Amazon Music Review prediction using BERT$_{base}$

Yingying Zhou University of Toronto Faculty of Information yingying.zhou@mail.utoronto.ca

1. Data Preprocessing, Exploratory Data Analysis & Feature Selection

We build a recommender system model to make predictions on Amazon music review ratings. The dataset ('train.csv') contains 150,000 observations about Amazon music product reviews from 1998 to 2018. The target variable is overall *rating* scores by reviewers. Amongst 10 features, there are 2 commentary textual variables *reviewText*, and *summary* containing sentiment information submitted by reviewers. Product related variables are *category, price* and *itemID*. The remaining features are review logs information including *overall, reviewTime, reviewerID* etc.

EDA

Figure 1 depicts that overall rating of 5.0 and 4.0 accounts for the majority of all reviews. Pop and Alternative Rock seem to be the more popular genres with the most purchase records and high ratings. Figure 2 shows that the average rating over the years does not evolve over time, which is flat around 4.5/5.0 since 1998, thus the feature *reviewTime* is excluded from our model.

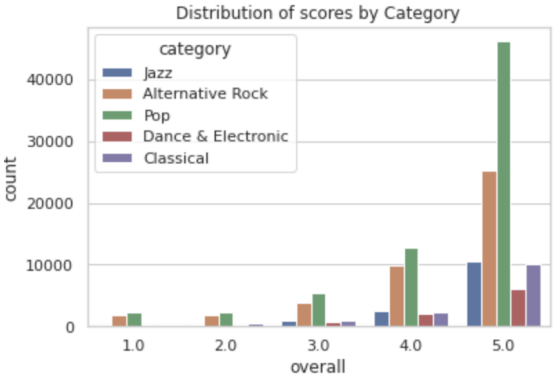

Figure 1. Distribution of scores by Category

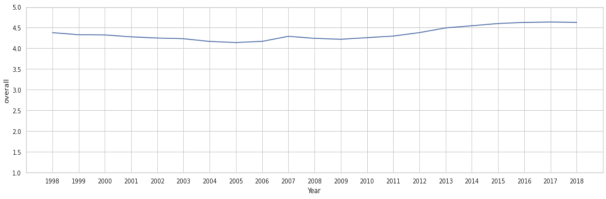

Figure 2. Rating change since 1998

Figure 3 and 4 are word clouds for *reviewText* and *summary*. Most of the frequently mentioned words have positive sentiments such as *great, love, best* etc.

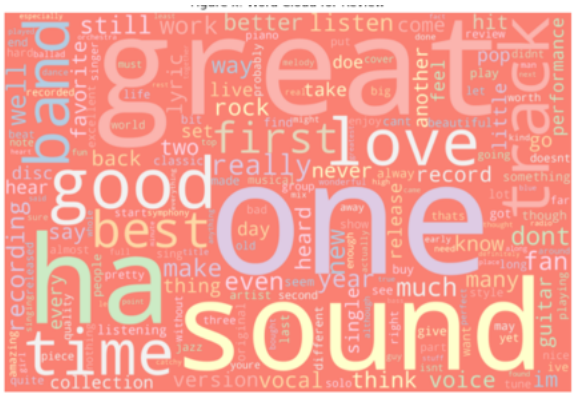

Figure 3. Word Cloud for ReviewText

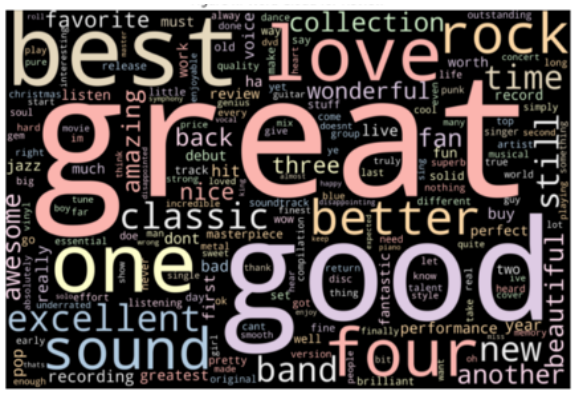

Figure 4. Word Cloud for Summary

More specifically, the results from top 20 most frequent keywords (Figure 5, 6) are consistent with the word cloud where most of the words are positive with more informative sentiment context to investigate further in our model.

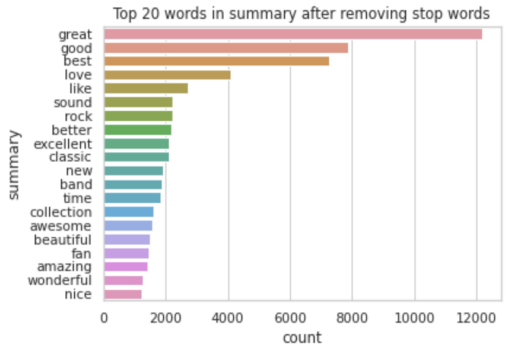

Fig 5. Top 20 words in Summary

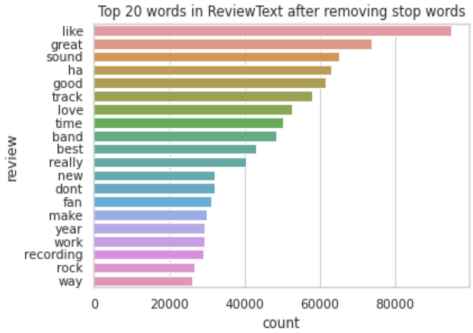

Fig 6. Top 20 words in Review

category	overall
Alternative Rock	4.292575
Classical	4.522408
Dance & Electronic	4.413300
Jazz	4.546700
Pop	4.428281

Fig 7. Average rating by category

Feature selection

Empirically speaking, *category, price, reviewText*, and *summary* might be of interest to our research purpose. However, the product price is not correlated with music rating. There is a negligible correlation between *price* and the target variable *overall* is (0.02). In addition, the average *overall* scores by category are close to each other (Fig 7). Thereby, *price* and *category* are excluded from our model. The general public has no preference over one particular music category, nor do they judge the album by its price tag.

After deselecting irrelevant variables, we are left with the two textual features in the model, *reviewText* and *summary*. Data cleaning is conducted before model construction. Rows with missing values in the *summary* column are removed in the training set. We performed text preprocessing on *reviewText* and *summary* where texts are lowercased, concatenated, lemmatized and url, punctuation and stop words are removed. We then split the training and validation datasets in a proportion of 80:20.

2. Model description

Consequently, the research question boils down to music ratings prediction based on text classification. To incorporate textual data, (a.k.a natural language) into a machine learning classifier, texts need to be processed into a format valid for model input. We started off with the TFIDF[1] (Term Frequency-Inverse Document Frequency) model with max feature of 1000, which was designed to be a baseline model. The model takes text data and outputs a term-weighted matrix, which is subsequently fed into a fully connected neural network as an input feature set. During training, the Multi-layer Perceptron[2] (MLP) model was fine-tuned to have three hidden layers with 10 neurons in each layer, ReLU activation function, Adam solver and 10 epochs. The best model performance of MSE was 0.72, which beat the Kaggle[3] weak baseline and approximated the strong baseline. We tried more hidden layers and more epochs, but the MSE for validation set suffered from overfitting. Besides MLP, we have attempted other machine learning classifiers such as Random Forest and SVM. But their model performances are pretty close to each other (MSE≈1.05), significantly underperforming the MLP model.

Notice that although TFIDF featurization is a powerful measure of word relevance for document topics, it does not take semantic meanings into account, rendering it to be one of the model limitations. In addition, a further exam reveals that there were quite a few zeroes in the feature matrix. One plausible cause to the sparsity problem is that the model training was confined by the corpus size, which is built on the text data we fed to the TFIDF vectorizer. Then, we intend to use a higher-level pre-trained NLP model BERT[4] for text classification.

BERT (Bidirectional Encoder Representations from Transformers) is a transfer learning language model utilizing the Transformer encoder, whose input is a sequence of tokenized text to be embedded into vectors and processed in the neural network, and outputs encoded vectors corresponding to input token.

The 12 layers, 12 attention heads, and 110 million parameters of the base BERT model are sufficient for our purpose. An untrained classification layer will be added to the transformer output to realize text classification. Subsequently, the new integrated model will be trained. In order to simplify the steps on code implementation, we intend to use huggingface Trainer API in Pytorch.

During the training process, hyper-parameters (batch size = 32, epoch = 2, etc.) were fine-tuned for optimal performance. We initially set epoch = 3, but the result shows overfitting. The training loss had already converged before the training ended. Besides, the validation loss was larger than the training loss (0.526 vs. 0.479). After changing the epoch to 2, the evaluation loss dropped to 0.525. There is a problem that the buffered logging records would stop when it reached the maximum output size. It is mainly due to the storage capacity on Google Colab. Therefore, we fixed this problem by changing logging_steps and save_steps.

In the BertForSequenceClasification model setup process, the number of labels was initially set to 5 in order to match the target rating classes. The MSE of the classifier is 0.66 which is almost beating the strong baseline. Furthermore, when changing the number of labels to 1 so that the task became rating prediction with a final regression layer, the model performance boosted to MSE as low as 0.535 on the test set. This alternative parameter choice can be argued that the 5-level rating classification can be seen as a regression problem on discrete ratings (number of stars) prediction.

Because of its well-trained attention model weights, the BERT model can produce satisfactory accuracy. It also took a lot less time to train the fine-tuned model than it would have if we were to develop a deep learning model from scratch. On the other hand, the BERT model has a weakness in that its processing time is heavily dependent on computing resources, necessitating the use of a powerful GPU if faster runtime is desired.

3. Results & conclusions

Model performance comparison

The best performance of the baseline model, TFIDF, was MSE of 0.72 on the validation set; in contrast, our pre-trained BERT base model delivered a more accurate rating prediction at MSE = 0.535 that sufficiently beat the strong baseline of 0.69897.

Significance of results

The significance of the final result is that with only one variable, music review summary in plain text, the BERT text classifier model is able to predict customer ratings accurately. The parsimony nature of the model and ease of implementation would make the result reproducible, and the model deployable to commercial use for rating prediction tasks.

Feature Representations

Initially, consideration was given for the combination of both summary and review columns for feature representation using TFIDF. But due to its inferior MSE performance when the verbose review texts are in the model, only summary data was utilized in the TFIDF model. *Summary* has

been cleaned and converted into a matrix using TFIDF vectorizer which gives better results compared to a combination of reviews and summary.
In the BERT model, since the maximum sentence length is 512 tokens, fell short of the longest the *reviewText* or combination of *summary* and *reviewText*, we decided to use summary only as the feature input. Although the transformer outputs (final hidden states) can be concatenated to get sentence encoded representation and then fed into other machine learning models as feature inputs, we see no need to experiment further.

Interpretation of model parameters
TFIDF vectorizer was defined on parameters like maximum dimension of the output features, document frequency and choice of stop words, to name a few. Maximum feature for TF-IDF vectorizer is set to 1000, with other parameters being default settings. For the classification layer using MLP neural network, our baseline model is finalized with 3 hidden layers and 10 neurons each after hyperparameter tuning. ReLU activation is used to avoid vanishing gradient problems. Optimizer Adam is chosen since it converges faster compared to other alternatives.
The base BERT model has 110M parameters in the neural network, which is technically uninterpretable. While the added classification layer has a new parameter (num_labels), standing for the number of classifier labels except when the value = 1, it is for regression. Number of labels of 1 was chosen in order to predict music ratings, in other words, the number of stars customers will give to the music.

Reasons for the proposed model success and others failure
BERT delivers a satisfactory accuracy performance thanks to its extensively pretrained attention model weights[5] which encode far more information content than the simplistic Bag-of-Words (TF-IDF) or Word2Vec word embedding algorithms. BERT accounts for context-dependent lexical relationships for each occurrence of a word. Thus, the added classification layer was able to extract the informative input and produce precise rating predictions with some fine-tuning.

Business Insight and next steps
Our deployed model builds a recommender system to help customers and Amazon customer service to better understand customer ratings and feedback related to music reviews. Specifically, given the information of the user, item and associated review data, the model is able to predict the review's rating star accurately. Furthermore, based on the numeric difference between predicted rating and the actual rating values, a filter could be built to determine how useful a rating record is. It saves effort for online shoppers who spend time manually browsing all the feedback records and also increases the efficiency for Amazon client relationship management.

In the future, Amazon can develop a UI Tab on review pages called Music Intelligent Recommender that groups reviews by frequent keywords from review summary for each music album and displays the distribution of rating next to the selected keyword. Based on the keywords the customers are interested in, the system can recommend some related music sharing similar keywords in the reviews to help enhance the customer shopping experience.

Appendix References:

1. TFIDF: K. Sparck Jones. "A statistical interpretation of term specificity and its application in retrieval". Journal of Documentation, 28 (1). 1972.
2. MLP: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/
3. Kaggle: https://www.kaggle.com/c/mie1624winter2021/leaderboard
4. BERT: Devlin, Chang, Lee, & Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. arXiv.org, 2019.
5. BERT Attention weights: “Deconstructing BERT: Visualizing the Inner Workings of Attention”. Towards Data Science. https://towardsdatascience.com/deconstructing-bert-part-2-visualizing-the-inner-workings-of-attention-60a16d86b5c1
6. “The Illustrated BERT”. http://jalammar.github.io/illustrated-bert/