Using Machine Learning and Natural Language Processing to Predict Sentiment of IMDb User Reviews

Samuel Sofela
University of Calgary, Samuel.sofela@ucalgary.ca
Chris DiMattia
University of Calgary, christopher.dimattia@ucalgary.ca
Sam Rainbow
University of Calgary, sam.rainbow@ucalgary.ca

SUMMARY OF CONTRIBUTION

1) Data Collection – All three members labelled the exact same set of 1000 user movie reviews from IMDb.
2) Each team member was responsible for a different type of NLP input. Chris was responsible for bi-gram and the combined feature set, Samuel Bag-of-words (unigram), and Sam TF-IDF. Each member respective Databricks published workbook can be found below:
    a. Chris – Chris_DB-combined, Chris_DB-bigram
    b. Samuel – Samuel_DB
    c. Sam – Sam_DB

    It should be noted that the majority of the original code was created by all three members in unison and then split up for specific feature sets, hence the similarity in code amongst all DataBricks notebooks.

    For the writeup all team members wrote various sections and reviewed and edited all sections. In particular:

    - Chris was responsible for Introduction: Motivation, Discussion: Marketing strategy, Results: Data Preprocessing, Naive Bayes, Conclusion
    - Samuel was responsible for Abstract, Discussion: YouTube comments, Results: Data Labeling, Logistic Regression
    - Sam was responsible for Introduction: Background, Results: Misclassification, Discussion: Early Access Review, Results: Random Forest.

DATABRICK NOTEBOOKS

Sentiment Analysis using Bag of Words: https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/4958791141435139/840977525057 01/7171596124991136/latest.html

Sentiment Analysis using TF-IDF: https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/5215055879510205/1301799772487842/3090267558923303/latest.html

Sentiment Analysis using bi-grams: https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/8667545747930695/4207301225666013/2110337704883208/latest.html

Sentiment Analysis using Bag of Words + TF-IDF + bi-grams: https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/8667545747930695/3082022085872169/2110337704883208/latest.html

1. Abstract

Context
The increasing influence of social media in our everyday lives continues to highlight the importance of customer feedback. In the movie industry, the degree of viewership of a movie or television show is often determined by sentiments of the early reviews received by the movie. An overall negative user review can mar the performance of a movie on the big screen while a positive review can propel it to being a blockbuster. With the plethora of textual data used in movie reviews, there is a great opportunity to utilize machine learning models that can extract sentiment from reviews and apply the results to guide the decision making of the production, advertising, marketing, and funding of films, news and media.

Objective
The objective of this paper is to explore different machine learning models, feature sets and preprocessing techniques and to determine the best hyper parameters for models to predict the sentiments of movie reviews obtained by scraping the IMDb website.

Method
1000 movie reviews were collected from IMDb website through web scraping using the BeautifulSoup library. The sentiments of the reviews were manually labeled by each team member as positive, negative or neutral. The reviews were cleaned and tokenized. Lemmatization and Snowball stemming were utilized in preprocessing the tokens. To vectorize the tokens, four techniques were explored: Bag of Words, Term Frequency-Inverse Document Frequency (TF-IDF), bi-grams and a combination of these three vectorization techniques. Subsequently, machine learning classification models were investigated to find the optimal hyperparameter for accurate classification of the sentiments in the review. The models explored were: Logistic Regression, Naive Bayes and Random Forest Classifier. The best model was then used to classify user reviews and misclassified reviews were evaluated.

Results
Our results revealed that the choice of preprocessing technique, lemmatization or snowball stemming, had little impact on the performance of the models. Across the three models explored, TF-IDF outperformed the other vectorization techniques, Bag of Words, bi-gram, and a combination of the three techniques. Overall, Naive Bayes model on TF-IDF vectors produced the best F1-score of 0.705.

Conclusion
In conclusion, this study highlights the importance of hyper parameter tuning, model selection and feature set selection with TF-IDF having a significantly higher validation score than the other feature sets. The study explored various preprocessing techniques, vectorization techniques, and

machine learning models to determine the optimal approach for sentiment analysis of movie reviews. The results showed that the Naive Bayes was the best performing model with an F1-score of 0.705 and was also the least computationally expensive. It was also discovered that a dataset of 1000 samples was likely a limiting factor in providing accurate results in that even small changes to the hyper parameters or the model seed could greatly influence the model accuracy. Furthermore, the study found that the choice between stemming and lemmatization had little impact on the model's performance. Overall, this study provides a valuable starting point for further development into researching sentiment analysis for movie related text data.

2. Introduction

2.1 Motivation

Data analysis plays a major role in shaping and advancing our society by driving innovation, making the economy more efficient, informing governments, influencing business and individual decisions, and advancing research in every scientific field. A major barrier to further advancement is that the majority of the data that is produced is textual, while computers struggle to analyze numerical data and so NLP (natural language processing) has become a major focus of software engineers and computer scientists. One of the most studied NLP problems is sentiment analysis which is the use of NLP models to identify, extract and quantify a subjective aspect of textual data such as intensity (bad vs horrible) or emotion (anger, sadness, etc) but most often it's used to identify emotional polarity as positive, neutral or negative [1,2]. The ability to identify which preprocessing methods, models and techniques lead to the best sentiment models is critical to advancing NLP and tools which can be used in a wide variety of fields such as election prediction [3], consumer sentiment [4], healthcare feedback and financial sentiment about the economy [5] or a company to name a few.

2.2 Background

The film industry is a multi-billion dollar industry that consistently struggles to determine if a movie will be profitable and why it may receive the level of success it does because the reception and interest of films is highly subjective and not well defined [6]. Before the advent of YouTube, Twitter and other social media platforms movie producers had to rely on a limited amount of feedback and even if they were able to extra large amounts of feedback they still had the challenge of processing a large corpus of textual data.

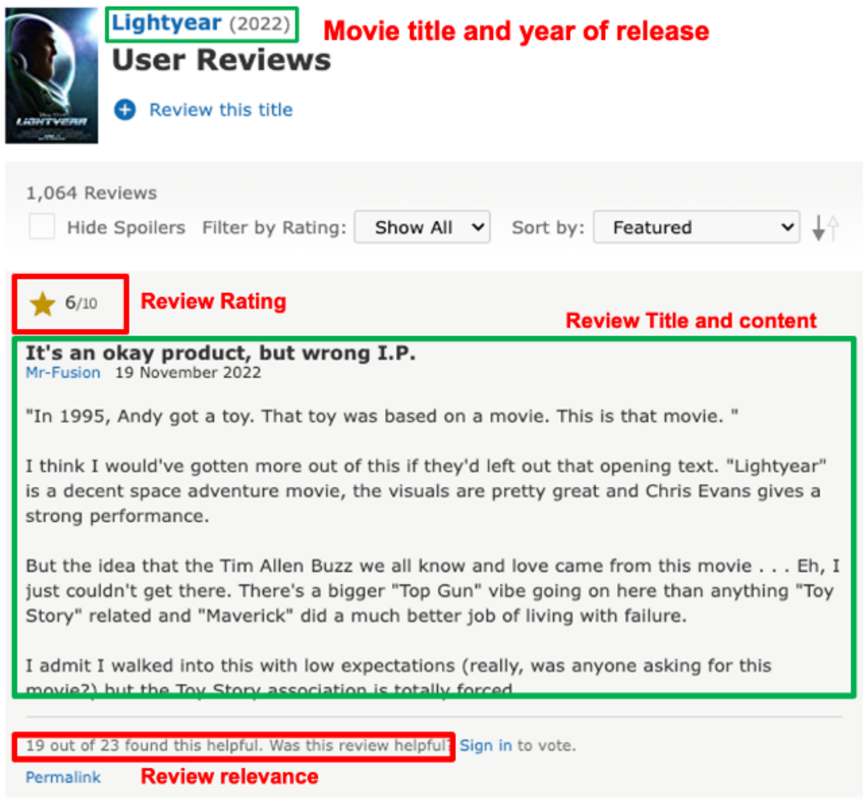
In this project we extracted and labeled 1000 movie reviews from IMDb and created a NLP model that can perform sentiment analysis for movie reviews. The movies were all released in 2022 and are from a wide array of genres. After labeling was completed, the month of the review, movie genre and textual data from the review was preprocessed to be viable for model input and featurization. The movie reviews were broken into multiple feature sets including bag of words, bi-gram, TF-IDF and a combination of the three for a total of 4 feature sets. After featurization, all the feature sets were used as inputs into several ML models including Naive Bayes, Logistic Regression and Random Forest. The data was then split into an 80/20 training/testing set. Hyper parameter tuning was performed with 5-fold cross validation to identify optimal parameters after which the model was then tested against the unseen test set. The final validation metrics were a weighted f1 score, weighted recall and weighted precision.

3. Results

3.1 Data Labeling

Approach

One thousand movie data was collected from the IMDb website. We collected reviews, the title of the review, review ratings, release year and genre of the movie. To make up the 1000 reviews required for this project, we collected 25 reviews each from 40 movies released in 2022. The movies released in 2022 were chosen to get the most recent data. This data was collected by scraping the movie pages on the IMDb website using the BeautifulSoup library. Data was collected from movies that had at least 50 reviews. On the review page, each review is voted as helpful or otherwise by website visitors. In order to avoid "troll" or useless reviews which could introduce bias into our model, we used the top helpful review. This implies that the top 25 reviews voted as the most helpful were used. After collection, the sentiment of the review was labeled using sentiment values -1, 0 or 1 which represent negative, neutral, and positive sentiments respectively. The title of the review and content of the review were used in labeling the sentiment of the review and each group member labeled all 1000 reviews. A majority rating system was used to decide the final review. This means that if a review was given the same ratings by two or more raters, that rating is used as the final rating for that review. In cases where the three raters rated the review differently, the review was revisited and collectively rated.



It's an okay product, but wrong I.P.
Mr-Fusion 19 November 2022

"In 1995, Andy got a toy. That toy was based on a movie. This is that movie. "

I think I would've gotten more out of this if they'd left out that opening text. "Lightyear" is a decent space adventure movie, the visuals are pretty great and Chris Evans gives a strong performance.

But the idea that the Tim Allen Buzz we all know and love came from this movie . . . Eh, I just couldn't get there. There's a bigger "Top Gun" vibe going on here than anything "Toy Story" related and "Maverick" did a much better job of living with failure.

I admit I walked into this with low expectations (really, was anyone asking for this movie?) but the Toy Story association is totally forced

19 out of 23 found this helpful. Was this review helpful?

Figure 1: Image showing snapshot of the review page on IMDb for the movie, Lightyear. The colored boxes highlight features that were used to scrape data from the website.

Results

After labeling the movie reviews, the degree of agreement between the raters was evaluated using two inter-rater agreement indices: Cohen-Kappa and Fleiss Kappa agreement score. Cohen-Kappa score is used to evaluate the agreement between 2 raters. The table below summarizes the agreement among the 3 raters.

Table 1: Summary of inter-rater agreement scores using Cohen-Kappa.

Cohen-Kappa Agreement percentage	
Rater 1 and Rater 2	72%
Rater 1 and Rater 3	68%
Rater 2 and Rater 3	66%

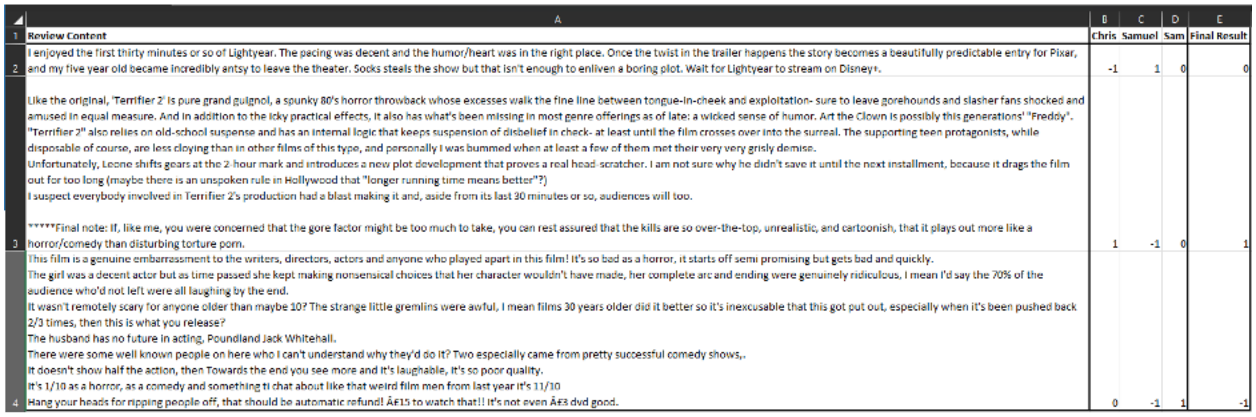
All the percentages are above 60%, which indicates substantial agreement between the three group members [7].

The Fleiss-Kappa was also used because it can be used to evaluate inter-rater agreement for more than two raters. The Fleiss-Kappa agreement score was 69% which indicated substantial agreement among the three team members [8] and aligns with the Cohen-Kappa score.

Out of the 1,000 reviews that were labeled, the team members had different ratings for 14 reviews. The reviews were reviewed collectively, and individual sentiment labels were revised to reach an agreement. The main reason for the different rating was due to ambiguous and inconclusive reviews by the author. For instance, a review says:

> *"I enjoyed the first thirty minutes or so of Lightyear. The pacing was decent and the humor/heart was in the right place. Once the twist in the trailer happens the story becomes a beautifully predictable entry for Pixar, and my five year old became incredibly antsy to leave the theater."*

The reviewer enjoyed the first 30 mins but didn't like the predictability of the rest of the movie. This is quite ambiguous, and any sentiment would be suitable for this review. This review was eventually given a neutral sentiment after collective discussion.



	A	B	C	D	E
1	**Review Content**	**Chris**	**Samuel**	**Sam**	**Final Result**
2	I enjoyed the first thirty minutes or so of Lightyear. The pacing was decent and the humor/heart was in the right place. Once the twist in the trailer happens the story becomes a beautifully predictable entry for Pixar, and my five year old became incredibly antsy to leave the theater. Socks steals the show but that isn't enough to enliven a boring plot. Wait for Lightyear to stream on Disney+.	-1	1	0	0
3	Like the original, 'Terrifier 2' is pure grand guignol, a spunky 80's horror throwback whose excesses walk the fine line between tongue-in-cheek and exploitation- sure to leave gorehounds and slasher fans shocked and amused in equal measure. And in addition to the icky practical effects, it also has what's been missing in most genre offerings as of late: a wicked sense of humor. Art the Clown is possibly this generations' "Freddy". "Terrifier 2" also relies on old-school suspense and has an internal logic that keeps suspension of disbelief in check- at least until the film crosses over into the surreal. The supporting teen protagonists, while disposable of course, are less cloying than in other films of this type, and personally I was bummed when at least a few of them met their very very grisly demise. Unfortunately, Leone shifts gears at the 2-hour mark and introduces a new plot development that proves a real head-scratcher. I am not sure why he didn't save it until the next installment, because it drags the film out for too long (maybe there is an unspoken rule in Hollywood that "longer running time means better"?) I suspect everybody involved in Terrifier 2's production had a blast making it and, aside from its last 30 minutes or so, audiences will too. *****Final note: If, like me, you were concerned that the gore factor might be too much to take, you can rest assured that the kills are so over-the-top, unrealistic, and cartoonish, that it plays out more like a horror/comedy than disturbing torture porn.	1	-1	0	1
4	This film is a genuine embarrassment to the writers, directors, actors and anyone who played apart in this film! It's so bad as a horror, it starts off semi promising but gets bad and quickly. The girl was a decent actor but as time passed she kept making nonsensical choices that her character wouldn't have made, her complete arc and ending were genuinely ridiculous, I mean I'd say the 70% of the audience who'd not left were all laughing by the end. It wasn't remotely scary for anyone older than maybe 10? The strange little gremlins were awful, I mean films 30 years older did it better so it's inexcusable that this got put out, especially when it's been pushed back 2/3 times, then this is what you release? The husband has no future in acting, Poundland Jack Whitehall. There were some well known people on here who I can't understand why they'd do it? Two especially came from pretty successful comedy shows,. It doesn't show half the action, then Towards the end you see more and it's laughable, it's so poor quality. It's 1/10 as a horror, as a comedy and something ti chat about like that weird film men from last year it's 11/10 Hang your heads for ripping people off, that should be automatic refund! Â£15 to watch that!! It's not even Â£3 dvd good.	0	-1	1	-1

Figure 2: Example of reviews that had different ratings from each team member.

After collective resolving differing ratings, the data had 501 positive reviews, 170 neutral reviews and 329 negative reviews. Below is a distribution of the individual ratings and final rating:

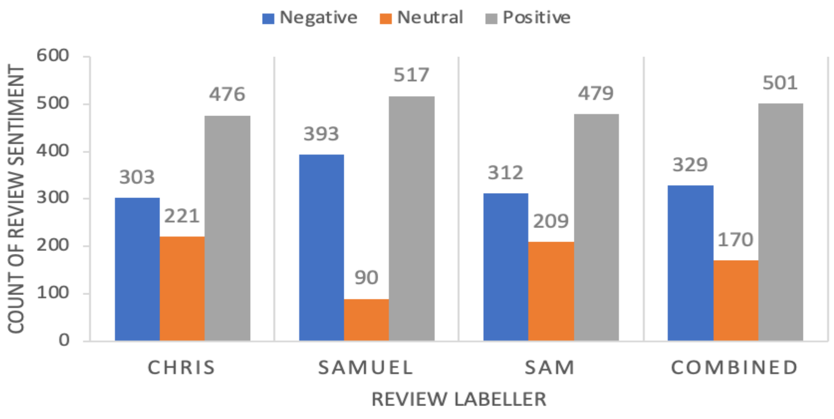


Figure 3: Bar chart showing the count of review sentiment for each reviewer and the final rating.

3.2 Pre-processing the Data

Approach

There were two main categories of data: text data from a movie review and month and genre data. The text data was preprocessed in two main stages. In the first stage the data was made ready for input into a model via six main steps:

- Concatenating the movie review title with the movie review.
- General cleaning which includes changing uppercase to lowercase, removing non-alphabetic characters, removing punctuation, and removing white spaces while tokenizing words.
- Removal of stop words.

- Lemmatizing or stemming words. Lemmatizing was done using a pos tagger which involves tagging a word as an adjective, verb, noun, or adverb and then applying lemmatizing to it.
    - The lemmatizing used WordNet as its lexical database.
    - Snowball stemming was used for stemming.

After the data was preprocessed then feature extraction was performed by converting the data into four feature sets including:

- Bag of words (uni-gram)
- Bi-gram
- TF-IDF (term frequency-inverse document frequency)
- Combining above features for some but not all models

All feature sets were vectorized using PySpark's built in CountVectorizer function. The CountVectorizer was chosen over the PySparks hash function HashingTF. The reason being that HashingTF has the potential for collisions and while CountVectorizer is computationally more expensive it was important to eliminate data lost which would reduce the model's ability to generalize, especially given that the types of collisions would be random (as opposed to stemming or lemmatization).

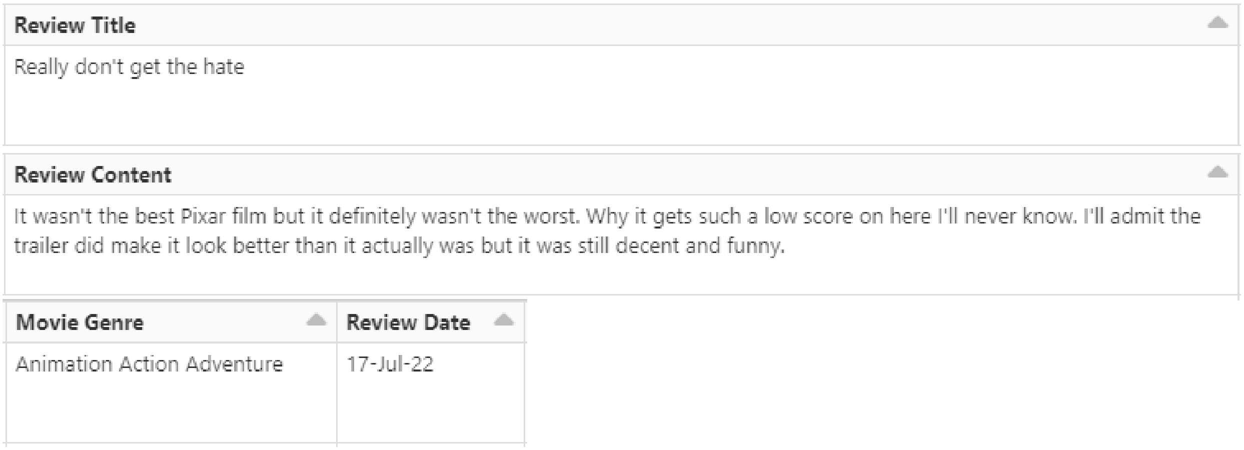
The genre was also included as a feature in the ML models and concatenated with the textual feature sets. They were processed in the same procedure as above but without the stop word removal. The month of the review was also included in the feature set. The month was converted from its text into a number (eg. January → 1, February → 2, etc) and then vectorized.

The sentiment for the 1000 records was converted from 1, 0, -1 to 2, 1 and 0 to satisfy PySparks evaluator which does not accept negative values.

Every processing step was accomplished within PySpark and with the usage of the nltk library, embedded PySpark functions or generic PySpark programming.

Results

The results of preprocessing can be seen below for a single movie review.

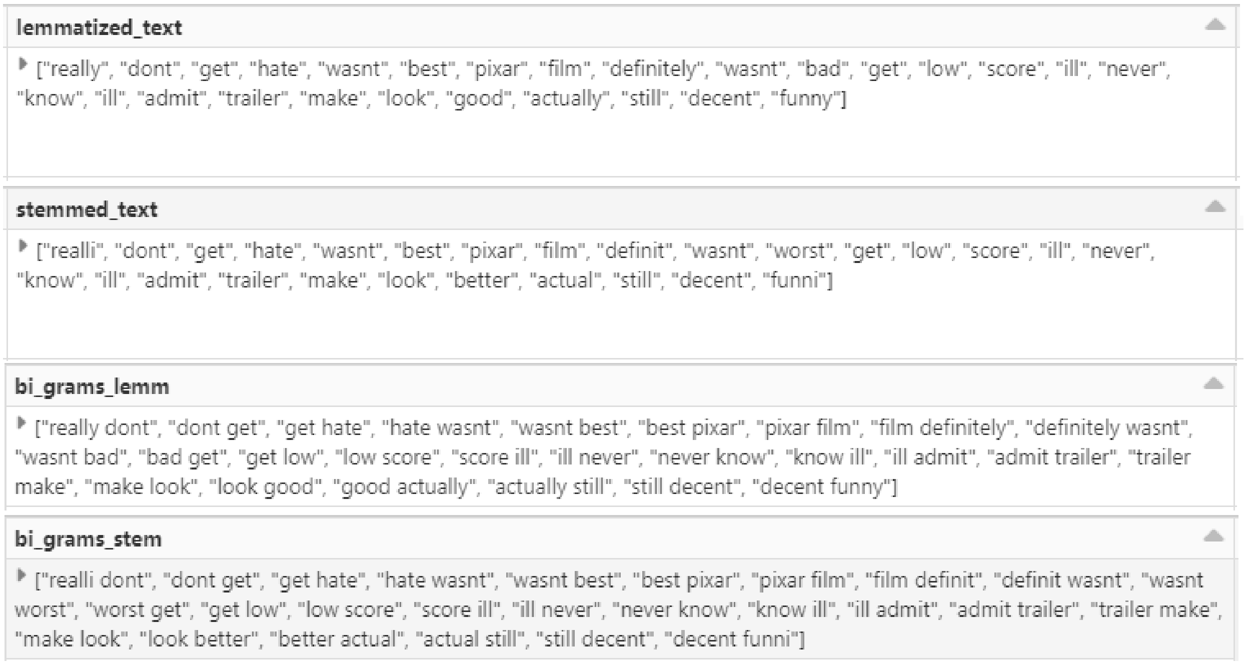


Review Title
Really don't get the hate

Review Content
It wasn't the best Pixar film but it definitely wasn't the worst. Why it gets such a low score on here I'll never know. I'll admit the trailer did make it look better than it actually was but it was still decent and funny.

Movie Genre	Review Date
Animation Action Adventure	17-Jul-22

Figure 4: Unprocessed data of movie review:



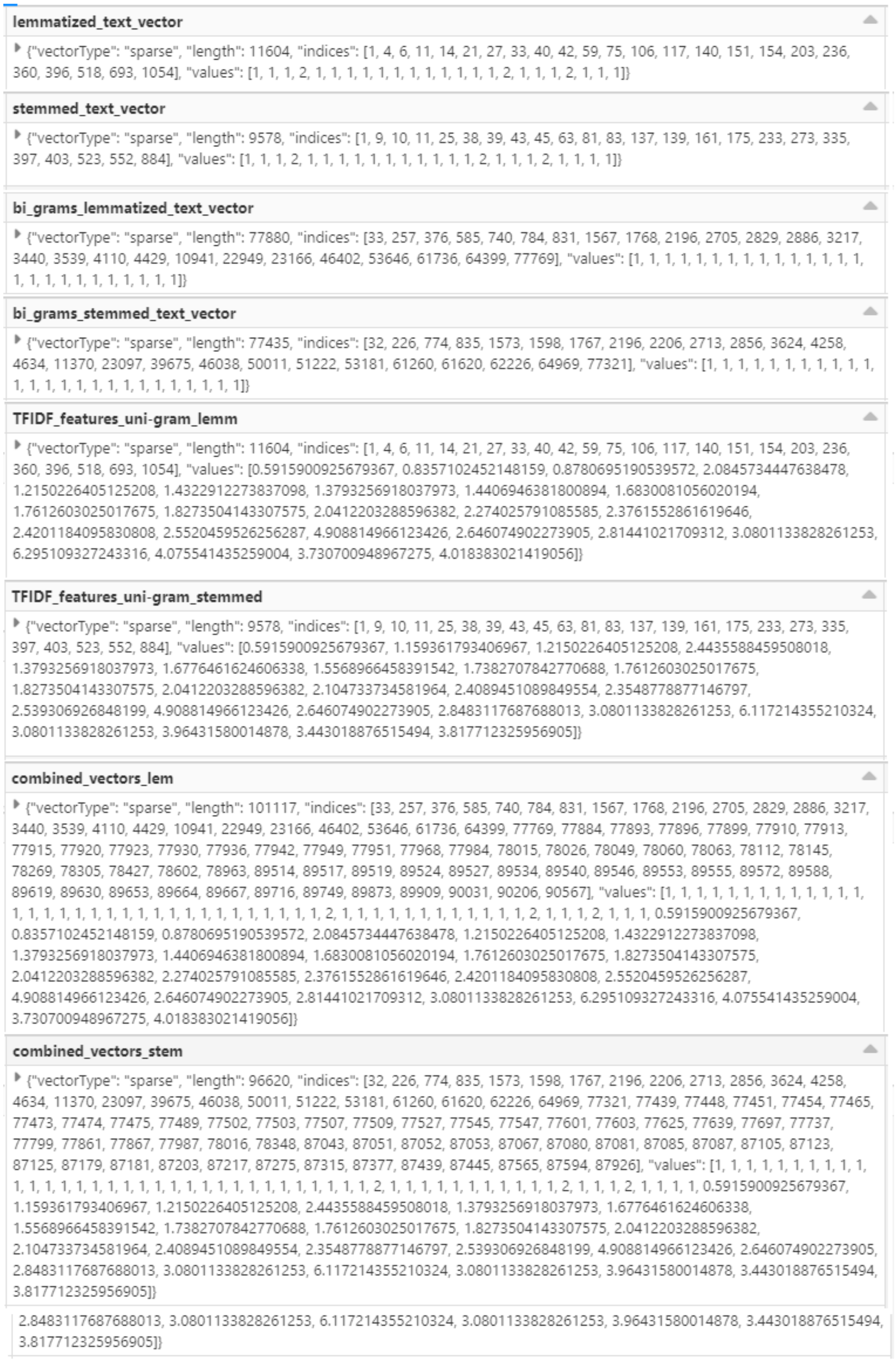
lemmatized_text
["really", "dont", "get", "hate", "wasnt", "best", "pixar", "film", "definitely", "wasnt", "bad", "get", "low", "score", "ill", "never", "know", "ill", "admit", "trailer", "make", "look", "good", "actually", "still", "decent", "funny"]

stemmed_text
["realli", "dont", "get", "hate", "wasnt", "best", "pixar", "film", "definit", "wasnt", "worst", "get", "low", "score", "ill", "never", "know", "ill", "admit", "trailer", "make", "look", "better", "actual", "still", "decent", "funni"]

bi_grams_lemm
["really dont", "dont get", "get hate", "hate wasnt", "wasnt best", "best pixar", "pixar film", "film definitely", "definitely wasnt", "wasnt bad", "bad get", "get low", "low score", "score ill", "ill never", "never know", "know ill", "ill admit", "admit trailer", "trailer make", "make look", "look good", "good actually", "actually still", "still decent", "decent funny"]

bi_grams_stem
["realli dont", "dont get", "get hate", "hate wasnt", "wasnt best", "best pixar", "pixar film", "film definit", "definit wasnt", "wasnt worst", "worst get", "get low", "low score", "score ill", "ill never", "never know", "know ill", "ill admit", "admit trailer", "trailer make", "make look", "look better", "better actual", "actual still", "still decent", "decent funni"]

Figure 5: Movie review after concatenating review title and review content, cleaning, removal of stop words and tokenization or stemming for bag of words and bi-gram.



lemmatized_text_vector

```
{"vectorType": "sparse", "length": 11604, "indices": [1, 4, 6, 11, 14, 21, 27, 33, 40, 42, 59, 75, 106, 117, 140, 151, 154, 203, 236,
360, 396, 518, 693, 1054], "values": [1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1]}
```

stemmed_text_vector

```
{"vectorType": "sparse", "length": 9578, "indices": [1, 9, 10, 11, 25, 38, 39, 43, 45, 63, 81, 83, 137, 139, 161, 175, 233, 273, 335,
397, 403, 523, 552, 884], "values": [1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1]}
```

bi_grams_lemmatized_text_vector

```
{"vectorType": "sparse", "length": 77880, "indices": [33, 257, 376, 585, 740, 784, 831, 1567, 1768, 2196, 2705, 2829, 2886, 3217,
3440, 3539, 4110, 4429, 10941, 22949, 23166, 46402, 53646, 61736, 64399, 77769], "values": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

bi_grams_stemmed_text_vector

```
{"vectorType": "sparse", "length": 77435, "indices": [32, 226, 774, 835, 1573, 1598, 1767, 2196, 2206, 2713, 2856, 3624, 4258,
4634, 11370, 23097, 39675, 46038, 50011, 51222, 53181, 61260, 61620, 62226, 64969, 77321], "values": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

TFIDF_features_uni-gram_lemm

```
{"vectorType": "sparse", "length": 11604, "indices": [1, 4, 6, 11, 14, 21, 27, 33, 40, 42, 59, 75, 106, 117, 140, 151, 154, 203, 236,
360, 396, 518, 693, 1054], "values": [0.5915900925679367, 0.8357102452148159, 0.8780695190539572, 2.0845734447638478,
1.2150226405125208, 1.4322912273837098, 1.3793256918037973, 1.4406946381800894, 1.6830081056020194,
1.7612603025017675, 1.8273504143307575, 2.0412203288596382, 2.274025791085585, 2.3761552861619646,
2.4201184095830808, 2.5520459526256287, 4.908814966123426, 2.646074902273905, 2.81441021709312, 3.0801133828261253,
6.295109327243316, 4.075541435259004, 3.730700948967275, 4.018383021419056]}
```

TFIDF_features_uni-gram_stemmed

```
{"vectorType": "sparse", "length": 9578, "indices": [1, 9, 10, 11, 25, 38, 39, 43, 45, 63, 81, 83, 137, 139, 161, 175, 233, 273, 335,
397, 403, 523, 552, 884], "values": [0.5915900925679367, 1.159361793406967, 1.2150226405125208, 2.4435588459508018,
1.3793256918037973, 1.6776461624606338, 1.5568966458391542, 1.7382707842770688, 1.7612603025017675,
1.8273504143307575, 2.0412203288596382, 2.104733734581964, 2.4089451089849554, 2.3548778877146797,
2.539306926848199, 4.908814966123426, 2.646074902273905, 2.8483117687688013, 3.0801133828261253, 6.117214355210324,
3.0801133828261253, 3.96431580014878, 3.443018876515494, 3.817712325956905]}
```

combined_vectors_lem

```
{"vectorType": "sparse", "length": 101117, "indices": [33, 257, 376, 585, 740, 784, 831, 1567, 1768, 2196, 2705, 2829, 2886, 3217,
3440, 3539, 4110, 4429, 10941, 22949, 23166, 46402, 53646, 61736, 64399, 77769, 77884, 77893, 77896, 77899, 77910, 77913,
77915, 77920, 77923, 77930, 77936, 77942, 77949, 77951, 77968, 77984, 78015, 78026, 78049, 78060, 78063, 78112, 78145,
78269, 78305, 78427, 78602, 78963, 89514, 89517, 89519, 89524, 89527, 89534, 89540, 89546, 89553, 89555, 89572, 89588,
89619, 89630, 89653, 89664, 89667, 89716, 89749, 89873, 89909, 90031, 90206, 90567], "values": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 0.5915900925679367,
0.8357102452148159, 0.8780695190539572, 2.0845734447638478, 1.2150226405125208, 1.4322912273837098,
1.3793256918037973, 1.4406946381800894, 1.6830081056020194, 1.7612603025017675, 1.8273504143307575,
2.0412203288596382, 2.274025791085585, 2.3761552861619646, 2.4201184095830808, 2.5520459526256287,
4.908814966123426, 2.646074902273905, 2.81441021709312, 3.0801133828261253, 6.295109327243316, 4.075541435259004,
3.730700948967275, 4.018383021419056]}
```

combined_vectors_stem

```
{"vectorType": "sparse", "length": 96620, "indices": [32, 226, 774, 835, 1573, 1598, 1767, 2196, 2206, 2713, 2856, 3624, 4258,
4634, 11370, 23097, 39675, 46038, 50011, 51222, 53181, 61260, 61620, 62226, 64969, 77321, 77439, 77448, 77451, 77454, 77465,
77473, 77474, 77475, 77489, 77502, 77503, 77507, 77509, 77527, 77545, 77547, 77601, 77603, 77625, 77639, 77697, 77737,
77799, 77861, 77867, 77987, 78016, 78348, 87043, 87051, 87052, 87053, 87067, 87080, 87081, 87085, 87087, 87105, 87123,
87125, 87179, 87181, 87203, 87217, 87275, 87315, 87377, 87439, 87445, 87565, 87594, 87926], "values": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 0.5915900925679367,
1.159361793406967, 1.2150226405125208, 2.4435588459508018, 1.3793256918037973, 1.6776461624606338,
1.5568966458391542, 1.7382707842770688, 1.7612603025017675, 1.8273504143307575, 2.0412203288596382,
2.104733734581964, 2.4089451089849554, 2.3548778877146797, 2.539306926848199, 4.908814966123426, 2.646074902273905,
2.8483117687688013, 3.0801133828261253, 6.117214355210324, 3.0801133828261253, 3.96431580014878, 3.443018876515494,
3.817712325956905]}
```

```
2.8483117687688013, 3.0801133828261253, 6.117214355210324, 3.0801133828261253, 3.96431580014878, 3.443018876515494,
3.817712325956905]}
```

Figure 6: All feature sets of the movie review after vectorization including bag of words (lemmatized_text_vector), bi-gram, TF-IDF and a combination of the previous three.

The corpus size increased significantly when using a bi-gram feature set or when combining multiple feature sets together. It should also be noted that stemming reduced the word count more than lemmatization. Below is a table indicating the difference in vector size for each feature set:

- Bag of words and TF-IDF. Lemmatization: 11,604. Stemming: 9,678
- Bi-gram. Lemmatization: 77,880. Stemming: 77,435
- Combined. Lemmatization: 101,117. Stemming: 96,620

This indicates that stemming may but will not nessisarily produce better results due to more concentrated corpus, but there may be lose of information in which the model may be less able to generalize. It was found that both methods produced similar validation scores. The table below contains a comparision for the bi-gram and combined feature sets difference in pre-processing validation scores for Naive Bayes. For the full comparision of all models see the above DataBricks notebooks.

Table 2:Comparision between stemming and lemmatization for Naive Bayes f1 scores

	Smoothing- Naive Bayes	**F1 - Score (lemmatization)**	**F1-Score (stemming)**
Bi-Gram	0.0	0.164	0.165
	7.0	0.513	0.524
	10.0	0.496	0.506
Combined	0.0001	0.583	0.613
	0.1	0.629	0.621
	1.0	0.623	0.605
	5.0	0.440	0.447

3.3 Hyper Parameters Effect on Model Performance

It should be noted that this project attempted to optimize its selection of feature sets and models, so not all models were run on all feature sets due to the extreme size of many of the feature sets and the difference in performance between models. In short it was deemed unproductive to run logistic regression and random forest models on the bi-gram and combined feature sets based on the comparison between other feature sets and models.

To optimize the model selection all the models were run on the bag of words feature set which includes Naive Bayes, Logistic Regression and Random Forest. After determining that the Naive Bayes was the best model due to its strong validation score and fast computation time, it was applied to all the feature sets (TF-IDF, bi-gram and the combination). It was then determined that the strongest feature set was TF-IDF. All models (Naive Bayes, Random Forest & Logistic Regression) were then run on the TF-IDF feature set to find the best model. The results of which are presented below.

3.3.1 Naive Bayes

Approach

The main parameter chosen for tuning with Naïve Bayes was alpha (called nb.smoothing in PySpark) and the type of Naïve Bayes model selected was multinomial as opposed to Bernoulli. We did not consider Bernoulli because the Bernoulli model assumes a Boolean feature set whereas multinomial assumes the features follow a multinomial distribution (count of words in a text) which our data set does.

It should also be noted that our models are optimized for a weighted f-score. While not a hyper parameter in the strict sense it is a parameter within the model that changes how the model optimizes and directly impacts the final f-score on the testing set because weighted the f-score accounts for unbalanced datasets. A normal f-score could not account for the unbalanced data and would have biased the results.

The following hyper parameters can be tuned but were deemed not necessary to tune for the following reasons:

- Prior probabilities. We did not modify this because the default model works under the assumption that each class is equally as likely which we assume to be true. There is no strong indication that movie reviews are more likely positive or negative and while our data set was skewed, we did not compile statistics from a dataset large enough to confidently change the prior probability within the Naive Bayes model.
- Binning. We did not apply binning because it was not viable given the large corpus of words and complexity involved in properly binning words.
- Threshold tuning. Multinomial threshold tuning is not supported in PySpark and to attempt it would require a one-vs-all approach which would not work because the data is unbalanced. A one-vs-all approach may have worked had more than 1000 records been used and there was freedom to eliminate records and balance the dataset without affecting the final results but eliminating records on an already small dataset would adversely impact results and the model's ability to generalize.

The alpha or smoothing function was varied by starting with a logarithmic scale such as (0.1, 1, 10, 100) to identify the best scale. After an appropriate scale was identified the range was narrowed until a local maximum was found. For all the models except bi-gram it was found that the best alpha value was roughly 0.1-0.5. For the bi-gram the optimal alpha was found to be 7. A higher alpha indicates that more emphasis is placed on the prior probabilities of the classes and less on the likelihood probabilities of the features given the class, so a higher alpha lead to a "smoother" or more general model (underfitting). This is suspected to be the case because the bi-gram model has a far larger range and sparsity of features to bag of words, TF-IDF and the combined feature set so overfitting would be highly selected against during cross validation.

Results

The below results show the weight f-1, weighted recall and weighted precision score. It should be noted that these values may be slightly different than those found when running the code due to the random seed selected.

Table 3: Validation scores for all feature sets for Naive Bayes model

	Smoothing	**F1-Score**	**Precision**	**Recall**
Bag of words	0.00	0.206	0.341	0.339
	0.10	0.651	0.652	0.649
	0.15	0.648	0.649	0.648
	0.20	0.646	0.645	0.646
	0.50	0.639	0.662	0.629
	1.00	0.626	0.679	0.650
	10.00	0.333	0.493	0.579
	20.00	0.326	0.490	0.579
Bi-Gram	0.0	0.164	0.110	0.331
	7.0	0.513	0.484	0.582
	10.0	0.496	0.489	0.575
TF-IDF	0.0	0.218	0.559	0.347
	0.2	0.696	0.696	0.698
	0.5	0.705	0.711	0.701
	0.6	0.703	0.709	0.698
	0.8	0.702	0.710	0.698
Combined	0.0001	0.583	0.584	0.618
	0.1	0.629	0.624	0.638
	1.0	0.623	0.623	0.665
	5.0	0.440	0.553	0.539

3.3.2 Logistic Regression

Approach

In logistic regression, two hyperparameters were considered: elasticNetParam which determines the ratio of L1 to L2 regularization to be used, and regParam which determines the degree of regularization. The regParam performs the same function as C in regularization. For elasticNetParam, we used the default value of zero which is L2 regularization because L1 regularization is used when few of the features are considered important which we assumed to be false given the large text corpus. The regularization parameter (regParam) was varied on a logarithmic scale. The ParamGridBuilder library was used with the CrossValidator library to evaluate the model and determine the best hyperparameter value for the model. For the Bag of Words set, regParam values of 0.00001, 0.001, 0.005, 0.01, 0.1, 1 and 10 were used. For TF-IDF, regParam values of 0.005, 0.008, 0.009, 0.01, and 0.011 were used. The weighted f1-score, weighted precision and weighted recall was evaluated for all cases and the f1-score was selected as the primary validation metric.

Result

Logistic Regression models were investigated using datasets that were featurized using Bag of Words and TF-IDF. With Bag of Words, the regParam of 0.001 gave the best F1 score of 0.648 while the corresponding precision and recall values were 0.697 and 0.673 respectively. As shown in Table 4, other hyperparameter values for regParam resulted in lower f1 scores. Movie review data that were featurized using TF-IDF resulted in higher f1-score of 0.679 for a regParam value of 0.008. The corresponding precision and recall were 0.701 and 0.739 respectively.

Table 4: Summary of weighted f1-score, precision and recall obtained for review data processed as bag of words using logistic regression.

Alpha	F1-Score	Precision	Recall
0.0001	0.641	0.691	0.661
0.001	0.648	0.697	0.673
0.005	0.643	0.693	0.667
0.01	0.636	0.686	0.660
0.1	0.634	0.689	0.664
1	0.601	0.672	0.569
10	0.429	0.544	0.544

Table 5: Summary of weighted F1-score, precision and recall obtained for review data processed using TF-IDF with logistic regression.

Alpha	F1-Score	Precision	Recall
0.005	0.673	0.695	0.732
0.008	0.679	0.701	0.739
0.009	0.678	0.728	0.739
0.01	0.675	0.725	0.735
0.011	0.672	0.722	0.732

3.3.3 Random Forest

Approach

For the random forest model two parameters were selected for tuning: max depth and number of trees. Max depth was chosen because it has several benefits, mainly that it directly controls model complexity so optimizing model complexity is achieved, it limits computation time by restricting how deep a tree can grow and it is easier to interpret results if analyzed. The number of trees was chosen because the larger the number of trees, the more accurate the model, but at the cost of increased computation time. Thus, this parameter is easy to tune because the number of trees can be increased until computational capacity or diminish returns stop further increases.

Other hyper parameters could be optimized such as the minimum number of samples required to create a split and maximum number of features, but these also control model complexity similar to max depth and given the already strained computation time it was deemed unnecessary for the scope of this project.

Results

Below are the weighted f1, weighted recall and weighted precision score for the TF-IDF and bag of words feature set. The bi-gram and combined feature set did not perform well with the naive bayes models and was not considered for future testing. This was decided as the random forest only provided marginal improvements based on the bag of words test set and was computationally expensive.

Bag of Words

Table 6: Summary of weighted F1-score, precision and recall obtained for review data processed using Bag of Words with random forest regression.

Num of trees			**Max Depth**		
F1	5	10	15	20	30
5	0.436	0.475	0.493	0.503	0.527
10	0.416	0.51	0.525	0.562	0.581
30	0.363	0.45	0.498	0.551	0.596
50	0.363	0.447	0.529	0.576	0.605
Recall	5	10	15	20	30
5	0.54	0.55	0.561	0.557	0.567
10	0.543	0.592	0.595	0.623	0.629
30	0.519	0.561	0.588	0.626	0.664
50	0.519	0.561	0.612	0.647	0.671
Precision	5	10	15	20	30
5	0.491	0.481	0.491	0.482	0.508
10	0.547	0.593	0.561	0.579	0.562
30	0.484	0.51	0.541	0.556	0.582
50	0.593	0.585	0.573	0.592	0.597

TF-IDF

Table 7: Summary of weighted F1-score, precision and recall obtained for review data processed using TF-IDF with random forest regression.

Num of Trees	**Max-Depth**		
F1	5	10	15
10	0.411	0.481	0.531
30	0.399	0.513	0.542
50	0.351	0.434	0.496
Recall	5	10	15
10	0.536	0.57	0.605
30	0.533	0.601	0.622
50	0.509	0.564	0.591
Precision			
10	0.654	0.502	0.612
30	0.586	0.579	0.578
50	0.472	0.564	0.556

3.4 Misclassifications

We determined that the best model was Naive Bayes using TF-IDF, month, and genre vectors as inputs. We then completed hyper parameter tuning via an iterative grid search and found the best performing model. This ended up being Naive Bayes with 0.5 smoothing and lemmatized text (rather than stemmed text) in the TF-IDF vector. Using these models, we predicted the sentiment of the testing set containing approximately 300 reviews. The model predicted a 0,1,2 for negative, neutral, and positive sentiment, respectively.

We were able to determine the misclassified sentiment of the model's prediction to our manually coded sentiment and found out of 272 reviews in the testing set the model misclassified 117 of them. Resulting in the model being correct approximately 57% of the time. Below we can see a breakdown of the results of the model on the testing set by incorrect prediction:

Table 8: Results of the misclassifications of the testing set predicted by Naive Bayes using TF-IDF, month, and genre as inputs.

	Model 0	***Model 1***	***Model 2***
Rater 0	-	26	11
Rater 1	16	-	7
Rater 2	23	34	-

The model struggled the most with predicting the neutral classification (rating of 1) making up 51% of the misclassifications. This is expected since movie reviews tend to be quite polarized – users either love or hate a movie. The few users who do have a neutral review of a movie typically talk about parts they like and parts they didn't like which is difficult for a model to understand context. Another explanation for the poor neutral prediction is that the target variable in the training set was skewed as seen in Figure 3.

The model was particularly good at predicting positive sentiment, which only represented ~15% of its misclassifications. The model was trained on mostly positive reviews (Figure 3) and as such it makes sense that this is its strongest prediction class. The polarizing nature of movie reviews also plays into it in that people who enjoyed a movie usually really enjoy a movie and are motivated to write a review.

The model predicted negative sentiment (rating of 0) with a misclassification rate of 33%. This is likely due to the skewed labeling mentioned previously.

After reviewing the misclassifications, we were able to come up with three main reasons why the model may have predicted the sentiment incorrectly. Of the three the most likely was due to "Difficult context by using double negatives to describe sentiment, slang, colloquialisms, or sarcasm.", followed by "Vocabulary is complex, and the model is likely not well to it due to the amount of data", and finally "Users review is contradictory saying they like/hate the movie and then talk about the parts they did/didn't like". When users have complex vocabulary, the model is not trained on a lot of this data and is likely why the model is not accurate at predicting sentiment from this. Similarly, when users use double negatives like "this was not a bad movie" it is very difficult for NLP to handle. Reviews that use slang ("this movie is bogus") can also be difficult to train a model on since they are not common words to indicate sentiment and vary a lot from user to user (i.e., the model is not exposed to these words enough).

Overall, the model would get much better at predicting sentiment of difficult types of user reviews by being exposed to more of these types of reviews, which could be done with large amounts of data.

4. Discussion

YouTube comments

The model developed in this project can be used to analyze sentiments of the audience of social media advertisements about the product being advertised. For instance, the trailer for a movie is typically released on social media platforms such as Youtube about 6 - 9 months before the movie's release date. Analyzing the comments made on the trailer using our model can give movie producers an insight on potential performance of a movie upon release. This will inform their decision on appropriation of the marketing budget. Positive sentiments would encourage a large marketing budget. This is not limited to only movie trailers but also any product to be released. This can be expanded to social feedback on the beta version of an application or a product prototype.

Early Access Reviews

Top influencers within an industry (such as movie critics in the film industry or international chefs in the food industry) typically get early access to a product before it is released to the public. A model can be used to analyze their reviews and help to manage expectations upon release of the product. Also, movies are usually produced in different cuts and sentiment analysis from early access reviews can be used to determine which cut has the potential to be more appealing to the audience.

Marketing Strategy

Film Studios can use sentiment analysis to determine which aspects of a movie resonates with the audience and focus their marketing efforts accordingly. For instance, sentiment analysis can reveal that viewers of drama movies are more interested in the plot of the movie than the graphics or acting performance. The marketing team can use such information to channel advertising campaigns with emphasis on the movie's plot. This would also inform directors of the aspect of a movie (plot) that should be focused on in subsequent releases of the movie or season of a TV show. Also, sentiment analysis can also be used to evaluate the performance of advertisement campaigns and determine which marketing efforts resonate more with the audience.

5. Conclusion

In conclusion, this study was successful in creating a machine learning model that could accurately predict user sentiment for textual movie reviews despite being trained on a relatively small data set of 1000 samples. The most successful model was identified as a Naive Bayes model with a TF-IDF feature set and an alpha value of 0.5 which was able to obtain a F1-score of 0.705. Furthermore, the study was successful in exploring different models and capturing the differences in how hyper parameters affect the model validation scores. This study was also able to confirm that the difference in F1 scores between stemming and lemmatization as a preprocessing technique was noticeable but not significant with an average difference of 0.5% between the two. The success of this study has implications for the film industry in that more sophisticated and well trained models have the opportunity to accurately predict sentiment for marketing, producing and advertising purposes ranging from helping producers identify which pre-screening cuts are the most liked, guiding financiers on expected returns from movies based on previous reviews and providing insight to advertisers on which ads are the most successful based on YouTube movie comments.

References

[1] Bravo-Marquez, F., Mendoza, M. and Poblete, B., 2014. Meta-level sentiment models for big social data analysis. *Knowledge-based systems*, *69*, pp.86-99.

[2] Yao, J., 2019, April. Automated sentiment analysis of text data with NLTK. In *Journal of Physics: Conference Series* (Vol. 1187, No. 5, p. 052020). IOP Publishing.

[3] Parackal, M., Mather, D. and Holdsworth, D., 2018. Value-based prediction of election results using natural language processing: A case of the New Zealand general election. *International Journal of Market Research*, *60*(2), pp.156-168

[4] Biswas, B., Sengupta, P., Kumar, A., Delen, D. and Gupta, S., 2022. A critical assessment of consumer reviews: A hybrid NLP-based methodology. *Decision Support Systems*, *159*, p.113799.

[5] Mishev, K., Gjorgjevikj, A., Vodenska, I., Chitkushev, L.T. and Trajanov, D., 2020. Evaluation of sentiment analysis in finance: from lexicons to transformers. *IEEE access*, *8*, pp.131662-131682.

[6] Gunter, B., 2018. *Predicting movie success at the box office*. Springer.

[7] Blackman, N.J.M. and Koval, J.J., 2000. Interval estimation for Cohen's kappa as a measure of agreement. *Statistics in medicine*, *19*(5), pp.723-741.

[8] Landis, J.R. and Koch, G.G., 1977. The measurement of observer agreement for categorical data. *biometrics*, pp.159-174.