Help - Yelp Midterm Report

Team Name: EMANON **Kaggle Name:** EMANON

Jiang, Song	Wang, Yewen	Xiao, Zhiping	Xu, Pengcheng
605222804	905229899	604775684	704872496

The project focuses on predicting the score rated by a given user on a given business, with prerequisite knowledge of some known scores rated by certain users on certain businesses.

To be specific, we focus solely on the Yelp dataset, where users rate the businesses by stars, with discrete distribution on finite set $\{1, 2, 3, 4, 5\}$. Users and businesses are denoted by their unique hash IDs. Knowing some existing ratings of certain users on certain businesses, we're going to predict the corresponding scores regarding N pairs of user IDs and business IDs.

The formal definition of the problem is as follows:

Definition 1.1 (*Learning Algorithm*) Assume that we have a set of n users U, and a business set B of size m, and for some user $u_i \in U$ ($i \in \{1, 2, \dots p\}$) and some business $b_j \in B$ ($j \in \{1, 2, \dots q\}$), we have a record of score $s(u_i, b_j)$ associated with u_i and b_j. We assume that there exists a mapping function h^* such that $h^*(u_i, b_j) = s(u_i, b_j)$. All the recorded scores $s(u_i, b_j)$ form a score set $S(U, B) = \cup_{i=1,j=1}^{p,q}\{s(u_i, b_j)\}$ Given training data set $\mathcal{T} = \{((U, B), S(U, B))\}$, hypothesis class $\mathcal{H} = \{h : (U, B) \to S(U, B)\}$, and loss function L defined by RMSE [1], learn:

$$h^* = \arg\min_{h \in \mathcal{H}} L_{\mathcal{T}}(h) \tag{1}$$

Using the same annotations, we formally define the prediction task as the follows:

Definition 1.2 (*Prediction Task*) Given N unseen pairs of users and businesses, denoted by $\mathcal{T}_{test} = \{U_{test}, B_{test}\}$, which means that although $U_{test} \subseteq U$ and $B_{test} \subseteq B$, $\forall (u_{test_i}, b_{test_i}) \in \mathcal{T}_{test}$ hasn't appeared together in any sample point from the training set $\mathcal{T}$. Output the series of predictions labeled by their index i in the test set (starting from 0), predictions $= \{(i, h^*(u_{test_i}, b_{test_i}))\}$.

[1] For the details of RMSE, see section 2.1.

1 Proposed Method

1.1 Sub-Tasks Formalization

Generally, this problem could be broke into 5 sub-tasks as formalized follows: *data preprocessing, feature engineering, model development, evaluation, and prediction.*

In data preprocessing stage, we'll analyze the provided data set, and transform the raw data set into an understandable prepared data set by conducting data cleansing and data editing.

In feature engineering stage, we'll remove the less related, or redundant features, combine and transform some existing features to better fit the models.

In model development stage, we'll first brainstorm all the possible applicable methods and analyze their pros and cons; second, we'll conduct experiment on those feasible method and train the corresponding models; last, we'll assign weight to each model according to their performance and apply ensemble learning.

In evaluation stage, we'll apply our selected model on test data set to evaluate its performance.

After finishing all those sub-tasks, we could proceed to make predictions on the test data set.

1.2 Data Preparation

In this subsection, we'll give insight of the provided data set and discuss how to process them to meet the needs in each sub-task.

The provided useful data set includes a labeled training set of size $150k \times 9$, a labeled validation set of size $50.1k \times 4$, an unlabeled test set of size $50.1k \times 2$, and 2 features set of size $12.1k \times 61$ and $41.7k \times 22$ including features of businesses and users respectively.

For sub-task feature engineering, we use the 2 features set *business.csv* and *users.csv*. Here we first transform all the attributes to numerical attributes, then regularize those attributes, and conduct preliminary feature selection with domain-specific knowledge to achieve some features that have the potentioal to better represent the underlying problem to the predictive models.

For sub-task model development, we use the training set *train_reviews.csv*. Here, since we can only use shared attributes of training set, validation set and test set, so we only need to use column with title "business_id", "user_id", and "stars" in the training set. Thus, we delete the other columns in this data set.

For sub-task evaluation, we use the validation set *validate_queries.csv*. Due to the same reason mentioned in last paragraph, here we delete the first column in this data set.

For sub-task prediction, we apply the test set *test_queries.csv* directly without any processing.

1.3 Algorithms for Sub-Tasks

As is mentioned above in section 1.1, we divide the task into four sub-tasks, namely *data preprocessing*, *feature engineering model development* and *model evaluation*. The *data preprocessing* task is well-explained in section 1.2.

The feature engineering task aims at coming up with a set of original features and derived features, by diving into the latent relevance in behind the data attributes. Beside domain-specific knowledge, there are specific methods to help us select the features automatically. A mathematical way of doing feature engineering is *Analysis of Correlation*, such as using Pearson, Spearman, Kendall, or Gamma statistics. We could also learn it computationally, for example, using the famous XGBoost [1], which is directly inherited from the classical gradient boosted decision trees (GBDT) model [2], but significantly improved in both efficiency and accuracy.

Collaborative filtering, no matter item-based [9], user-based [13], or combined [10] models, are widely used in recommendation systems. There are various models based on the standard collaborative filtering model, which has a long-lasting and wide-spread influence.

Coming together with the "gold rush" in deep learning in the past few years, recommendation systems using neural networks have been a hot topic, as is discussed in [12]. Among them, RNN[2] models are widely known as effective ways of integrating Neural Networks into recommendation systems [4].

Generally speaking, we are planning to use an ensemble learning model that consists of multiple basic models as mentioned above. There are multiple ways of combining the components, including simply do averaging of all results, applying weight to each of them, or even try introducing attention mechanism if time permitted.

For details on *model evaluation* task, see section 2.

[2]Recently, most RNN models use LSTM.

2 Experiment and Evaluation

2.1 Evaluation Metric

Root Mean Squared Error (RMSE) is used to evaluate model performance as required. It is defined as

$$RMSE = \sqrt{\sum_{i=1}^{N}(\hat{S}_i - S_i)^2/N} \tag{2}$$

where N is the example number of test set, S is the set of predicted scores and $\hat{S}$ is the set of actual scores.

2.2 Model Selection

XGBoost

Ensemble method [3] is a paradigm of supervised learning that trains and combines a set of base (weak) learners to obtain better predictive performance. Recently, this appealing method has played a role in many fields such as computer security, remote sensing and emotion recognition.

Among commonly-used ensemble methods (Bayes optimal classifier, bootstrap aggregating (bagging), stacking), boosting does the training sequentially by correcting errors made by early-step learner in later-step one. Gradient boosting [6] is a subset of boosting which identifies pseudo-residuals (usually negative gradient of loss function) as the "shortcoming" to be improved (AdaBoost could be seen as a subset of gradient boosting use exponential loss function), and it usually uses decision tree (regression tree) as base learners because of its effect of feature selection, interpretability and scalability[2].

XGBoost[1], developed by Tianqi Chen and coworkers, is an effective and popular implementation of gradient boosting decision tree (GBDT). (Actually, base learner could be some linear classifiers.) XGBoost's loss function (with regularization terms) takes into account second derivatives in addition to first derivative, which improves accuracy and ability to predict. Other advantages include efficient finding of best split for decision trees, parallel processing of feature vectors, economical use of drive space. It is the choice of many machine learning competitions (e.g., Kaggle) and has been integrated with other packages for the ease to use.

Recursive neural network (RNN)

RNN ,which can describe time behavior, is one of the artificial neural network. Because RNN cyclically transfers the state in its network, it can accept the more inputs of extensive time series[11].

Simple RNN cannot capture long-term relation, so we usually use modified RNN - Long Short-Term Memory (LSTM), which can solve this problem well, to tackle

problems. LSTM can precisely express or simulate human's behavior, logical development, and recognition of neural networks[11].

Although LSTM has several advantages, it can only memorize long-term memory for about 100 s instead of 1,000 s or 10,000 s. Furthermore, RNN needs large computation resources, which make higher costs. If series treatment is unavoidable, we will need the computing unit that can predict forward and review backward to solve the problem[11].

Collaborative filtering (CF)

Collaborative filtering is a widely-used technique used by recommendation systems. Unlike context-based recommendation that considering the properties of items only, it utilizes an information filtering process taking into account of collaboration among users.

Based on the ideas that similar users share same interests and that alike items are liked by a user, this personalized technique mainly includes user-based, item-based and mixed methodologies[9][10][13]. If a user's preference on items has strong correlation with her taste and personality, user-based method would work appropriately (e.g., artworks, books); if internal relations among items obviously influence users' shopping decisions, item-based model might be suitable (e.g., online-shopping). (Another approach is model-based [8]: data mining algorithms are applied to train models for predicting users' rating on items.)

There are some challenges confronted by collaborative filtering such as data sparsity, synonyms and shilling attacks. Correspondingly, scholar are continuously making innovations to address these problems. For example, associative retrieval techniques [5] could mitigate sparsity, and the addition of probabilistic latent semantic analysis [7] could make the recommendation algorithm less vulnerable to attacks.

2.3 Current Performance

Now our training error is 1.14855, validation error is 1.261453 and online test result is 1.26192. We can observe that the model has a not bad generalization property but still needs to improve accuracy.

3 Discussion

In this paper, we tackle the Yelp Review Rating Prediction problem. We treat it as a regression problem, and examine various feature extraction and XGBoost as supervised machine learning method to construct the prediction system.

We find that "business_star" and "user_star", which means the average star rating of a business and a user separately, have the most importance on the prediction result.

In the next work, we will improve the model in following ways. 1)Feature Engineering. Now the feature engineering is very rough, so we can extract more useful features in detail. For example, features related to business category. 2)Model construction. Now we only use XGBoost as our main model, which is insufficient. To fix this issue. We will try more models such as linear model, Bayes based model and SVM to better model the data and reduce the inductive bias of single model.

4 Schedule

Due date	Milestones
11/9	Model version 1.0 implementation
11/10	Model version 1.0 evaluation
11/10	Improving model design
11/11	Model version 2.0 implementation
11/12	Submit Model 2.0
11/19	Evaluation Model 2.0
11/26	Further improvements
12/3	Implementation of the final model
12/9	Submit final version model

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[2] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[3] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

[4] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[5] Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filter-

ing. *ACM Transactions on Information Systems (TOIS)*, 22(1):116–142, 2004.

[6] Ping Li, Qiang Wu, and Christopher J Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2008.

[7] Bamshad Mobasher, Robin Burke, and Jeff J Sandvig. Model-based collaborative filtering as a defense against profile injection attacks. In *AAAI*, volume 6, page 1388, 2006.

[8] David M Pennock, Eric Horvitz, Steve Lawrence, and C Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 473–480. Morgan Kaufmann Publishers Inc., 2000.

[9] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[10] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

[11] Wikipedia contributors. Rnn — Wikipedia, the free encyclopedia, 2018. [Online; accessed 3-Nov-2018].

[12] Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*, 2017.

[13] Zhi-Dan Zhao and Ming-Sheng Shang. User-based collaborative-filtering recommendation algorithms on hadoop. In *Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on*, pages 478–481. IEEE, 2010.