Regression Implementation of BERT Model for Amazon Rating Prediction

Xiang Cao
Institute for Aerospace Study
University of Toronto
Toronto, ON, Canada
Tonyxiang.cao@Mail.utoronto.ca

Abstract

This report describes a Bidirectional Encoder Representations from Transformers (BERT) model implementation for competing in the Kaggle competition of Amazon rating prediction, in fulfillment for University of Toronto CSC 2515 Machine Learning final project. The machine learning task is to predict the rating scores of reviews on Amazon, given the dataset that contains features such as review text, reviewer ID, Item ID. A pre-trained BERT model (bert-large-cased) is used to pre-process, tokenize, and train the review text. To predict rating of the Amazon reviews, a customized regression output layer is added on top of the BERT model for final output. The final performance has achieved a MSE of 0.31, surpassing the strong baseline of MSE 0.65, and ranked no.5 in the leaderboard. Author's Kaggle team ID is Xiang Cao.

1 Data Preprocessing

1.1 Amazon Music Review Dataset

In the CSC2515 Kaggle competition, 200,000 samples of Amazon music product reviews were given as THE training set. The ground truth is the overall rating, ranging from 1-5. The available input features including reviewtime, reviewerID, reviewText, summary, unixReviewTime, category, price, itemID, and reviewHash. The testset contains 10,000 samples of reviews with same input feature except missing the overall rating. The task for the Kaggle competition is to generate predictive overall rating score for those 10,000 test set, and evaluate the MSE (mean square error) as performance metric, which measures if the predictive rating is close to the actual ratings. A snippet of the training dataset is shown in Figure 1.

	overall	reviewTime	reviewerID	reviewText	summary	unixReviewTime	category	price	itemID	reviewHash
0	4.0	08 24, 2010	u04428712	So is Katy Perry's new album "Teenage Dream" c...	Amazing that I Actually Bought This...More Ama...	1282608000	Pop	$35.93	p70761125	85559980
1	5.0	10 31, 2009	u06946603	I got this CD almost 10 years ago, and given t...	Excellent album	1256947200	Alternative Rock	$11.28	p85427891	41699565
2	4.0	10 13, 2015	u92735614	I REALLY enjoy this pairing of Anderson and Po...	Love the Music, Hate the Light Show	1444694400	Pop	$89.86	p82172532	24751194
3	5.0	06 28, 2017	u35112935	Finally got it . It was everything thought it ...	Great	1498608000	Pop	$11.89	p15255251	22820631
4	4.0	10 12, 2015	u07141505	Look at all star cast. Outstanding record, pl...	Love these guys.	1444608000	Jazz	$15.24	p82618188	53377470

Figure 1 Training Data Overview

1.2 Format conversion

The original data is given in the format of json file, organized in paired structure between the feature name and feature values. Since the author decided to use Tensorflow and Keras framework for this machine learning project, DataFrame format from Python Pandas library would be easier to work with. The author used an online JSON to CSV converter to convert the given train.json and test.json to train.csv and test.csv [1]. The csv files are uploaded on Google Drive, and imported to Google Colab as Pandas Dataframe data structure for pre-processing and training.

1.3 Data pre-processing

One of the challenges of the Amazon Review dataset is its variety of data forms. It contains natural language data(reviewText), categorical data (category, reviewerID and itermID, time (reviewTime), numerical value(overall and price). Considering the reviewText and summary column has the most important information to predict overall rating, the author decide to convert several features into string format, and feed it for NLP (natural language processing) models. The author joined several column togethers into the reviewText column, which now includes category, summary, reviewerID, itemID, reviewText. In addition, the processed data failed to be processed in the model initially, since there are samples with empty text. The author dropped the rows with missing values using DataFrame.dropna function.

2 BERT Model

2.1 Model Selection

Without any prior experience in NLP, the author read several papers and found Bidirectional Encoder Representations from Transformers (BERT) model have state of the art performance in many NLLP task including general language understanding (GLUE), questions answering (SQuAD), Natural Language Inference(MNLI) [2]. The BERT model uses transformers model, which learns contextual relations between words in bidirectional way. The transformer encoder read text input, and tokenize them into a sequence of vectors. Those tokens are processed and trained in the pre-trained BERT deep neural network, and then a decoder output a sequence of vectors, with same index from the input token, and lastly generate prediction depending on the task. Figure 2 shows the overall structure of BERT model, starting from word feature embedding, onto transformer encoder and eventually classification output [3].

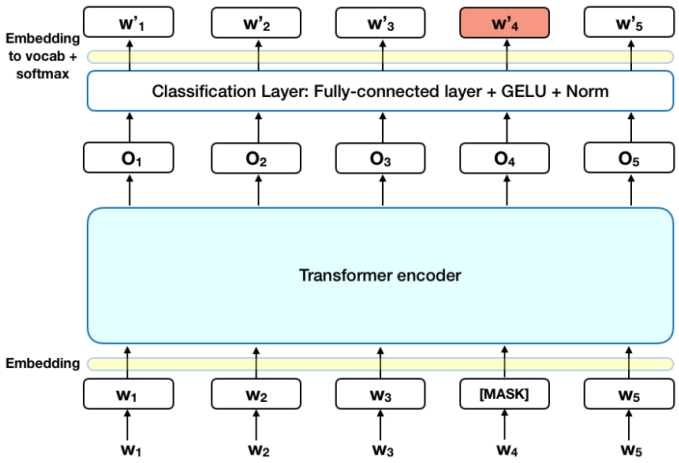

Figure 2 BERT Model Structure

2.2 Implementation and Optimization of BERT

One of the advantage of using BERT model is that it has been already pre-trained on a large BooksCorpus with 800 million words, and English Wikipedia with 2500 million words. The author thus leverage the power of transfer learning, and directly import the bert-large-cased model which contains 24 layer and 336 million parameters [4]. The author implemented the BERT model in his final submission with Tensorflow and Keras in the following steps:

1. Use pre-trained bert-large-cased model to tokenize the preprocessed Amazon review text, where the prediction label is the overall score in the training set.
2. Split training data randomly in training set(90%) and test set(10%).
3. Import the TFBertForSequenceClassification model from the Transformer library, load the pretrained bert-large-cased for training.
4. Add a customized Dense output layer after the imported BERT model for the regression predictive task in this Kaggle competition. The overall model summary is shown in Figure 3.

```
Model: "tf_bert_for_sequence_classification"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bert (TFBertMainLayer)       multiple                  333579264
_________________________________________________________________
dropout_73 (Dropout)         multiple                  0
_________________________________________________________________
classifier (Dense)           multiple                  1025
=================================================================
Total params: 333,580,289
Trainable params: 333,580,289
Non-trainable params: 0
```

Figure 3 Deep Learning Model Structure using BERT

5. The model is trained for 2 epochs when it has reached optimum as the MSE loss no longer decrease during the 2nd epoch. The training took 6 hours in total.
6. The trained model is used to predict on the 10 % test set (20,000) samples. Comparing the ground truth of the test set, the prediction on test set has a MSE of 0.315.
7. Import the submission test set (10,000 samples). Complete identical pre-processing, and tokenize the reviewtext in submission set. Use the trained model to generate prediction and download the final ouput csv.

2.3 Challenges Encountered

One of the biggest challenges encountered during the project is the size of the BERT model. The author initially used an NVIDIA RTX 2060 GPU on personal laptop to train the model but the model is much bigger than the 6GB of memory equipped. The author then use Google Colab to train the model, and the bert-big-cased model still exceed the GPU memory limit. The author then reduced batch size from 32 to 16 and was able fit the model within the 16GB GPU memory of Google Colab.

2.4 Previous Attempts

Since BERT model has state of art performance in NLP tasks, the author started his first attempt using BERT model and got 0.38 MSE as an initial result. The initial attempt directly import a smaller bert-base-uncased model without any fine tuning, and used the original classification final output. The initial attempt output overall rating as classification task with 5 categories of 1 ,2 ,3 ,4 ,5 instead of a regression task outputting float value from 1 to 5. The BERT model delivered satisfactory performance, and the author didn't use any other NLP model or treated features as numerical values and category ID.

3. Results and Conclusion

3.1 Performance and Fine Tunning

Although classification method generated a satisfactory result of 0.38 MSE, the author added a regression layer to make the prediction more precise by outputting floating point value. The performance improved to 0.31 MSE after the author completed the following fine tuning:

1. Added the regression output layer to the model. Use MSE as loss function, Adam as optimizer.
2. Tune the learning rate of Adam optimizer to a smaller value of 2e-5, to learn in a smaller steps.
3. Added column of category, reviewerID and itemID to the training set words. In fact, the reviewID and itemID are not natural languages and are most likely not within the 336 million parameters of BERT. If those IDs are not tokenized by BERT, they will not increase the performance.
4. For prediction output larger than 5, capping the value to equal to 5.

3.2 Result

The final results of this model sachieved a MSE score of 0.30875 on public test set and 0.32887 on the private test set. Both results are ranked No.5 in this CSC 2515 Kaggle competition. This performance is achieved by leveraging the BERT model, which use bi-directional transformer to learn context before and after the token. Its capability of understanding various text helped the model to understand the semantic of reviewer's review text. As an attention based Transformer model, BERT model often outperforms earlier NLP models such as representation models including bag-of-words, Word2Vec, GloVe, and earlier deep learning models processing sequence data including RNN, LSTM and GRU.

There are several ways to potentially further improve the performance. Further preprocessing such as fully integrating the reviewerID and itemID into the BERT model could improve the prediction. Early stopping can be added into the training to prevent over-fitting. Once we can make sure there is no over-fitting, we can also skip the 10% test set split, and use that 10% test set data for training, which will provide more information.

References

[1] Data Design Group, Inc., "Convert JSON to CSV," 2020. [Online]. Available: http://convertcsv.com/json-to-csv.htm. [Accessed 1 December 2020].

[2] M.-W. C. K. L. K. T. Jacob Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of The North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, Minneapolis, Minnesota, 2018.

[3] R. Horev, "BERT Explained: State of the art language model for NLP," 10 November 2018. [Online]. Available: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270. [Accessed 3 December 2020].

[4] Huggingface, "Transformers-Pretrained models," 2020. [Online]. Available: https://huggingface.co/transformers/pretrained_models.html. [Accessed 5 December 2020].