Text classification on comments of Genshin Impact via non-deep learning and deep learning models

732A92 Text Mining

Chenjian Shi (chesh532)

Jan 15,2022

Abstract

This project deal with a text classification problem of comments on Genshin Impact through using non-deep learning and deep learning models, which includes the whole process of data from data accessing to results analysis.

Contents

1 Introduction

Genshin Impact (Mihoyo n.d.) is an open-world ARPG developed by a Chinese company named Mihoyo. This game was recently awarded many international awards in the game fields, like The Game Award for Best Mobile Game, etc. (Awards n.d.)

In this case, more and more people comment on this game on Twitter, Reddit, etc. But for now, it is hard to classify these comments manually since there are thousands of data. So, here the text classification could help us understand whether these comments belong to the categories, like Extremely Negative, Negative, Neutral, Positive, Extremely Positive.

In this paper, I would use some typical methods/models below:

1. Non-deep learning:

- Linear Support Vector Classification
- Logistic Regression

2. Deep learning:

- Recurrent neural networks
- Bidirectional Encoder Representations from Transformers

I would try to analyze the results and compare the performance of these methods/models based on the results, data quality, etc.

2 Data

In this section, I would introduce how to get access to the data and the way I choose to do the pre-processing(data cleaning)

2.1 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[./main.py], 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:AOqpTOE0Iy5S9Je1F8W1BgCl6l_TCFP_QN4qGtRATX3PeB5VV9aZu
#                                           6UHfMWdYFF1at4qZ59xxLNHFqYLql5SL-k"
# },
```

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

2.2 Data Overview

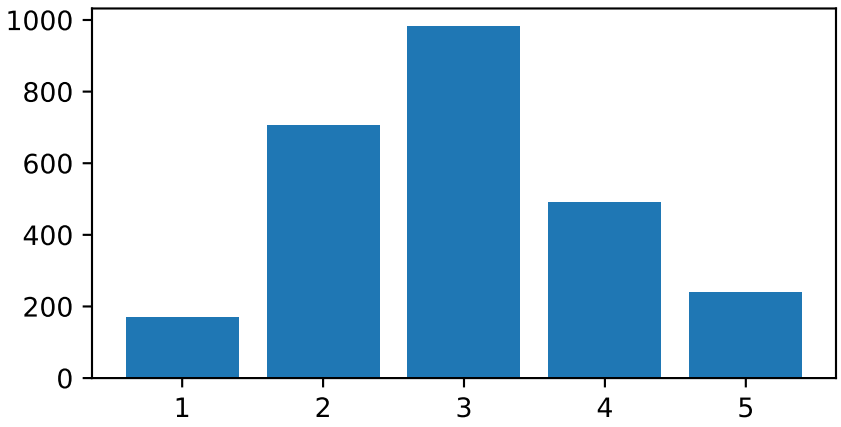
```
##                      Time  Score                        Content
## 0     2022-01-05 04:22:08      5  I literally love this game...
## 1     2022-01-07 04:54:02      5  The game itself is wonderf...
## 2     2022-01-08 00:08:37      5  This is an amazing game it...
## 3     2022-01-04 23:52:11      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 of my computer or other cloud platforms like Colab, it is hard for me to use all these data this time since it takes me about 10 hours to clean 3000 records of data. So, I had to sample the data 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, which may lead to a low-performance model.

Here I would use wordcloud(Oesper et al. 2011) and matplotlib(Hunter 2007) for data visualization.

According to the figures below, the train data are both imbalanced, which is necessary to make it balanced before training, this would make bias to the results. And, the word cloud for the data, the word 'game' is the highest frequency word for all the labels.

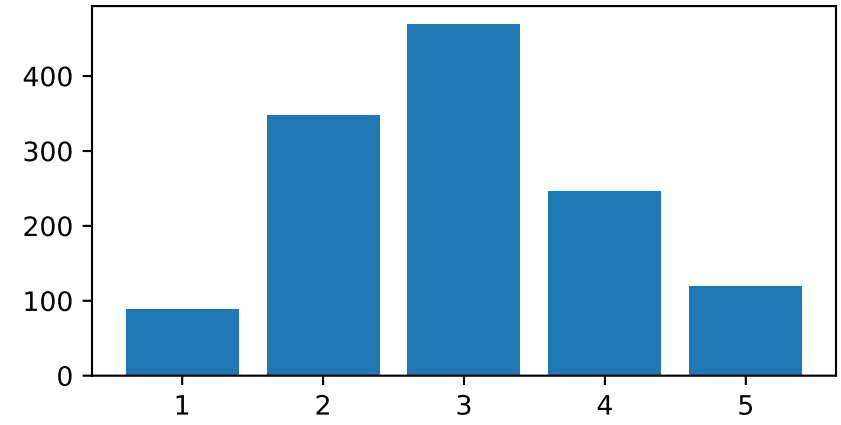
```
## The shape of train data is 2586 x 3
## The count per Class of labels
```




```
## The wordclod per class of labels
```

Figure 1: The wordclod per class of labels for train data

```
## The shape of test data is 1274 x 3
## The count per Class of labels
```




```
## The wordclod per class of labels
```

Figure 2: The wordclod per class of labels for test data

2.3 Data Cleaning

In practice, the most important and troubling part of deploying a machine learning model might be data cleaning. It is because 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. But 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 is 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 contain 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.) This is a package from the spacy universe, which uses the BERT model to do the correction of the spelling of the words. 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 be also used for language detection in a way.

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

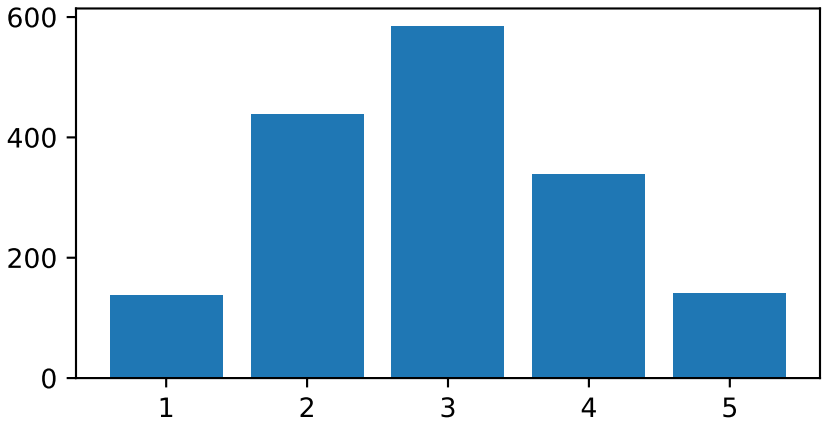
```
# stackoverflow.com/questions/33404752/removing-emojis-from-a-string-in-python
def remove_emoji(string):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002500-\U00002BEF"  # chinese char
                               u"\U00002702-\U000027B0"
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               u"\U0001f926-\U0001f937"
                               u"\U00010000-\U0010ffff"
                               u"\u2640-\u2642"
                               u"\u2600-\u2B55"
                               u"\u200d"
                               u"\u23cf"
                               u"\u23e9"
                               u"\u231a"
                               u"\ufe0f"  # dingbats
                               u"\u3030"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', string)
```

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

Since the processing time of this cleaning takes too much time (10 hours for 3000 records of data). I save the processed data into Excel files and use them for the later model deploying.

Like I introduced above, there maybe some data is not in english or not spell 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.

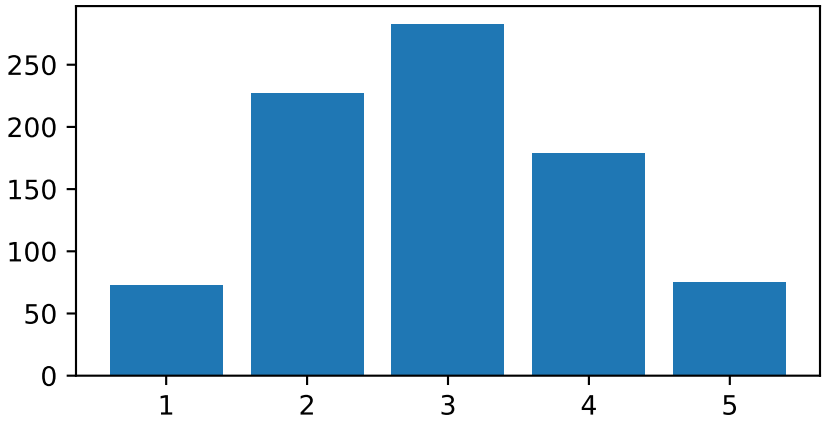
```
## The shape of train data is 1640 x 3
## The count per Class of labels
```




```
## The wordclod per class of labels
```

Figure 3: The wordclod per class of labels for train data

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




```
## The wordclod per class of labels
```

Figure 4: The wordclod per class of labels for test data

3 Non-Deep Learning

In this section, I would introduce preprocessing of the data (data balance and vectorizer) and the two classical non-deep learning models for text classification. For both the data vectorizer and the model training, I would use API from the scikit-learn[Pedregosa et al. (2011)](Buitinck et al. 2013)

3.1 Data preprocessing

3.1.1 Data balance

According to the previous introduction, 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, I would use the Synthetic Minority Over-sampling Technique (SMOTE)(Chawla et al. 2002), which may be the most popular over-sampling 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 the distance from it 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 on the data sets.

3.1.2 Vectorizer

Vectorization is a process that translates the given text into numerical vectors, which transform the text feature into a way that the machine learning algorithms could understand and learn. In this paper, I would use the CountVectorizer, which not only implements tokenization but also the occurrence counting[Pedregosa et al. (2011)](Buitinck et al. 2013). Besides, it also needs a long time for Vectorization, in this case, I also use the cache from the previous processing for time saving.

3.2 Linear Support Vector Classification

3.2.1 Theory

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 "one-vs-the-rest" multi-class strategy, thus training n classes.[Pedregosa et al. (2011)](Buitinck et al. 2013)

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

Figure 5: Linear Support Vector Classification formula

3.2.2 Result

	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

3.3 Logistic Regression

3.3.1 Theory

Logistic regression It 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 6: Logistic Regression formula

3.3.2 Result

	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

3.4 Summary

According to the results above, we could find that the logistic regression has better performance than the LSVC does, however, the total accuracy for both models is not good enough. Although both of them are linear models, the LSVC has based on the calculation of the distance between the data, which means the normalization is important for it, but when we are doing the text classification, it is hard to do the normalization. As for logistic regression, it has no problem about it, so it performs better.

4 Deep Learning

In this section, I would introduce preprocessing of the data (data balance and tokenizer) and the two classical deep learning models for text classification.

All the processes of the data pre-processing and model training would use the API from tensorflow(Abadi et al. 2015) and keras(Chollet et al. 2015) and scikit-learn[Pedregosa et al. (2011)](Buitinck et al. 2013). Besides, due to the limitation of the computation resources, I would try to build the model as simple as possible but still have better performance than the models in the previous section.

4.1 Data Preprocessing

4.1.1 Data Balance

For the deep learning model, especially the neural network, the data balancing is also an important task for having better performance. In this paper, I would like to use the most classical one, which is compute the class weight by using the function compute_class_weight(Pedregosa et al. 2011) from scikit-learn. In this case, all the data would have a different initial weights for neural network.

4.1.2 Tokenizer

Like the Vectorizer we did in the previous section, the neural network also cannot learn the data directly without any pre-processing. It is a method of transforming the given text into a small unit, which is named token. In this case, a token would be divided into three different types, which include word, character, and subword. This is important for neural network because the neural network need to use these tokens to get the marks of words and prepare the vocabulary for them. Besides, after the tokenizer, it is also important to make the length aligned for both training data and the test data.

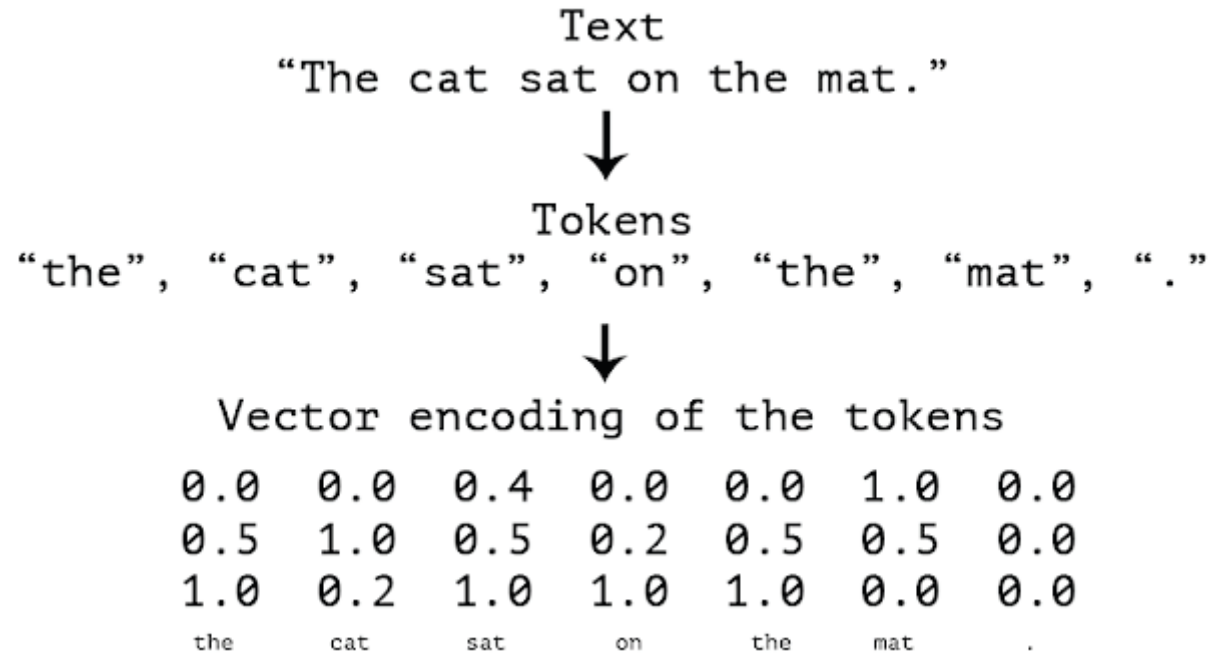


Figure 7: Tokenizer Theory

(Manning Publications 2018)

4.2 Recurrent neural networks

4.2.1 Theory

Recurrent neural network (RNN) is a kind a neural networks that process sequential data.(Goodfellow, Bengio, and Courville 2016) Technically,RNN is takes a loop for iterating over the state changing steps,the RNN use the same w in the same time for the whole training process,

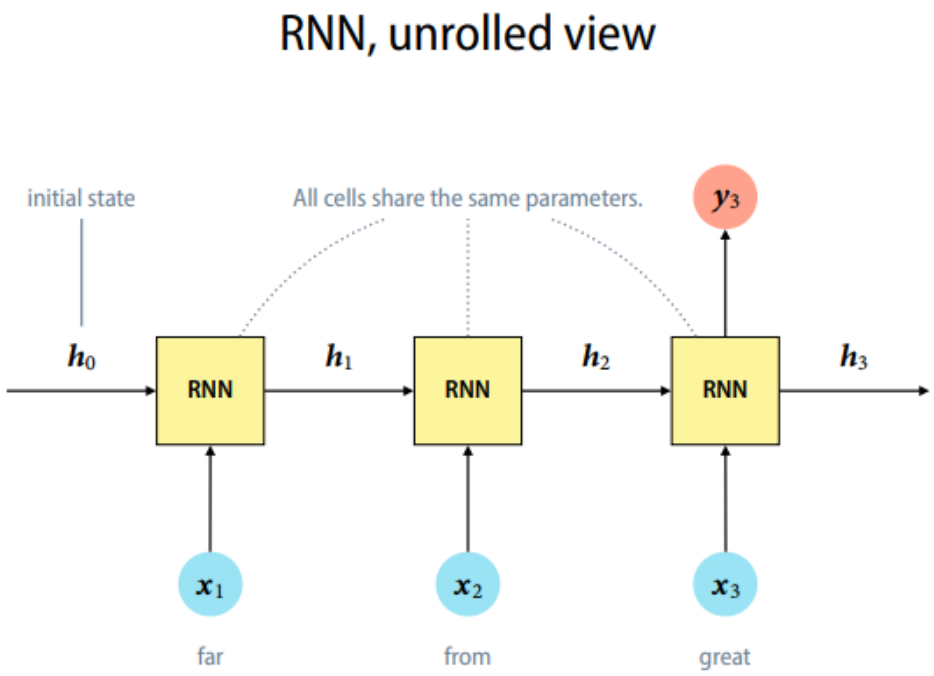


Figure 8: RNN, unrolled view

(Figure from 732A78 Deep Learning,Lecture 5,May 16,2021)

The model used in the paper is build based on the torturial from the Tensorflow(Abadi et al. 2015) and keras(Chollet et al. 2015)

https://www.tensorflow.org/text/tutorials/text_classification_rnn

https://www.tensorflow.org/guide/keras/rnn

4.2.2 Result

The accuracy of this simple RNN is about 0.311, which is not actually better than the performance of the model in the previous section.

4.3 Long Short-Term Memory

4.3.1 Theory

The Long Short-Term Memory (LSTM) actually is a mutaion of the RNN, which was designed for solving the gradients problems.(Hochreiter and Schmidhuber 1997)

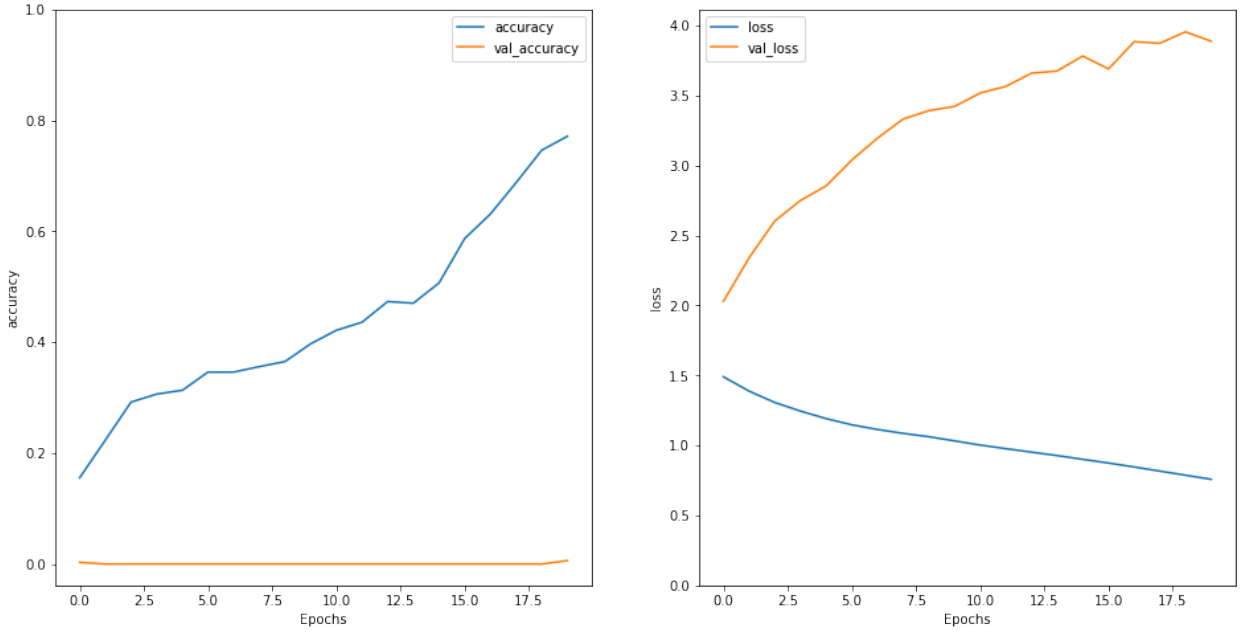
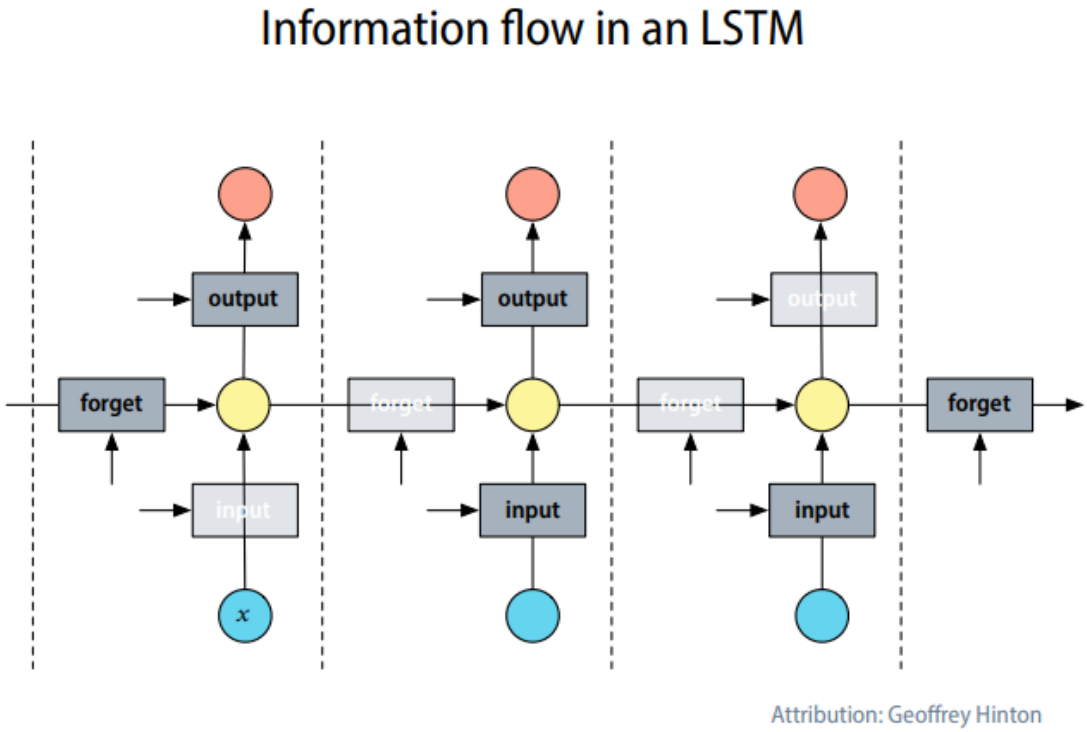
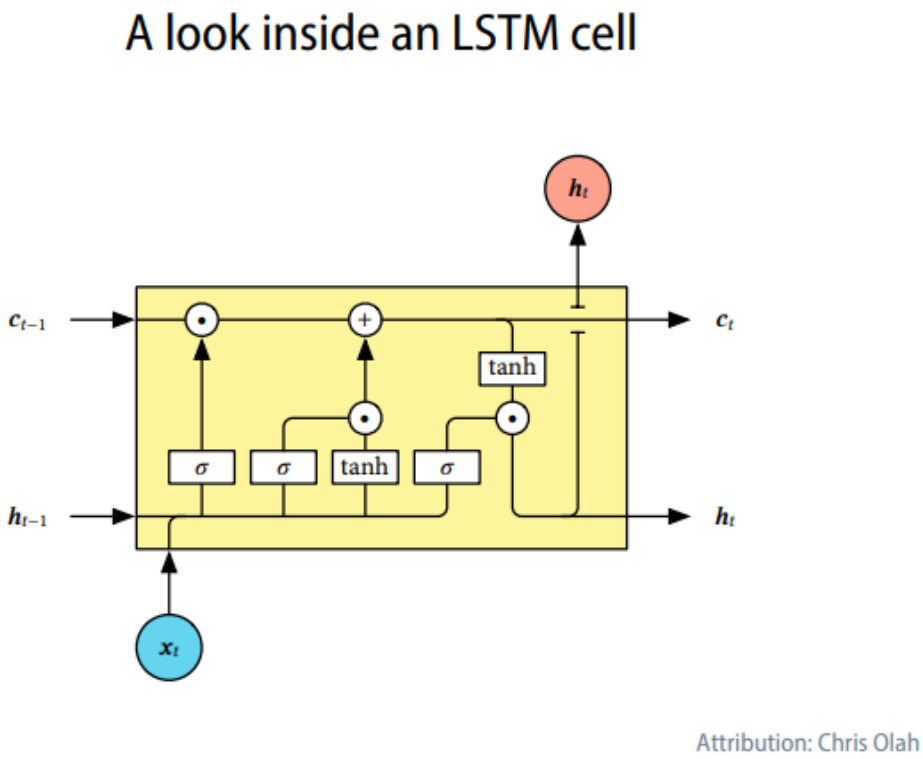


Figure 9: Result for RNN



(Figure from 732A78 Deep Learning,Lecture 5,May 16,2021)



(Figure from 732A78 Deep Learning,Lecture 5,May 16,2021)

4.3.2 Result

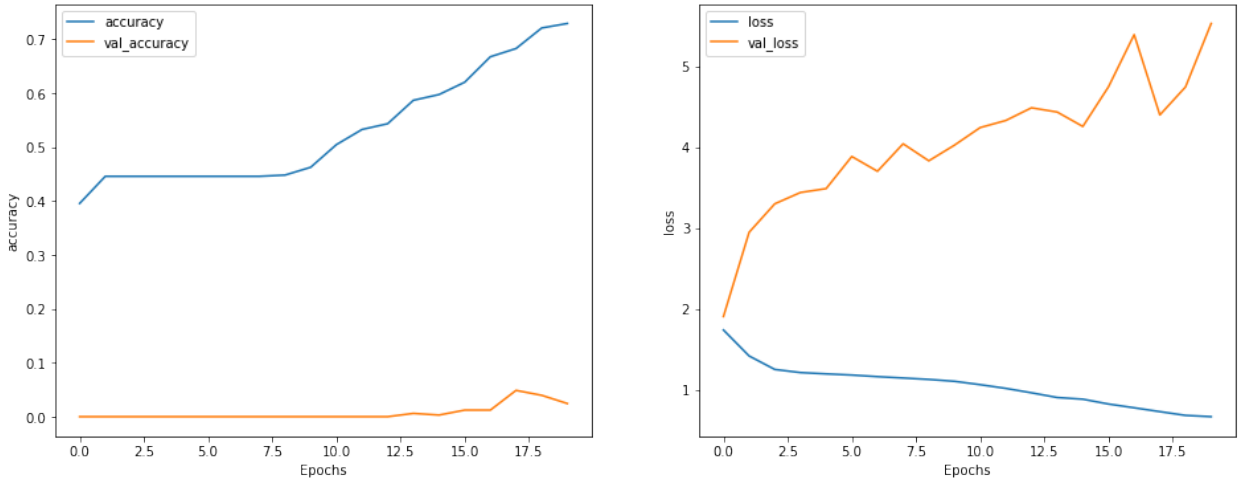


Figure 10: Result for LSTM

The accuracy of LSTM is about 0.311, which is not close to the RNNs'.

4.4 Bidirectional Encoder Representations from Transformers

4.4.1 Theory

Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from an unlabeled text by jointly conditioning on both left and right context in all layers.(Devlin et al. 2019)

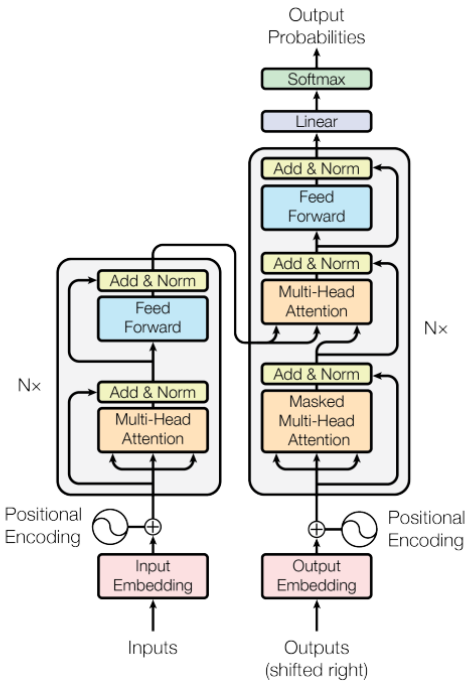


Figure 11: BERT-theory

(Figure from (Vaswani et al. 2017))

In this section, we use the BERT model which is pre-trained from Tensorflow[Turc et al. (2019)](Abadi et al. 2015)

4.4.2 Result

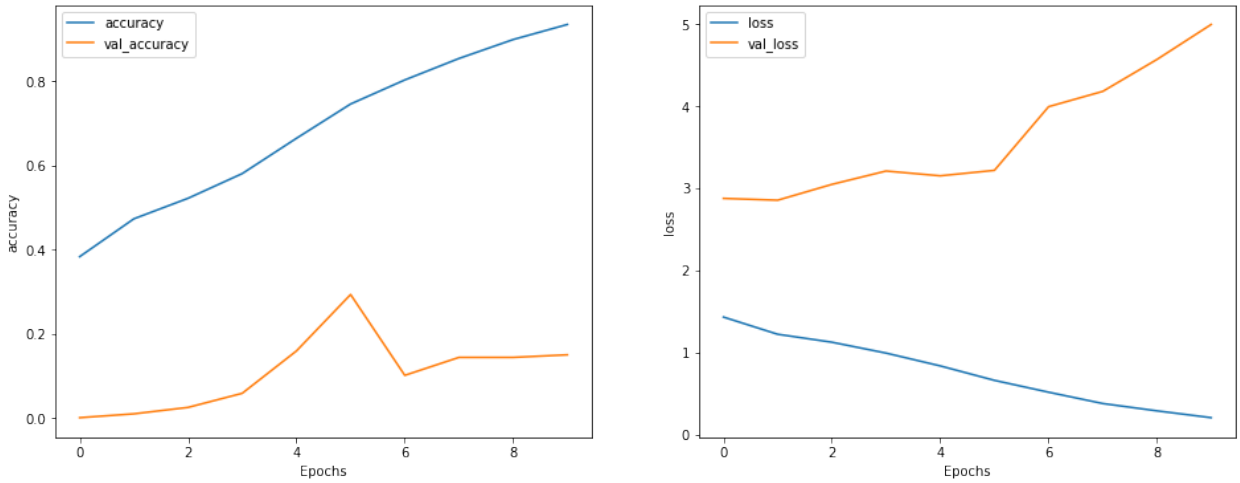


Figure 12: Result for BERT

Although this BERT only trained with 10 epochs, which is less than the RNN's 20 epochs, the accuracy get huge improved compared with the previous RNN models, which increased from 0.311 to 0.415.

4.5 Summary

By comparing these deep-learning models, it is obvious that the BERT achieved better performance than RNN. If we train the BERT model with more epochs, there will be further performance improvement.

5 Conclusion

In general, the models previously mentioned do not perform well because of low accuracy, which means it is hard to do the job of comment sentiment analysis. I think the reason which leads to these results like below:

- The data size is not large enough, which makes the data lack some features.
- Lacking computation resources. Non-deep learning models may have the same performance compared with the simple deep learning models. But only if the computation resources are enough and the data size(feature) is large enough, the deep learning model could have better performance.

By comparing to the official documents/tutorial in the Tensorflow(Abadi et al. 2015), which gets a high accuracy of about 0.85, there is also a important reason:

- The data quality is not good enough, for example, the data is imbalanced in this paper and the content may still contain some non-English words which may lead to the low performance of data.

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.

Buitinck, Lars, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, et al. 2013. "API Design for Machine Learning Software: Experiences from the Scikit-Learn Project." In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–22.

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.

Chollet, François et al. 2015. "Keras." https://keras.io.

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

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." https://arxiv.org/abs/1810.04805.

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

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

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.

Manning Publications. 2018. "Deep Learning for Text ." https://freecontent.manning.com/deep-learning-for-text/.

Mihoyo. n.d. "Genshin Impact – Step into a Vast Magical World of Adventure." Accessed January 14, 2022. genshin.mihoyo.com/en/home.

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.

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.

Turc, Iulia, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "Well-Read Students Learn Better: On the Importance of Pre-Training Compact Models." *arXiv Preprint arXiv:1908.08962v2*.

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.