Feature Engineering and Classifier Ensemble for KDD Cup 2010

Hsiang-Fu Yu, Hung-Yi Lo, Hsun-Ping Hsieh, Jing-Kai Lou, Todd G. McKenzie, Jung-Wei Chou, Po-Han Chung, Chia-Hua Ho, Chun-Fu Chang, Yin-Hsuan Wei, Jui-Yu Weng, En-Syu Yan, Che-Wei Chang, Tsung-Ting Kuo, Yi-Chen Lo, Po Tzu Chang, Chieh Po, Chien-Yuan Wang, Yi-Hung Huang, Chen-Wei Hung, Yu-Xun Ruan, Yu-Shi Lin, Shou-de Lin, Hsuan-Tien Lin, Chih-Jen Lin
Department of Computer Science and Information Engineering, National Taiwan University
Taipei 106, Taiwan

Editor:

Abstract

KDD Cup 2010 is an educational data mining competition. Participants are asked to learn a model from students' past behavior and then predict their future performance. At National Taiwan University, we organized a course for this competition. Most student sub-teams expanded features by various binarization and discretization techniques. The resulting sparse feature sets were trained by logistic regression (using LIBLINEAR). One sub-team considered condensed features using simple statistical techniques and applied Random Forest (through Weka) for training. Initial development was conducted on an internal split of training data for training and validation. We identified some useful feature combinations to improve performance. For the final submission, we combined results of student sub-teams by regularized linear regression. Our team is the first prize winner of both tracks (all teams and student teams) of KDD Cup 2010.

Keywords: educational data mining, feature engineering, classifier ensemble, linear classification, logistic regression, Random Forest

1. Introduction

KDD Cup 2010 is an educational data mining competition in which participants are charged with predicting student algebraic problem performance given information regarding past performance. Specifically, participants were provided with summaries of the logs of student interaction with intelligent tutoring systems. Two datasets are available: algebra_2008_2009 and bridge_to_algebra_2008_2009. In the rest of this paper, we refer to them as A89 and B89, respectively. Each dataset contains logs for a large number of interaction steps. There are 8,918,055 steps in A89, while there are 20,012,499 steps in B89. Some interaction log fields are included in both the training and testing sets, such as student ID, problem hierarchy including step name, problem name, unit name, section name, as well as knowledge components (KC) which were used in the problem and the number of times a problem has been viewed. However, there are some log fields which are only available in the training set: whether the student was correct on the first attempt for this step (CFA), number of hints requested (hint) and step duration information.

The competition regards CFA, which could be 0 (i.e., incorrect on the first attempt) or 1, as the label in the classification task. For each dataset, a training set with known CFA is available to participants, but a testing set of unknown CFA is left for evaluation. Subsequently we call the training and testing set T and $\tilde{T}$, respectively. The evaluation criterion used is the root mean squared error (RMSE). During the competition, participants submitted prediction results on the testing set to a web server, where the RMSE calculated based on a small subset of the testing data is publicly reported. This web page of displaying participants' results is called the "Leaderboard."

At National Taiwan University, we organized a course for KDD Cup 2010. Our members include three instructors, two TAs, 19 students and one RA. The students were split into six sub-teams. Every week, each sub-team presented their progress and participated in discussion with other sub-teams. The TAs helped to build an internal competition environment such that each sub-team could try their ideas before submission to the external competition website.

In the beginning we suspected that this year's competition would be very different from several past KDD Cups. Domain knowledge seems to be quite important for educational systems. Indeed, Educational Data Mining is an entire research field in itself with specialized models and techniques to address the unique types of data that are collected in educational settings.[1]

Contrary to several past KDD Cups, the meanings of the dataset entries were made available this year. This means that feature engineering is able to play a vital role. Also, temporal information is important as students are learning during the tutoring process of which the logs are a record. This property seems to differentiate the task from traditional classification, which assumes data independence. Interestingly, after exploring some temporal approaches such as Bayesian networks, we found that it was easier to incorporate useful information into a vector-space model and then train a traditional classifier. Therefore, our focus was shifted to identifying useful features by taking domain knowledge, temporal information, and other concepts into consideration.

The following sections describe our solution in detail. The architecture of our approach is outlined in Section 2. Sections 3 and 4 focus on diversified feature engineering. Ensemble techniques are described in Section 5. Section 6 shows our experimental results with some additional observations in Section 7.

2. Initial Settings and Our Approach

This section discusses the use of a validation set and introduces our solution architecture.

2.1 Validation Set Generation

Due to the nature of competition, participants tend to optimize the scores on the leaderboard. However, overfitting leaderboard results is undesired as the leaderboard shows the RMSE of only a small portion of the testing set. To avoid overfitting the leaderboard, our sub-teams focused on our internal competition environment, in which validation sets were

1. Major educational data mining conferences include, for example, International Conference on Intelligent Tutoring Systems and International Conference on Educational Data Mining.

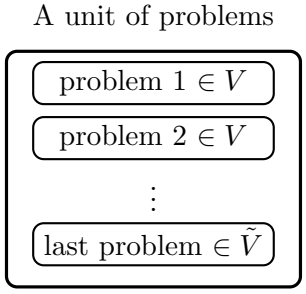

Figure 1: Generation of the validation set $\tilde{V}$. For each unit of problems, the last problem was used for validation, while all others were for training.

	A89	B89
Internal training (V)	8,407,752	19,264,097
Internal validation ($\tilde{V}$)	510,303	748,402
External training (T)	8,918,055	20,012,499
External testing ($\tilde{T}$)	508,913	756,387

Table 1: Number of steps in internal and external datasets

provided by splitting the training set (T) into two parts: an internal training set (V) and an internal testing set ($\tilde{V}$). Then, T and $\tilde{T}$ are referred to as external training and testing sets, respectively.

Care must be taken when choosing a validation technique, as some are only appropriate when certain criteria are met. For example, cross validation (CV) is a common technique applied in document classification problems. However, for this competition, simple CV is not a suitable approach due to the temporal property of the data. To generate a validation pair $(V, \tilde{V})$ with a distribution similar to the distribution of the challenge training and testing pair $(T, \tilde{T})$, we set the last problem in each unit in T as $\tilde{V}$, while calling the remaining part V. Figure 1 shows the partition of a unit of data. The procedure is same as the generation of $(T, \tilde{T})$, thus a similar distribution as achieved. Sizes of validation sets for A89 and B89 are given in Table 1.

2.2 Architecture of Our Solution

Because of the temporal relationship between data points, initially we tried methods such as Bayesian networks. However, we found that it was easier to incorporate information into independent vectors of data instances and train a traditional classifier. Therefore, the main task was to extract useful features and apply suitable classifiers. Our approach can be split into the following steps. First, each student sub-team extracted different features from the datasets according to their analysis and interpretation of the data, and chose different classifiers for learning based on the internal set. The feature engineering approaches can be categorized into two types: sparse feature sets generated by binarization and discretization techniques, and condensed feature sets using simple statistics on the data. Finally, we applied ensemble methods on the testing results from sub-teams. The procedure is shown in Figure 2. The detailed description of each step is presented in the following sections.

3. Feature Engineering: Sparse Features

We describe our approach of transforming logs of student interaction to sparse feature vectors and applying linear classifiers.

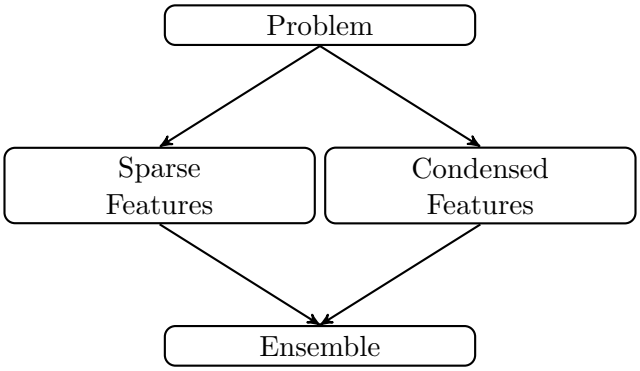

Figure 2: Solution architecture of our approach

3.1 Basic Sparse Features (Our Baseline Setting)

Some basic yet important features considered in our early experiments can be categorized into two types: student name, unit name, section name, problem name, step name and KC are categorical features, while problem view and opportunity are numerical features.[2]

We chose to expand a categorical feature into a set of binary features. For example, there are 3,310 students in A89. The feature vector then contains 3,310 binary features to indicate the student who finished the step.

For numerical features, due to large value ranges, we applied various scaling methods. One simple way is to linearly scale each numerical feature to the range $[0, 1]$. We have also checked nonlinear scaling methods, where for generating Figure 3 later in the experiment, we considered $\log(1 + x)$.

Using the above procedure, the resulting numbers of features for A89 and B89 are about 1 million and 200 thousand, respectively. Owing to the categorical expansion, each instance contains very few non-zero elements, yielding a sparse dataset. Though five out of six student sub-teams took this approach of feature expansion, their own baseline implementations were slightly different from the one described above. They may use only a subset of features or apply different scaling methods for numerical features.

3.2 Feature Combination

Due to the large training size, nonlinear classifiers like kernel training methods are perhaps not an ideal solution due to training time restrictions. Alternatively, linear classifiers can be used; see details in Section 3.5. However, linear classifiers are not capable of exploiting possible feature dependence. Following the polynomial mapping in kernel methods or bigram/trigram in natural language processing, we can use feature combinations to indicate these relationships. Because all feature meanings are available, we are able to manually identify some useful pairs of features. For example, hierarchical information can be modeled by indicating the occurrences of the following pairs: (student name, unit name), (unit name, section name), (section name, problem name) and (problem name, step name). In addition to two-feature combinations, we have also explored combinations of higher-order features. Re-

2. These two features have integer values, so they can also be considered categorical.

sults in the experiment section show that feature combinations effectively improve RMSE. We released two datasets using feature combinations at the LIBSVM dataset repository.[3]

3.3 Temporal Context as Features

Because learning is a process of skill-improving over time, temporal information should be taken into consideration. There are some well-established techniques, utilizing a quite different data model than traditional classification problems, to model student latent attributes such as knowledge tracing and performance factor analysis (Gong et al., 2010). We considered a simple and common approach to embed temporal context into feature vectors. For each step, step name and KC values from the previous few steps were added as features.

3.4 Other Ways for Generating Features

During the competition, we tried many other ways to generate features. Here we describe some of these methods.

In our baseline setting in Section 3.1, KCs are obtained by splitting the KC string by "~~" following the suggestion at the "Data Format" page on the competition website. Then binary features are used to indicate the KCs associated with each step. However, this setting results in many similar KCs. For example, "Write expression, positive one slope" and "Write expression, positive slope" indicate similar concepts, but are considered different. To remedy this problem, we tokenized KC strings and used each token as a binary feature. For example a KC "Write expression, positive slope" will cause four features, "Write," "expression," "positive" and "slope" to have true values. Our experiences indicated that this method for generating KC features is very useful for the dataset A89. We also used techniques such as regular expression matching to parse knowledge components, but did not reach a setting clearly better than the one used for the baseline.

For problem name and step name, we tried to group similar names together. For example, two step names "$-18 + x = 15$" and " $5 + x = -39$" differ only in their numbers. By replacing numbers with a symbol, they become the same string and are then considered to have the same step name. This simple approach effectively reduced the number of features without deteriorating the performance, and a majority of our student sub-teams applied this approach. One or two student sub-teams tried more sophisticated methods to group similar names, but did not reach concrete conclusions.

We also tried to model the learning experience of students. A student's performance may depend on whether that student had previously encountered the same step or problem. We added features to indicate such information. Results showed that this information slightly improves the performance. Conversely, we also added a reverse cumulative features, recording the occurrence of future problems. The logic here is that the appearance of future problems will depend on the performance of previous problems.

Earlier problem view was considered as a numerical feature. In some situations, we treated it as a categorical one and expanded it to several binary features. This setting is possible because there are not too many problem view values in the training and testing sets. One student sub-team reported that this modification slightly improved RMSE.

3. See `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. We thank Carnegie Learning and Datashop for allowing us to release these processed datasets.

3.5 Linear Classification and Probability Outputs

In our classification process, for each step we used its CFA as label y_i

$$y_i = \begin{cases} 1 & \text{if CFA} = 1, \\ -1 & \text{if CFA} = 0, \end{cases}$$

and extracted a sparse feature vector $\boldsymbol{x}_i$. Hence, we assume the training set includes $(\boldsymbol{x}_i, y_i),\ i = 1, \ldots, l$. Due to the high dimensionality of $\boldsymbol{x}_i$, we considered only linear classifiers. In particular, we used logistic regression, which assumes the following probability model:

$$\mathcal{P}(y \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-y\boldsymbol{w}^T\boldsymbol{x})}.$$

Then, regularized logistic regression solves the following optimization problem

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l} \log\left(1 + e^{-y_i\boldsymbol{w}^T\boldsymbol{x}_i}\right), \tag{1}$$

where $\boldsymbol{w}$ is the weight vector of the decision function, $\boldsymbol{w}^T\boldsymbol{w}/2$ is the L2-regularization term, and C is a penalty parameter. The parameter C is often decided by a validation procedure. In our experiments, we used $C = 1$ most of the time.

L2 regularization leads to a dense vector $\boldsymbol{w}$, so we have also considered L1-regularized logistic regression to obtain a sparse $\boldsymbol{w}$:

$$\min_{\boldsymbol{w}} \quad \|\boldsymbol{w}\|_1 + C\sum_{i=1}^{l} \log\left(1 + e^{-y_i\boldsymbol{w}^T\boldsymbol{x}_i}\right). \tag{2}$$

Once $\boldsymbol{w}$ is obtained, we submitted either labels $y = 1$ if $\boldsymbol{w}^T\boldsymbol{x} \geq 0$ and 0 otherwise, or probability values $1/(1 + \exp(-\boldsymbol{w}^T\boldsymbol{x}))$ as our predictions. Results showed that using probability values gives a smaller RMSE. We can easily explain this result. Assume the true label is 0. For a wrongly predicted data point, the error incurred when predicting a false label and a probability of this occurrence are 1 and $(1 - p_1)^2$, respectively, where $p_1 \geq 0.5$ is the predicted probability. If a data point is correctly predicted, then this error and probability are 0 and p_2^2, respectively, where $p_2 \leq 0.5$. Since the quadratic function is increasing in $[0, 1]$, the gain of reducing 1 to p_1^2 for a wrongly predicted data is often larger than the loss of increasing 0 to p_2^2 for a correctly predicted data.

In our development, we used a large-scale linear solver LIBLINEAR (Fan et al., 2008), which can effectively train very large data. It took only about one hour to train our largest setting for the data B89, which contains more than 20 million instances and 30 million features.

We also checked linear support vector machine (SVM) solvers in LIBLINEAR, but the result was slightly worse than logistic regression.

4. Feature Engineering: Condensed Features

In contrast to generating sparse features in Section 3, this section discusses an approach that exploits only a small feature set.

4.1 Correct First Attempt Rates as Feature Values

We proposed replacing each categorical feature with a numerical one by using the "correct first attempt rate" (CFAR). Take the student name feature as an example. The CFAR of a specific student name "sid" is defined as

$$\text{CFAR} \equiv \frac{\#\text{steps with student name} = \text{sid and CFA} = 1}{\#\text{steps with student name} = \text{sid}}.$$

For each step whose student name is sid, the above CFAR would be used as the corresponding feature value. This setting directly connects a feature and CFA, which is now the target for prediction. We have considered CFARs for student name, step name, problem name, KC, (problem name, step name), (student name, problem name), (student name, unit name) and (student name, KC). Feature combinations are considered following the discussion in Section 3.2 for sparse features. For the KC feature, we consider the whole string as the ID in calculating CFAR. We cannot split the string to several KC features as in Section 3.

4.2 Learning Temporal Information

We developed two approaches to extract temporal information.

First, we suspect that the performance on the current problem is related to the same student's past performances on similar types of problems. Thus we use the average CFA and average hint on the student's previous records (up to six depending on the availability) with the same KC to model the student's recent performance on similar problems.[4] A binary feature is also added to indicate whether such previous records exist.

Second, as students have higher chance to forget older events, the time elapsed since the student saw a problem with the same KC may reveal the performance of the student. For the current step, we find the KC and check if the student has seen a step with the same KC within

same day, 1-6 days, 7-30 days, or more than 30 days.

Therefore, four binary features are used to indicate how familiar a student is with a given KC.

4.3 Training by Various Classifiers

In addition to features mentioned above, two numerical features opportunity and problem view were first scaled by $x/(x+1)$ and then linearly scaled to the interval $[0, 1]$. Thus the feature set contains 17 features:

- Eight CFARs; see Section 4.1.
- Seven temporal features; see Section 4.2.
- Two scaled numerical features for opportunity and problem view.

Due to the small feature size, we were able to apply various classifiers in Weka including Random Forest (Breiman, 2001) and AdaBoost (Freund and Schapire, 1997). Random Forest is an ensemble classifier consisting of many decision trees, while AdaBoost is a

4. In the testing set, neither CFA nor hint is available. Because testing data was generated using the last problem of a unit (see Figure 1), finding CFA and hint values in the student's previous records (with the same KC) is not a problem.

method which adaptively improves the performance by a series of weak classifiers (e.g., decision stumps). We also used logistic regression via LIBLINEAR. To save training time, we considered a subset of training data and split the classification task into several independent sets according to unit name. Specifically, for each unit name, we collected the last problem of each unit to form its training set. A classification method was then applied to build a model. In testing, we checked the testing point's unit name to know which model to use. Results showed that Random Forest technique gives the best result. Regarding the training time, due to the small feature size, we could efficiently train a Random Forest on the training subset of a unit in a few minutes.

5. Classifier Ensemble

Many past competitions (e.g., Netflix Prize) showed that an ensemble of results from different methods can often boost the performance. The ensemble approach we adopted was to find a weight vector to linearly combine the predicted probabilities from student sub-teams. We did not use a nonlinear ensemble because a complex ensemble may increase the chance of overfitting. We checked several linear models including simple averaging, linear SVM, linear regression and logistic regression. Probably because linear regression directly minimizes RMSE, which is now the evaluation criterion, we found that it gives the best leaderboard result. Given l testing steps and k prediction probabilities $\boldsymbol{p}_i \in [0,1]^l,\ i = 1, \ldots, k$, regularized linear regression solves the following optimization problem:

$$\min_{\boldsymbol{w}} \quad \|\boldsymbol{y} - P\boldsymbol{w}\|^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2, \tag{3}$$

where λ is the regularization parameter, $\boldsymbol{y}$ is the CFA vector, and $P = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k]$. If $\lambda = 0$, Equation (3) becomes a standard least-square problem. In SVM or logistic regression, sometimes we add a bias term b so that $P\boldsymbol{w}$ becomes $P\boldsymbol{w} + b\mathbf{1}$, where $\mathbf{1} = [1, \ldots, 1]^T$. In our implementation, we also replaced $\|\boldsymbol{w}\|^2$ with $\|\boldsymbol{w}\|^2 + b^2$.

The obtained weight $\boldsymbol{w}$ is used to calculate $P\boldsymbol{w}$ for combining prediction results. However, as components of $P\boldsymbol{w}$ may be out of the interval $[0, 1]$, we employ a simple truncation to avoid such a situation:

$$\min(\mathbf{1}, \max(\mathbf{0}, P\boldsymbol{w})), \tag{4}$$

where $\mathbf{1}$ is the vector with all ones, and $\mathbf{0}$ is the vector with all zeros. We also explored sigmoid transformation and linear scaling for modifying $P\boldsymbol{w}$ to $[0,1]^l$, but results did not improve on these measures.

The analytical solution of (3) is

$$\boldsymbol{w} = (P^T P + \frac{\lambda}{2} I)^{-1} P^T \boldsymbol{y}, \tag{5}$$

where I is the identity matrix. The main difficulty to apply (5) is that $\boldsymbol{y}$ is unknown. We considered the following two approaches:

1. Use validation data to estimate $\boldsymbol{w}$. Recall that we split the training set T to two sets V and $\bar{V}$ for internal training an validation, respectively; see Section 2.1. Our student

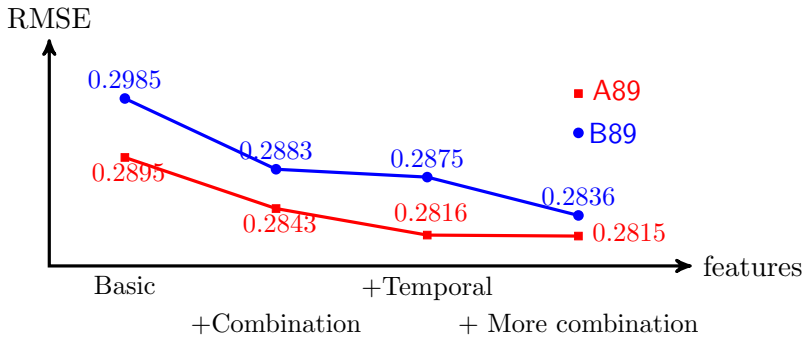

Figure 3: Experiment result of how incremental sparse features improve RMSE. Reported RMSEs are leaderboard results.

sub-teams generated two prediction results on $\tilde{V}$ and $\tilde{T}$:

$$\begin{aligned} \text{Train } V \Rightarrow &\text{ Predict } \tilde{V} \text{ to obtain } \tilde{\boldsymbol{p}}_i, \\ \text{Train } T \Rightarrow &\text{ Predict } \tilde{T} \text{ to obtain } \boldsymbol{p}_i. \end{aligned}$$

Let $\tilde{P}$ be the matrix of collecting all predictions on the validation set and we have the corresponding true labels $\tilde{\boldsymbol{y}}$. Then in (3) we used $\tilde{\boldsymbol{y}}$ and $\tilde{P}$ to obtain $\boldsymbol{w}$. For the final prediction, we calculated $P\boldsymbol{w}$ and applied the truncation in (4).

2. Use leaderboard information to estimate $P^T\boldsymbol{y}$ in (5). This approach follows from Töscher and Jahrer (2009). By the definition of RMSE,

$$r_i \equiv \sqrt{\frac{\|\boldsymbol{p}_i - \boldsymbol{y}\|^2}{l}},$$

so

$$\boldsymbol{p}_i^T\boldsymbol{y} = \frac{\|\boldsymbol{p}_i\|^2 + \|\boldsymbol{y}\|^2 - lr_i^2}{2}. \tag{6}$$

Because r_i and $\|\boldsymbol{y}\|$ are unavailable, we estimated them by

$$r_i \approx \hat{r}_i \quad \text{and} \quad \|\boldsymbol{y}\|^2 \approx l\hat{r}_{\boldsymbol{0}}^2,$$

where $\hat{r}_i$ is the RMSE on the leaderboard by submitting $\boldsymbol{p}_i$ as the prediction result and $\hat{r}_{\boldsymbol{0}}$ is the RMSE by submitting the zero vector.

We conducted experiments to check which of the above two methods performs better. See Section 6 for our setting (including the selection of λ) for the final submission.

6. Experiments and Final Results

We used the leaderboard to test the effectiveness of different types of features. We checked four settings by incrementally adding basic sparse features, combined features, temporal features and additional combined features. See details of added features in Table 2(a). The number of features for each setting is shown in Table 2(b). We conducted this experiment

Table 2: Detailed information of our feature combinations and our temporal features.

(a) List of features

+Combination	(student name, unit name), (unit name, section name), (section name, problem name), (problem name, step name), (student name, unit name, section name), (unit name, section name, problem name), (section name, problem name, step name), (student name, unit name, section name, problem name) and (unit name, section name, problem name, step name)
+Temporal	Given a student and a problem, add KCs and step name in each previous three steps as temporal features.
+More combination	(student name, section name), (student name, problem name), (student name, step name), (student name, KC) and (student name, unit name, section name, problem name, step name)

(b) Number of features

Features	A89	B89
Basic	1,118,985	245,776
+Combination	6,569,589	4,083,376
+Temporal	8,752,836	4,476,520
+More combination	21,684,170	30,971,151

Table 3: RMSE on the leaderboard after adding important features gradually.

#features	A89	B89
Basic	0.2895	0.2985
+ (problem name, step name)	0.2851	0.2941
+ (student name, unit name)	0.2881	0.2942
+ (problem name, step name) and (student name, unit name)	0.2842	0.2898
+ Combination; see Table 2(a)	0.2843	0.2883

with the L1-regularized logistic regression solver in LIBLINEAR with $C = 1$. The leaderboard results are shown in Figure 3. Clearly, some feature combinations provide additional information to basic features, so the RMSE is significantly improved. In particular, the RMSE of B89 is reduced from 0.2985 to 0.2883. In contrast, from Figure 3, temporal features are more useful for A89 than B89. We managed to further improve B89's RMSE by adding more feature combinations, combining the five features together in additional ways. The improvement is, however, less dramatic which is likely due to the fact that much of the information captured by these new features had already been realized by earlier feature combinations.

During the competition, our sub-teams reported some feature combinations were very useful such as (problem name, step name) and (student name, unit name). To further confirm this observation, we designed an experiment to see the impact of these two combinations. Table 3 shows the leaderboard results of basic sparse features, basic sparse features plus

Table 4: Leaderboard results by different approaches.

	A89	B89	Avg.
Basic sparse features	0.2895	0.2985	0.2940
Best sparse features	0.2784	0.2830	0.2807
Best condensed features	0.2824	0.2847	0.2835
Best ensemble	0.2756	0.2780	0.2768
Best leaderboard	0.2759	0.2777	0.2768

one feature combination (either (problem name, step name) or (student name, unit name)), and basic sparse features plus both feature combinations. From this table, we can see that these two feature combinations clearly improve the leaderboard RMSE, especially for B89.

Next, Table 4 lists our leaderboard results submitted during the competition. "Basic sparse features" means that we used the baseline setting described in Section 3.1. For "Best sparse features," the improvement of reducing RMSE by around 0.015 is very significant. For the condensed representation, the best result comes from training Random Forest with depth 7. It is surprising that the performance (in particular, on B89) of using 17 features is competitive with the sparse approach with millions of features.

For the classifier ensemble, we collect 19 results from 7 sub-teams (six student sub-teams and one RA sub-team). We make sure that each result comes from training a single classifier instead of combining several predictions. While not shown in the table, results are improved by various ensemble strategies including simple averaging. For the "Best ensemble results" in Table 4, we use linear regression for combining 19 results. To select λ in (3), we gradually increased λ until the leaderboard result started to decline. This procedure, conducted in the last several hours before the deadline, was not very systematic, but this is unavoidable in participating in a competition. Our best A89 result is by estimating $P^T\boldsymbol{y}$ in (5) and using $\lambda = 10$. See the second method in Section 5. For the B89, the best result is by using the validation set to estimate $\boldsymbol{w}$, which corresponds to the first method in Section 5. The parameter λ is zero (i.e., no regularization).

In Table 4, we also show the best leaderboard result (by another team). Our team ranked 2nd on the leaderboard.[5] However, from the last two rows of Table 4, the difference between our result and the best leaderboard result is very small. At that time, we hoped that our result suffered from a lesser degree of overfitting and might yield better results on the complete challenge set.

Table 5 showed the final KDD Cup results of the top five teams. Note that, in general, final KDD Cup scores are slightly better than the leaderboard results.

7. Discussion and Conclusions

There were many submissions in the last week before the deadline. In particular, in the last two hours as each top team, including ourselves, tried to outperform the other top teams on the leaderboard. Whether such a large number of submissions resulted in overfitting remains a concern.

5. More precisely, our submission ranked 10th, but the 1st to the 9th submissions on the leaderboard were from the same team.

Table 5: Challenge final result.

Rank	Team name	Leaderboard	Cup
1	National Taiwan University	0.276803	0.272952
2	Zhang and Su	0.276790	0.273692
3	BigChaos @ KDD	0.279046	0.274556
4	Zach A. Pardos	0.279695	0.276590
5	Old Dogs With New Tricks	0.281163	0.277864

We believe that a crucial point to our ensemble's success is feature diversity. Different sub-teams tried various ideas guided by their own intuition. Our feature combination helps to enrich the feature set. However, while we have identified some important features, a detailed study on feature selection is needed to obtain more definitive results.

Once we decided to use a vector-space model and apply standard classifiers such as logistic regression, the techniques which were subsequently developed in this paper seem to be reasonable. As indicted in Section 2.2, we initially tried Bayesian networks to directly model the domain knowledge, but failed to obtain good results. Therefore, a traditional classification approach may be more suitable for this competition. Whether the same conclusion holds for other education data mining tasks is an interesting future issue.

In summary, feature engineering and classifier ensembling were useful techniques used in our approach for winning KDD Cup 2010.

Acknowledgements

We thank the organizers for offering this interesting and fruitful competition. We also thank National Taiwan University for providing a stimulating research environment.

References

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9: 1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Yue Gong, Joseph E. Beck, and Neil T. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. *Intelligent Tutoring Systems*, 2010.

Andreas Töscher and Michael Jahrer. The BigChaos solution to the Netflix grand prize. 2009.