Table of content

Abstract

In recent years, a huge volume of textual information has been produced every day with the rapid development of social media and e-commerce. That textual information, especially the customer reviews, can significantly affect the decision-making of businesses and future customers since they can reveal previous users' experiences and attitudes towards the products or services. Therefore, there is growing demand in sentiment analysis for the purpose of text mining and analysis.

In this project, our group expects to conduct sentiment analysis for rating prediction on a Kindle book according to the corresponding customer reviews. We start with data cleaning and processing to make the textual information tidier and clearer. Then a bunch of text feature engineerings, which includes BOW, TFIDF, as well as embedding layer building, have been applied for the preparation of model training. At the next step, a series of machine learning algorithms, such as random forest, RNN, LSTM, BLSTM and transformer model BERT, will be trained for rating prediction. In addition, the experiment details, including prediction method selection (regression or classification) and hyperparameter tuning, will also be compared and discussed in this report. Finally, BERT is selected as the best predictive model based on the size of the weighted F-1 score.

Introduction

Sentiment analysis, which is also named opinion mining, has become one of the most popular research areas in natural language processing since the 21st century (Pozzi et al., 2017). It is used to analyse people's opinions, sentiments, attitudes and emotions from written language. Nowadays, the increasing significance of sentiment analysis has coincided with the accelerating growth of social platforms. Therefore, for the first time in human history, we now have a tremendous amount of opinionated data recorded in digital form for analysis (Liu, 2012).

In this report, a series of machine learning algorithms, such as random forest, RNN, LSTM, BLSTM, as well as the transformer model BERT, will be applied to conduct sentiment analysis for rating prediction on a Kindle book. After taking into account the interpretability and predictability, the model performing the best in rating prediction will be recommended to potential practitioners in relevant industries.

Literature Review

There is a long history of text analysis techniques, and we briefly review the widely used models in this section.

Random forest is a tree-based machine learning model for solving classification and regression tasks (Ho, 1995). It was proposed by Ho to overcome the overfitting problem of decision tree (Fawagreh, Gaber, & Elyan, 2014). Neural networks are different techniques from Random Forest which have been shown to perform well in solving complex problems such as natural language processing.

(Recurrent neural networks) RNN is one of neural networks, dealing with sequential data where data points have defined ordering (Schuster & Paliwal, 1997). RNN was inspired by David Rumelhart's work in 1986 (Rumelhart, Hinton, & Williams, 1986). There is an obvious

limitation in RNN. The outputs mostly rely on previous input information since RNN processes data in sequential order (Graves & Schmidhuber, 2005). This limitation is overcomed in bidirectional recurrent neural networks (Schuster & Paliwal, 1997). BRNN was introduced by Schuster and Paliwal in 1997. RNN can be trained simultaneously in positive and negative time direction using all available input information from the past and future of a given time frame (Schuster & Paliwal, 1997).

Long Short Term Memory (LSTM) was invented in 1997 to solve another limitation of RNN (Hochreiter & Schmidhuber, 1997). RNN has difficulty dealing with long sequences since it can face issues such as exploding or gradient vanishing (Hochreiter, Bengio, Frasconi, & Schmidhuber, 2001). LSTM allows passed information to skip the processing of the current cell and move on to the next which enables it to retain the memory for a long sequence (Ghoshal, n.d.). Bidirectional Long Short Term Memory (BLSTM) was proposed by Graves and Schmidhuber in 2005 to solve the phoneme classification problem (Graves & Schmidhuber, 2005). Similar to the bidirectional feature of BRNN, BLSTM runs inputs in two directions so that information from both past and future can be preserved.

Transformer neural network architecture was introduced in 2017 by a team at Google Brain (Vaswani et al., 2017). The difference between Transformer and RNN is that Transformer can process the entire input in parallel instead of sequentially since its attention mechanism provides context for any position in the input sequence (Wikipedia, n.d.). Therefore, it can greatly reduce training time. Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based model introduced by Google in 2019 (Devlin, Chang, Lee & Toutanova, 2018). It is pre-trained on huge amounts of unlabeled data from BooksCorpus and Wikipedia and it learns information from a sequence of words in two directions in all layers. As a result, the pre-trained BERT can be fine-tuned with only one additional output layer to build models with great performance (Devlin et al., 2018).

Model

Random Forest

Random Forest works by generating multiple classifiers/models, and they learn and make predictions independently. One model is a tree, and features are randomly chosen at each decision node. Then, all the trees form a random forest. When different models have different prediction results, they will vote (voting method), and the result with the most votes shall prevail. For example, if someone trains 10 trees, 8 of them are True and 2 are False, then the result will be True.

RNN, LSTM, & BLSTM

Three models were used in "task B", RNN, LSTM, and BLSTM. RNN (recurrent neural network) is a class of artificial neural networks where directed, or undirected graphs were formed along a temporal sequence by the connections between nodes (Dupond, 2019). RNNs are ideal for applications with natural language processing as they are good at dealing with sequential data (Ankit, 2020). However, RNNs were suffering from long-distance dependence problems (Bender et al. 2003) because of the vanishing gradient (Basodi et al. 2020).

The solution to this problem is the LSTM model (Long short-term memory) (Ankit, 2020). LSTM were invented by Hochreiter and Schmidhuber in 1997 (Hochreiter & Schmidhuber,

1997). LSTM stores previous input information in memory units during each temporal state. By doing so, LSTM effectively alleviated the long-distance dependence problem and, therefore, improved large-vocabulary speech recognition (Sak et al. 2014). However, LSTM ignores future information, which provides room for improvement (Ankit, 2020).

BLSTM improves LSTM by using the bidirectional propagation mechanism (Graves & Schmidhuber,2005), concatenating the outputs of two LSTMs, including both historical and future information. Bi- LSTM consistently achieve better performance than LSTM (Ankit, 2020).

BERT

Bidirectional Encoder Representation for Transformer (BERT) is an NLP model developed by Google Research in 2018 (Moonat, 2021). By pre-training on vast amounts of text data (like Wikipedia), its fine-tuning will be faster and more accurate on small datasets. (Venelin, 2021). So, we use BERT as an advanced improvement for the NLP-sentiment analysis.

It exploits a Transformer, an attention mechanism that learns contextual relations between words in a text. But different from transformer architecture, BERT is just an encoder stack to read the text input without the decoder stack to predict(Horev, 2018). Masked LM (MLM) and Next Sentence Prediction (NSP) are two strategies of BERT to read the text. The sentiment analysis in this project is a classification task similar to the NSP, and we can build a classification layer on top of the Transformer output for the [CLS] token to achieve (Horev, 2018).

The BERT model version in the experiment is BERTBASE which has 12 layers,768 hidden layers as well as 12 attention layers, so the parameters are approximately 110 million (Winastwan, 2022)

Experiment and Result Analysis

Data Processing

As the reviews and [summaries are represent] similar sentiment expression, combining these two features as one variable. Besides, to get the performance of each selected model, it is necessary to split the 9000 pieces of data into a training set (7200)and a test set(1800), and the ratio is 0.8:0.2. This means that the comparison of the performance of the different following models will use the same data for testing, which guarantees consistency. Besides, the training set will be divided into a training set and a validation set later to select the best model.

Each model executes on the provided data, but messy data can lead to degraded performance or even erroneous results, while clean data is a prerequisite for good model performance. In this experiment, as the reviews come from the network, the raw data contains some browser information, such as the websites and useless information after the sign '@' (emails), these information need to be cleaned from the data. Besides, the stop words and punctuations are not helpful to the review sentiment analysis, so they also need to be removed. Furthermore, as some numbers in the reviews represent the rating scores, numbers are not deleted from the raw data. For further accurate model prediction, converting all uppercase to lowercase, splitting words according to the semantics (tokenize) and making lemmatization. Therefore, the cleaning

methods contain removing Html, things after '@', stop words and punctuations, converting to lowercase, tokenizing and making lemmatisation.

As for the target values, the labels of classification are encoded as 0 to 4, so the range of rating should be deducted by 1, which means rating 0 represents the previous rating 1, and rating 4 represents the previous rating 5.

Task A

Model Preparation

The engineered text features (BoW and TF-IDF) need to be chosen to train the Random Forest model. BoW is a kind of representation of sample data. It produces a dictionary, and each word is represented by the number of times the word appears in the text, while TF-IDF considers the term frequency as well as the importance of a word to a document.

According to previous experience (CoreJT, 2019), trigrams are used the most. Nonetheless, in principle, if it can be solved with bigrams, then never with trigrams. It is rare for n to take ≥4. Thus, combined with comprehensive considerations, the hyperparameter (ngram_range) in BoW and TF-IDF will be set 1 and 2. In order to remove the words that appear in almost all texts and prevent misspelling, the min_df and max_df are set as 2 and 0.9 separately.

Split Data into Train and Validation

As the best model needs to be selected, the training set will be split again. The test size will be set as 0.2, which means there are 20% of training data become validation data.

Modelling

Through the previous preprocessing, the raw data has been represented in BOW or TF-IDF. Next step is to train the chosen model --- Random Forest. However, it should be demonstrated whether regression or classification is more suitable for this case.

All of the different types of Random Forest models are trained in different combinations of hyperparameters by GridSearchCV. This means that it will train all hyperparameter combinations, and evaluate the best hyperparameter combination according to cross-validation.

As for the training of the Random Forest in this experiment, 'n_estimators', 'max_features' and 'max_depth' will be the hyperparameters that can be controlled in the Random Forest. 'n_estimators' is the number of decision trees being built in the forest. Generally speaking, if 'n_estimator' is too small, it is easy to underfit. However, if it is too large, it is hard for calculation (Lisenpy, 2021). Therefore, it is selected around the default value (100). To be specific, it sets in 80, 100 and 120. 'max_features' is the maximum number of features used for node splitting process. It will choose from 200, 250 and 300 depending on the common type(sqrt), which means at most sqrt(total features) is considered when dividing (Pavan, 2020). The common values of max depth can be between 10 to 100 (Lisenpy, 2021), so it is set as 20,60, and 100. In summary, there are 27 combinations for each Random Forest model. After gird searching, using the best hyperparameter combination to fit each different type of model. For each model, take the corresponding validation set to predict the result. For making a comparison, F1-score is used for evaluating which engineered text feature (BoW or TF-IDF) and which way (regression or classification) is appropriate.

Comparison

According to the "Table 1", the best Random Forest model is the type of classification with TF-IDF, which means multi-classification is more suitable for the whole sentiment analysis in this case.

	F1-score
RF (Classification) with BOW	0.4085
RF (Classification) with TF-IDF	0.4091
RF (Regression) with BOW	0.2119
RF (Regression) with TF-IDF	0.2068

Table (1)

Model evaluation

Putting the test data（1800） in the adjusted model trained by the combination of train data and validation data. Then the F1-score of the classification Random Forest model with TF-IDF is 0.4014.

Task B

Model Selection

Three models were built in "Task B", the RNN, LSTM, and BLSTM. As mentioned in the last section, BLSTMs can always perform better than LSTMs, while LSTMs always perform better than RNNs. Therefore, the first step in "Task B" is to verify the theories and conjunctively consider the theories and the experimental results to select the most suitable model for the Amazon Kindle book review project. Three groups of contrast experiments were conducted. In each group of contract experiments, the hyper-parameters were kept the same for the three models. The f1 score was compared with the increase in the models' complexity. The results are shown in "Table 2."

According to "Table 2", BLSTM always provide better f1 scores compared with RNN and LSTM, which is also in line with the theories. In conclusion, BLSTM is an ideal model, and it will be trained to increase its accuracy and be used for prediction.

Model training

The second step is to train the BLSTM model. Specifically, that is to adjust the number of epochs, the dropout rate, the number of embedding and hidden layers, the batch size, the learning rate and the length of the sentence. The goal of doing so is to eliminate the underfitting or overfitting of the model and help the model predict more accurately.

The increase in the number of embedding and hidden layers, the increase in the length of the sentence, and the decrease in the dropout rate will increase the complexity of the model. If the model is too complex, it will cause an overfitting problem and will hurt the accuracy of the model. Similarly, if the model is too simple, it will cause an underfitting problem, decreasing the accuracy of the model prediction. The learning rate controls the speed of the model training. Lower learning rates require more training epochs to reach the optimal solution, whereas larger learning rates require fewer training epochs. If the learning rate is too large, the model will quickly converge to a suboptimal solution (Jason, 2019). Therefore, a combination of a lower learning rate and more epochs may increase the accuracy of the model prediction.

	RNN	LSTM	BLSTM
Total epoch = 20 Dropout rate= 0.5 Embedding = 16 Hidden = 32 Batch = 128 Learning rate = 0.001 Length of the sentence = 256	Train f1 score = 0.1276, Validation f1 score = 0.1035, Underfitting	Train f1 score = 0.1165, Validation f1 score = 0.0990, Underfitting	Train f1 score = 0.5569, Validation f1 score = 0.4134, Underfitting
Total epoch = 20 Dropout rate= 0.3 Embedding = 16 Hidden = 32 Batch = 128 Learning rate = 0.004 Length of the sentence = 256	Train f1 score = 0.1646, Validation f1 score = 0.1445, Underfitting	Train f1 score = 0.1468, Validation f1 score = 0.1000, Underfitting	Train f1 score = 0.9389, Validation f1 score = 0.4429, overfitting
Total epoch = 20 Dropout rate= 0.2 Embedding = 32 Hidden = 64 Batch = 128 Learning rate = 0.004 Length of the sentence = 256	Train f1 score = 0.1188, Validation f1 score = 0.1130, Underfitting	Train f1 score = 0.7246, Validation f1 score = 0.4081, overfitting	Train f1 score = 0.9946, Validation f1 score = 0.4392, overfitting

Table (2)

In this project, hundreds of tries have been attempted. After carefully adjusting the parameters, the best model was shown where the dropout equals 0.5, the number of embedding and hidden layers equal 32 and 16 separately, the batch size equals 128, and the learning rate equals 4e-3 with 20 epochs. The validation f1 score of the model is 0.4477 while the test f1 score is 0.4626

Task C

Dataset class

First, we define a Dataset class to generate our input by inheriting this class. In the customized process, we import BertTokenizer from transformer packages as a tokenizer to transform the input sentence in the text column into a sequence of tokens that BERT expects (Winastwan, 2022). Considering that the maximum length of a sequence allowed for BERT is 512, we have truncated those sequences that exceed the number limit and pad vacant space. The sequence has also been added [CLS], [SEP], and [PAD] tokens automatically to form a new input which is convenient for the model to recognize. (Winastwan, 2022).

Model building

We use 'bert-base-cased' to build the pre-trained model. The simple Neural Network model with a linear layer and Relu activation function is enough for prediction since the complexity of the BERT is enough for our small training dataset (5760).

Training Loop

For fine-tuning the pre-trained model, we focus on three hyperparameters that can influence the model performance: epoch, learning rate and batch size (Kamsetty et al., 2020). Initially,

HYPERPARAMETER	SEARCH SPACE
Number of epochs:	2,3,4
Batch size:	16,32
Learning rate (adam):	2e-5 3e-5 5e-5

Table (3)

inspired by the previous research (Winastwan, 2022), we set epoch=10, a larger epoch, to explore the validation performance on different epochs with a fixed learning rate =1e-6 and batch size=2. Then, the epoch should not over 6 since the validation loss increases obviously after six. Thus, we use the new setting (epoch=6, learning rate=1e-6, batch size=2) to train model and use it as a benchmark to explore other two hyperparameters. The epoch four seems to have the lowest validation loss in this configure but may not be the best for others. The BERT author also recommended a search space as followings(table)to find the optimal hyperparameter that works well across all tasks (Devlin et al., 2019)

Interestingly, all the learning rates are all larger than 1e-6. A high learning rate will cause training errors (Bengio, 2012), so we compare the performance of 1e-6 and 2e-5 to testify and find that the F1 score (0.5612) is a little higher than our benchmark (0.5518) but the validation loss (0.9131) is also too larger than benchmark (0.5224). Next, we only change the batch size to 16 to compare the different performances. The larger batch would not improve the model with a lower F1 score (0.4660) and higher validation loss (1.1448). We also replace the optimizer Adam with SGD to compare performance, and the results suggests that the Adam is better.

Combination	Validation F1 score	Validation Loss	Validation Accuracy
epoch=6,batchsize=2, LR=1e-6, Adam **(Benchmark)**	0.5518	0.5224	0.5556
epoch=6,batchsize=2, LR=2e-5, Adam	0.5612	0.9131	0.5646
epoch=6,batchsize=16,LR=1e-6,Adam	0.4660	1.1448	0.5007
epoch=6,batchsize=2, LR=1e-6, SGD	0.1899	0.8111	0.2118

Table (4)

Model evaluation

We put the remained test data (1800) in the adjusted model trained by the training and validation data. Then the F1 score of the BERT model is 0.5770, and the final model's accuracy in predicting the sentiment of the book review is 58.39%.

Result Analysis

Our experimental results show that BERT achieves the highest result with the F1 score of 0.5770. The great performance comes from the advantage of pre-training. BERT has made a significant improvement in creating new state-of-the-art results for many natural language processing tasks. However, it is not absolute that BERT is better than other models discussed in the report considering that the performance of a model may depend on the size of data and the specific tasks. We have also found that performances of the models were improved through retraining. The validation set was added to the training set when retraining data. It is reasonable that models perform better as the size of training data increases.

Model	Epoch	Learning rate	Test set F1 score
Random forest (TF-IDF)	N/A	N/A	0.4014
BLSTM	20	4.00E-03	0.4580
BERT	4	1.00E-06	0.5770

Table (5)

Limitation and Assumptions

As for Random Forest, although the hyperparameters are selected according to the papers and it also provides reasonable results, this selected model has the problem of overfitting. Cross-validation may be a significant tuning tool for model performance, and we could apply this on the Random Forest, BLSTM.

Early Stopping is a better method to halt the training of Neural Networks at the right time and avoid overfitting based on the validation loss.

For BERT, we fine-tune the model by hand like a simple grid search given research for our limitation in coding ability, we can fine-tune BERT using more advanced search algorithms like Bayesian Optimization and Population Based Training (Kamsetty et al., 2022). The package Ray tune may help to improve our BERT model.

We find that the combination order of 'summary' and 'Review text' has influenced our final model predicted ability. The Performance of BLSTM in unseen data, the remained 20% test and Kaggle show an obvious difference, The potential reason need to explore more.

	'Combine'='Review text' +'summary'	Combine'='summary'+ Review text'
F1 score for test	0.4353	0.4514
F1 score for Kaggle test	0.4240	0.4586

Table (6)

Conclusion

In conclusion, the transformer model BERT has the highest performance in terms of prediction accuracy amongst the models we have discussed above. The outstanding forecasting performance mainly attributes to pre-training of the model on a huge volume of textual data and combination of Mask Language Model (MLM) and Next Sentence Prediction (NSP) techniques. Moreover, training NLP models usually takes hundreds of hours of training time, but using a pre-trained model like BERT can be less time-consuming. Although the prediction results of BERT are not interpretable easily, our primary goal is to classify the customer review texts accurately and precisely thus model interpretability is not the most important factor we consider in this case. Therefore, BERT is selected as the best predictive model, and it will be recommended to potential practitioners in relevant industries.

Reference

Ankit, A. (2020). *Sentiment Analysis using Bi-Directional LSTM https://www.linkedin.com/pulse/sentiment-analysis-using-bi-directional-lstm-ankit-agarwal/*

Basodi, S., Ji, C., Zhang, H., & Pan, Y. (2020). *"Gradient amplification: An efficient way to train deep neural networks".* Big Data Mining and Analytics. 3 (3): 198. doi:10.26599/BDMA.2020.9020004. ISSN 2096-0654.

Bender, E., Sag, A., & Wasow, T. (2003).*Syntactic Theory: a formal introduction(Second Edition).*CSLI Publications.

Bengio, Y. (2012). Practical recommendations for gradient-based training of Deep Architectures. *Lecture Notes in Computer Science*, 437–478. https://doi.org/10.1007/978-3-642-35289-8_26

CoreJT. (2019, February 10). Natural Language Processing | (11) N-gram language model and its application [Blog post]. Retrieved from https://blog.csdn.net/sdu_hao/article/details/86893580

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). *Bert: Pre-training of deep bidirectional Transformers for language understanding*. arXiv.org. Retrieved May 21, 2022, from https://arxiv.org/abs/1810.04805

Dupond, S. (2019). *"A thorough review on the current advance of neural network structures".* Annual Reviews in Control. 14: 200–230.

EasyAI. (2020). *Random Forest.* Retrieved May 13, 2022, from https://easyai.tech/ai-definition/random-forest/

Fawagreh, K., Gaber, M. M., & Elyan, E. (2014). Random forests: from early developments to recent advancements. *Systems Science & Control Engineering: An Open Access Journal*, *2*(1), 602-609.

Gers, A., Schraudolph, N., Schmidhuber, J. (2002). *"Learning Precise Timing with LSTM Recurrent Networks"* . Journal of Machine Learning Research. 3: 115–143.

Ghoshal, A. (n.d.). Part 1: Journey of BERT. Retrieved from https://www.arpanghoshal.com/post/part-1-journey-of-bert

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, *18*(5-6), 602-610.

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Horev, R. (2018, November 17). *Bert explained: State of the art language model for NLP*. Medium. Retrieved May 21, 2022, from https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Ho, T. K. (1995). Random decision forests. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=598994

Jason, B. (2019). *Understand the Impact of Learning Rate on Neural Network Performance, https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/#:~:text=The%20learning%20rate%20controls%20how,and%20require%20fewer%20training%20epochs.*

Kamsetty, A., Fricke, K., & Liaw, R. (2020, October 6). *Hyperparameter optimization for 🤗transformers: A guide*. Medium. Retrieved May 21, 2022, from https://medium.com/distributed-computing-with-ray/hyperparameter-optimization-for-transformers-a-guide-c4e32c6c989b

Lisenpy. (2021, May 19). Description of random forest parameters [Blog post]. Retrieved from https://blog.csdn.net/lisenby/article/details/117037374

Liu, B. (2012). *Sentiment analysis and opinion mining.* San Rafael: Morgan & Claypool Publishers. Retrieved from https://www-morganclaypool-com.ezproxy.library.sydney.edu.au/doi/pdf/10.2200/S00416ED1V01Y201204HLT016

Moonat, D. (2021, December 28). *Fine-tune Bert model for sentiment analysis in Google Colab*. Analytics Vidhya. Retrieved May 21, 2022, from https://www.analyticsvidhya.com/blog/2021/12/fine-tune-bert-model-for-sentiment-analysis-in-google-colab/

Pavan, V. (2020). *Random Forest Hyperparameter Tuning: Processes Explained with Coding*. Retrieved from https://www.upgrad.com/blog/random-forest-hyperparameter-tuning/

Pozzi, F., & Pozzi, F. A. (2017). *Sentiment analysis in social networks* (1st edition) .Retrieved from https://www-sciencedirect-com.ezproxy.library.sydney.edu.au/book/9780128044124/sentiment-analysis-in-social-networks

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533-536. doi: 10.1038/323533a

Sak, H., Senior, A., & Beaufays, F. (2014). *"Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling"* .

Schuster, M., & Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions On Signal Processing*, *45*(11), 2673-2681. doi: 10.1109/78.650093

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Venelin. (2021, April 26). *Multi-label text classification with Bert and Pytorch Lightning*. Curiousily. Retrieved May 21, 2022, from https://curiousily.com/posts/multi-label-text-classification-with-bert-and-pytorch-lightning/

Wikipedia. (n.d.). Transformer (machine learning model). Retrieved from https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)

Winastwan, R. (2022). *Text Classification with BERT in PyTorch*. Medium. Retrieved 21 May 2022, from https://towardsdatascience.com/text-classification-with-bert-in-pytorch-887965e5820f.