Combination of Diverse Ranking Models for Personalized Expedia Hotel Searches

Article · November 2013

Source: arXiv

8 authors, including:

Bin Wang

Chinese Academy of Sciences

55 PUBLICATIONS **381** CITATIONS

SEE PROFILE

Combination of Diverse Ranking Models for Personalized Expedia Hotel Searches

Xudong Liu*,[1], Bing Xu*,[2], Yuyu Zhang*,[3], Qiang Yan*,[4], Liang Pang*,[3], Qiang Li[3], Hanxiao Sun[3], and Bin Wang†,[3]

[1]Institute of Automation, Chinese Academy of Sciences
[2]University of Alberta
[3]Institute of Computing Technology, Chinese Academy of Sciences
[4]Taobao Inc.
[1,2,3]{lxdwinwin, antinucleon, zhangyuyu2008, pl8787}@gmail.com
[4]yanqiang.yq@taobao.com

***Abstract*— The ICDM Challenge 2013 is to apply machine learning to the problem of hotel ranking, aiming to maximize purchases according to given hotel characteristics, location attractiveness of hotels, users aggregated purchase history and competitive online travel agency (OTA) information for each potential hotel choice. This paper describes the solution of team "binghsu & MLRush & BrickMover". We conduct simple feature engineering work and train different models by each individual team member. Afterwards, we use listwise ensemble method to combine each model's output. Besides describing effective model and features, we will discuss about the lessons we learned while using deep learning in this competition.**

I. Introduction

ICDM Challenge 2013 requires learning to rank hotels to maximize purchases for given hotel queries by Expedia.com. The dataset which is provided by Expedia.com, contains hotel characteristics, location attractiveness of hotels, users aggregate purchase history and competitive OTA information for each search_id-hotel pair. Hotels for each user query are assigned relevance grades as following: 5 for user purchased a room; 1 for user clicked the information of the hotel and 0 for user neither click nor book. The data is split by organizers by randomly split.

The training data contains 399,344 unique search lists and 9,917,530 points. The test data contains 266,230 search lists and 6,622,629 points. 25% of the test data is used for evaluating in public leaderboard and the remaining 75% is used as final private test data.[1]

The evaluation metric for this competition is Normalized Discounted Cumulative Gain (NDCG), which is commonly used in ranking[1]. According to the announced result, our approach achieved 5^{th} on the private leaderboard with 0.53102 NDCG@38 score.

The paper is organized as follows. Section 2 outlines the framework of our approaches. Section 3 discusses the preprocessing and feature engineering. Section 4 introduces the single effective models we use. Section 5 describes the ensemble experiments we use to boost our performance. Finally we conclude this paper and further discuss lessons we learned in Section 6.

II. Framework

This section introduces the architecture and softwares we used in our system. Then it discusses the self-split internal validation set from the training set, which is important in model evaluating and combining different models.

A. System Overview

Our system can be divided into three parts: data infrastructure, training individual models and ensemble as shown in Figure 1. The data infrastructure is based on pandas[4]. In this step we use pandas to store data. And we do some feature engineering in this stage. The output of the data infrastructure can be in different format like numpy binary array, SVMRank[5] text format and LIBSVM[6] text format. In the second stage, we explore diverse approaches to generate various models, including logistic regression, support vector machine, random forest, gradient boosting machine, factorization machine, LambdaMART and deep neural network. In the last step, we combine all different results on the internal validation set and test set by using listwise approach, linear approach and deep neural network approach.

We use LIBLINEAR[7] and SVMRank[5]for pairwise logistic regression; Random Forest[8] from scikit-learn[2]; Ranking algorithms like AdaRank[9], LambdaMART[10] from RankLib[2]. And we also use Gradient Boosting Machine[3][12], Extremely Randomized Trees[4][13] from R, deep neural net-

* These authors equally contributed to this work.
† Team advisor
[1]Complete leaderboard can be found at http://www.kaggle.com/c/expedia-personalized-sort/leaderboard

[2]http://sourceforge.net/p/lemur/wiki/RankLib/
[3]http://cran.r-project.org/web/packages/gbm/index.html
[4]http://cran.r-project.org/web/packages/extraTrees/index.html

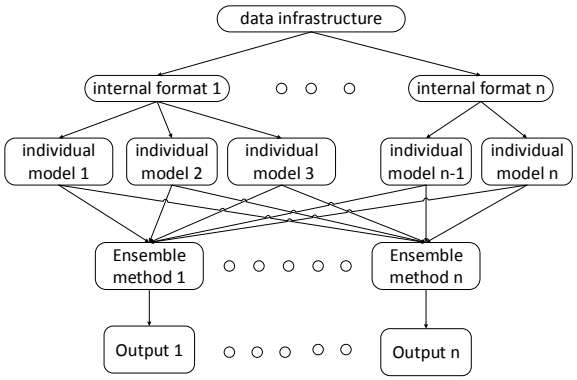

Fig. 1. The overall architecture of our approach.

work implementation from PyLearn2[3] and Factorization Machine libFM[5][15].

B. Internal Validation Set

A validation set can be used to evaluate single model without submitting the test result to the leaderboard. And validation set is very important for combining different models. Usually, the training set can be divided in the ratio 6:2:2. But in this dataset the validation result is quite robust so we just keep 10% data as the validation. We use the rule $srch\_id\%10 == 1$ to generate validation set and others to be sub training set.

III. Preprocessing

A. Listwise Feature for Point/Pairwise Model

Some of the feature's rank in the list is used as a single feature for each hotel choice. Here are the most important listwise ranking features.

- price_usd
- prop_starrating
- prop_location_score2

Other listwise ranking features we proposed but had no time to evaluate including: rank of exp(prop_log_historical_price)-price_usd, rank of click/booking bias and so on.

Listwise features is a bridge to bring listwise information to point/pair wise models.

B. Composite Features

Composite features is a method that combine two different features. For example, now we combine $srch\_room\_count$ and $srch\_booking\_window$, the $count\_window$ feature equals $srch\_room\_count * \max(srch\_booking\_window) + srch\_booking\_window$.

[5]http://www.libfm.org/

C. Dealing Missing Feature Values

There are many missing feature values in the data such as prop_location_score2. We use the first quartile calculated by the country which the data point located in to represent the missing data.

D. Use 10% data

We randomly sample 10% of the data by srch_id to generate new training data. Using the new training data we can train a model with very small difference from the model trained by total training data.

E. Use Balanced Data

Balanced data is used in training random forest and deep neural network. For there are only 4.4% positive data points among the 9.9 million data points, we choose one positive example and randomly choose one negative example. In this way we can train tree-based models with a large amount of trees in a reasonable time.

F. Split Data by prop_country_id

Based on prop_country_id we split the data into 172 pieces and train independent models on each piece. This method greatly reduces time on training tree based models.

G. Use Bucket to Binarize Float Feature

Bucket is a strong rounding method to binarize the float feature. The bucket algorithm can be described as Algorithm 1. By using bucket, the float features are in smaller variance.

Algorithm 1 $Bucket(feature, bucket\_number)$

Require: An integer $BUCKET > 0$
  $description$ = {}
  $binary\_feature$ = zeros($(feature.size, bucket\_number)$)

  for $i = 1$ to $bucket\_number$ **do**
    $description[i] = feature.quantile(i/bucket\_number)$
  end for
  for $i = 0$ to $feature.size$ **do**
    $j = 1$
    while $feature.at(i) < description[j]$ **do**
      $j = j + 1$
    end while
    $binary\_feature[i][j] = 1$
  end for
  return $binary\_feature$

IV. Models

A. Logistic Regression

As a classical model for binary classification, logistic regression is used as our initial attempt in this competition. We tried both the binary logistic regression and the multinomial logistic regression, while the former one performs obviously better. So in this part, we will only introduce our approach on binary logistic regression.

We firstly pre-process the data by merging the clicked and booked items within each query as positive instances, while all the left items are regarded as negative instances. With some feature engineering work, which will be discussed in details later, all the instances can be represented as a series of feature vectors. Now it becomes a standard binary classification problem, though the data here is very unbalanced since the negative instances are overwhelming. Therefore, we then adjust the weight in the cross-entropy error function (also known as negative log-likelihood function) to tackle the issue of data unbalance. The revised error function is shown as follows:

$$\begin{aligned} CEE &= -\sum_{n=1}^{N} \log[\mu_i^{\alpha \mathbb{I}_{(y_i=1)}} \cdot (1-\mu_i)^{\mathbb{I}_{(y_i=0)}}] \\ &= -\sum_{n=1}^{N} [\alpha y_i \log \mu_i + (1-y_i)\log(1-\mu_i)] \end{aligned} \tag{1}$$

where α is the parameter of class weight decided by input data, μ is the output of sigmoid function representing the probability to be positive instance, μ_i is for the i^{th} instance.

Since the cross-entropy error above is convex, it has a unique global minimum, we use gradient descent approach to find the optimal weight vectors or model parameters. Surprisingly, although the model of logistic regression is simple and the model itself is designed for classification rather than object ranking, the performance is fairly good, which can achieve over 0.52 in terms of NDCG on public leaderboard.

B. Pairwise Logistic Regression

In order to use full train set as pairwise in logistic regression model, we use FTRL-Proximal algorithm and liblinear[7] and SVMRank[5] build in function. The FTRL-Proximal algorithm [18], can be seen as a hybrid of FOBOS and RDA algorithms, and significantly outperforms both on a large, realworld dataset.

We use the whole training set to train this model with simply seven features. The featurelist is srch_id, prop_id, srch_destination_id, prop_starrating, prop_location_score1, prop_location_score2, price_usd. And this single FTRL model archives 0.51273 NDCG@38 on validation set.

C. Random Forest

After forming a split dataset in 172 pieces by using prop_country_id, we balanced each piece. For each unique balanced prop_country_id data piece we train an independent random forest model [8] with 3200 trees. With listwise ranking features, the 172 random forest models achieve nearly 0.51 NDCG@38 score in internal validation set. Some failed cases happened in predicting for some countries, so to make the example count of test and internal validation set equal we simply combine the score with a pairwise logistic regression trained by liblinear[7] model with 0.47 NDCG@38 score on validation, then the mixture balanced random forest model achieves nearly 0.52 NDCG@38 on validation set.

TABLE I. THE TOP 20 RELEVANCE FEATURES.

Feature	Rel.	Feature	Rel.
fm_score	50.35	random_bool	1.00
lr_score	12.99	srch_destination_id_1_cnt	0.84
prop_location_score2	10.19	date_time	0.63
ump	3.58	per_fee	0.61
price_diff	2.40	price_usd	0.54
random_bool_1_cnt	2.26	score2ma	0.50
random_bool_f	1.57	prop_location_score2_1_cnt	0.49
starrating_diff	1.39	prop_review_score	0.49
prop_id_1_cnt	1.25	total_fee	0.47
score1d2	1.06	orig_destination_distance	0.47

D. Gradient Boosting Machine

Gradient Boosting Machine(GBM) [12] is a machine learning technique for regression problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The gradient boosting method can also be used for classification problems by reducing them to regression with a suitable loss function.

We use GBM in R language with the balanced data. The most 20 relevant feature show in Table I. Fm_score and Lr_Score is the rank score that simple Factorization Machine and Linear Regression predicted using only visitor and query features. Date time is transform to unixstamp as a continuous feature. Feature name with the suffix _cnt means one-way count of the feature in the combine of train and test set. The feature ump, price_diff, starrating_diff, per_fee, score2ma, total_fee and score1d2 is generated by the formula below.

$$ump = exp(prop\_log\_historical\_price) - price\_usd$$

$$price\_diff = visitor\_hist\_adr\_usd - price\_usd$$

$$starrating\_diff = visitor\_hist\_starrating - prop\_starrating$$

$$per\_fee = \frac{price\_usd * srch\_room\_count}{srch\_adults\_count + srch\_children\_count}$$

$$score2ma = prop\_location\_score2 * srch\_query\_affinity\_score$$

$$total\_fee = price\_usd * srch\_room\_count$$

$$score1d2 = \frac{prop\_location\_score2 + 0.0001}{prop\_location\_score1 + 0.0001}$$

This single GBM model archives 0.52477 NDCG@38 on validation set.

Without one-way count features the GBM model archives 0.50099 NDCG@38 on validation set which is useful in ensemble process.

E. Extreme Randomized Trees

Extremely Randomized Trees (ERT) is proposed by [13].This method is similar to the Random Forests algorithm in the sense that it is based on selecting at each node a random subset of K features to decide on the split. Unlike in the Random Forests method, each tree is built from the complete learning sample (no bootstrap copying) and, most importantly, for each of the features (randomly selected at each interior node) a discretization threshold (cut-point) is selected at random to define a split, instead of choosing the best cut-point based on the local sample (as in Tree Bagging or in the Random Forests method).

ERT model using in learning to rank task is proposed by [14]. We also use this model with the same feature set as GBM model and archives 0.51699 NDCG@38 on validation set.

TABLE II. FEATURES FOR FACTORIZATION MACHINE.

Bin_ID Feature	Normalized Feature	Ranking Feature
prop_id	price_usd	price_rank
srch_destination_id	prop_location_score1	price_diff_rank
srch_room_count	prop_location_score2	star_rank

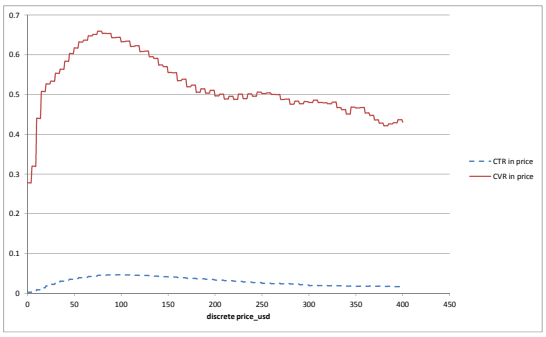

Fig. 2. The CTR and CVR in discrete price_usd scale.

TABLE III. FEATURES FOR LAMBDAMART.

Normalized Feature	Ranking Feature	Result Feature
prop_starrating	price_rank	fm_score
prop_location_score1	price_diff_rank	lr_score
prop_location_score2	star_rank	

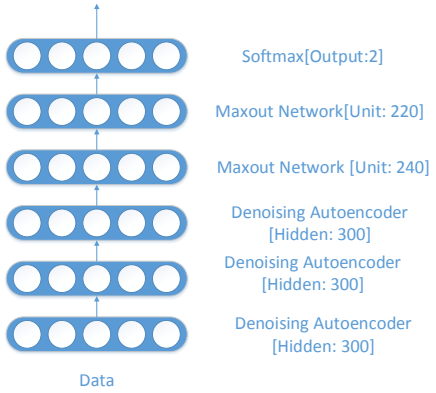

Fig. 3. The architecture of deep neural network.

F. Factorization Machine

Factorization Machine [15] is widely used in Recommender System. It is also a kind of Regression model. So it can be used as a pointwise model. We did a lot of work in feature engineering and the single model archives 0.5171 NDCG@38 on validation set. As the feature engineering and the model itself bring much diverse, it works well when ensemble. Some of the features are listed in Tab. II.

For the model of Factorization Machine, our features are built in different ways. Some of the features are normalized (e.g. pro_location_score1), others are divided into bins (e.g. srch_booking_window). The ranking feature (e.g. price_rank) means the rank value of the identical query, which works fairly well in this model. The key point of this model is that features should be used in the right way and ranking feature brings listwise information to this pointwise model.

G. LambdaMART

LambdaMART model [10] [11] with CTR(clickthrough rates Eq 2) and CVR(Booking rates Eq 3) features and original features in full train set. CTR and CVR are calculate in two scale, prop_id and discrete price_usd (Fig. 2). This model archives 0.51149 NDCG@38 on validation set.

$$CTR_i = \frac{\#(Click_i)}{\#(Presentation_i)} \tag{2}$$

$$CVR_i = \frac{\#(Booking_i)}{\#(Click_i)} \tag{3}$$

Two of the team members use LambdaMART independently. The other LambdaMART model is based on normalized features, ranking features and result features. Some features are listed in Tab. III and this single LambdaMART model archives 0.5243 NDCG@38 on validation set All these features except fm_score and lr_score are introduced in the Factorization Machine part. Fm_score and lr_score are learnt by visitor and query features. These features won't work in pairwise or listwise models as they have the same value in one query. But in this way, these features contributes its bias in pairwise or listwise models.

H. Deep Learning Approach

Deep Learning has been successfully used in many fields such as image processing and acoustic processing. It is an open question to know whether the complex network and non-convex model can improve ranking and recommendation system. In practice, we first transform selected features into binary by using Algorithm 1. To make it simpler, we choose to train independent models for each country by using same network framework in the figure. 3. We choose Maxout network[16] because it provides much more regularization than other network layer. We discover the following situations while pretraining the denoising autoencoders:

1) Reconstruction error fixed around a large number (in most of cases)
2) Reconstruction error gradually become small (especially for prop_country_id=219)

As the result shown before, composite feature is important. We want to use 3 layer denoising autoencoder to find at least 3-level and robust composite features. But for most of cases, the training data is not enough to make training process jump out of local minimum. And for the largest No.219 model, although pretraining looks fine, but while the training, the error rate on validation sucks to 4.3%, which is quite near the positive ratio of the data. The mean of parameters of the softmax layer stay in less than 10^{-4}. It means this deep neural network can not accurate predict the unbalanced data. But it achieves 0.48 NDCG@38 on local validation set, which means it learns something out. After switching to balanced data, the training process is a little smooth. The error rate reduces from 50% to near 10%. But the local NDCG@38 score only improve to near 0.49. Vertical ensemble[19] helps improve a little but still can not make deep network as a strong single model. Linear embedding with other models can improve 0.004 NDCG@38 score on the local validation set of No. 219. We will provide further discussion in later chapter.

V. ENSEMBLE MODELS

A. z-score

As mentioned, we split the test set into two parts, by query. 10% of the data served for a validation set, which helps us to choose

and combine rankers. We experimented with several linear methods to combine and the linear combining is nothing more than selecting several effective rankers and assign a set of parameters to them that the combination can achieve a better result than ever single model.

The simplest, also work best, method we try was z-score ensemble Eq 4. We added corresponding score in each ranker into the ensemble ranker in the beginning, but find it was not working well. For the rankers were derived from different models, the scope of their scores vary greatly. It is unreasonable to simply add them without normalization, so we employed z-score to normalize the rank scores.

$$Z(x) = \frac{x - \bar{x}}{\sigma(x)} = \frac{x - \bar{x}}{\sqrt{(x - \bar{x})^2/n}} \tag{4}$$

We tried two ways of z-score, global z-score and query z-score. For the reason that we just compared the scores in each query, we calculated the z-score of every each query.

In addition, instead of using greedy search, we tuned parameter manually. Lacking of a local test set to do cross validation and for fear of overfitting on local test set, we have to do it manually. Manually tuning could avoid excessive parameters, so, to some extent, it is a way of simple normalization, even not beautiful enough. Global z-score was the way we finally choose to ensemble and also the way achieve the highest NDCG score on private board.

B. GBM Ensemble

Treat each model's output on local test set as a feature of GBM model [12]. The target to learn is the click ground truth in local test set. In order to get a better result, we also include some significant features, such as prop_location_score1, prop_location_score2 and price_usd.

We use 30 models and 120 trees, and it archives 0.53573 NDCG@38 on the local test set, but only 0.53053 NDCG@38 on online test set.

C. Deep Learning

We want to check whether the deep composite of the models output can make any progress. By using models with local NDCG score of 0.505, 0.508, 0.510, 0.511, 0.512, 0.513,0.519, 0.521. With ReLU[17] network or Maxout network, our model achieves around 0.526 on private leaderboard. And dropout logistic regression achieves best result of 0.52729 NDCG@38 on private leaderboard. They are still weaker than other ensemble methods.

D. Listwise Ensemble

Previous ensemble methods don't involve the listwise information. So we are trying to use LambdaMART to ensemble all the models. By using z-score normalization, it achieves our final score: 0.53249 NDCG@38 on public leaderboard, and 0.53102 on the final private leaderboard.

VI. Conclusion and Further Discussion

A. Conclusion

In this paper, we present our approaches in ICDM Contest 2013. Due to the large size of data provided by Expedia, we use both random sampling and balanced sampling methods to construct reliable validation set. Diverse ranking models are described in this paper, including modified logistic regression, random forest, gradient boosting machine, extreme randomized trees, factorization machine, and lambdaMART. We also attempt to adopt deep learning approach, which will be further discussed in next sub-section. With these individual ranking models, we introduce our ensemble methods to combine individual models, including z-score, GBM, deep learning and listwise ensemble. The combination of models significantly improves the ranking accuracy in terms of NDCG on both public and private leaderboards.

B. Lessons Learned in Deep Learning

Based on the split data, there is a serious problem of lacking training data. And reconstruction error may not reflect the best optimization direction. We suggest in later practice it is better to check whether pretraining learns out distributed representations other than only rely on the reconstruction error. Bucket may not be a optimized solution to normalize feature input A better normalization may lead to better generalization capacity for the deep networks.

Acknowledgment

We thank to Prof. Russ Greiner from University of Alberta for his advice while writing this report. And we also thank to Prof. Hong Hu from Institute of Computing Technology for his advice on data analysis.

References

[1] Liu, Tie-Yan. *Learning to rank for information retrieval*. Foundations and Trends in Information Retrieval 3, no. 3 (2009): 225-331

[2] Pedregosa, Fabian, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al. *Scikit-learn: Machine learning in Python*. The Journal of Machine Learning Research 12 (2011): 2825-2830.

[3] Goodfellow, Ian J., David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frdric Bastien, and Yoshua Bengio. *Pylearn2: a machine learning research library*. arXiv preprint, arXiv:1308.4214, 2013.

[4] McKinney, Wes. *pandas: a Foundational Python Library for Data Analysis and Statistics.*

[5] T. Joachims. *Training Linear SVMs in Linear Time.* Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), 2006

[6] Chang, Chih-Chung, and Chih-Jen Lin. *LIBSVM: a library for support vector machines.* ACM Transactions on Intelligent Systems and Technology (TIST) 2.3 (2011): 27.

[7] Fan, Rong-En, et al. *LIBLINEAR: A library for large linear classification.* The Journal of Machine Learning Research 9 (2008): 1871-1874.

[8] L. Breiman. *Random Forests.* Machine Learning 45 (1): 532, 2001.

[9] Xu, Jun, and Hang Li. *Adarank: a boosting algorithm for information retrieval.* Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007.

[10] Burges, Chris. *From ranknet to lambdarank to lambdamart: An overview.* Learning 11 (2010): 23-581.

[11] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. *Adapting boosting for information retrieval measures.* Information Retrieval 13.3 (2010): 254-270.

[12] Friedman, Jerome H. *Greedy function approximation: a gradient boosting machine.* Annals of Statistics (2001): 1189-1232.

[13] Geurts, Pierre, Damien Ernst, and Louis Wehenkel. *Extremely randomized trees.* Machine learning 63.1 (2006): 3-42.

[14] Geurts, Pierre, and Gilles Louppe. *Learning to rank with extremely randomized trees.* JMLR: Workshop and Conference Proceedings 14 (2011).

[15] Rendle, Steffen. *Factorization machines with libFM.* ACM Transactions on Intelligent Systems and Technology (TIST) 3.3 (2012): 57.

[16] Goodfellow, Ian J., et al. *Maxout networks.* arXiv preprint arXiv:1302.4389 (2013).

[17] Nair, Vinod, and Geoffrey E. Hinton. *Rectified linear units improve restricted boltzmann machines.* Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010

[18] McMahan, H. Brendan. *Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization.* International Conference on Artificial Intelligence and Statistics (2011).

[19] Xie, Jingjing, Bing Xu, and Zhang Chuang. *Horizontal and Vertical Ensemble with Deep Representation for Classification.* arXiv preprint arXiv:1306.2759 (2013).