Application Scoring using Linear Regression

Thanh Vu and Michael Trenell

NIHR Innovation Observatory
{thanh.vu,michael.trenell}@io.nihr.ac.uk

1 Introduction

Correctly predicting the score of an application helps save time to not reviewing low-quality applications which are normally associated with low scores. In this paper, we propose to use Linear Regression [1] to handle this task with various types of input features including binary, lexical and semantic features. We then evaluate the proposed system on the dataset of 120 reviewed application with 5-fold cross-validation setting using Root Mean Square Error (RMSE) [4]. The initial result is promising with the best RMSE of 1.53 using the proposed features.

The remainder of this paper is organized as follows: We describe our system in Section 2. The experimental results and conclusion are detail in Section 3 and Section 4, respectively.

2 Our approach

2.1 Dataset

The dataset contains 120 applications which had been scored by reviewers with different review types, such as "Peer", "Public" and "Sub Panel Review". The score ranges from 0 to 10. We used the average score of the scores as the final single score for an application. Each application contains 383 sections in which the applicant need to provide the content (normally in the free text form).

2.2 Linear Regression

As the score is a number ranging from 0 to 10, we propose to use the Linear Regression (LR) model [1] to handle the application scoring task. Figure 1 shows an overview of our model architecture including an input layer, LR layer and output layer. Given an application, the input layer represents the application by a feature vector which concatenates features, such as binary, lexical, semantic features. The LR layer takes the input feature vector and selects the most important features which are then used to compute the predicted score of that application.

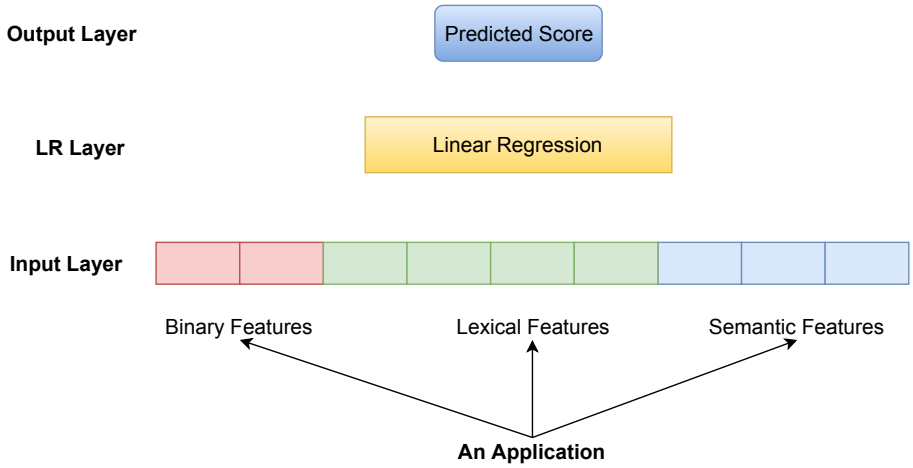

Fig. 1. Overview of our model architecture for application scoring.

Name	# Features
Binary features	383
Lexical features	5,000
Semantic features	300
Total	**5,683**

Table 1. Number of features used in our model

2.3 Features

Table 1 shows the number of binary, lexical and semantic features used in our model.

Binary features: We extract the binary feature for each section to indicate that whether the applicant had provided the content for that section. There are 383 binary features which are equivalent to the 383 application sections detailed in the Dataset section. The value of each feature is either 0 (without content) or 1 (with content).

Lexical features: Our lexical features include unigram in word level. We only leverage top 1,000 unigrams based on the term frequency-inverse document frequency (tf-idf) values. That is, each word appearing in an application becomes an entry in the feature vector with the corresponding feature value tf-idf.

Semantic features: We employ 300-dimensional pre-trained word embeddings from GloVe Pennington et al.,[3] to compute an application embedding as the average of the embeddings of words in the application.

3 Experiments

We use scikit-learn [2] to implement our model with 5-fold cross-validation setting. First, we randomly split the dataset into 5 folds. Each time, we combine

Model	Features used
Bin	Binary features
Sem	Semantic features
Lex	Lexical features
BinSem	Binary and Semantic features
BinLex	Binary and Lexical features
SemLex	Semantic and Lexical features
All	All the proposed features

Table 2. Model description

4 folds for training a linear regression model and use the remaining fold for evaluating the model. In total, we have 5 linear regression models with different results. The final result is the mean of the results.

3.1 Metric

The metric we use to evaluate our model is Root Mean Square Error (RMSE) [4] as it is the popular metric to evaluate the performance of a linear regression model. The lower RMSE value means the better performance.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(predicted\_score - actual\_score)^2}{T}} \quad (1)$$

3.2 Settings

We propose several combinations of proposed features to train different linear regression models. The proposed models are detailed in Table 2:

3.3 Results

Table 3 shows the results for different models. The combination between binary and lexical features produces the best performance of 1.53 RMSE. With 5000 features, Lex is the best single feature model. Using binary features (383 features) alone, we can still get the comparative performance. Semantic features seem to not help to improve the performance. We hypothesise that the embeddings were trained using general text domain which probably is much different from the application domain. In addition, as the dataset is very small we need to test on a larger-scale dataset to draw better/stronger conclusions.

Table 4 shows the scores predicted by the BinLex model.

4 Conclusion

In this paper, we proposed to use Linear Regression to handle the task of predicting application score using various features including binary, lexical and semantic features. Our system achieved the best performance using the combination of binary and lexical features.

Model	Mean_{RMSE}	Std_{RMSE}
Bin	1.65	0.38
Sem	2.17	0.25
Lex	1.54	**0.12**
BinSem	2.64	0.46
BinLex	**1.53**	0.15
SemLex	1.56	0.19
All	1.58	0.21

Table 3. Results

Reference	Actual Score	Predicted Score	Diff
II-LA-1116-20001	6.2	5.9	0.3
II-LB-1116-20003	7.0	6.5	0.5
II-OL-1116-10007	8.2	7.1	1.1
II-OL-0417-10017	5.8	5.7	0.1
II-LA-0417-20004	4.4	4.7	0.3
II-LB-0417-20004	5.2	4.8	0.4
II-LA-1116-20007	6.3	7.1	0.8
II-OL-0417-10008	3.5	4.2	0.7
II-OL-0417-10028	6.3	6.9	0.6

Table 4. Scores generated by BinLex

This tool can be used to filter out applications with lower predicted scores. Another application of this tool is to identify bias in the process of review. For example, if the predicted score of an application is much different from the actual score given by the reviewers, we can hypothesize that there is some bias in the reviewing process.

The data with reviewing scores is very small (120 applications). It leads to the fact that the conclusions in our report are weak and can be changed with a large-scale dataset. That is why we need to access larger-scale dataset in the future to draw stronger conclusions.

For the next step, we plan to evaluate the model on a large-scale dataset. Different machine learning techniques, such as tree-based and deep learning will be also explored in our future work.

References

1. J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. *Applied linear statistical models*, volume 4. Irwin Chicago, 1996.
2. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
3. J. Pennington, R. Socher, and C. Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

4. C. J. Willmott and K. Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.