NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CE/CZ4045 Natural Language Processing
AY22/23 S1 - Group 28

Sentiment Analysis on Game-related Tweets

Submitted by:

Chen Yiting   U2022489K

Hoo Jia Kai   U1921944C

Isabelle Ong Ee Ling   U1921109A

Lee Jian En   U1921762A

Low Soo Yee Calvin   U1920636D

Thach Sarai   N2202272D

6 November 2022

Table of Contents

Introduction

Gaming is one of the most common hobbies in the world, even in Singapore. This is corroborated by a recent survey conducted by Statista [1], where 28% of respondents indicated that one of their hobbies was video gaming, as seen in Figure 1 below.
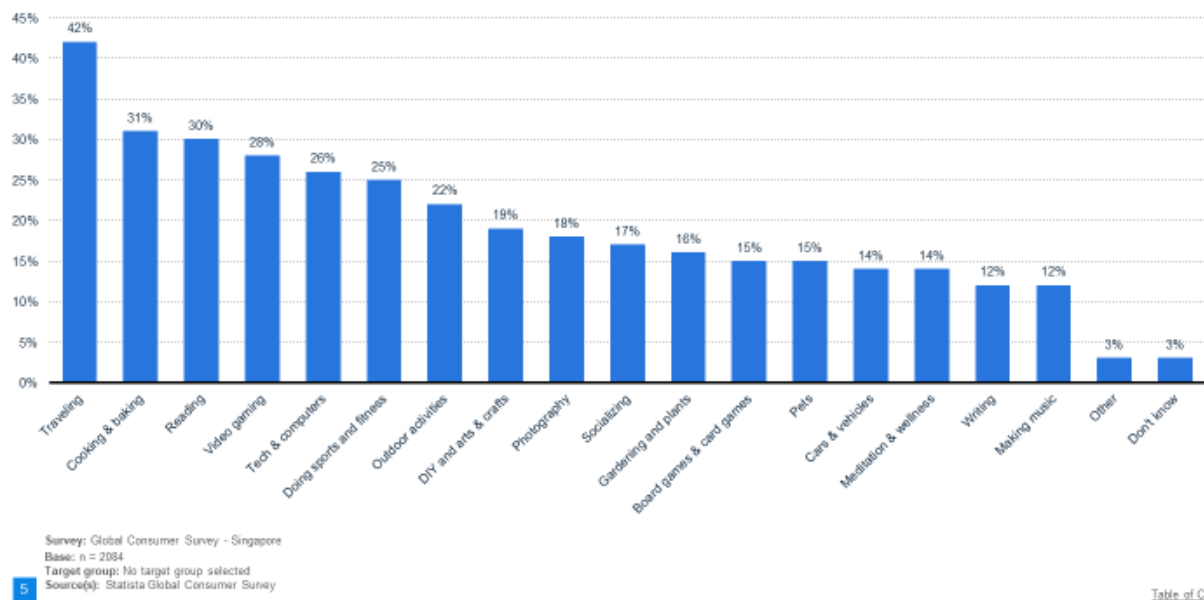

Figure 1. "What are your personal hobbies and activities?" Global Consumer Survey - Singapore

Game reviews provide crucial information to both players, current and potential, and developers. Potential players may look at reviews when considering if they should purchase them. Developers use reviews to understand what their users enjoy and hate. While there are many platforms that already allow users to post reviews, there is still ground to be covered on the social media front.

Social media platforms like Twitter and Reddit are a gold mine for text-based posts. By extracting relevant posts, it is possible to understand how users generally feel about a game. Thus, the goal of this project is to develop a model that is able to conduct sentiment analysis on game-related tweets.

Crawling

Before scraping Twitter for game-related tweets, we decided to narrow down our search to 10 games. We used the top 10 most played games on Steam as of 29 September 2022. As shown in Figure 2 below, the 10 games are: Counter-Strike: Global Offensive, Dota 2, PUBG: Battlegrounds, Apex Legends, Lost Ark, Naraka: Bladepoint, Grand Theft Auto V, Team Fortress 2, Cyberpunk 2077, and Rust. The wallpaper engine was excluded as it is an application that allows enables users to apply a live wallpaper on their desktop.
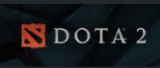

Figure 2. Top 10 most played games on Steam (not including Wallpaper Engine)

Crawling Tools and Labelling

We divided the scrapping workload amongst our team. For each game that we had been assigned, we had to crawl 1,000 tweets from Twitter and label 100 of them using the respective subjectivity and polarity categories.

Twint was used for the crawling process. It is a python-based Twitter scraping tool, which does not utilise Twitter's API. The package makes use of Twitter's search function instead, making it possible to retrieve tweets from specific users and hashtags.

After the scraping and labelling process was completed, the tweets were collated in a single CSV file (compiled_tweets.csv).

Inter-Annotator Agreement Score

Among 11,042 crawled tweets, our team has manually labelled 1,000 tweets. As these tweets are labelled by different team members, it is important to determine how clear the annotation guidelines are, how uniform the annotators understood it, and how reproducible the annotation task is. This is calculated using the Cohen Kappa measure, where 0 stands for chance agreement while 1 stands for total agreement.

Our inter-annotator agreement between 2 annotators is 0.825. This is thus a good (>0.8) agreement. Hence, this shows that both annotators can make almost the same annotation decision for a certain category.

Summary of Dataset

Below is a summary of the crawled tweets without any preprocessing:

Number of Tweets	11,042
Number of Words	275,649
Number of Unique Words	33,511

In the crawled tweets, there were some patterns that we have observed:

- Many streamers use Twitter to announce that they have started their streams on the respective platforms. However, it should not affect the sentiment of a tweet.
- Special preprocessing is required for the crawled tweets. The majority of tweets contain hashtags, mentions, retweets, and URLs which often do not contribute to sentiment analysis.
- Tweets can also contain emojis (i.e. 🐼), acronyms (i.e. LOL), and multiple consecutive punctuations (i.e. !!!). All of which can affect the extent of the polarity of the tweets.

Preprocessing

As aforementioned, we had to perform special preprocessing of the tweets due to the "noisy" nature of tweets. This allows us to transform the tweets into a more standardised form for the training of the models later. The preprocessing includes case folding, tokenization, stopword removal, noise removal, stemming, and lemmatization.

Before performing the preprocessing, we first identify the different languages used in the tweet datasets, and only select those in English, as it is our main communication language and thus of most interest.

Case Folding

Case folding is performed to reduce all letters to lowercase. Even though words with all capitals display the emotions of users, such as being agitated and excited, in the form of shouting, we believe that the words alone are sufficient to infer such emotions to determine their subjectivity and polarity. As such, case folding is conducted in our preprocessing.

Tokenization and Stopword Removal

Tokenization is also conducted to separate a tweet into smaller units or tokens so that each token can be easily analysed and assigned its meanings. Stopword removal is also done to common words that have little semantic content and thus bring little value to our sentimental analysis of tweets.

Noise Removal

As mentioned earlier, the crawled tweets contained hashtags and URLs. Thus, noise removal is required. The approach used for this was to create a function to remove the following:

- Any usage of carriage return (\r) and line feed (\n)
- Any URLs
- Any non-UTF-8 or ASCII characters (i.e.Ã±ã¼â»§)
- Any hashtags, mentions (i.e. @nlp) and retweets (i.e ^RT I agree)

Stemming and Lemmatization

Our team decided to conduct both stemming and lemmatization for our preprocessing. NLTK Porter Stemmer is used to stem words, while the NLTK WordNet Lemmatizer is used to lemmatize words. For words that are not found in WordNet, the input word is returned in its original form.

However, the processes of stemming and lemmatization were not done simultaneously. This is because we wanted to experiment and see which yielded better results, hence we decided to have 3 methods of preprocessing:

1. Case folding, tokenization, noise removal, and stopword removal
2. Case folding, tokenization, noise removal, stopword removal, stemming
3. Case folding, tokenization, noise removal, stopword removal, and lemmatization

We will compare the performances among these 3 methods under the Classification section below.

Balance Skewed Dataset with Random Undersampling

Lastly, we realised that our subjectivity and polarity classes are highly unbalanced, where there are too many opinionated and negative tweets. In order to get a better performance, we decided to balance the dataset first. This is done by doing random undersampling, where random samples will be selected from the majority classes (opinionated and negative), and removed from the training dataset. Hence a balanced dataset is achieved with the same amount of both classes for subjectivity and polarity. We chose to perform undersampling instead of oversampling, as oversampling results in overfitting, which we want to avoid at all costs. Although undersampling may result in the loss of valuable information from the majority classes, we believe that the resultant samples are representative of the original dataset and thus are not of huge concern, as compared to having an overfitted model.

Vectorization

After balancing the dataset, we then split it into training, testing, and cross-validation sets. Vectorization is then performed on all the split sets, to convert text data into numerical data.

This is done via Count Vectorization and Term Frequency - Inverse Document Frequency (TF-IDF) Vectorization. A count vectorizer converts strings into a representation of frequency, according to the frequency of the composed words. A TF-IDF vector determines the importance of a text depending on the rarity of the composed words.

Classification

Classification Methods

For our classification approach, we chose to use 4 different types of classification methods. They are Naive Bayes, K-Nearest Neighbour (KNN), Support Vector Machine (SVM), and decision tree classifications.

Motivation of Classification Methods

Naive Bayes Classification

Naive Bayes classification is a simple and common supervised learning algorithm to classify data. It is heavily used for text classification and analysis, due to its higher success rates as compared to other algorithms. This is attributed to its superior performance in multi-class problems and conditional independence assumption, which is highly applicable to our project here. Also, this method is used as the baseline model in many other classification tasks, due to its great time performance and accuracy. Hence, we have decided to use the Naive Bayes classification as the baseline model to compare with the other 3 classification methods.

K-Nearest Neighbours (KNN) Classification

K-Nearest Neighbours classification is a supervised machine learning algorithm, which uses proximity to make classifications or predictions about the classes of data. This is based on distance functions, such as Euclidean distance, Manhattan distance, and Cosine similarity of TF-IDF weighted vectors. As all our tweets revolve around video games, similar words or vocabulary are used and the K-Nearest Neighbours can be identified by calculating the similarity score, before classifying them according to their subjectivity and polarity.

Support Vector Machine (SVM) Classification

Support Vector Machine classification uses a hyperplane that creates a boundary to distinguish between two classes to classify data. This can be used in the sentiment analysis of our project as each tweet can be classified into one of two classes: neutral or opinionated and positive or negative, making it a linearly separable problem. Although this algorithm takes a relatively long time to train, with a time complexity of at least $O(n^2)$, it is one of the better methods for text classification as it can generalise well in high-dimensional feature spaces and eliminate the need for feature selection, hence making the classification easier and more robust.

Decision Tree Classification

Decision Tree classification is a supervised algorithm that continuously splits the data according to a certain condition. Since a tweet can be either neutral or opinionated, and

either positive or negative, we can use this method to split and classify the data for sentiment analysis.

Classification Details

For each classification method, we find the optimal hyperparameters that can give the most accurate predictions, using GridSearchCV. These parameters are then saved to train each model.

Models were then tested on an evaluation set, which had been preprocessed with the above 3 methods. These 3 methods' performances are then compared to the baseline model of Naive Bayes classification, as mentioned earlier.

Evaluation Metrics

Finally, after testing the models on the evaluation set and comparing the results, we found that pairing stemming and SVM classification had the best performance.

From the table below, we can see that the differences in the metric scores among the pre-processing types for subjectivity classification are minimal. Although stopword removal has the best scores, apart from recall, the method does worse than stemming and lemmatization for polarity classification. Out of the 3 methods, stemming achieved the best scores for polarity and relatively high scores for subjectivity. Thus, it was chosen as our preprocessing method.

Metrics (Subjectivity)	Stemming	Lemmatization	Stopword Removal
F1 Score	0.857	0.857	0.858
Precision Score	0.813	0.816	0.819
Recall Score	0.907	0.903	0.900
Average Precision-Recall Score	0.784	0.785	0.788
Metrics (Polarity)	**Stemming**	**Lemmatization**	**Stopword Removal**
F1 Score	0.771	0.769	0.759
Precision Score	0.774	0.769	0.762
Recall Score	0.768	0.768	0.756
Average Precision-Recall Score	0.710	0.707	0.698

Moving on to the types of evaluation metrics we used, they are namely F1 score, precision score, and recall score. We will be discussing the performance of our SVM model with respect to the baseline model, Navies Bayes classification. The rationale of why we chose the SVM model, and not the KNN and Decision Tree models, can be found in the Appendix section below, for neater structuring purposes.

F1 Score

F1 Score is the weighted average of precision and recall. The metric is especially important in situations where both false positives and negatives are intolerable.

$$F1\ Score\ =\ \frac{2*Precision*Recall}{Precision+\ Recall}$$

As shown in Figure. 3 and 4 below, the SVM model's F1 score is 0.857 and 0.771 for subjectivity and polarity respectively. It outperforms the baseline model for subjectivity but attained the same F1 score for polarity.

Precision Score

Precision Score, also called positive predictive value, is the ratio of true positive observations to predicted positive ones.

$$Precision\ Score\ =\ \frac{True\ Positives}{True\ Positives+False\ Positives}$$

Positive observations refer to opinionated tweets for subjectivity and tweets with positive sentiment for polarity. The individual precision scores for subjectivity and polarity are 0.813 and 0.774, as seen in Figure. 3 and 4 below respectively. The precision score of the baseline model barely edges that of the SVM model for subjectivity. Whereas the SVM model's score for polarity is slightly better than the baseline.
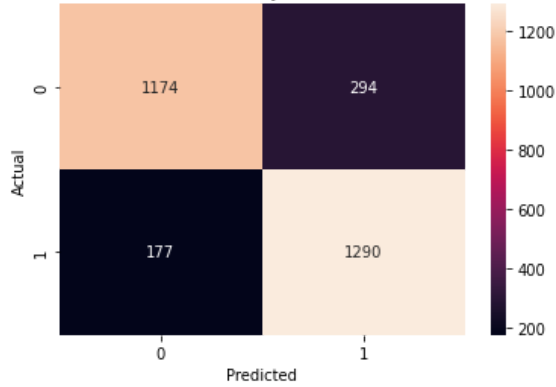
Recall Score

Recall Score, also known as sensitivity or true positive rate, is the percentage of positive observations that have been correctly classified by the model.

$$Recall\ Score\ =\ \frac{True\ Positives}{True\ Positives+False\ Negatives}$$

The recall score is 0.907 for subjectivity and 0.768 for polarity, as seen in Figure. 3 and 4 below respectively. The results reflect that tweets are correctly classified as opinionated approximately 90% of the time. However, the performance for polarity is much poorer and the SVM model does not perform as well as our baseline model.

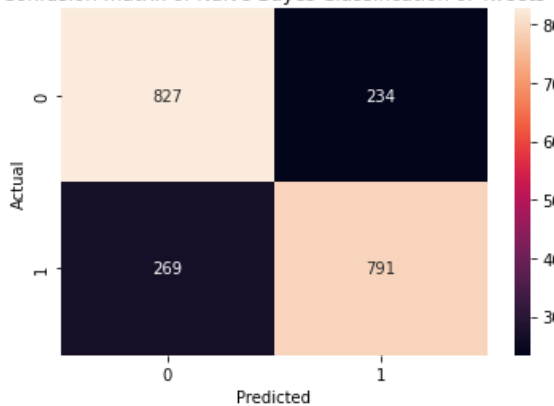Confusion matrix of Naive Bayes Classification of Tweets

Confusion matrix of Decision Tree Classification of Tweets
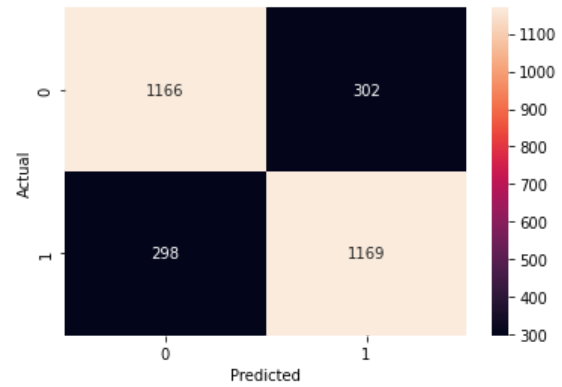
Metrics (Subjectivity)	Baseline	SVM
F1 Score	0.846	0.857
Precision Score	0.814	0.813
Recall Score	0.879	0.907
Average Precision-Recall Score	0.776	0.784

Figure. 3 Performance Comparison - Baseline vs SVM (Subjectivity)

Confusion matrix of Naive Bayes Classification of Tweets

Confusion matrix of SVM Classification of Tweets
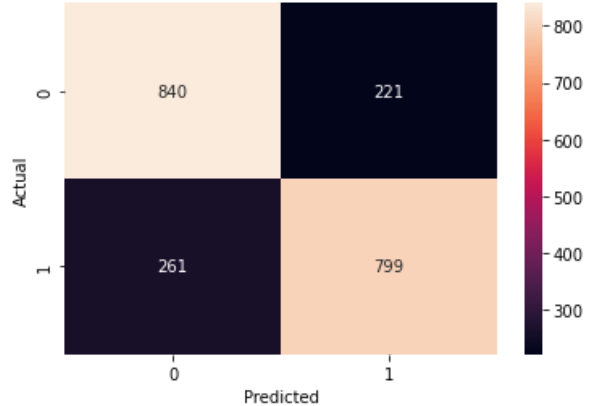
Metrics (Polarity)	Baseline	SVM
F1 Score	0.771	0.771
Precision Score	0.769	0.774
Recall Score	0.772	0.768
Average Precision-Recall Score	0.708	0.710

Figure. 4 Performance comparison - Baseline vs SVM (Polarity)

To summarise, we can conclude from the evaluation scores that the SVM model's ability to classify polarity is relatively poor as compared to subjectivity. This trend is constant regardless of the classification method and pre-processing method used. We can classify

the current methods used as sub-symbolic AI, which is effective even when there is noise and can attain high-performance levels, for example, classification speed. However, as tweets can be representative of a person's thoughts, using symbolic AI can be more effective. Thus, our team decided to employ a hybrid classification approach, combining symbolic and sub-symbolic AI, through the use of BERT, which will be explained further in the Innovations section below.

Performance Metrics

To measure the performance of the SVM model, we recorded the time taken for the model to predict the subjectivity and polarity for a set amount of tweets. This starts at 100 and the number of tweets gradually increases to 2000 in steps of 50. The model was tested on 2000 random tweets taken from the unlabeled self-crawled dataset and then used as input data.

Number of Tweets	Time taken by SVM model
100	0.1043s
200	0.2164s
500	0.5145s
1000	1.0269s
1500	1.8715s
1950	1.7504s

From the sample of recorded times above, we can see that the model performs classification very quickly. However, SVM models can be resource-intensive, thus they might not be as scalable as other models like Naive Bayes or decision trees.
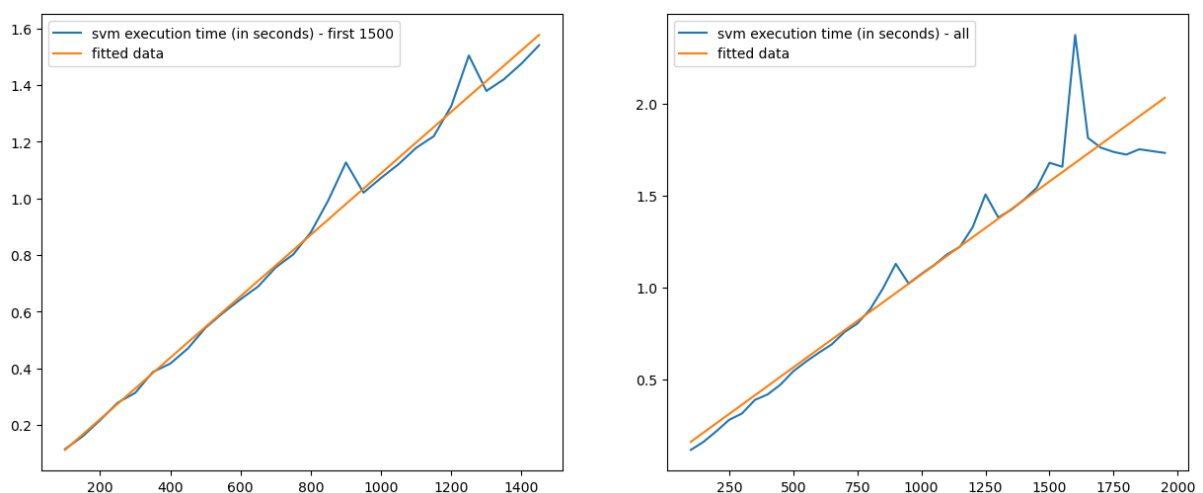

Figure. 5 Performance Metrics - Fitted with degree 1

Innovations

Bidirectional Encoder Representations from Transformers (BERT)

As seen earlier, current traditional models (Naive Bayes, KNN, SVM, and Decision Tree) perform better in the prediction of subjectivity, as compared to polarity. As such, we decided to build a model that can perform well in both tasks of subjectivity and polarity detection. Furthermore, this model needs to be easy to train, fine-tune, and migrate.

This can be achieved with Bidirectional Encoder Representations from Transformers (BERT), which is a symbolic AI and deep neural network algorithm. Instead of only looking at the text sequence unidirectionally, BERT can train bidirectionally to have a deeper understanding of the context of a word based on its surrounding words, and thus achieve better performance.

For this model, we used the first method of the preprocessed dataset (case folding, tokenization, noise removal, and stopword removal). This is because BERT uses byte-pair encoding to shrink its vocabulary size, which means that words like "run" and "running" will ultimately be decoded to "run + ##ing". Hence, such additional information from other forms of the lemma or stem is crucial for the model, and so lemmatization and stemming should not be performed.

We used Keras to train the BERT model and chose the small_bert pre-trained model. Different layers of the model are then specified, which kicks off with the input text layer, followed by the preprocessing layer that allows the BERT model to understand. Then, we encode the preprocessed text input and pass it through the dropout later to filter out the noise, which is done randomly to prevent the overfitting of the model. Lastly, it is then passed into the final layer of a dense layer to produce the raw trained model, with no activation functions used. This raw model does not perform well with our dataset as we have yet to tweak the model to cater to our project, as seen in Figure. 6 below.
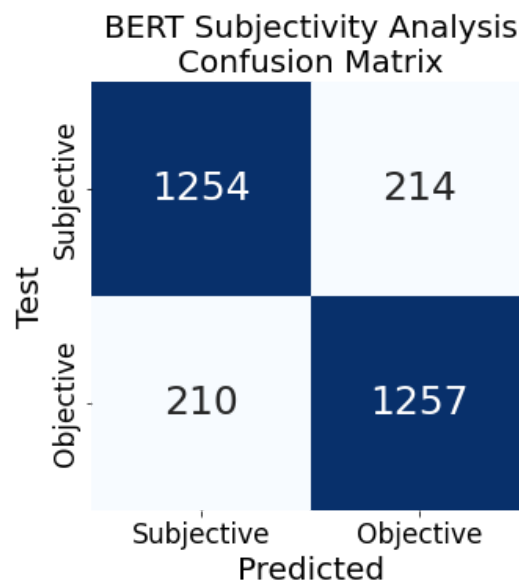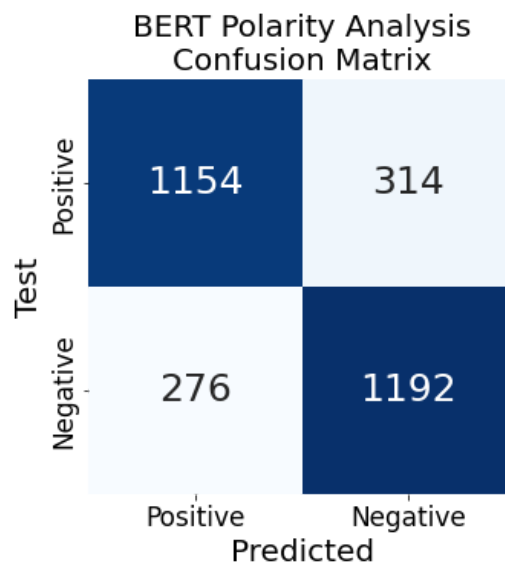

Figure. 6 Performance of raw BERT model (Subjectivity)

We then proceed to tweak the BERT model to suit our dataset. This is done by incorporating an Adam optimizer with weighted decay (AdamW) and early stopping to avoid overfitting from occurring. AdamW is an algorithm that is commonly used in conjunction with BERT models. It determines model parameters that generate the best fit between actual and predicted values, using adaptive estimation of first-order and second-order moments with an added method to decay weights. Early stopping is a method supported by Keras to check at end of every epoch whether the loss is no longer decreasing. The loss function is continuously monitored and if the loss increases for consecutive 5 epochs, the model will stop training more epochs and return to the lowest loss value epoch. The parameters of this epoch are then stored to save for future use.

After we have trained our customised BERT model, we tested it and achieved excellent performance for both subjectivity and polarity detections as seen in Figures. 7 and 8 respectively below.

Figure. 7 Performance of our tweaked BERT model (Subjectivity)

Figure .8 Performance of our tweaked BERT model (Polarity)

A summary of the results are as follows, comparing the baseline (Naive Bayes), SVM and BERT models:

Metrics (Subjectivity)	Baseline	SVM	BERT
F1 Score	0.732	0.733	0.86
Precision Score	0.689	0.684	0.86
Recall Score	0.779	0.791	0.86

Metrics (Polarity)	Baseline	SVM	BERT
F1 Score	0.759	0.768	0.80
Precision Score	0.772	0.783	0.80
Recall Score	0.746	0.754	0.80

Hence, with this innovation, we are now able to perform both detection tasks well, which were unachievable by the previous models.

Performance Metrics

We used the same performance metrics as before to compare the two models.

Number of Tweets	Time taken by SVM model	Time taken by BERT model
100	0.1043s	0.9175s
200	0.2164s	2.8941s
500	0.5145s	15.7501s
1000	1.0269s	60.9072s
1500	1.8715s	131.6300s
1950	1.7504s	150.8671s

We can see that the BERT model takes significantly more time to classify the tweets, even at smaller scales of tweets. The long execution time may be caused by the preprocessing layer that's built as part of the model. As an application, users will definitely be put off by such long waiting times. However, this becomes a non-issue if the application is modified to behave more like a dashboard. For example, the scrapping and classification of tweets for games can be scheduled at 12 am and the results can be stored on a database that can be accessed by the users.
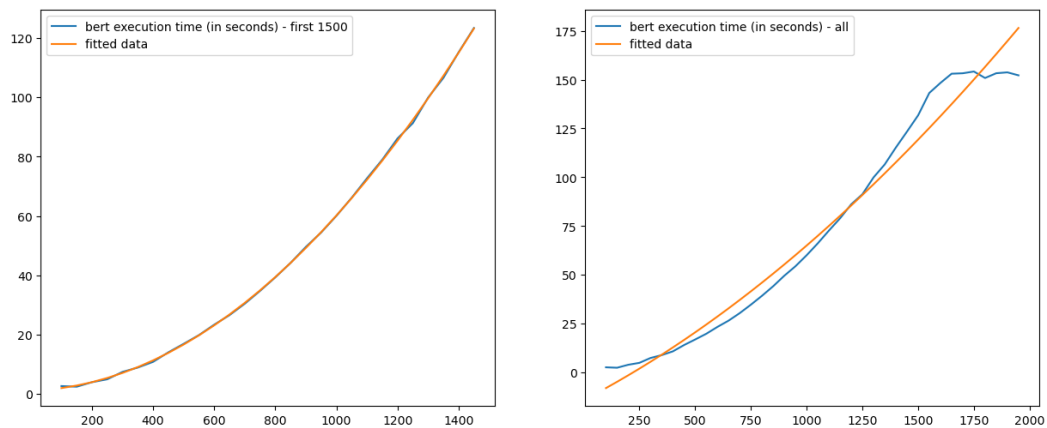
Figure. 9 Performance Metrics - Fitted with degree 2

Possible future Innovation: Aspect-based sentiment analysis (ABSA)

When dealing with reviews, it is likely users will talk about different features of a product. For example, in Figure. 10, we can have the user comment about how fun the solo and multiplayer features are. However, the user also talks about the poor game-saving system.

Figure. 10 Steam Review on PlateUp!

Common aspect categories such as gameplay, story, graphics, music, community, and general/others have been found in game review websites [5]. However, tweets are generally shorter and often less informative than game reviews. Thus, a tweet might only talk about one aspect of a game or not at all.

It is also important to note that there are many lingos and abbreviations that are exclusive to specific games or genres. The term "ads" can be interpreted as advertisements but it actually stands for "aim down sight" which is a mechanic in first-person shooter games. This means that a game-based NER would be helpful for determining aspect candidates. Another possible way of identifying them would be to handle common nouns and abbreviations as aspect candidates [6].

As different users have different priorities and preferences, ABSA can help potential users decide if the game is worth purchasing or spending the time to learn the mechanics.

Front End

As game reviews are the main source of information for both players and developers to make more informed decisions, we felt that a user-friendly web application can be made to better aid anyone in choosing a game to play or developing a game.
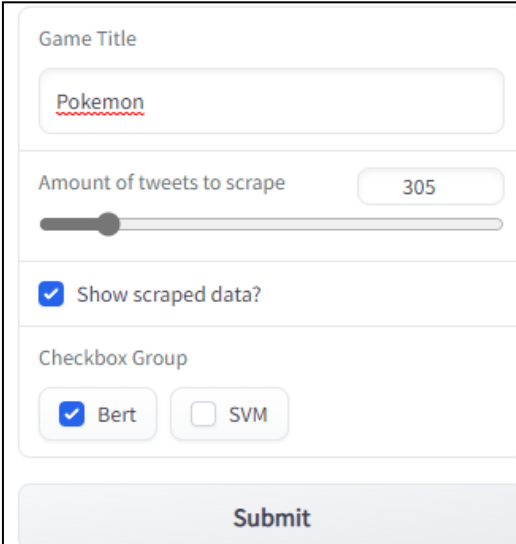
As such, our team created an all-in-one web application, using the Gradio library, for anyone to use for sentiment analysis in general. The overall user interface can be seen in Figure. 11 below.

Figure. 11 Overall user interface for our web application

This allows users to easily scrap tweets of their desired game and test them using our shortlisted SVM and BERT models. One can simply enter the topic in the search box, customize the number of tweets scrapped, and choose the types of models to run the scrapped tweets against. An example of possible user inputs can be seen in Figure. 12 below.

Figure. 12 An example of possible user inputs

After submitting the user's customised inputs, the desired topic will be scrapped from Twitter, and the relevant tweets will be put into the selected models to run the sentiment analysis.

The results will be displayed in the section below, in the format of a table, as seen in Figures. 13 and 14.

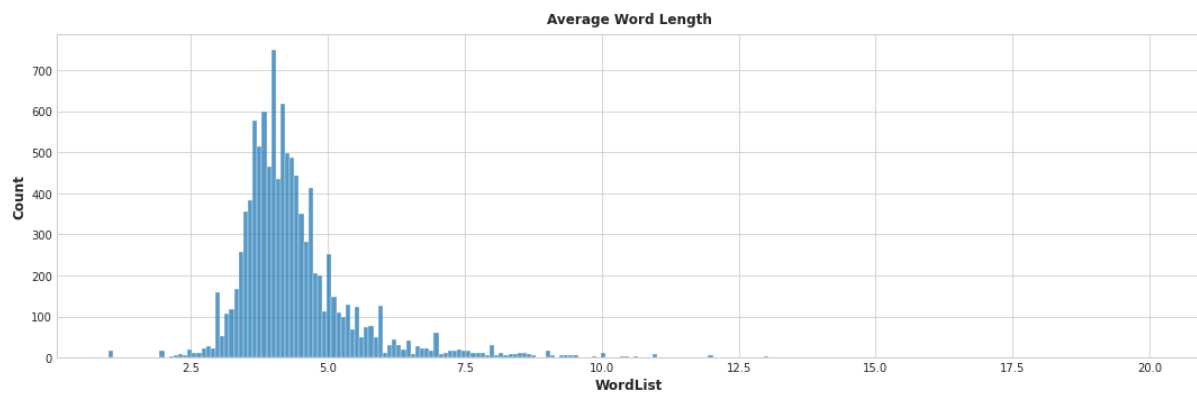*Figure. 13 Results: scrapped tweets*

Figure. 14 Results: sentiment analysis on scrapped tweets

Appendix

A. Basic Data Visualization
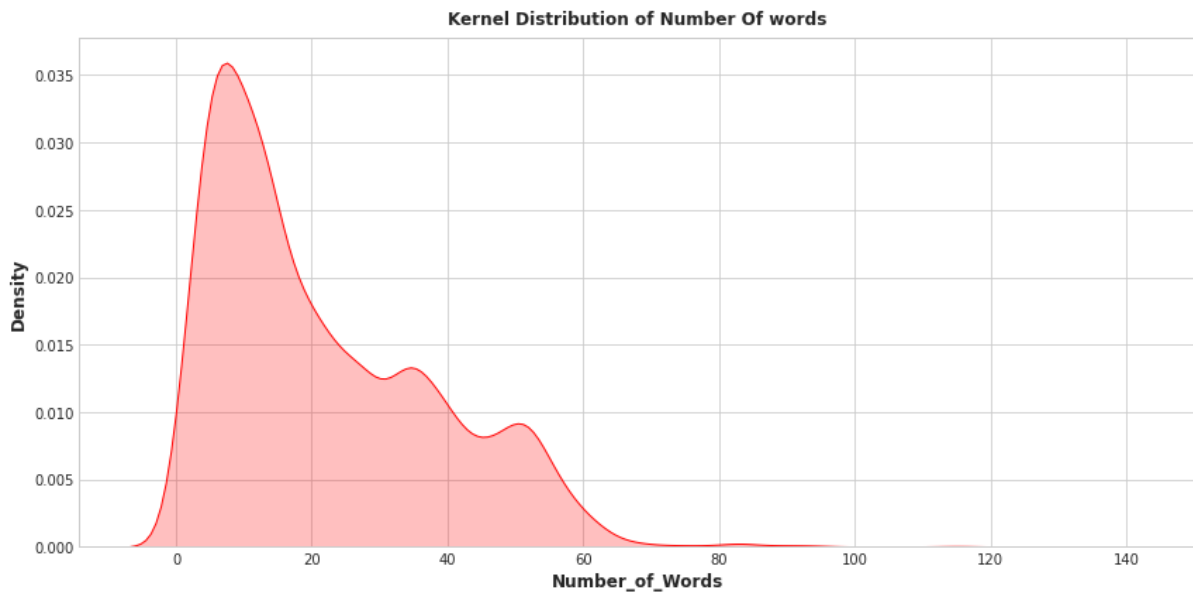
Distribution of word count in a tweet in the corpus

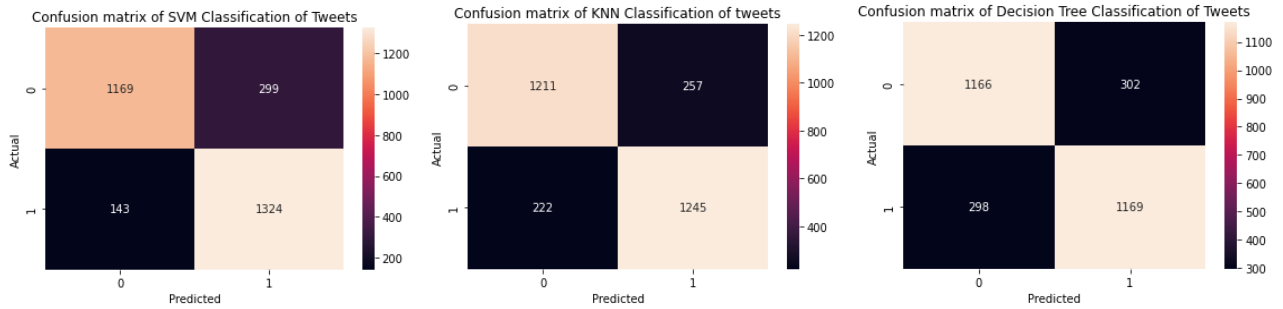*Distribution of word length in the corpus*

	Common_words	count
0	to	4929
1	the	4850
2	I	3944
3	and	3522
4	a	3344
5	is	2635
6	in	2364
7	of	2256
8	for	2187
9	it	2155
10	game	2056
11	b	2009
12	x9f	1930
13	2	1904
14	xf0	1894

Top 15 most common words in the corpus

WordCloud of Tweets

Word Cloud of tweets in the corpus

B. SVM vs KNN vs Decision Tree

Metrics (Subjectivity)	SVM	KNN	Decision Tree
F1 Score	0.857	0.839	0.796
Precision Score	0.816	0.829	0.795
Recall Score	0.903	0.849	0.797
Average Precision-Recall Score	0.785	0.779	0.735

Performance Comparison - SVM vs KNN vs Decision Tree (Subjectivity)
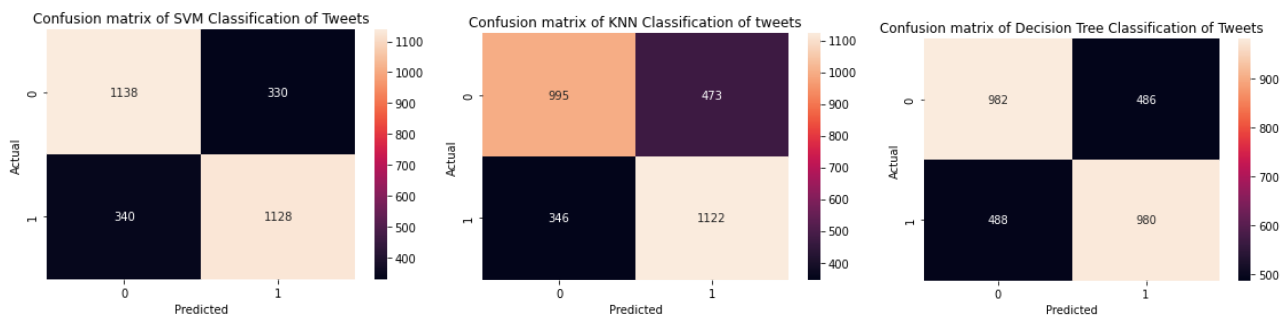
Metrics (Subjectivity)	SVM	KNN	Decision Tree
F1 Score	0.771	0.733	0.668
Precision Score	0.774	0.703	0.668
Recall Score	0.768	0.764	0.668
Average Precision-Recall Score	0.710	0.655	0.612

Performance Comparison - SVM vs KNN vs Decision Tree (Polarity)

References

1. https://www.statista.com.remotexs.ntu.edu.sg/global-consumer-survey/tool/10/gcs_sg p_202200?bars=1&index=0&absolute=0&missing=0&heatmap=0&rows%5B0%5D=v 8884_demo_hobbies&tgeditor=0&pendo=0

2. https://store.steampowered.com/charts/mostplayed

3. https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algo rithms-24f9ff6ab222#:~:text=Decision%20tree%20vs%20naive%20Bayes,the%20acc uracy%20for%20a%20toss.

4. https://steamcommunity.com/profiles/76561198105947930/recommended/1599600/

5. https://courses.media.mit.edu/2016spring/mass63/wp-content/uploads/sites/40/2016/ 02/Symbolic-vs.-Subsymbolic.pptx_.pdf

6. https://arxiv.org/abs/1711.05101

7. https://ieeexplore.ieee.org.remotexs.ntu.edu.sg/document/8090997

8. https://ieeexplore.ieee.org.remotexs.ntu.edu.sg/document/6735273

9. SenticNet 6: Ensemble Application of Symbolic and Subsymbolic AI for Sentiment Analysis