Sentiment classification in the review of Genshin Impact

732A92 Text Mining

Chenjian Shi (chesh532)

Mar 18,2022

Abstract

Nowadays, video games are a vital part of the entertainment industry, and mobile games play an essential role. Recently, a mobile game named Genshin Impact gets more and more popular and has achieved many international awards. People talk about this game on Twitter and Reddit. Therefore, we want to do the sentiment analysis on the comments of Twitter and Reddit. However, we do not have labeled data of comments in Twitter and Reddit, so in this paper, we would like to use the reviews of Genshin Impact from the Google play store as data to build a sentiment classifier for classifying the comments on Twitter and Reddit. Finally, we found that the classifier based on the BERT has the best performance among the models in this paper.

Contents

1 Introduction

Nowadays, video games are a vital part of the entertainment industry, and more and more people choose this way to relax. Meanwhile, people would like to play mobile games instead of traditional PC games or console games due to mobile phones' convenience and increasing performance. So they could play the games they like anywhere they like, for example, in the subway, or the Cafe, instead of playing games at a fixed place. Genshin impact, first published in Sep 2020, has become more popular. According to the report from BBC news(Baggs 2021), Genshin Impact stands at the top of mobile games, which earns $2 billion after 'unheard of' success in the first year. Moreover, this game was recently awarded many international awards in the game fields, like The Game Award for Best Mobile Game, Etc. (Awards n.d.). A newborn game could rarely get these achievements.

As we know, Twitter and Reddit are the most famous discussion places for people. So we could try to learn why Genshin Impact is so popular with people by analyzing the comments from Twitter and Reddit. However, currently, we do not have a data set with a sentiment label to do the analysis, which means it is hard for us to learn the reason from a comment since we do not know it is positive or not. So we need to find a way to get a model that could do the sentiment classification for the comments on Twitter and Reddit. Fortunately, we have the Google play store. The Google play store is one of the most popular video game stores on mobile phones. There are massive reviews of games about the players' views on the games, which could be used for studying the player's sentiment classification. Besides, we could use its score system as the label.

In this paper, we will try to apply these several machine learning and text mining skills to analyze the reviews from the Google play store to do the sentiment classification.

2 Theory.

2.1 Data Balancing

2.1.1 Synthetic Minority Over-sampling Technique (SMOTE)

Usually, the data in the natural lie is unbalanced, which means the number of each class/label is different. We need to balance the data to meet the requirement of the models, which requires the data has an equal number per class of labels. In this case, the Synthetic Minority Over-sampling Technique (SMOTE)(Chawla et al. 2002) would be used for balancing the data. It may be the most popular oversampling method to do the data balancing. SMOTE could not only do over-sampling the minority class but also do the under-sampling of the majority class. The algorithm of SMOTE is processing as below,

For each sample, x in the minority class, use the Euclidean distance as the standard to calculate its distance to all samples in the minority class sample set S and get its k nearest neighbors.

Then a sampling ratio is set according to the sample imbalance ratio for helping determine the sampling ratio N. For each minority class sample x, several samples are randomly selected from its k nearest neighbors, assuming that the selected nearest neighbor is xn.

At last, for each randomly selected xn, then construct a new sample with the original sample according to the following formula: $x_{new} = x + rand(0, 1) * |x - xn|$

However, there would create bias in the data sets.

2.2 Machine Learning Models

2.2.1 Non-deep Learning

Linear Support Vector Classification (LSVC) is a kind of supervised learning model and related learning algorithm for analyzing data in classification and regression analysis.(Cortes and Vapnik 1995) Here the LSVC implements a "one-vs-the-rest" multi-class strategy, thus training n classes.(Pedregosa et al. 2011)

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1} \max(0, 1 - y_i(w^T \phi(x_i) + b)),$$

Figure 1: Linear Support Vector Classification formula

Logistic regression (LR) is a generalized linear regression analysis model, which is often used in data mining, automatic disease diagnosis, economic forecasting, and other fields.(Pedregosa et al. 2011)

$$w^T x + b = ln \frac{P(Y = 1|x)}{1 - P(Y = 1|x)}$$
$$P(Y = 1|x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

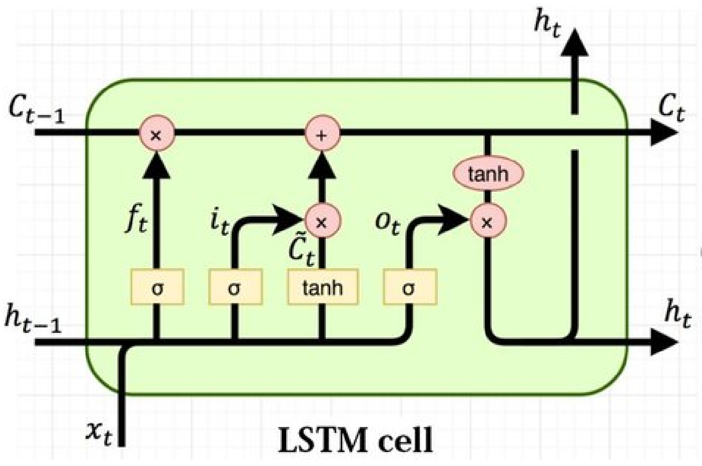
Figure 2: Logistic Regression formula

2.2.2 Deep Learning

Bidirectional Long Short-Term Memory (BiLSTM) :

Long short-term memory (LSTM) is a special recurrent neural network (RNN), mainly to solve the problem of gradient disappearance and gradient explosion during long sequence training. Simply put, LSTM can perform better in longer sequences than ordinary RNNs.

The LSTM model is composed of the input word x_t at time t, the cell state C_t, the temporary cell state $\hat{C}_t$, the hidden layer state h_t, the forgetting gate f_t, the memory gate i_t, and the output gate o_t. For the calculation process of LSTM, first, forgetting the information in the cell state and memorizing new information, the valuable information for subsequent calculations is transmitted. Then discard useless information, and the hidden layer state h_t is output at each step. Besides, the forgetting, memory, and output are controlled by the forgetting gate f_t, memory gate i_t, and the output gate o_t, which are calculated by the last time hidden layer state h_{t-1} and current input X_t.(Hochreiter and Schmidhuber 1997)



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Figure 3: LSTM-theory

(Varsamopoulos, Bertels, and Almudever 2018)

Bidirectional Long Short-Term Memory (BiLSTM) is composed of forwarding LSTM and backward LSTM. (Cornegruta et al. 2016)

For example, input "I", "Love", "You" in turn to get three vectors $[h_{L0}, h_{L1}, h_{L2}]$. Enter "you", "love", and "me" in turn to get three vectors $[h_{R0}, h_{R1}, h_{R2}]$. Finally, the forward and backward hidden vectors are spliced to obtain$[(h_{L0}, h_{R0}), (h_{L1}, h_{R1}), (h_{L2}, h_{R2})]$, namely $[h_0, h_1, h_2]$

For sentiment classification tasks, the representation of the sentences we use is often $[(h_{L2}, h_{R2})]$. Because it contains all the forward and backward information

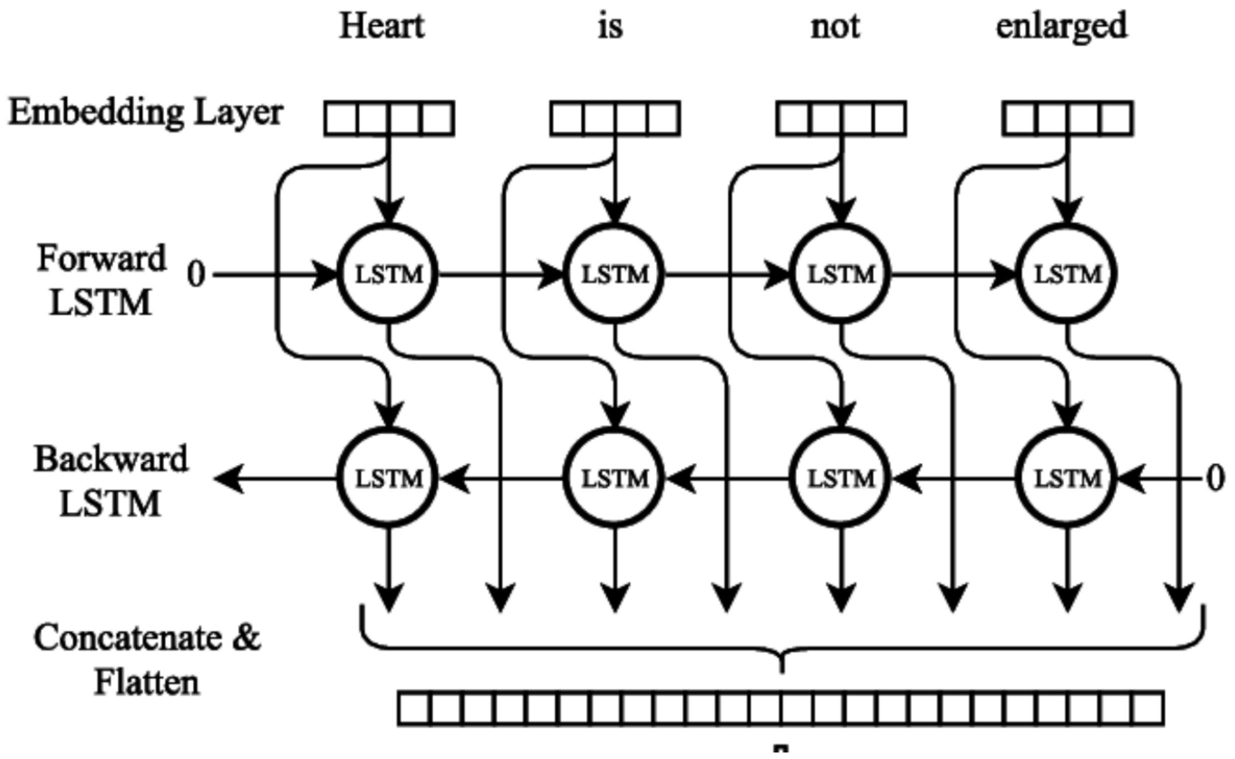


Figure 4: BiLSTM

(Cornegruta et al. 2016)

Bidirectional Encoder Representations from Transformers (BERT) :

BERT, a transformer-based bidirectional encoding representation, is a pre-training model. The two tasks during model training are to predict the words that are masked in the sentence and determine whether the two input sentences are upper and lower sentences. After the pre-trained BERT model is added with the corresponding network according to the specific task, the downstream tasks of NLP can be completed, such as text classification, machine translation, Etc.(Vaswani et al. 2017)

Although BERT is based on the transformer, it only uses the encoder part of the transformer, and its overall framework is formed by stacking the encoders of multiple layers of transformers. The encoder of each layer is composed of a layer of muti-head-attention and a layer of feed-forward. The large model has 24 layers, each layer has 16 attention, and the miniature model has 12 layers. Each layer has 12 attention. The main role of each attention is to re-encode the target word through the relevance of the target word to all words in the sentence. Therefore, calculating each attention includes three steps: calculating the correlation between words, normalizing the correlation, and obtaining the encoding of the target word through the weighted summation of the correlation and the encoding of all words.

When calculating the correlation between words through attention, first linearly transform the input sequence vector (512*768) through three weight matrices, respectively generate three new sequence vectors of query, key, and value, and use each word. The query vector is multiplied with the key vector of all words in the sequence to obtain the correlation between words, and then this correlation is normalized by softmax. The normalized weight is summed with the value weight. Finally, get a new encoding for each word.

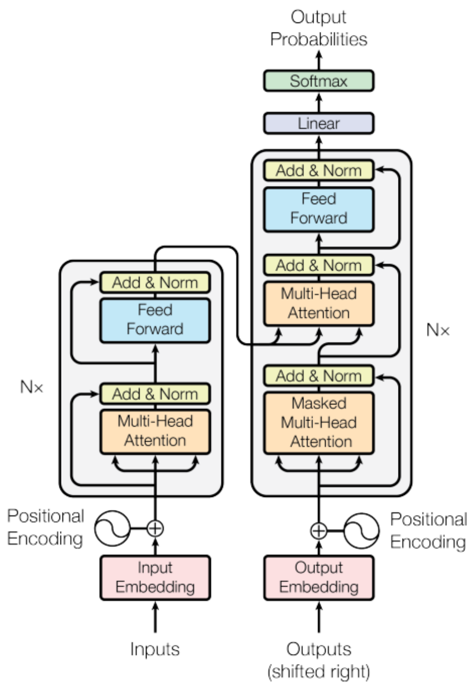


Figure 5: BERT-theory

(Figure from (Vaswani et al. 2017))

In BERT, the input vector is summed by three different embeddings, namely(Paul and Saha 2020):

Wordpiece embedding: The vector representation of the word itself. WordPiece refers to the division of words into a limited set of standard subword units, which can compromise the effectiveness of words and the flexibility of characters.

Positional Embedding: Encodes the positional information of words into feature vectors. Because our network structure does not have RNN or LSTM, the position information of the sequence cannot be obtained, so the position embedding needs to be built. There are two ways to build positional embeddings: BERT initializes a positional embedding and then learns it through training, while Transformer builds positional embeddings by formulating rules.

Segment Embedding: A vector representation used to distinguish two sentences. It is used to distinguish asymmetric sentences, such as questions and answers.

The input of the BERT model is wordpiece token embedding + segment embedding + position embedding, as shown in the figure:

(Figure from (Paul and Saha 2020))

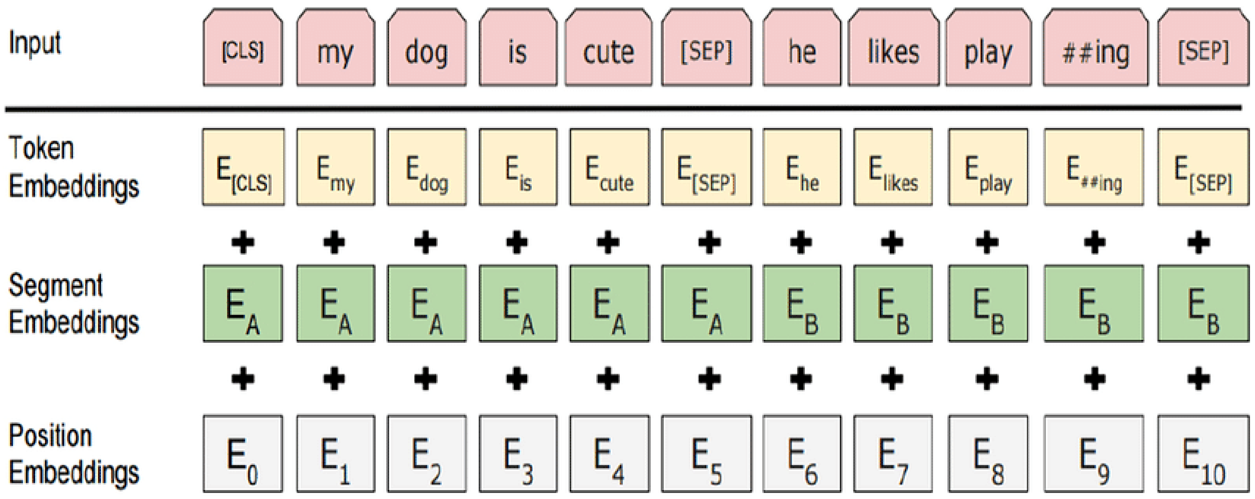


Figure 6: BERT-Input

3 Data.

3.1 Data access

The data was scraped from the reviews of Genshin Impact on Google play(Google n.d.) on Jan 09th, 2022 by python script, which is implemented from a package named google-play-scraper(JoMingyu n.d.). The original data is in the JSON form below:

```
# {
#         "userName": "Alyssa Williams",
#         "userImage": "https://lh3.googleusercontent.com/-cVEHKr7mzv8/AAAAAAAAA
#                        AI/AAAAAAAAAAA/AKF05nB2r3GUkji31m0tC4ylFNiVMpmNWA/photo.
#                        jpg",
#         "content": "This is literally the best idle game I have ever played. T
#                   he penguins waddle around and live their best lives in the
#                   cutest little outfits. I just unlocked the little penguins
#                   and I have been sobbing uncontrollably for ten minutes beca
#                   use they are so adorable. There are only two suggestions I
#                   have for this game: more of the penguin info ads. I love th
#                   em. I have learned so much about all the teeny fellas. Seco
#                   ndly, I would like to be able to name my 'guins so I can te
#                   ll them apart.",
#         "score": 5,
#         "thumbsUpCount": 54,
#         "reviewCreatedVersion": "1.16",
#         "at": datetime.datetime(2020, 2, 24, 17, 19, 34),
#         "replyContent": "Hello, We will gradually improve the various systems
#                         in the game to enhance the player's game experience.
#                         We have recorded your suggestions and feedback to the
#                         planner. If you have any other suggestions and ideas,
#                         please feel free to contact us at penguinisle@habby.
#                         com.Thank you for playing!",
#         "repliedAt": datetime.datetime(2020, 2, 24, 18, 30, 42),
#         "reviewId": "gp:AOqpTOEOIy5S9Je1F8W1BgCl6l_TCFP_QN4qGtRATX3PeB5VV9aZu
#                                          6UHfMWdYFF1at4qZ59xxLNHFqYLql5SL-k"
# },
```

According to the aim of this report, only choose the data of "at", "score" and "content," which stand for the date, content, and the score of one review, and save them as Excel files. Besides, the score range in the Google play store is from 1 to 5, which could be regarded as a different attitude to things, like 1 (Extremely Negative), 2 (Negative), 3 (Neutral), 4 (Positive), 5 (Extremely Positive).

3.2 Data Overview

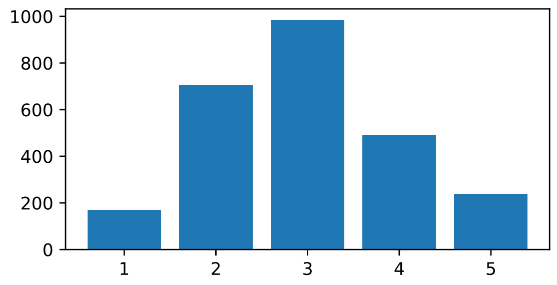
```
##                        Time  Score                          Content
## 0     2022-01-05 04:22:08      5  I literally love this game...
## 1     2022-01-07 04:54:02      5  The game itself is wonderf...
## 2     2022-01-08 00:08:37      5  This is an amazing game it...
## 3     2022-01-04 23:52:11      5  Full fledged game that you...
## 4     2022-01-05 01:08:32      5  This game is so fun! The s...
## ...                   ...    ...                            ...
## 38593 2021-10-14 09:23:25      1  I can't even play it.I jus...
## 38594 2021-09-29 11:21:58      1  The game has been fun so f...
## 38595 2021-09-29 05:19:42      1  Aniversary awards sucks as...
## 38596 2021-09-28 20:33:23      1  Overall a very good game w...
## 38597 2021-09-29 11:05:05      1  This is the only game in m...
##
## [38598 rows x 3 columns]
```

This data frame shows the shape of the original data with (38598 rows x 3 columns). Due to the limited resources or other cloud platforms like Colab, it is hard to use all these data since it takes about 10 hours to clean 3000 records. So, the data had to be sampled according to the original data's distribution with a ratio rate of 10%. The sampled data would be split into training and test sets. This data size for the training model may not be sufficient, leading to a low-performance model.

Wordcloud(Oesper et al. 2011) and matplotlib(Hunter 2007) would be used for data visualization.

```
## The shape of train data is 2586 x 3
```

```
## The count per Class of labels
```



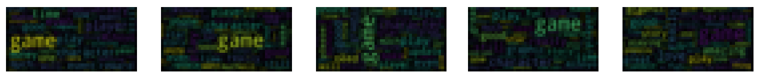
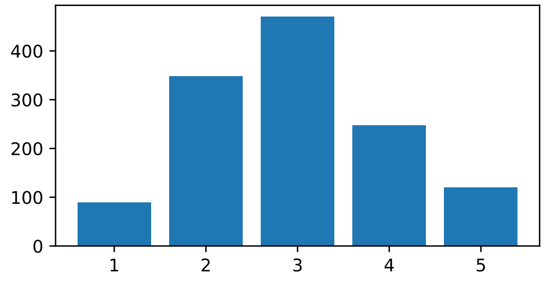


Figure 7: The wordclod per class of labels for train data

```
## The shape of test data is 1274 x 3
```

```
## The count per Class of labels
```



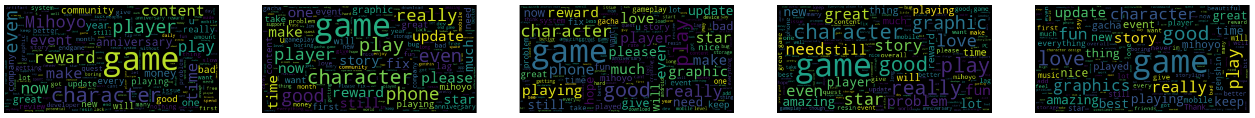


Figure 8: The wordclod per class of labels for test data

According to the figures above, we could find that the train data are imbalanced, which is necessary to balance before training. However, this data balancing would make bias to the results.

Besides, by observing the word cloud, which stands for the frequency of words per class, we could find there are some unrelated words need to be cleaned.

3.3 Data Cleaning

In practice, the most important and troubling part of deploying a machine learning model might be data cleaning. When we are deploying a machine learning model, the choice of model and the parameter adjustment depends on the objective performance of the model, which usually has standards rules to follow. However, the data we used in the model cannot be changed if the performance is not good. Besides, if the data is of low quality, for example, the data size is too small, or the label/features are not correctly written in the data sets, all these would lead to a bad performance of a model.

As for the data we use here, the content contains much informal spelling or abbreviation. Besides, even if we set the language to 'en,' there may still be non-English words or emojis. So to get a higher quality of data, we would use the way introduced below to do the data cleaning:

a) Contextual Spell Check(Goel n.d.) It is a package from the spacy universe, which uses the BERT model to correct the spelling of the words. So it could be added to the spacy pipeline. Here the doc._.outcome_spellCheck would be used as the correct content after the process. Besides, it could also be used for language detection in a way.

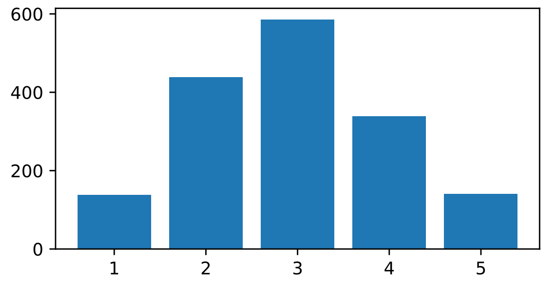
b) remove_emoji(Karim Omaya n.d.) This function removes the emoji, non-English characters, and other symbols.

The cleaning work is implemented by using the spacy(Honnibal et al. 2020) pipeline.

Since the processing time of this cleaning takes too much time (10 hours for 3000 records). The data would be processed and saved as Excel files and used to deploy later.

As introduced above, some data may not be in English or not spelled correctly. In this case, the return of the doc._.outcome_spellCheck would be null, which means some data contains the NA value in the data set. So we need to drop the NA value.

```
## The shape of train data is 1640 x 3
## The count per Class of labels for trian data
```



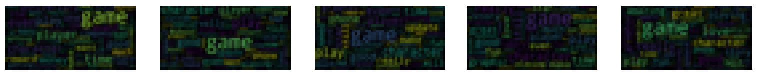
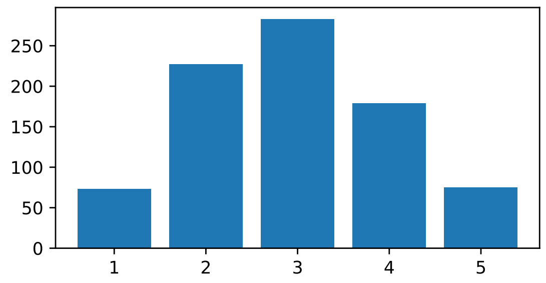


Figure 9: The wordclod per class of labels for train data

```
## The shape of test data is 837 x 3
## The count per Class of labels for test data
```



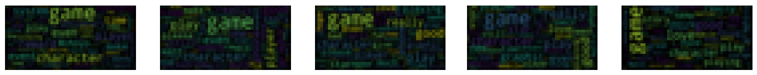


Figure 10: The wordclod per class of labels for test data

By comparing the wordcloud and shape of data before cleaning, we could find that, although the highest frequency 'key' word for the different label would still be 'game', however, the frequency is not higher than before, and other words like 'reward' get higher frequency.

4 Method.

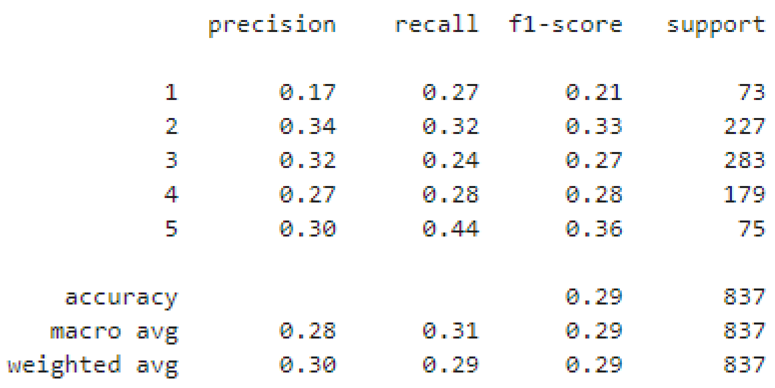
First, the data has already been divided into training and test data. Then, the SMOTE mentioned previously would be used on train data to balance the data.

After we have balanced training data, we could do the necessary data preprocessing based on different models' needs. For example, for a non-deep learning model, the preprocessing would be vectorization. Moreover, for the deep learning model, the preprocessing would be tokenization.

Then we could first apply the non-deep learning models, like LSVC and Logistic regression. Then applies the deep learning models.

5 Results.

5.1 LSVC



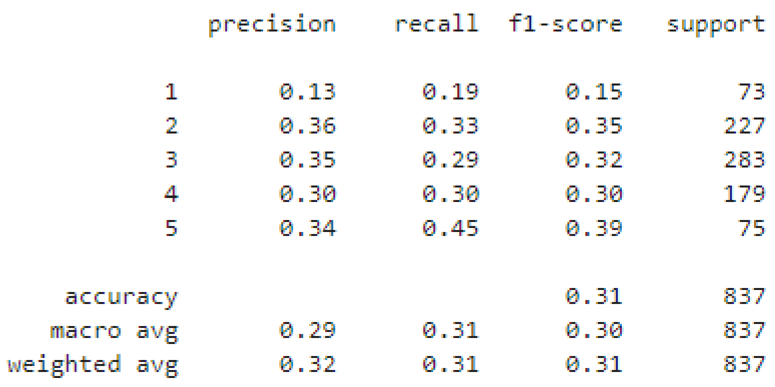
```
              precision    recall  f1-score   support

           1       0.17      0.27      0.21        73
           2       0.34      0.32      0.33       227
           3       0.32      0.24      0.27       283
           4       0.27      0.28      0.28       179
           5       0.30      0.44      0.36        75

    accuracy                           0.29       837
   macro avg       0.28      0.31      0.29       837
weighted avg       0.30      0.29      0.29       837
```

Figure 11: LSVC-result

5.2 LR



```
              precision    recall  f1-score   support

           1       0.13      0.19      0.15        73
           2       0.36      0.33      0.35       227
           3       0.35      0.29      0.32       283
           4       0.30      0.30      0.30       179
           5       0.34      0.45      0.39        75

    accuracy                           0.31       837
   macro avg       0.29      0.31      0.30       837
weighted avg       0.32      0.31      0.31       837
```

Figure 12: LR-result

5.3 BiLSTM

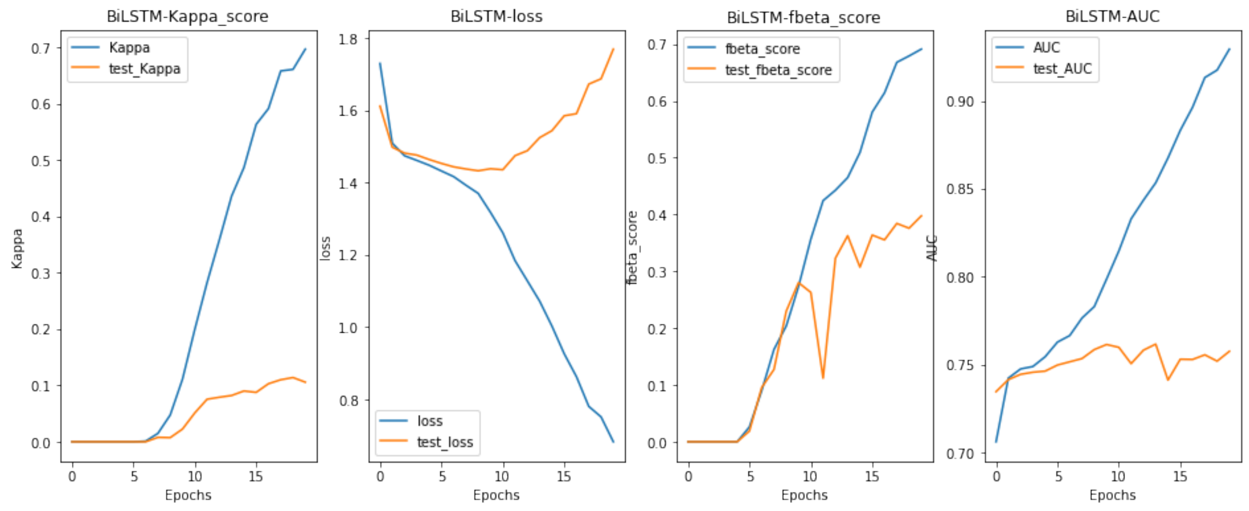


Figure 13: BiLSTM-result

5.4 BERT

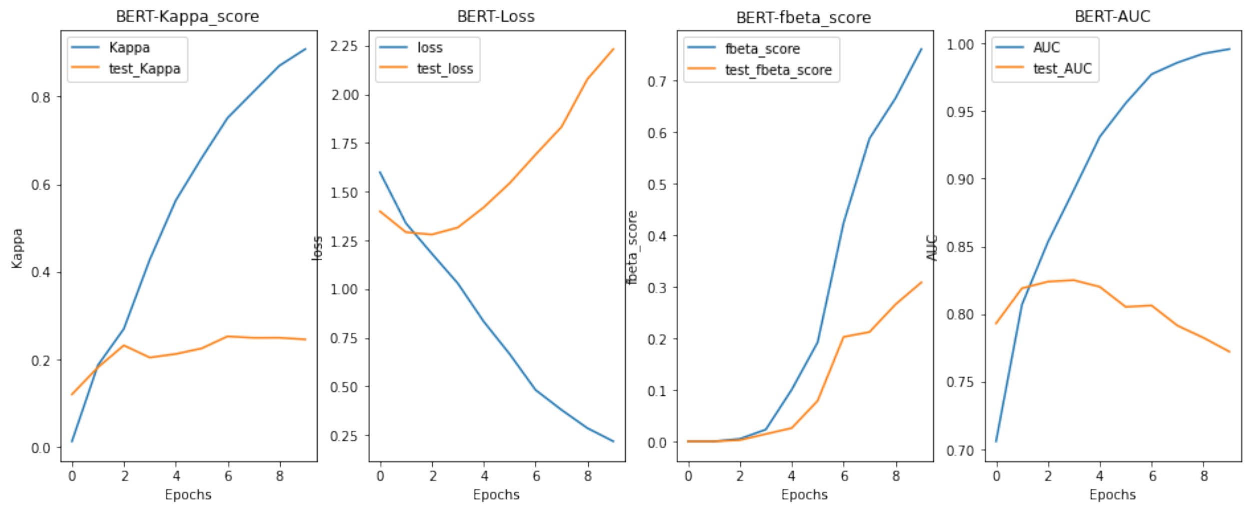


Figure 14: BERT-result

6 Discussion.

6.1 Non-deep learning models:

- Figure 11 shows that the average macro f1-score of LSVC is about 0.29, precision is 0.28, and recall is 0.31.
- Figure 12 shows that the average macro f1-score of LR is about 0.30, precision is 0.29, and recall is 0.31.

Based on the definition below:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$F1 - score = 2 * \frac{precision\ *\ recall}{precision\ +\ recall}$$

We could find that LSVC and LR do not perform well on these data sets.

6.2 Deep learning models:

- Figure 13 shows that, for BiLSTM on test data, the Kappa score is about 0.10584, the loss is about 1.76934, fbeta-score is about 0.39763, and AUC is about 0.75750.
- Figure 14 shows that, for BERT on test data, the Kappa score is about 0.24542, the loss is about 2.23189, the fbeta-score is about 0.30858, and AUC is about 0.77219.

Based on the definition below(Abadi et al. 2015):

- Kappa score: The score lies in the range [-1, 1]. A score of -1 represents complete disagreement between two raters, whereas a score of 1 represents the complete agreement between the two raters. A score of 0 means agreement by chance.
- fbeta-score: It is the weighted harmonic mean of precision and recall. Output range is [0, 1]. Works for both multi-class and multi-label classification. In order to compare the result between non-deep learning models and deep learning models, here we set the $\beta = 1$ in the fbeta-score so that the fbeta-score would be theoretically the same as the f1-score.

$$F_\beta = (1 + \beta^2) * \frac{precision\ *\ recall}{(\beta^2\ *\ precision)\ +\ recall}$$

- Area Under the ROC Curve (AUC): AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.("Classification: Roc Curve and AUC | Machine Learning Crash Course | Google Developers," n.d.)

We could find that, currently, BiLSTM and BERT do not perform well on these data sets.

We could find that although the loss score and the fbeta-score of the BERT are lower than the BiLSTM, the Kappe score and AUC of BERT are higher than BiLSTM.

In this case, it seems that the BiLSTM has a similar performance to the BERT. However, according to the result plots, we could find that for the BiLSTM, the most scores become stable, but for BERT, they are still growing. Besides, the final results for the BiLSTM and BERT are not the best. However, they are still better than the non-deep learning model.

7 Conclusion.

By comparing the results above, we could find that all these models we used in this paper do not have a good performance on these data. However, the BERT is the best model among them.

If we want to get a well-performance machine learning model, it is essential to have a high quality and large enough data set for training and testing since it is the basis of a model. The model learns the feature(information) from this data set and applies it to other places. However, due to the limitation of the computer resources, we have to use a small size data set, which may not contain enough features for models to learn. Besides, the quality of the data is also not good enough. Although it is common in real life that the data is imbalanced, the method we used here, oversampling, is not enough for balancing the data, which also brings a bias to the data. For further work, maybe we need to consider using the combination of oversampling and undersampling to create more balanced data.

What is more, although the deep learning model usually performs better than the non-deep learning model when applying to some big data analysis, it takes time and needs to find the best parameters for it, like in this paper, the performance of the four models is quite similar. However, the non-deep learning model results much quicker and easier than the deep learning model.

8 Further work

In order to improve the performance of the model, we could have a try on the method below:

- 1) Try to use the combination of oversampling and undersampling to create more balanced data.
- 2) Try to use other measurements(metrics) for models.
- 3) Try to use other kinds of models, for example, CNN, Etc.
- 4) If possible, try to use a complete data set.

Reference

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2015. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." https://www.tensorflow.org/.

Awards, The Game. n.d. "Best Mobile Game | Nominees | the Game Awards." Accessed January 14, 2022. thegameawards.com/nominees/best-mobile-game.

Baggs, Michael. 2021. *BBC News*. BBC. https://www.bbc.com/news/newsbeat-58707297.

Chawla, N. V., K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. "SMOTE: Synthetic Minority over-Sampling Technique." *Journal of Artificial Intelligence Research* 16 (June): 321–57. https://doi.org/10.1613/jair.953.

"Classification: Roc Curve and AUC | Machine Learning Crash Course | Google Developers." n.d. *Google*. Google. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=zh_cn.

Cornegruta, Savelie, Robert Bakewell, Samuel Withey, and Giovanni Montana. 2016. "Modelling Radiological Language with Bidirectional Long Short-Term Memory Networks." *CoRR* abs/1609.08409. http://arxiv.org/abs/1609.08409.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-Vector Networks." *Machine Learning* 20 (3): 273–97.

Goel, Rajat. n.d. "Contextual Spell Check spaCy Universe." Accessed January 9, 2022. spacy.io/universe/project/contextualS

Google. n.d. "Genshin Impact - Apps on Google Play." Accessed January 9, 2022. play.google.com/store/apps/details?id=com

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8): 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

Honnibal, Matthew, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. "spaCy: Industrial-strength Natural Language Processing in Python." https://doi.org/10.5281/zenodo.1212303.

Hunter, J. D. 2007. "Matplotlib: A 2d Graphics Environment." *Computing in Science & Engineering* 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.

JoMingyu, etc, kluhan. n.d. “Google-Play-Scraper.” Accessed January 9, 2022. github.com/JoMingyu/google-play-scraper.

Karim Omaya, Denis da Mata. n.d. “Removing Emojis from a String in Python - Stack Overflow.” Accessed February 15, 2021. https://stackoverflow.com/questions/33404752/removing-emojis-from-a-string-in-python.

Oesper, Layla, Daniele Merico, Ruth Isserlin, and Gary D Bader. 2011. “WordCloud: A Cytoscape Plugin to Create a Visual Semantic Summary of Networks.” *Source Code for Biology and Medicine* 6 (1): 7.

Paul, Sayanta, and Sriparna Saha. 2020. “CyberBERT: BERT for Cyberbullying Identification.” *Multimedia Systems*, November, 1–8. https://doi.org/10.1007/s00530-020-00710-4.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12: 2825–30.

Varsamopoulos, Savvas, Koen Bertels, and Carmen Almudever. 2018. “Designing Neural Network Based Decoders for Surface Codes,” November.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need.” https://arxiv.org/abs/1706.03762.