Learning to Estimate the Authors of Double-blind Submission in Scholarly Networks

Jielin Qiu

Shanghai Jiao Tong University

Department of Electrical Engineering

Email: Qiu-Jielin@sjtu.edu.cn

- OUTLINE

The problem is to estimate the authors of a double-blind submission paper with the help of scholar networks.

- First, we conduct the feature engineering to transform the various provided text information into different features.

- Second, we train classification and ranking models using these features.

- Last, we combine our individual models to boost the performance by using results on the Valid set.

INTRODUCTION

The problem lies in how to estimate the author of papers which were submitted to double-blind process. The dataset, which is provided by Microsoft (Acemap Group), contains the information of confirmation and deletion of authors. The confirmation means the authors acknowledge they are the authors of the given paper; in contrast, the deletion means the paper has no information about its authors, institutions and any other information related to their personality. The confirmation and deletion information are split into three parts, including Train, Valid, an Test sets based on paper and author IDs.

- FRAMEWORK

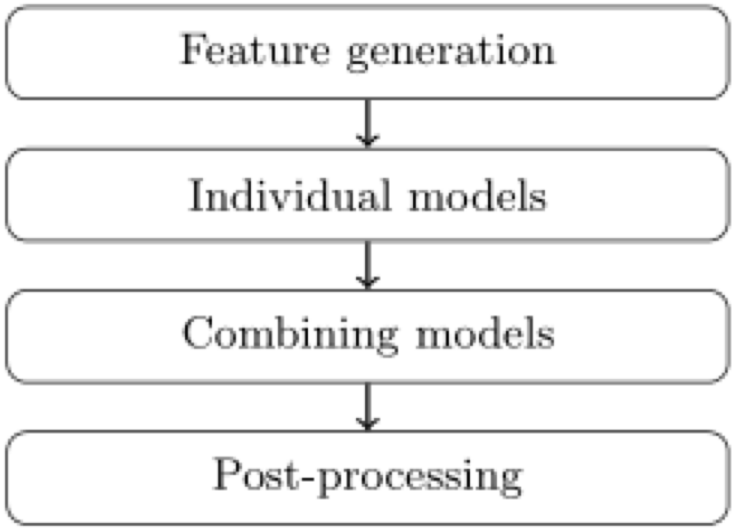


Fig. The architecture of our approach.

FEATURE ENGINEERING

- Preprocessing

Since many features are mainly based on string matching, we conduct simple preprocessing to clean the data. We first remove or replace non-ascii characters. Then, we remove stop words in affiliations, titles and keywords, where the stop-word list is obtained from the NLTK package. Finally, we convert all characters into lowercase before comparison.

FEATURE ENGINEERING

- Features Networks

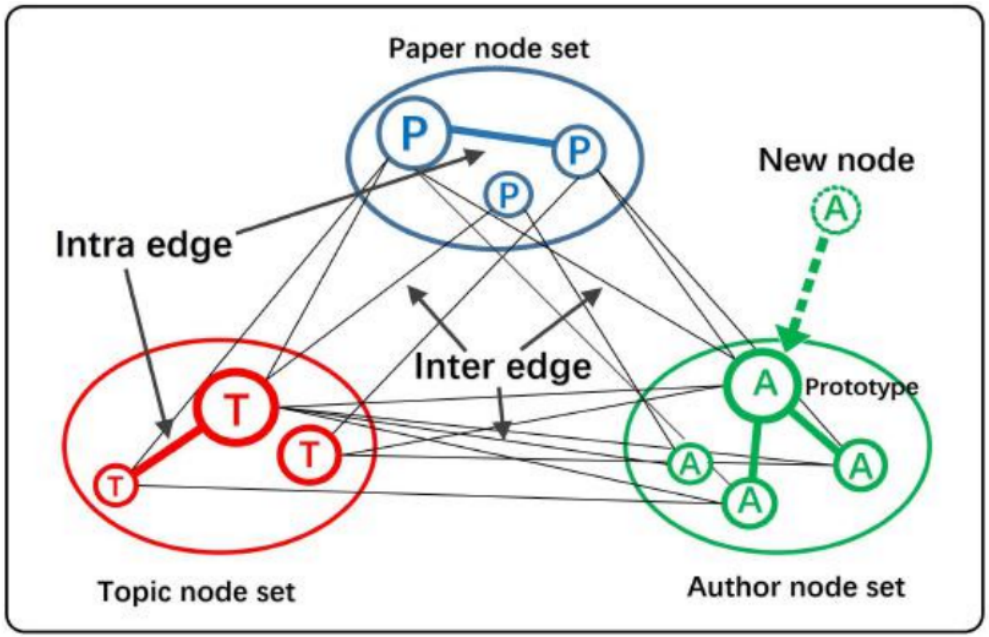


Fig. Relationship among paper, author and topic.

MODELS

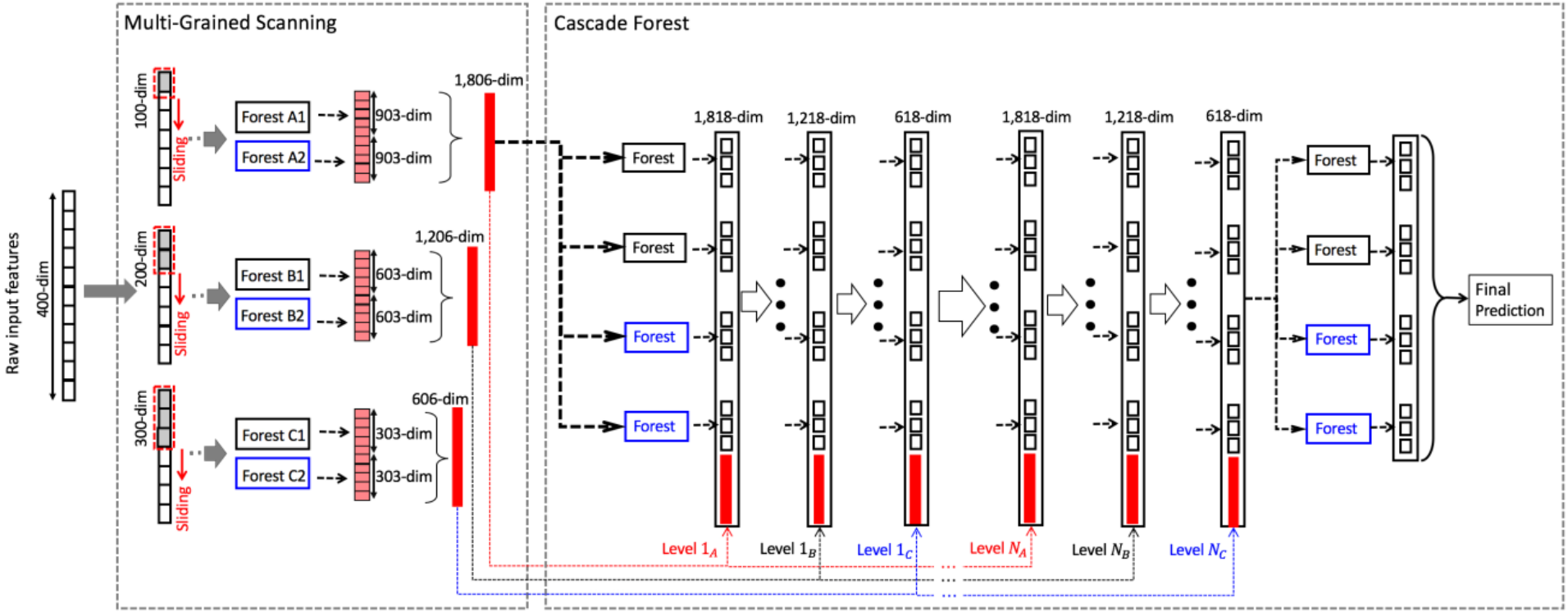
- Random Forests

Random Forests is a tree based learning method introduced by Leo Breiman. The algorithm constructs multiple decision trees using randomly sub-sampled features and outputs the result by averaging the prediction of individual trees. The use of multiple trees reduces the variance of prediction, so Random Forests are robust and useful in this project.

We use the implementation in the scikit-learn package. The package provides a parallel module to significantly speed up the tree building process. Note that the scikit-learn implementation combines classifiers by averaging probabilistic prediction instead of a voting mechanism.

- MODELS

  - gcForest



MODELS

- Gradient Boosting Decision Tree

Gradient Boosting Decision Tree (GBDT) is also a tree-based learning algorithm. We use the same package scikit-learn. The optimization goal of GBDT is to optimize "deviance" which is same as logistic regression. Unlike Random Forests, GBDT combines different tree estimators in a boosting way. A GBDT model is built sequentially by using weak decision tree learners on reweighted data. Then it combines built trees to generate a powerful learner. The main disadvantage of GBDT is that it cannot be trained in parallel, so we only use 300 trees to build the final ensemble model of GBDT. This is much smaller than 12,000 for Random Forests.

MODELS

- LambdaMart

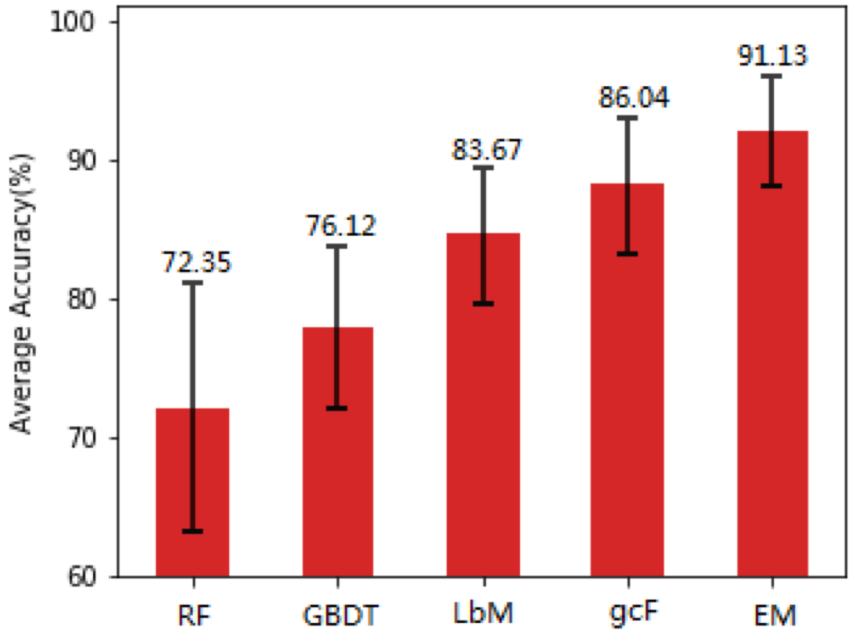
We choose LambdaMART because of its recent success on. LambdaMART is the combination of GBDT and LambdaRank. The main advantage is that LambdaMART use LambdaRank gradients to consider highly non-smooth ranking metrics. We use the implementation in the Jforests, which optimizes the NDCG metric. To avoid over-fitting, we train 10 LambdaMART models with random seeds from 0 to 9, and average the output confidence scores. The number of leaves is set to 32, the feature sample rate is 0.3, the minimum instance percentage per leaf is 0.01, and the learning rate is 0.1.

ENSEMBLE MODELS

In our system, we calculate the simple weighted average after scaling the decision values to be between 0 and 1. Because only four models, we search a grid of weights to find the best setting.

To see the performance under a setting of weights, we check the results on the Valid set. We train three models on the internal training set, and predict on the internal validation set. Then we combine the results according to the weights to check the improvement. Similarly, we train three models on the Train set (internal train set + internal valid set) and predict on the Valid set. Then we check whether results are further improved. The final weights are 1 for both Gradient Boosting Decision Tree and LambdaMART, 4 for gcForest, and 5 for Random Forests.

- RESULTS



RF is short for random forest, GBDT is short for Gradient Boosting Decision Tree, LbM is short for LambdaMART, gcF is short for gcForest, and EM is short for ensemble method. As it is shown, gcForest achieved better results than the other three and ensemble method took advantage of complementarity of different approaches and achieved better results than single model.

Thanks!