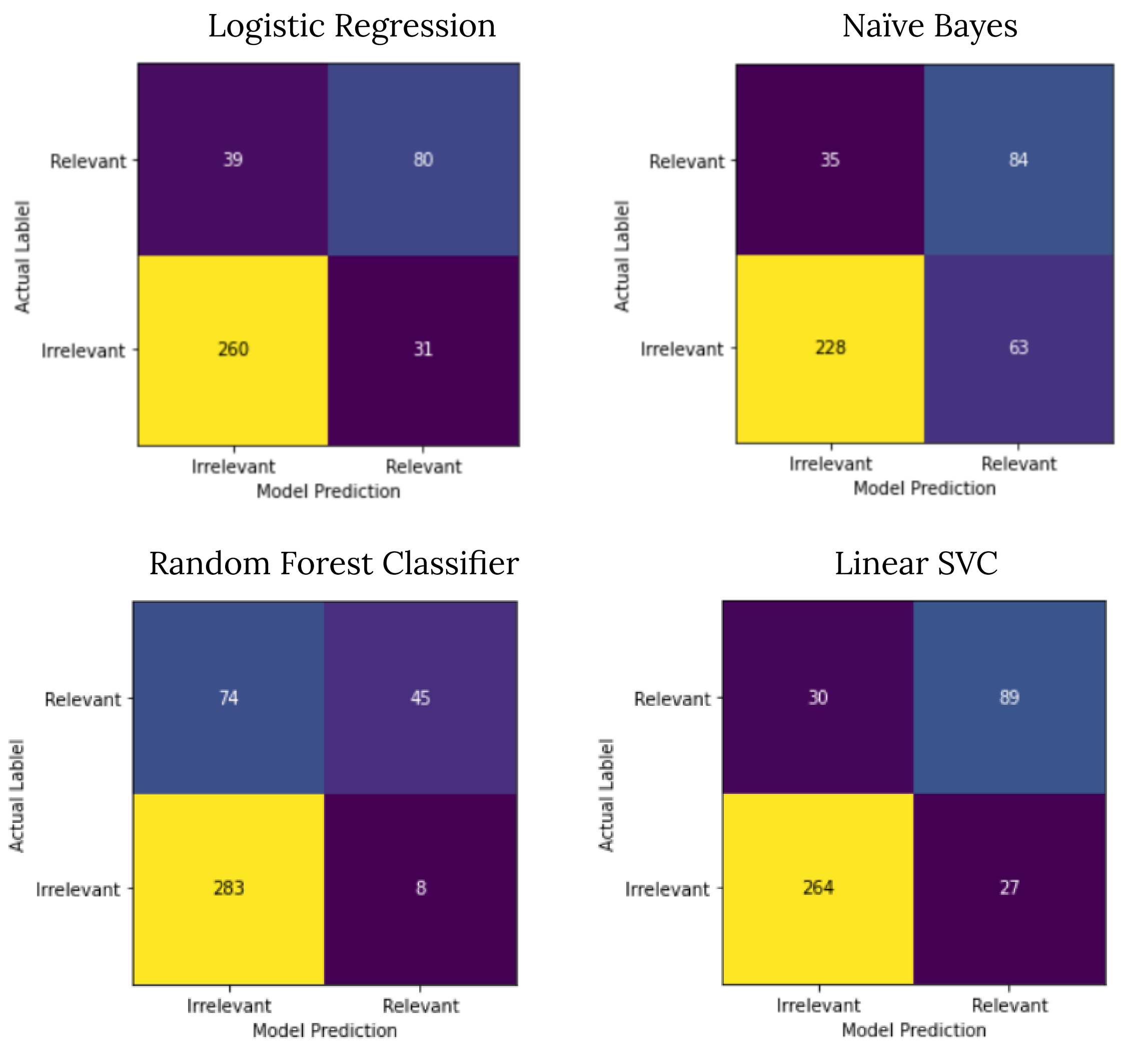


Figure 11: Confusion Matrices of Trained Models.

To better understand the performance of the different models the accuracy of the results was calculated, and are shown in Table 1. These results are for the data that was processed using lemmatization which improved the accuracy approximately 3% for models over using stemming.

Table 1: Models Performance with Default Parameters

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	82.9%	87.0%	89.3%	82.7%
Naïve Bayes	76.1 %	86.7%	78.4%	76.8%
Random Forest Classifier	80.0%	79.3%	97.3%	77.2%
Linear SVC (SVM)	86.1%	89.8%	90.7%	86.0%

For classifying game reviews as relevant F1-Score is the preferred accuracy metric, and will be used to determine the best model. F1 is preferred since neither false positives, or false negatives are critical to avoid, and using the harmonic mean will better account for the overrepresentation of irrelevant reviews in the training set then using accuracy. Based on F1-Score the models in order from best to worst are SVM, Logistic Regression, Random Forest, and Naïve Bayes; the same ranking is found using accuracy. Based on Precision the Naïve Bayes performs substantially better since it doesn't falsely label many reviews as irrelevant, if this was the primary focus Naïve Bayes would be a good model to investigate. Based on Recall the Random Forest model performs extremely well since it only falsely labels 8 relevant rows. If the primary goal was collecting all relevant reviews this would be a good model to investigate.

Overall the best scoring models were SVM and Logistic regression. SVM is known to work well in problems with many dimensions like a bag of words, which likely contributes to better performance in this case. The training set for these results only contains the TF-IDF results as features. The naivety of Naïve Bayes could make it harder to determine context of words that could be either relevant or irrelevant depending on context, which SVM can analyze better. The models cannot be compared to a previous study since we are not working off an existing paper.

The better performing Logistic Regression and SVM models were also trained using the following extended parameters to improve the accuracy:

- Weighted Vote Score
- Number of Reviews
- Game Name
- Number of Games Owned
- Playtime at Review
- Playtime Forever

The accuracies with this extended feature set included are shown in Table 2. In both cases adding additional features reduced the F1-Score, and precision but increased Recall. The additional features help the model avoid falsely labeling reviews as relevant. This is confirmed by the low number in the bottom right quadrants of Figure 12. The models overall generate fewer correct results since there are more reviews falsely labeled as irrelevant. This is an improvement for the goal of filtering out irrelevant reviews but not general accuracy.

Table 2: Models Performance with Extended Features

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression- Original	82.9%	87.0%	89.3%	82.7%
Logistic Regression	82.0%	81.3%	96.5%	80.2%
Linear SVC (SVM) - Original	86.1%	89.8%	90.7%	86.0%
Linear SVC (SVM)	84.1%	85.0%	94.1%	83.4%

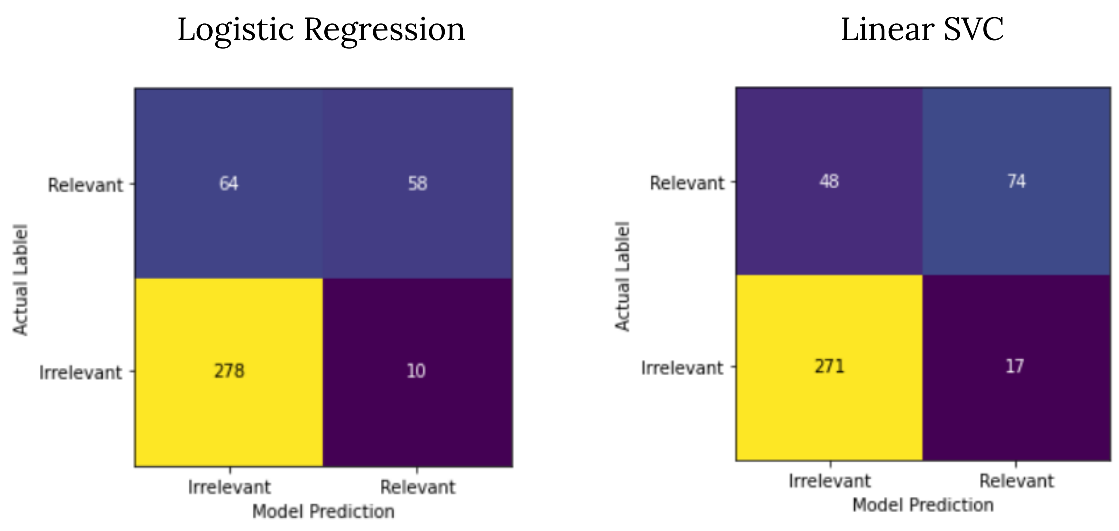


Figure 12: Confusion Matrices of Models Trained On Additional Features.

3.5 Hyper-parameter Tuning

The hyper-parameters were tuned for the best performing models of SVM and Logistic Regression to improve the accuracy. The tuning was completed using a grid search to evaluate how the model performed for different parameters. At each grid point a 5-fold cross validation was performed and the best performing model was kept.

For Logistic Regression the hyper-parameters that were tuned were the regularization and the elastic net parameters. The regularization parameter adjusts the acceptable amount of error, and the elastic net parameter defines how the L1 and L2 errors are combined when training the model [3]. The best results were found at Elastic Net = 0.0 and Regularization = 1.0. A heat map showing the accuracy at different points is shown in Figure 13 below. The heat map shows a large grid of results that are all 71%. The cause of this is the model always predicting the review is irrelevant in this case. These models are not useful and no further tuning is required along this boundary of parameters. The parameters on the left and right edges of the heatmap perform much better and the parameters cannot be increased further in these directions; this is likely the optimal set of hyper-parameters.

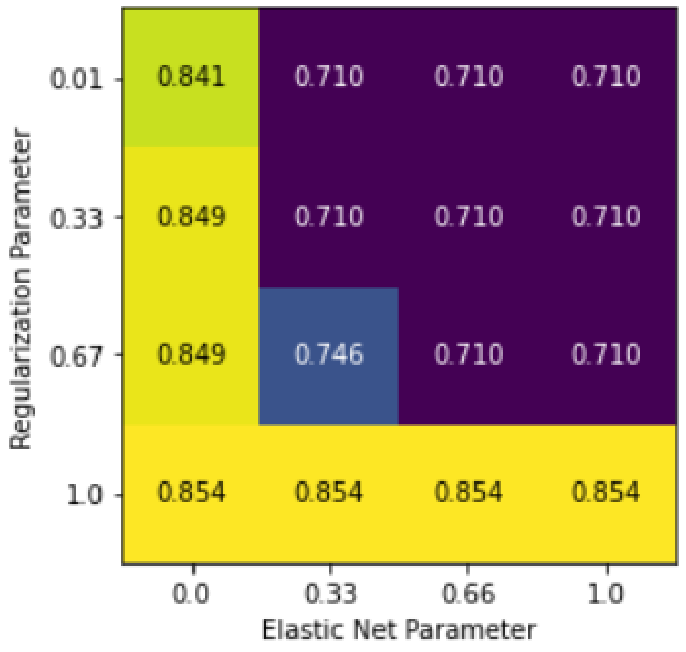


Figure 13: Logistic Regression Grid Search Heatmap.

For the SVM model, the regularization parameter and aggregation depth parameters were used to tune the model. The regularization parameter adjusts the acceptable amount of error, and the aggregation depth is the number of layers for the Tree Aggregate [4]. The best model was found with a regularization parameter of 1.0 and an aggregation depth of 2, which aligns with the default parameters. The performance is not dependent on the aggregation depth as shown in Figure 14, where accuracy does not change across rows that have a fixed regularization parameter. Potentially, higher aggregations depths could improve accuracy but that would increase the complexity more with at most minimal improvements. These are likely the optimal hyper-parameters for this problem.

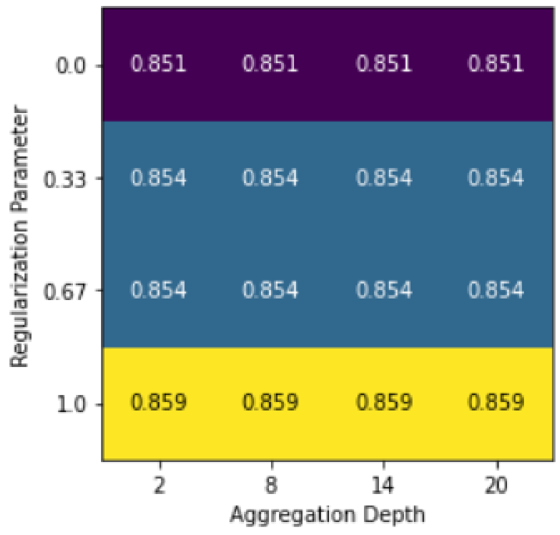


Figure 14: SVM Grid Search Heatmap.

3.6 Misclassifications

Misclassifications are evaluated based on the best models after hyper-parameter tuning which resulted in the best F1-Score. The percent of misclassifications is low and generally the misclassifications are falsely identifying reviews as irrelevant when they are labeled relevant (80% of errors for Logistic Regression and 60% for SVM). Most games have many reviews and it is better to get fewer reviews that are actually irrelevant being considered relevant, so this is the preferred bias.

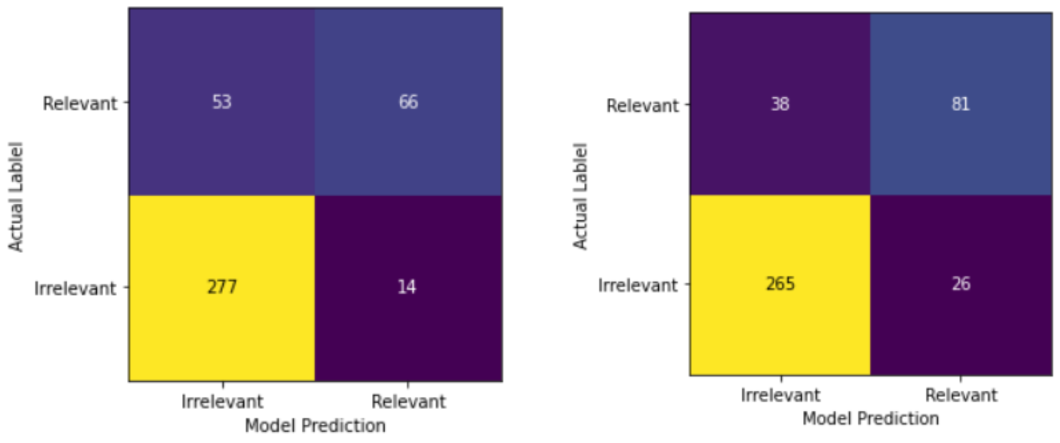


Figure 15: Confusion Matrices (Left = Logistic Regression, Right = SVM Model).

The confusion matrices for SVM and Logistic Regression are shown in Figure 15. SVM had an overall higher accuracy. However, Logistic Regression may be a better model for filtering out irrelevant reviews since it makes fewer mistaken relevant predictions. This could be better than accepting a larger ratio of irrelevant reviews being considered relevant, since getting rid of irrelevant reviews is the purpose of the model.

Misclassifications were assessed into three major categories. High-effort sarcasm, negative words in relevant reviews, and over fitting due to unique words. The high sensitivity to unique words was the main reason for misclassification and is caused by the emphasis TF-IDF puts on unique words. Some of the misclassifications came from tags that were not effectively being removed, these are remnants of the unique markdown language steam reviews use, however this accounted for less than 2% of misclassifications. The types of misclassifications are described in more detail below.

Example 1: High-Effort Sarcasm

An example of a review that contains high effort sarcasm is shown in Table 3. This review demonstrates a case where a user will leave high-effort comedically slanted reviews. Usually the goal of this user is to mock, and a clever posting might be quite elaborate. This review was labeled as irrelevant since it is a personal story not based on the game. However the model picks up many of the relevant looking words and sentiment leading to it misclassifying the review as relevant.

Table 3: Misclassification Example, High-Effort Sarcasm

Body of review	Labeled	Predicted	Type
I read The Hobbit as a Freshman in highschool in 1979. I started playing Advanced D&D later that year, and Dungeon Mastered my first game on Dec. 31, 1979. We played well into the morning and everyone crashed in chairs and couches after the sun had come up. We didn't play any other than AD&D until the mid to late 80's but when we did we tried them all. Paladium and all of their stuff from Rifts to Beyond the Supernatural, Shadown Run, Champions you name it. We picked up Warhammer Fantasy Roleplay and ate it up. During that time we played a lot of Games Workshop stuff. Talisman, Hero's Quest, Battle Master, and of course Warhammer Fantasy Battle. Games Workshop always put out great stuff with awesome back stories that set the perfect atmosphere of each adventure. When computers rolled around in the 90's we saw the potential and would fantasize about the possibilities of going to these places virtually and see all the places we had played for so many years. We started in Ultima Online I stayed there way past their peak, then ended up in WOW for 7 years or so. Everyone wants to complain about the price of DLC's. I paid 15 a month for WOW for years and always thought it was worth the money. Now one of those games we used to play back in the day has come out. Total War Warhammer has captured those awesome backstories and set that perfect atmosphere that great games always do. The game is completely flexible to allow expansion and easily changed to make it what ever you think it should be. Add some units with a mod, or pay for a DLC if you want, but you don't have to. I absolutely love this game and I want it to be successful. I don't have any problem paying for DLC as long as they don't get too expensive. So far so good! I like what I see! They have made an old gamer happy! Hopefully I'll retire one day with the complete world at my fingertips and ride off into a magical sunset. Pretty scary thought. Anyway, I do recommend this game!	Irrelevant	Relevant	High Effort Sarcasm

Example 2: Negative-Words In a Relevant Posting

An example of a review containing many negative words is shown in Table 4. Here the challenge is that negative sentiment was over represented in irrelevant reviews in the training set, since many reviewers will complain without adding any relevant content to the review. This caused the models to over associate negative sentiment with an irrelevant review. As a result some reviews where reviewers are leaving relevant complaints to be misclassified as irrelevant, due to the negativity. This is not a common error as most relevant reviews tend to use proactive language even when complaining. This is the most concerning misclassification since it leads the model to suppress critical reviews.

Table 4: Misclassification Example, Negative-Words in Review

Body of review	Labeled	Predicted	Type
Game worked pretty good the first few days, but after the new update the game wont let me control my players when I play as the home team. Tried everything and every fix, nothing works. Game is literally unplayable. Do not buy this game until they fix these issues. Had to quit 6 games in a row after the ps4 controller suddenly stops working ingame. In menu the controller has no issues. Its obvious that EA doesn't care about their PC community, but didn't expect it to be this bad. Unfortunately we cant expect anything better from this careless, moneysucking company. Absolute disgrace.	Relevant	Irrelevant	Negative wordings

Example 3: Overfitting by word uniqueness

An example of a review containing many negative words is shown in Table 5. This review has several phrases words that are processed into very unique words do to incorrect grammar. Some of the unique words are: “PICK-UP-AND-PLAY” -> “pickupandplay”, “driveclub”, “thumbsdown”, “simracing”. Since this review is from the test set and these words do not show up in the training set the model was not trained properly against them, and is considering them relevant like other unique words. Model accuracy could potentially be improved by removing words that have single or few occurrences in the entire set of reviews.

Table 5: Misclassification Example, Overfitting Unique Words

Body of review	Labeled	Predicted	Type
THIS IS A GAMEPAD PICK-UP-AND-PLAY SORT OF RACER. If you are expecting PCars 1 or 2, stay far away. Really far away. However, if you felt burned by GRID 2019, and want a fun driving experience with a gamepad in the likes of Driveclub or PGR, this is your game. It really deserves a thumbsdown for what SMS and Ian Bell have done in terms of lying to the fans, but that doesn't take away from what this game is and what it sets out to do. I play a lot of simracing on a wheel with my G29 and Oculus Rift S, but I want a good racing game that runs great on a controller. This is that.	Irrelevant	Relevant	Overfitted Unique Words

The number of misclassifications for each category based on the 100 rows of misclassifications is shown in Table 6. The most common issue was overfitting due to word uniqueness, which accounts for 46% of the errors. The other two misclassifications occur approximately equally. The most pressing next step would be to reduce the misclassifications due to unique words. This could be done by filtering these words out of the bodies. That would be a valid approach since the model does not get enough chances to train on them and therefore they are difficult to draw conclusions from

Table 6: Misclassification Labeling Breakdown

Overfitting Due to Word Uniqueness	High-effort Sarcasm	Negative Wording	Garbage in Body	Total
46	22	29	3	100

4. Discussion

The model and methodologies developed could be applied in areas where the overall quality of the short text reviews is to be assessed. Our goal as stated in the introduction was to investigate filtering of irrelevant reviews of products, in this case games. This approach can be applied in a few real world scenarios after additional training:

4.1 Educational Mentorship - Justin

As StackOverflow demonstrates, low-effort responses that don't address the question can demoralize or mislead a user. Our model can be applied to moderate negative and low quality spam responses so a question asked could filter through the best answer(s). This model could also be used to introduce a system that would assess responses and reward respondents.

4.2 Digital Storefronts - Ben

Our original idea of filtering towards relevant reviews. Steam is not the only game store and this model could be adapted to other similar platforms like EA and Epic. Our model is well suited to be applied here without much adjustment. A big issue with public domain reviews is that sarcasm in the user base leaves an overwhelming amount of irrelevant reviews which can strongly mislead a potential customer who is not well versed with the game context, our model tries to address this.

4.3 Forum Moderation - Garnet

An area of concern in digital spaces is low quality responses in forums such as website forums or reddit style posting platforms. The model is currently specialized in gaming language so forums that dealt with other topics would need training on a broader language set to maintain current accuracy levels. Likewise certain features such as 'time_played_at_review' no longer apply so other applicable user features could be used in-place. However with some additional training the model and methodology could be extended to this space

5. Conclusion

We trained a model to determine the relevancy of reviews about games on the Steam store. The model was trained on 2000 rows of data which were manually labeled into categories of relevant or irrelevant. The reviews were processed to eliminate noise and enhanced using a bag-of-words and TF-IDF. Several classification models were trained on the data which had accuracies from the mid 70's to the mid 80's. The better performing models were further improved using a grid search to tune the hyper-parameters, which resulted in a small improvement in accuracy. Finally the model was trained using additional data about the review/reviewers like upvotes, funny tags, and hours played. Including these features lowered the overall accuracy but also decreased the number of false relevant predictions, which is helpful for this scenario.

After training, a Linear SVC version of SVM was the most accurate model with an F1-Score of 86%. The majority of the misclassifications were labeling relevant reviews as irrelevant, with about 2/3's as many of the opposite errors. The reasons for these misclassifications mainly were mainly a result of three issues: High-effort sarcasm, negative words in relevant reviews, and overfitting due to unique words. Future improvements could be made to reduce these errors by expanding the list of stopwords and improving the quality of preprocessing to improve the quality of the features. This was an excellent experience to take something of interest to the group and see how a machine learning model can be used to improve the service.

References

[1] A. Muller, “Scraping steam user reviews,” *Medium*, 17-Apr-2021. [Online]. Available: https://andrew-muller.medium.com/scraping-steam-user-reviews-9a43f9e38c92. [Accessed: 10-Dec-2022].

[2] F. Berhane, “Machine Learning with Text in PySpark”, *Data Science Enthusiast*, [Online]. Available: https://datascience-enthusiast.com/Python/PySpark_ML_with_Text_part1.html. [Accessed: 10-Dec-2022].

[3] Apache Spark, “Classification and regression” A*pache Spark*, [Online] Available: https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier. [Accessed: 12-Dec-2022].

[4] Apache Spark, “LinearSVC,” A*pache Spark*, [Online] Available: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark ml.classification.LinearSVC.html. [Accessed: 12-Dec-2022].