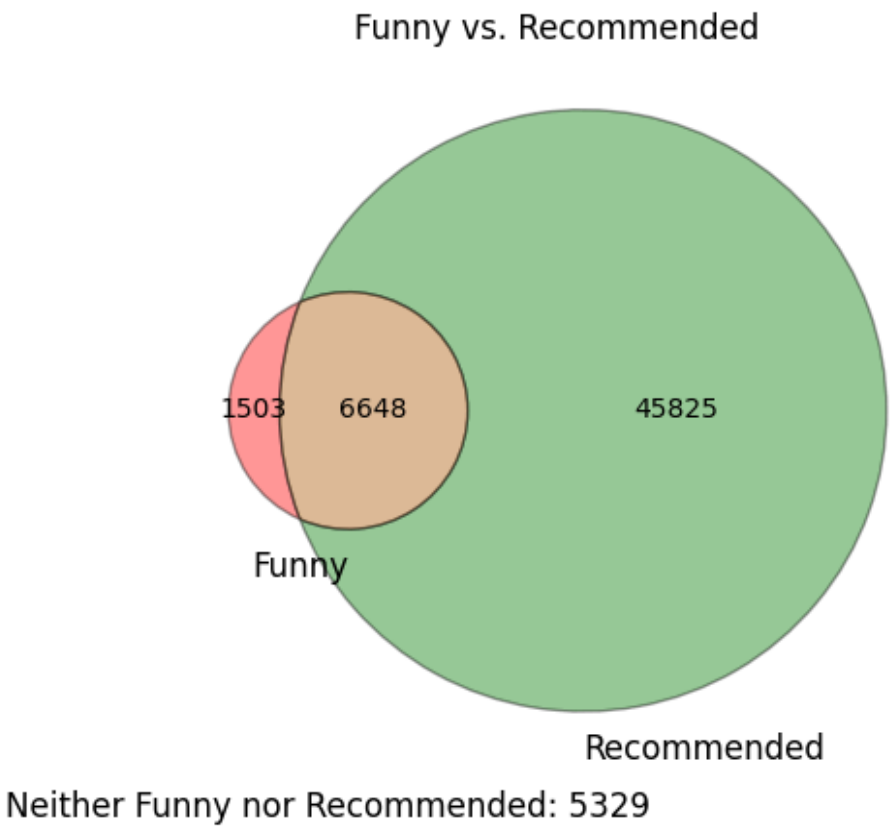
Predicting Whether a Review is Funny on Steam

1. Dataset

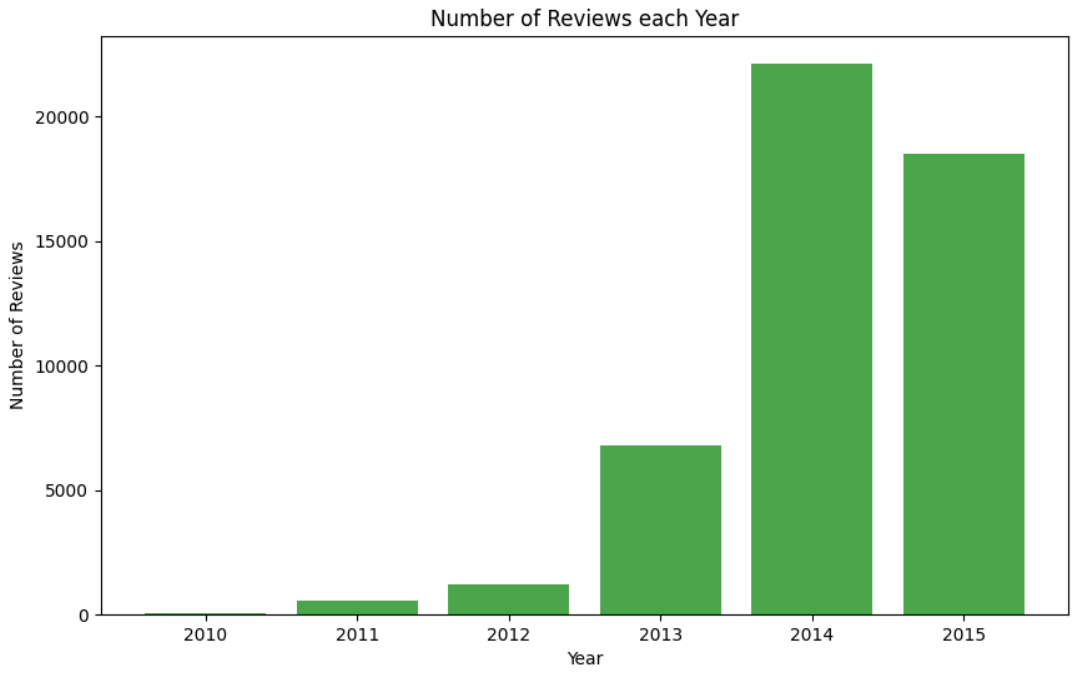
The dataset of this assignment is Steam Video Game [1,2,3] data, which includes reviews of Steam games. It contains 59,305 reviews, each with a user ID, item ID, post time, review text, whether the user recommends the game, and the number of other users who find the review helpful/unhelpful, or funny. There are 25,458 users and 3,682 games in the review dataset. We are particularly interested in whether a review is considered funny by other users; therefore, the exploration focuses on this aspect.

Now, let's take a deeper look at the review dataset. Out of 59,305 reviews, 52,473 reviews recommend the game, 30,168 reviews have been rated as helpful or unhelpful by at least one other person, and 8,151 reviews are considered funny by at least one other user. Below is a Venn diagram illustrating the relationship between recommendations and funniness:



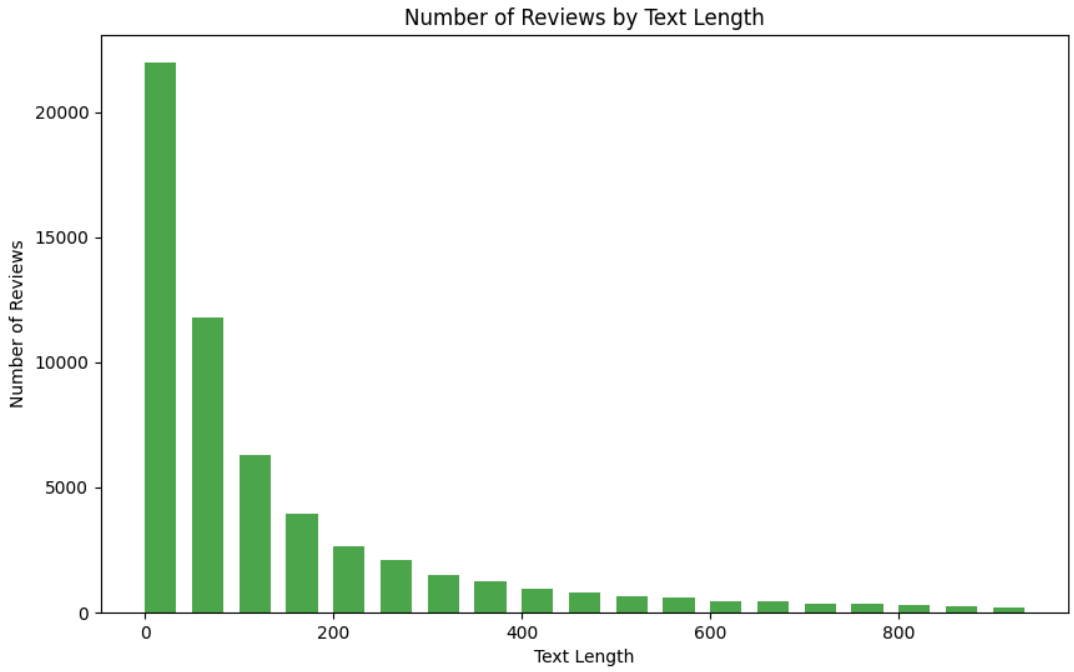
From the Venn diagram, we can conclude that reviews not recommending the game are more likely to be considered funny than those recommending the game. Specifically, **1,503 out of 6,832 (23%)** reviews that do not recommend the game are rated as funny, compared to **6,648 out of 52,473 (13%)** reviews that recommend the game.

Below is the distribution of the number of reviews for each year. The data shows a dramatic increase in the number of reviews over time, with the dataset being collected in 2015.



52.6% of users leave only one review, while the maximum number of reviews written by a single user is 20. 40.2% of games have only one review, while the maximum number of reviews for a single game is 3,759. This record belongs to *Counter-Strike: Global Offensive*, a popular first-person shooter game.

For the review text, the average length is 216 characters. Reviews considered funny have an average length of 299 characters, while those not considered funny have an average length of 202 characters. The maximum length of a review is 8,000 characters, while the minimum length is 0 characters. Below is the bar diagram showing the distribution of text length:



The review with the most funny votes has 956 votes. The review is: "Tutorial area took me 8 hours to complete. I regret nothing," for the game *The Witcher 3: Wild Hunt*.

The metadata for the games is also provided, including the title, genres, tags, release date, and more. We focus only on each game's genre, tags, and release date. Other metadata, such as title, developer, price, etc, are not very informative.

There are 333 different genres and tags for the games (some genres and tags are the same, so they are counted together). The genre/tag with the most games is "action," which has 2,058 games.

No user metadata is provided in the dataset; we only know the username associated with each review, which users can change.

2. Predictive task

The predictive task is to predict whether a review is considered funny by other users based on the review text and the game information. This is a binary classification task.

a. Clean up dataset

The first step is to clean up the dataset and remove any unnecessary data.

First, many reviews are very short, and humor cannot be properly established without sufficient content. Therefore, we remove any reviews that are under 10 characters.

Second, reviews are in multiple languages. Since the model cannot learn all the languages with such a small dataset, we focus only on English reviews. A review is considered an English review if more than 80% of its non-space characters are English letters, numbers, punctuation, symbols, or special characters.

Third, many reviews may be considered funny, but if no other users have seen them, the humor cannot be properly validated. Therefore, a review must have been rated as helpful or not helpful by at least two other users to ensure it has been seen by enough people.

After removing these reviews, the dataset has 16,942 reviews, of which 5,400 are considered funny.

b. Features

Next, we prepare the features for the predictive tasks.

The first feature is whether the user recommends the game. As shown above, the recommendation has some relationship with the funniness of the review.

The second feature is the length of the review text. As shown above, the length of the review text has some relationship with the funniness of the review.

Next, we use the genres/tags of the game in one-hot encoding. We choose genres over game IDs because many games are rated by only a single user, which does not provide enough data to establish a meaningful correlation between the game ID and whether the review is funny. On the other hand, genres/tags provide insight into the content of the game, and each genre/tag is associated with multiple games. Therefore, using genres/tags in one-hot encoding offers a better and more informative representation of the game.

We also include the time difference in years between the release of the game and the post time of the review. While the release time and post time alone do not convey meaningful information, the time difference may help the model determine whether the review is funny.

Lastly, the review text is the most important feature. We use different methods to process the review text to extract meaningful features for the model, which will be shown in the next section.

Other features are excluded because they convey little information for the predictive task, such as the username associated with the review. Whether other users find the review helpful cannot be used as a feature because this information is not available at the time the review is posted.

c. Metrics

The dataset is randomly shuffled into 80% of training and 20% of testing data.

Two metrics on the test dataset are used to evaluate the performance of the model: accuracy and balanced error rate (BER).

Accuracy measures the overall percentage of correct predictions, while balanced error rate (BER) accounts for the imbalance in class distributions. Since only about one-third of the reviews are considered funny, using both metrics ensures that the model's performance is evaluated not only in terms of overall correctness but also in terms of its ability to handle imbalanced data. This approach helps provide a more comprehensive evaluation, particularly for tasks with skewed class distributions.

d. Baseline models

Each model consists of two components: converting the review text into features and the predictive model.

Converting Review Text into Features: This component focuses on processing the raw text to extract relevant features. For the baseline approach, three methods are employed to convert the text into numerical features: Bag of Words (BOW), TF-IDF, and word2vec. For BOW

and TF-IDF, the vocabulary size is the 5,000 most frequent words. For word2vec, the vector size is set to 16, the window size is 3, and the model will use the skip-gram approach. To convert word2vec vectors into features, we use the mean of all vectors as features.

None of the approaches consider punctuation. For BOW and TF-IDF, morphological affixes are removed from words, and stop words are excluded from the vocabulary. For Word2Vec, only English letters and numbers are retained, while all other characters are removed.

Predictive Model: This component takes the extracted text features, along with additional features mentioned above, and applies a machine learning algorithm to make predictions. For the baseline model, logistic regression is used as the classifier. The inverse of regularization (C) is set to 0.01, and the class weight is balanced to address the imbalance inherent in the dataset.

The models and feature extraction methods are mentioned in the lecture. The parameters for each model have been carefully tuned to optimize test accuracy

3. **Model**

Three alternative models are implemented to outperform the baseline model.

   a. **Random Forest**

A Random Forest is an ensemble learning model that uses a collection of decision trees to make predictions.

A decision tree is a model that works like a flow chart, where each node makes a decision based on a subset of features. The tree starts at the root node and follows the branches based on feature values, ultimately reaching a leaf node where the final classification decision is made.

In a Random Forest model, each decision tree is trained using a random subset of the training data. When building each decision tree, the model randomly selects which features to

use at each node. The final classification result is determined by taking a majority vote across all the decision trees.

Random Forest can take inputs from each of the extracted text features, along with any other relevant features, to make predictions.

Strength:

1. Random forests scale well with high-dimensional data. For this predictive task, thousands of features are used for prediction. Since each node in a random forest considers only a random subset of features, the model is less affected by noisy or irrelevant features, such as irrelevant words in BOW or TF-IDF representations.
2. Due to the inherent randomness in random forests, they are less prone to overfitting. The final decision is made based on the majority vote of all trees, preventing the model from overfitting to specific training data.
3. Random Forest can select the most informative features to predict. After training, Random Forest evaluates feature importance by measuring how much each feature contributes to reducing the entropy at each split across all trees in the forest. This can be useful for improving model interpretability and understanding the underlying patterns in the data.
4. Random Forest is effective at handling imbalanced data. Since the final prediction is based on the majority vote, even if some trees are biased toward the majority class, the influence of other trees helps balance out this bias.

   Overall, Random Forest is a suitable choice for this predictive task.

Optimization:

During testing, we found that the optimal feature parameters for logistic regression also worked well for Random Forest. Therefore, no additional feature tuning is required for Random Forest.

We also enabled balanced class weights to address the issue of an imbalanced dataset.

Weakness:

1. Random Forest relies on the extracted text features as inputs. The extracted text features do not include punctuations and has a limited vocabulary size. Sometimes, it is difficult to understand the humor of the text using these text features.
2. Random Forest does not understand the meaning of the text, making it difficult to grasp nuances such as humor.

   To address the above limitations, we build another model.

b. BPE+LSTM+FC

To use all characters in the review text for prediction, we implemented Byte Pair Encoding (BPE). Instead of treating each word as a single token, BPE breaks each word into subword tokens. BPE starts with a vocabulary that contains all the characters in the training dataset. At each step, it identifies the most frequent pair of adjacent characters and adds it to the vocabulary, repeating this process until a maximum vocabulary size is reached. The review text is converted into tokens by finding the subwords in the vocabulary. This approach ensures that punctuation and other characters are included in the vocabulary and can be used for evaluation. We set the vocabulary size to be 10,000.

After converting the review text into tokens, they are passed through the word embedding layer using one-hot encoding. Word embedding, similar to word2vec, maps tokens to a continuous latent space. The word embeddings are then passed through an LSTM (Long Short-Term Memory) network to capture the meaning of the review text. LSTM is a type of recurrent neural network (RNN) that uses repeated layers to process sequential data. The output is the final hidden state of the LSTM.

Finally, the final hidden state is concatenated with other features and passed through several fully connected (FC) layers with activation functions to introduce non-linearity. The output layer has a size of 1, using Binary Cross-Entropy (BCE) as the loss function and AdamW for optimization.

Strength:

1. BPE can include all characters for evaluation. Many reviews contain emojis and smiley faces made with punctuation that convoy humor. The above text feature extraction methods cannot understand such content, but BPE can effectively address this issue.
2. BPE uses subwords instead of entire words, enabling it to capture the meaning of unseen words or typos. It is a state-of-the-art text feature extraction method widely used by large language models (LLMs) and other advanced language models.
3. LSTM is suitable for this task because the text length varies, and LSTM can process the text sequentially, allowing the latent space to effectively capture the meaning of the text.

Optimization:

During implementation, we find that when the text is too long, LSTM starts to lose focus, and the additional padding in samples within the same batch can degrade performance. As a result, we limit the maximum number of tokens per review to 512.

The vocabulary size is also a crucial hyperparameter for BPE. If the vocabulary size is too large, it becomes equivalent to word tokenization. If it is too small, it may miss many important subwords. We ultimately decided on a vocabulary size of 10,000, which is suitable for our dataset.

The word embedding layer size and hidden size of the LSTM is another important hyperparameter. If it is too small, the model may fail to capture the meaning of the text. If it is too large, the model may overfit quickly. After tuning, we chose a word embedding layer of 16 and a hidden size of 128 for LSTM.

To add non-linearity to the FC layers, we use GeLU as the activation layer because it is the state-of-the-art activation layer. The final output is connected to the Sigmoid activation layer.

To avoid overfitting, we save the model with the highest accuracy after each epoch during training.

Weakness:

1. We need to train both the word embedding and LSTM from scratch. However, the dataset size is not large enough for the model to effectively understand humor.
2. The text size variation is too large, which makes it challenging for the model to process sequences efficiently and maintain consistent performance across reviews of different lengths. We have to add paddings to the end of reviews to match the sequence length, which may have an impact on the performance.
3. LSTM is not the state-of-the-art language model for understanding human language, as newer models like transformers have surpassed LSTMs in capturing complex language patterns and dependencies.

   Therefore, to address these issues, we build our last model.

c. RoBERTa [4] +FC

RoBERTa (A Robustly Optimized BERT Pretraining Approach) is a variant of the BERT (Bidirectional Encoder Representations from Transformers) model.

BERT is a pre-trained language model that uses transformers for natural language processing. BERT is bidirectional, meaning it looks at the tokens before and after a token to determine the meaning of that token. This contextual ability allows it to better understand each token. BERT also uses self-attention mechanisms, allowing BERT to focus on related tokens when processing each token to better understand the meaning.

RoBERTa is similar to BERT, the details of RoBERTa will be explained in the next section.

For our model, the last hidden layer of RoBERTa, combined with other features, is connected to an FC network to predict results, which is the same as the last model. We also use early stopping to prevent overfitting.

Strength:

1. Using a pre-trained language model can help mitigate the impact of a small dataset because it is learned from a much larger text dataset.

2. RoBERTa takes advantage of BPE for token encoding. Additionally, BPE uses a larger vocabulary size of 30,000, which helps the model understand subwords that may not exist in the training dataset.
3. RoBERTa is a state-of-the-art language model for understanding text and converting it into a latent space representation.
4. RoBERTa uses masks for padding, which helps mitigate the effects of long padding in the input sequences.

Optimization:

There were several challenges during implementation. Firstly, RoBERTa is a very large model, making training expensive. The optimal approach would be to fine-tune the pre-trained RoBERTa on our dataset to make it task-specific. However, due to the high cost of training, we can only use the pre-trained RoBERTa as a feature extractor, generating latent space representations without further training RoBERTa.

We also tuned the FC layer hyperparameters to suit the output of RoBERTa better. Since RoBERTa has a much larger hidden size, we added an additional FC layer to help interpret its output more effectively.

Weakness:

1. The largest weakness is that we are unable to fine-tune it on our dataset. Although RoBERTa can effectively convert the text into latent space, it is not task-specific, which makes predicting whether the text is funny difficult.
2. The BPE tokenization in RoBERTa does not include punctuation and special characters, which makes it challenging to capture the nuances of humor conveyed by these symbols.

d. Other unsuccessful attempts

1. Instead of RoBERTa, we tried several similar models from Hugging Face, such as BERT, ALBERT, and DistilBERT, but none performed better than RoBERTa.

2. We tried using decision trees and SVMs instead of Random Forest but encountered different issues. A single decision tree tends to be too biased and prone to overfitting. SVM, being similar to logistic regression, also does not seem to be the optimal solution for understanding the text.

4. Literature

For state-of-the-art humor predictor, in this paper, **ColBERT: Using BERT Sentence Embedding in Parallel Neural Networks for Computational Humor** [5], the author proposed measuring humor with ColBERT. The author uses BERT to process each sentence in parallel, and then concatenates the latent space generated by all sentences to determine the congruity and other relationships between the sentences for the final prediction. This approach can effectively understand the meaning of each sentence and the relationship between sentences to predict humor. It is a state-of-the-art humor detector. This paper also introduces a novel dataset for humor prediction, consisting of 200,000 text samples with balanced data.

For the approach that we used In this paper, **RoBERTa: A Robustly Optimized BERT Pretraining Approach** [4], the author proposed RoBERTa, which is one of the methods we use. RoBERTa is similar to BERT, but it is trained on larger datasets and more optimized training configurations. It is proven to be more effective and robust than BERT. RoBERT also uses BPE as token encoding, while BERT uses Wordpiece.

For predicting users' posts, this paper, **Predicting the Popularity of Reddit Posts with AI** [6], predicts the popularity of Reddit posts. It uses linear regression, Random Forest, and a neural network for prediction. The result shows that Random Forest and the neural network are significantly better than linear regression, which matches the result of our predictive task.

Recognizing Humor and Predicting Humor Ratings in Short Texts [7] compares the performance of different text feature extraction and models on understanding the humor. It shows that fine-tuned BERT embedding with neural networks performs the best. Without

fine-tuned BERT embeddings, directly using pre-trained BERT embedding with neural networks performs the best, which aligns with the findings of this project.

The dataset we use comes from **Generating and personalizing bundle recommendations on Steam** [3]. This paper focuses on bundle recommendations. It proposed a modified BPR with bundle size and bundle correlation, achieving the best AUC.

5. Results

The following is the accuracy and BER of all models.

Text Feature	Predictor	Accuracy (%)	BER
BOW	Logistic Regression	62.08	0.3983
	Random Forest	69.99	0.4445
TF-IDF	Logistic Regression	61.64	0.4138
	Random Forest	69.84	0.4456
Word2vec	Logistic Regression	60.43	0.4027
	Random Forest	69.66	0.4516
BPE	LSTM+FC	69.76	0.4454
RoBERTa	FC	**70.76**	0.4379

Among all the models, RoBERTa+FC achieves the highest accuracy and demonstrates the best overall performance. Additionally, all of the proposed models show significant improvement compared to the baseline. Random Forest and BPE+LSTM+FC have higher BER values than RoBERTa+FC. Therefore, we conclude that RoBERTa+FC is the best-performing model among all models.

The results show that RoBERTa+FC can effectively extract meaningful semantic features from the review. The advanced language models can significantly outperform traditional machine learning approaches in natural language processing. However, it is not significantly better than Random Forest because RoBERTa is trained on formal English, while this dataset

contains a large number of Internet memes and colloquial English. If RoBERTa were fine-tuned on this dataset, it could likely achieve better results by adapting to the specific linguistic patterns of Steam users.

BPE+LSTM+FC performs worse than RoBERTa+FC in terms of both accuracy and BER, validating the effectiveness of using a pre-trained language model for this task. Training a word-embedded LSTM and FC layers from scratch does not yield strong performance because the dataset is too small for the model to understand the English language.

TF-IDF+Random Forest performs second best among all models. TF-IDF performs better than BOW and word2vec because it assigns importance to words based on their frequency across the dataset, which captures the relevance of specific words. Unlike BOW, which treats all words equally, TF-IDF reduces the influence of frequent but less informative words. Additionally, word2vec uses the mean of the vectors as text features, which can dilute the impact of important words and introduce noise from unrelated words, therefore, impacting the model's performance.

Random Forest outperforms Logistic Regression because it is a non-linear model, whereas Logistic Regression is linear. A linear model cannot capture the complex patterns present in language data, while Random Forest, can effectively model non-linear interactions between features.

All of our models have a slightly higher BER than the baseline. The effort to reduce BER and improve performance on the minority class was not very successful because the dataset is heavily imbalanced. While class balancing techniques were applied, such as weighting classes in logistic regression and Random Forest, they may not fully address the inherent challenges posed by the imbalance.

In addition to the text features, other features also contribute to the model's performance.

The feature indicating whether the review recommends the game can aid in the prediction, as reviews are more likely to be funny when the user does not recommend the game. For example, a funny review that says, "*The best part of this game is closing it...*" uses the word "best," which is positive and could be identified by text feature extraction methods. However, the fact that the review does not recommend the game suggests sarcasm, which is a key indicator of humor. This context helps the model recognize and predict whether the review is funny.

The time difference between the release of the game and the post time of the review may not be very helpful in determining whether the review is funny. The average time difference for funny reviews is 2.02 years, while the average for unfunny reviews is 1.97 years. Given that the difference is minimal, it suggests that the timing of the review relative to the game's release does not have a significant impact on the likelihood of the review being considered funny.

The feature for the review text length is helpful for the prediction. As mentioned earlier, the average text length of funny reviews is longer than that of unfunny reviews. Longer reviews contain more information, providing greater chances to be humorous. Therefore, the length of the review can serve as a useful indicator.

The feature of one-hot encoding of game tags/genres is also helpful. Different genres or tags may be associated with certain types of humor or user expectations, influencing the likelihood that a review will be considered funny. For example, reviews of games in genres like "comedy" might have a higher chance of containing humor.

The above observations are confirmed during the implementation of the model.

Now, let us examine whether our best-performing model can recognize sarcasm. A review from the test dataset states, *"Compared to this game, Bad Rats is Game of the Year,"* which is sarcastic. However, the output of RoBERTa+FC for this review is 0.3718, indicating that the model predicts it as not funny, but it is actually marked as funny by 31 users. This result

highlights a limitation of the model in identifying sarcasm, likely due to the lack of sarcastic reviews in the training dataset.

In conclusion, RoBERTa+FC is the best-performing model among all the models we tried. It achieved **70.76%** accuracy and **0.4379** BER. RoBERTa+FC is a valid approach to determine whether the review text is humorous.

Citations:

1. **Self-attentive sequential recommendation**

   Wang-Cheng Kang, Julian McAuley

   ICDM, 2018

2. **Item recommendation on monotonic behavior chains**

   Mengting Wan, Julian McAuley

   RecSys, 2018

3. **Generating and personalizing bundle recommendations on Steam**

   Apurva Pathak, Kshitiz Gupta, Julian McAuley

   SIGIR, 2017

4. **RoBERTa: A Robustly Optimized BERT Pretraining Approach**

   Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M.,

   Zettlemoyer, L., & Stoyanov, V.

   ArXiv, 2019

5. **ColBERT: Using BERT Sentence Embedding in Parallel Neural Networks for Computational Humor.**

   Annamoradnejad, I., & Zoghi, G.

   ArXiv, 2020

6. **Predicting the Popularity of Reddit Posts with AI.**

   Kim, J

   ArXiv, 2021

7. **Predicting Humor in Text using Contextual Embeddings**

   Daniel Berg Thomsen, Jean-Baptiste de la Bro¨ ıse, Eelis Mielonen

   ACL, 2022