CSC2515 Final Project Part1

Xinqi Shen, Kaggle Team ID: 5890857

Data Processing

Features that I use in my final model are **summary** and **category**, **overall** is treated as the rating label. I also use reviewerID, itemID and price as features in other models that I tried. To process the data, I first remove all the reviews that contain empty feature values or incorrect values. Then, I combine summary and category into a single feature by concatenating them together because they are both texts. Since I use the Bert model, I tokenize the sentences and break them up into words and subwords in the format BERT is comfortable with. After that, lists of tokens are padded to the same size and these paddings are masked. Before feeding into the model, I randomly split the data into training and validation set with a ratio of 90:10 due to the large dataset.

Model Description

I use the fine-tuning Bert model as my prediction model. For the transfer learning purpose, I use the pre-trained Bert model('bert-base-uncased') in order to get better performance and less training time. Since our task is to predict review ratings, I add a single linear layer(regression layer) on top of the normal BERT model to get the predictions. Due to the performance measured with MSE, I also use MSE as my loss function. The Bert model is one of the most powerful language representation models nowadays for natural language processing tasks. Our task is to predict ratings, the review summary is the most relevant feature first comes to my mind, and along with category as text-feature can be used in the Bert model. To optimize the result, I tune different hyperparameters many times and get the best setting with batch size(32), learning rate(2e-5) and the number of epochs(2). Noticed that, overfitting will occur if the number of epochs is greater than 2, due to the use of the pre-trained model. I will discuss other models that I used for comparisons.

0.1 Collaborative Filtering(CF)

Since we need to build a recommender system model, we can use the collaborative filtering algorithms, there are two types – memory-based and model-based. In terms of memory-based algorithm, it can be user-based or item-based, the two approaches are mathematically quite similar by first finding similarity and then calculating ratings. I use the KNN-based algorithm here. In terms of model-based algorithm, it involves a step to reduce the user-item matrix, I use matrix factorization, SVD here. The input features for the CF algorithm are reviewerID and itemID. CF algorithm takes advantage of the user and item relation, however, there are many reviewers and items in our dataset, it can not retrieve much information to accurately predict ratings.

0.2 Word Embedding + One hot encoding category + Price(WOP)

I want to use as many features as I can to increase the predictive power. Thus, I consider using summary, category and price. In order to build the feature map, it needs a different data processing step. For summary text, I convert it to word embedding by feeding the summary text into the pre-trained Bert model and slice the last hidden layer that corresponds to the first token of each sentence. Since the way Bert does sentence classification, is that it adds a token called [CLS] (for classification) at the beginning of every sentence. The output corresponding to that token can be thought of as an embedding for the entire sentence. Then, I use the one-hot encoding for category and convert the price feature to floating numbers. After that, I concatenating these three features together into a feature map. I use Random Forest, Gradient Boosting and MLP model to compare the performance. The performance(MSE) for these three models are similar. There is an unsuccessful attempt here, I try to encode the userID and itemID to word embedding and include them in the feature map, but it makes the result worse.

0.3 Ensemble

I implement two different bagging ensemble methods. For the first approach, I train the Bert model four times based on the different training sets that are uniformly sampled from the original dataset with replacement. Then, I get results based on the average of predicted ratings. For the second approach, I get results based on the average of predicted ratings from four different models: Bert, Random Forest, Gradient Boosting and SVD-based Collaborative Filtering.

Strengths and Weaknesses of the different models

- **Bert:** pre-trained model is powerful, but cannot embedding ID-based or numeric features
- **Collaborative Filtering:** utilize the user-item relationship, but the performance is limited by number of user/item
- **Random Forest/Gradient Boosting:** need extra feature engineering,
- **Multilayer Perceptron(MLP):** sensitive to scale of different features

Results and Conclusions

model	Bert	CF+KNN	CF+SVD	WOP+Random Forest
MSE in Test Set	0.53167	1.02626	0.82503	0.66314
model	WOP+Gradient Boosting	WOP+MLP	Bagging+Bert	Bagging+(Bert+RF+GB+SVD)
MSE in Test Set	0.66576	0.66800	0.51130	0.55168

The Bert model performed best among all the models, it gets much lower MSE than others. However, since different models may use different features, we cannot make a horizontal comparison directly. If using a collaborative filtering algorithm, the matrix factorization(SVD) algorithm outperforms the user/item-based KNN algorithm, improved almost 20%. In terms of using WOP features, the results for Random Forest, Gradient Boosting and MLP are quite similar, the MSE hover around 0.66. The ensemble method further improves the performance of any single model, the best result comes from the Bert bagging ensemble, reaches the 0.51130 MSE. According to the result, the summary feature is the most important feature in this prediction task, however, convert the summary text to word embedding by Bert model and combine with other features perform worse than directly using the Bert model. This word embedding may not be appropriate for training when using other models, we may find another way to get word embedding, or we should also embed category and price before concatenating with word embedding. Since the Bert model is the neural network model, the model's parameters are just weight, so we are unable to make any interpretation. The Bert model is the most powerful model in the NLP area and we make predictions based on text, it provides amazing performance. Due to the time limit and for future improvement, we can find a way to combine the NLP approach with collaborative filtering, it may reach a better result.

Reference

Bert.ipynb contains the code for fine-tuning the Bert model inspired by
https://mccormickml.com/2019/07/22/BERT-fine-tuning/#cite
get_word_embedding.ipynb contains the code for getting word embedding inspired by
https://github.com/jalammar/jalammar.github.io/blob/master/notebooks/bert/A_Visual_Notebook_to_Using_BERT_for_the_First_Time.ipynb