A Feature Engineering Approach for Click-Through Rate Prediction: KDD CUP Track 2

Lucas Silva
Belo Horizonte 30330-110, Brazil
lucas.eustaquio@gmail.com

Aaron Davis
DataLabs
Maryland 20876-4063, USA
aaron@datalabusa.com

Henrique Ribeiro
UFMG - ICB
Belo Horizonte 30330-220, Brazil
deassis.info@gmail.com

ABSTRACT
This paper describes detailed information about our approach to the task 2 of KDD Cup 2012. It presents the problem of ranking the click-through rate (CTR) of advertisements. The CTR is used in search advertising to rank ads and to price clicks.
We cast this task as a binary classification problem and addressed it utilizing gradient boosted decision trees. To make this possible, we created five sets of predictive features: ad-based statistical features, unknown-users imputation features, user-based statistical features, context-based statistical features and cross product statistical features. These statistical features were created using linear mixed effects and bag of words models. This approach granted us 4th place, above the performance mark of 0.8 (AUC).

Categories and Subject Descriptors
I.2.6 [**Machine Learning**]: Engineering applications

General Terms
Algorithms, Experimentation

Keywords
KDD Cup, feature engineering, linear-mixed-effects, gradient boosted decision trees, bag of words, machine learning, recommender systems.

1. INTRODUCTION
The "KDD Cup 2012 Track 2" contest posed the challenge of ranking advertisements' CTRs. A dataset containing ads' click history was provided by Tecent [12], and our task was to rank the CTRs of about 20 million ads-user sessions. A detailed description of the dataset can be found on [1].

A big challenge in this task was dealing with a massive dataset: there were 22,023,547 users, 641,707 ads over 149,639,105 sessions. For a dataset this size, commonly statistical software like R [14] cannot be efficiently used without proper big data counter-measures. As we used R in this project, we had to keep that constantly in mind.

Another huge challenge was to create efficient features. Feature creation is one of the most important steps in solving a supervised learning problem. To this end we developed many features which can be grouped into the following sets: ad-based statistical features, unknown-users imputation features, user-based statistical features, context-based statistical features and cross-product statistical features.

In this paper, we describe how we handled the challenges described above. Section 2 briefly describes the dataset and how we split this data for model training. Section 3 outlines the algorithms we used. Section 4 shows the performance measurement. Section 5 is about our approach to this task, it describes our training methods and features. Section 6 presents our results, section 7 discusses some aspects of our solution, and section 8 presents our conclusions.

2. DATASET
The provided dataset [1] consisted of two datasets, one for performance evaluation (test set) with 20,297,594 instances and another for training with 149,639,105 instances. The datasets structure is presented in Table 1.

Table 1. Description of KDD dataset.

Column	Description	Type
Click	Number of times an ad was clicked	Output (train only)
Impression	Number of times an ad was printed	Output (train only)
Depth	Number of printed ads in the user screen	Context-based
Position	Position of ad in the user screen	Context-based
DisplayURL	Hash of URL	Ad-based
AdID	Advertise id	Ad-based
AdvertiserID	Advertiser id	Ad-based
KeywordID	Keyword id	Ad-based
KeywordTokens	Word tokens expanded from KeywordID	Ad-based
TitleID	Title id	Ad-based
TitleTokens	Word tokens expanded from TitleID	Ad-based
DescriptionID	Description id	Ad-based
DescriptionTokens	Word tokens expanded from DescriptionID	Ad-based
UserID	User id	User-based
Age	Age of an user, expanded from UserID	User-based
Gender	Gender of an user, expanded from UserID.	User-based
QueryID	Query id	User-based
QueryTokens	Word tokens expanded from QueryID	User-based

Table 1 shows the dataset structure. As we can see KDD data was grouped in four categories:

- Output: the data to be predicted and presented only in training set.

- Context-based: represents interaction's properties between user and the advertisement
- Ad-based: features representing properties of the advertisement. Some of them point to a list of word tokens.
- User-based: features representing user properties. The UserID feature points to a file mapping age and gender and the QueryID points to a list of word tokens.

This classification scheme of input variables was very important in the feature creation step as it allowed us to define the cross-context interactions we used to build our final model.

2.1 Training set split

The training set was split in two parts, trying to replicate between these parts the same relationship the original training set and test set had. It was done to ensure that our training method had to deal with a significant number of ad/users with no history. That decision played a huge role in our final standing as it allowed our internal training sets to reflect properly the competition test set.

Since the instances didn't have a timestamp to do a temporal separation, this was done keeping the history distribution of ads and users as in test set. Figure 1 shows the distribution for the official test set, and Figure 2 shows the distribution of ads' history size for the internal sampled set.

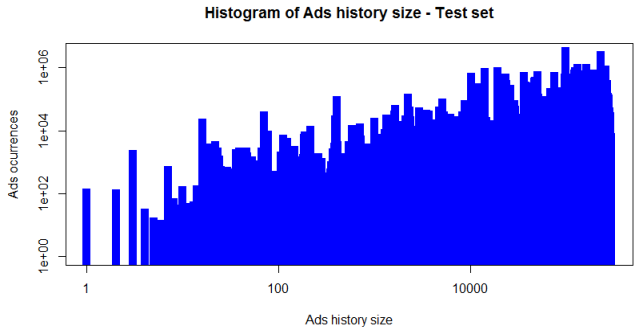

Figure 1. Histogram of previous occurrences of Ads - Test set

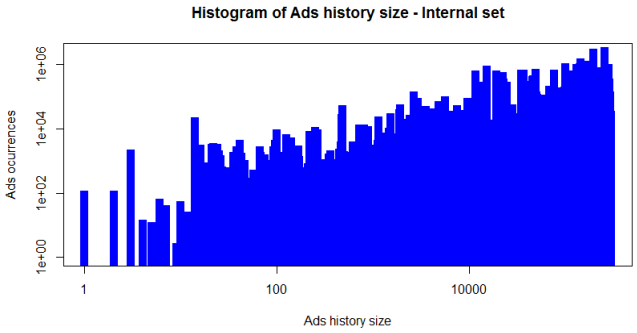

Figure 2. Histogram of previous occurs of Ads – Internal set

After doing this for the AdID feature, that history size likeliness followed for all other ad-related features. Unfortunately, that didn't happen for the user-based features QueryID and UserID. In that case, we synthetically anonymized the users of about 3 million instances. We randomly sampled in our internal set (changed their ids to an unknown one) and selecting about 1.5 million instances of queries independently using the same procedure.

3. ALGORITHMS

In this work we have used mainly three machine learning algorithms, most of them on the R statistical environment. Table 2 gives a brief description of how each algorithm was employed. The rest of this section outlines these algorithms in more detail.

Table 2. Used algorithms

Algorithm	Used for
Vowpal Wabbit [2]	Bag of words related models
LME [3]	Statistical features creation
GBM [4]	Model training

3.1 Vowpal Wabbit

The Vowpal Wabbit (VW) project is an intrinsically fast out-of-core learning system, sponsored by Yahoo! [2], that runs on a online learning platform. It provides several loss functions as well as learning algorithms. VW also provide a sparse matrix input format which easily allows a bag of words model.

In our solution it was used for two different tasks as described next.

3.1.1 One against all

VW provides built-in one against all (OOA) training mode (-ooa option). It just needs to be fed with a multi-class dataset. In this work we used OOA to do gender and age imputation. Squared loss function (1) was used for this:

$$l(p, y) = \frac{1}{2}(p - y)^2 \quad (1)$$

where

- p represents the prediction.
- y the desired output.
- l(p, y) represents the deviation between this values.

In OOA, as many models as distinct output classes are trained. Each model describes the probability of an output class occurring. The predicted class is chosen based on the highest probability found.

3.1.2 Bag of words

In a bag of words model, a sparse binary vector is used to represent the presence of a word. VW input format natively supports this.

In our approach, the bag of words model was used to predict the probability of each word based feature (word tokens) being present on a clicked instance. The loss function (1) was also used for this model.

3.2 LME – Linear Mixed Effects

The linear mixed effects (LME) model may be viewed as a generalization of the variance component and regression analysis models. When the number of categorical values is small and the number of observations per category is large, we treat the cluster-specific coefficients as fixed and ordinary regression analysis with dummy variables applies. Such a model is called a fixed effects model. Vice versa, when the number of categorical values is large but the number of observations per category is relatively small, and a random effects model would be more appropriate. Penalized likelihood is frequently used to cope with parameter multidimensionality [6].

We used the R nlme library [3] for statistical features creation. The lme (linear mixed effects) function in the nlme library employs the Laird-Ware form of the linear mixed model [7].

$$y_{ij} = \beta_1 x_{1ij} + \cdots + \beta_p x_{pij} + b_{i1} z_{1ij} + \cdots + b_{iq} z_{qij} + \varepsilon_{ij}$$
$$b_{ik} \sim N(0, \psi_k^2), Cov(b_k, b_{k'}) = \psi_{kk'} \quad (2)$$
$$\varepsilon_{ij} \sim N(0, \sigma^2 \lambda_{ijj}), Cov(\varepsilon_{ij}, \varepsilon_{ij'}) = \sigma^2 \lambda_{ijj'}$$

where

- y_{ij} is the value of the response variable.
- β are the fixed-effect coefficients.
- x are the fixed-effect regressors.
- b are the random-effect coefficients for each categorical value.
- z are the random-effect regressors.
- ψ^2_k are the variances and $\psi_{kk'}$ the covariances among the random effects.
- ε is the error.
- $\sigma^2\lambda_{ijj'}$ are the covariances between errors.

3.3 GBM – Generalized Boosted Models

GBM was used to train our final submission models. We used the GBM package [4] in the R environment with the AdaBoost distribution function [8].

The R GBM package implements boosting for models commonly used in statistics. The GBM algorithm is as follows [9]:

1. $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$
2. For $m = 1$ to M do:
3. $\quad \tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \; i = 1, N$
4. $\quad \mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
5. $\quad \rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6. $\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7. endFor

end Algorithm

Algorithm 1 - Boosting algorithm

where:

- F(x) is the function we want to minimize
- L(y, p) is the loss function
- y is the output
- p is the prediction
- M is number of trees
- h(x; a) is a "weak learner"or "base learner" usually trees

4. EVALUATION METRIC

The evaluation metric for task 2 was Area Under Curve (AUC), which can be viewed as the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one [15]. An AUC score of 1 indicates the perfect classifier and an AUC score of 0.5 indicates that a classifier is close to the random classifier and has no statistical value. More details about this metric can be found in [10].

5. OUR APPROACH

In this work, we split the training set as explained in section 2 generating two training subsets tr1 and tr2.

Features were created using tr1 subset for model training and using the full training set for prediction.

Models were trained using likelihood calculated with tr1 (history set) applied to tr2 (training/validation set). Next, submission outputs were predicted using likelihood calculated with the whole training set (submission history set) and the model previously trained. Figure 3 summarize this process.

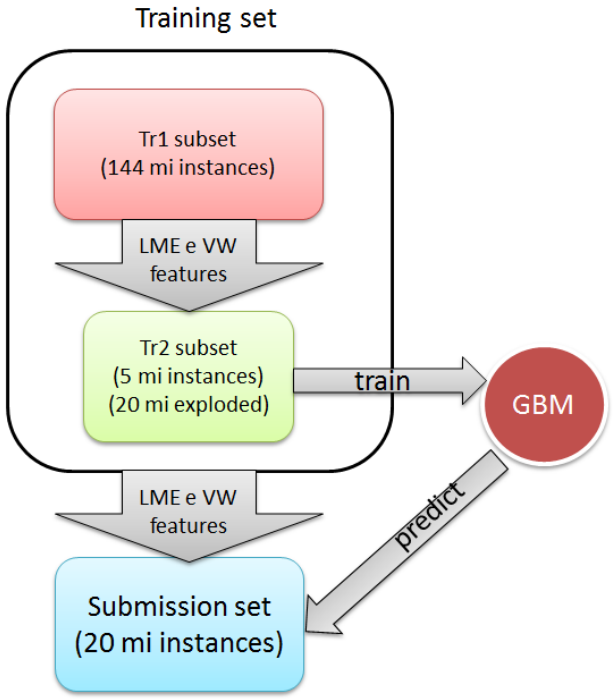

Figure 3. Diagram summarizing our approach.

5.1 Features creation

Features were created in three fashions: imputation, bag of words features and statistical history features. These three sets of features are explained next.

5.1.1 Imputation

Over 25% of the dataset consisted of unknown users. That's a considerable amount of data. To work around this, missing gender and age were imputed using the query word tokens since they were user generated. This prediction was made with the one against all option of the VW algorithm with default options. No feature pre-processing was needed because VW provides native support for this kind of input.

Known users in the submission set were used to train the imputation model. Accuracy was measured in the known users of the training set. Table 3 shows the imputation accuracy. The low level of accuracy for age and the large discrepancy in age/gender accuracy can be partially explained by the fact that gender has two distinct classes and age has six. Another reason behind this is the fact that the only feature we could use here was the QueryID.

Table 3. Imputation accuracy

Feature	Accuracy
Age	33.28%
Gender	62.85%

In order to not introduce some noise due to inaccuracy of imputation, the calculated values were not imputed as real class

values. Instead, new levels were introduced creating the concept of "maybe man", "maybe woman", maybe 0-12 years" and so on.

The unknown user ids were also replaced by a combination of imputed gender and age, effectively clustering them based on the query tokens.

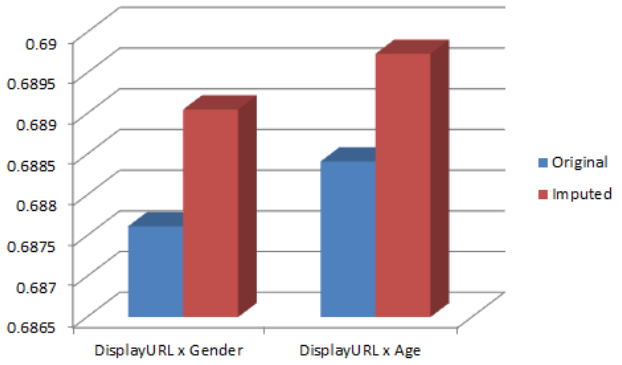

Figure 4. AUC of Original x Imputed cross feature

In Figure 4, we can see the AUC of a cross-product (DisplayURL x Age and DisplayURL x Gender) feature with both the original and imputed versions of Age and Gender. In both cases it shows an improvement of 0.135 in the AUC when using the imputed version. These values of AUC were measured in the internal validation set.

5.1.2 Bag of words

Likelihoods of keyword, query and title were calculated using their tokens lists with the VW algorithm. The likelihood of the cross-product (more on this soon) of these tokens with user-based variables (Gender, Age) and context-based variables (Deph, Position) were also calculated.

Figure 5 shows the AUC score difference between the original word feature (e.g. KeywordID) and the token based feature (KeywordTokens). The original feature usually scores higher, but for features with lower history it is better to use the token based one (shrinkage). That trend is also shown on the figure. The KeywordID feature has a higher average history size (125 previous ocurrences per id) than the QueryID feature (7.5 previous occurrences per ID); hence, the overall performance difference of the two versions of the KeywordID is higher when compared to the QueryID one.

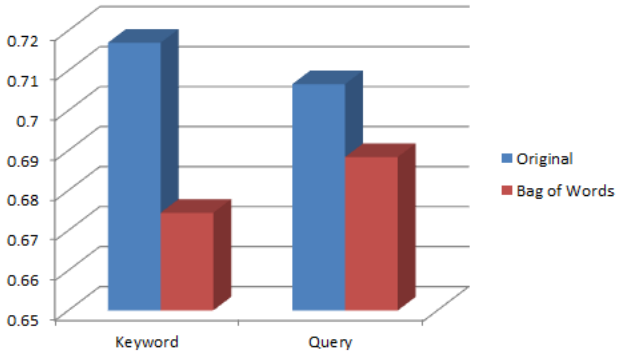

Figure 5. WordFeatureID x WordFeatureToken AUC

5.1.3 LME

LME were used to transform categorical variables with many values (more than 15) in likelihood using shrinkage. To check the quality of this shrunken likelihood, we compared the AUC of some these features with the equivalent benchmark scores posted on the leaderboard (raw likelihood benchmarks). It is shown on Figure 6.

As shown in Figure 6, most of the gain occurs in "UserID" and "QueryID". These two variables have lower average history sizes (7.5 previous occurrences per ID) and therefore lower confidence in calculated averages. On the other hand, "AdvertiserID" has the highest average history size (15,000 previous occurrences per ID).

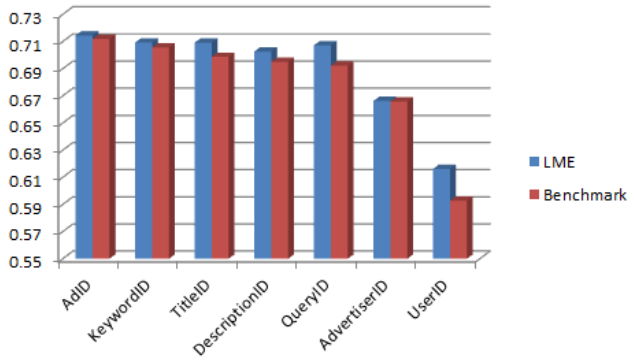

Figure 6. LME x Benchmark AUC comparison

Another reason to use the LME feature instead of raw averages was to gather the low confidence likelihoods around the same value, making it easier for the training method to identify these values and apply the shrinkage procedure. Table 4 presents this comparison. It compares some rank variation statistics between the LME and benchmark ad feature when the AdID have less than 6 previous occurrences. The most clickable variable is ranked 1 and the worst clickable instance 20,297,594. Their scores measured in AUC are about the same, but the LME places all of them closer: around the 7,652,317 position with a standard deviation of 641,480 positions. The raw maximum likelihood benchmark places them around the 7,962,020 position with a standard deviation of 4,081,248.

Table 4. LME x Benchmark rank variation for ads with history lower than 5. Rank 1 is best and 20,29,759 worst. Total of 1,207,387 instances

Method	Mean	STD	Min	Max
LME	7,652,317	641,480	333,855	9,288,149
Raw ML	7,962,020	4,081,248	1,067	19,828,987

5.1.4 Cross-product features

Cross-product features were made by combining categorical variables of large domain (all IDs) and the ones with little domain (Gender, Age, Depth, Position). These new variables were also used in the likelihood calculations as discussed previously.

Cross-products between two variables are made by collapsing their ids into a single composite id, e.g., AdID_1 (ad with ID 1) and Gender_1 (Gender with value 1) will produce the new composite id "AdID_1_Gender_1" which is different from the id produced by the cross-product of AdID_1 and Gender_2 ("AdID_1_Gender_2").

5.1.5 Feature List

Table 5 presents all calculated features, and the method used for calculating each. An underscore is used to indicate a cross-product between features. The features are grouped by the base feature used to produce it (Origin).

These features were inspired on [11], but instead of using decision rules, we decided to create cross-products and let the training method express that rule with coefficients.

Our motivation for generating such cross-product features was to get a glimpse of the most specific measurement for each instance (e.g., likelihood of a male teenager user when shown ad X at position Y …), and at the same time have more generic features to be used when that context didn't have enough history. That decision was left up to the training algorithm to detect once the low confidence values were placed around each other by the LME calculation.

Table 5. List of calculated features

Origin	Method	Features
AdID	LME	AdID, AdID_Gender, AdID_Gender_Age, AdID_Pos, AdID_Pos_Dep, AdID_Pos_Dep_Gender
DisplayURL	LME	DisplayURL, DisplayURL_Gender, DisplayURL_Gender_Age, DisplayURL_Pos, DisplayURL_Pos_Dep, DisplayURL_Pos_Dep_Gender
AdvertiserID	LME	AdvertiserID, AdvertiserID_Gender, AdvertiserID_Gender_Age, AdvertiserID_Pos, AdvertiserID_Pos_Dep, AdvertiserID_Pos_Dep_Gender
KeywordID	LME	KeywordID, KeywordID_Gender, KeywordID_Gender_Age, KeywordID_Pos, KeywordID_Pos_Dep, KeywordID_Pos_Dep_Gender
KeywordID Tokens	VW	KeywordTokens, KeywordTokens_Age, KeywordTokens_Gender, KeywordTokens_Pos_Dep_Gender
TitleID	LME	TitleID, TitleID_Gender, TitleID_Gender_Age, TitleID_Pos, TitleID_Pos_Dep, TitleID_Pos_Dep_Gender
TitleID Tokens	VW	TitleTokens, TitleTokens_Age, TitleTokens_Gender, TitleTokens_Pos_Dep_Gender
DescriptionID	LME	DescriptionID, DescriptionID_Gender, DescriptionID_Gender_Age, DescriptionID_Pos, DescriptionID_Pos_Dep, DescriptionID_Pos_Dep_Gender
DescriptionID Tokens	VW	DescriptionTokens, DescriptionTokens_Age, DescriptionTokens_Gender, DescriptionTokens_Pos_Dep_Gender
UserID	LME	UserID, UserID_Pos, UserID_Pos_Depth
QueryID	LME	QueryID, QueryID_Gender, QueryID_Gender_Age, QueryID_Pos, QueryID_Pos_Dep, QueryID_Pos_Dep_Gender
QueryID Tokens	VW	QueryTokens, QueryTokens_Age, QueryTokens_Gender, QueryTokens_Pos_Dep_Gender

5.2 Training Method

After creating the statistical features as described above, we trained our models with GBM. For this training, we built the features with tr1 subset only (Figure 3) and used them on tr2 subset for training and validation.

Before training, we exploded tr2 subset creating one instance per impression (i.e., click instances positive and the remainder negative). After this, the GBM training had about 20 million training instances of binary outputs.

That training was made in three flavors:

1. Using a sampling of four millions instances and the adaboost distribution, once it optimizes AUC [12] (Birutas's team).
2. With a downsampling of two millions with a proportion of one positive to one negative sample and used the Bernoulli distribution (Birutas's team).
3. Same sampling as before but with a different set of features and instances of the dataset to train on (Team DL).

The final model was an ensemble of them all.

5.3 Model Ensemble

The ensemble was introduced in our model a few days before the competition ended. At that time it wasn't possible to do a proper ensemble because we lost some internal data and had only the submitted data at reach. There wasn't enough time to generate it again.

Because of these constraints the ensemble was done using only the submission files for our models. We calculated the ranks of each prediction, 1 being the most likely to be clicked and 20,297,594 the most unlikely one. The final model was just a simple average of these ranks. It is very likely that this procedure produced a suboptimal ensemble and probably impacted our final standings. The results of this ensemble will be presented in the results section.

5.4 Implementation Details

As mentioned in the introduction, one of challenges of this task was to deal with a fairly large datasets. To handle it, we had to constantly use file-backed solutions and do some sampling. Actions we took to work this out:

- We always processed the data in chunks. In R we used the ff package [16] , and always processed it in chunks. As for VW, it is already an on-line learning platform with native chunk processing.
- LME requires loading the whole data in memory, and for this dataset it was not feasible. Our solution was to build a table with the raw counts of click and impression whenever we want to calculate a statistical feature with LME. The effect of this method was to group all occurrences of each id (composite included) into a single instance and use weights for each instance when training.
- For the GBM training we couldn't load all data into memory, so we sampled it. We did a uniform sampling of 4 million positive instances and a downsampling of the negative instances.

6. RESULTS

Figure 7 shows the milestones of our solution. As we built the features described in section 5, our score was slowly evolving.

As we can see in Figure 7, our first model was a single feature model, analogous to the advertiser benchmark, which scored 0.69. Then we made small adjustments until we beat the benchmark with a 0.72 AUC score. Next, we built more single features (no cross-product) and did a linear regression with them, reaching 0.74. The next milestone was very similar, except that the features were ensembled with GBM. It scored about 0.78. Finally we included query tokens features (bag of words) as a proof of

concept and the score went up again. So we added all cross-products and reached over 0.79. After this, we merged our teams (Birutas & Team DL) and ensembled our models, achieving 0.8. In our final attempt, we could only train a version of our imputated model once, without any parameter tuning, and achieved our final score of 0.80166.

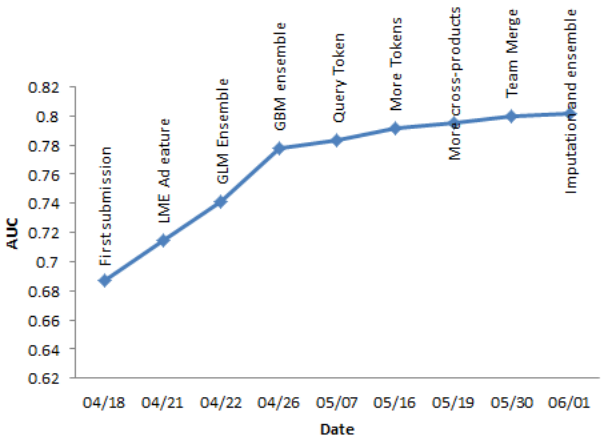

Figure 7. Score evolution

Our procedure generalizes well since the best solution has 0.80166 AUC score on private leaderboard (unknown during competition) against 0.79808 in public leaderboard (known during competition), a difference consistent with the one seem on most leading teams as we can see in Table 6.

Table 6. Final results

Rank	Team	Public Score	Final Score
1	Catch up	0.80697	0.80893
2	Opera Solutions	0.80524	0.80824
3	Steffen Rendle	0.79857	0.80178
4	Birutas & Team DL	0.79872	0.80166
5	UCenter	0.79801	0.79995

7. DISCUSSION

In this work, we provide a simple and efficient approach to predict CTR. Our solution was to create simple and composite statistical features, as described in section 5 and use them to train a GBM model. The authors of [11] suggest that CTR is influenced by an ads' content and its position. They propose an approach where the probability of an ad being clicked is calculated as a function of its position and its content independently and then multiplied to find the joint probability given both position and content. Our method was loosely based on the proposed method. We took into account not just the context-based variables (position and depth) and ad-based features but also user-based ones. We create several statistical features for each group (context, content and user) and then cross-product features were created mixing these three groups (Table 5). These cross-products features were the equivalent (not equal) in our work to the product of probabilities proposed in [11].

To calculate the likelihood of statistical features either simple or composite, we use VW and LME, mostly the latter (Table 5). As described in [6]. LME handles well large categorical variables with many values and data with low confidence by doing shrinkage on this data. The gain of using LME over raw likelihood of cross-products is expected to be greater than the gain for single features (red bars in Figure 6). That's because cross-product features in general have larger domain than the simple features. In addition to LME, we used VW to create statistical features based on bag of words and to do imputation. We choose VW for this task because it can handle huge datasets, and it has a very flexible input format.

Finally, we trained our models with GBM using adaboost and bernoulli distributions. Some of the models weren't trained with the same samples, and due to the deadline we couldn't generate features to do a proper ensemble, instead using a simple arithmetic mean of the instances' rank. The results for doing a proper ensemble using a good ML technique is expected to be greater.

8. CONCLUSION

Our proposed solution was quite simple and yet efficient. We have shown that with simple feature engineering and a good training algorithm it is possible to achieve high precision CTR prediction. The most important steps were choosing a reliable dataset for training, creating good composite features and handling large categorical variables with low average history. Some improvement was also achieved with the imputation procedure for unknown users. And finally, to sum it up, choosing a classification algorithm capable of discovering the relationship between the features was crucial to have some accuracy in predicting sessions for new users and advertisements.

9. ACKNOWLEDGMENTS

We would like to thank the organizers of this year's KDD Cup competition as well as Tecent for making the dataset available. We would also like to acknowledge DTI Sistemas for helping us to present at the KDD workshop.

10. REFERENCES

[1] Yanzhi Niu, Yi Wang, Gordon Sun, Aden Yue, Brian Dalessandro, Claudia Perlich, Ben Hamner "The Tencent Dataset and KDD-Cup'12". KDD-Cup Workshop, 2012.

[2] John Langford, Alekh Agarwal, Miroslav Dudik, Daniel Hsu, Nikos Karampatziakis, Olivier Chapelle, Paul Mineiro, Matt Hoffman, Jake Hofman, Sudarshan Lamkhede, Shubham Chopra, Ariel Faigon, Lihong Li, Gordon Rios, and Alex Strehl. 2012. Vowpal Wabbit a fast out-of-core learning system. https://github.com/JohnLangford/vowpal_wabbit/wiki.

[3] Douglas Bates, Martin Maechler and Ben Bolker. 2012. LME4: Linear mixed-effects models using S4 classes. http://cran.r-project.org/web/packages/lme4//index.html.

[4] Greg Ridgeway. gbm: Generalized Boosted Regression Models. http://cran.r-project.org/web/packages/gbm/index.html, 2012

[5] Salford Systems. TreeNet® Gradient Boosting. http://www.salford-systems.com/en/products/treenet, 2012

[6] Eugene Demidenko. Mixed Models: Theory and Applications. http://www.dartmouth.edu/~eugened/index.php?section=summary_points, 2012

[7] Laird, N. M. & J. H. Ware. 1982. "Random-Effects Models for Longitudinal Data." Biometrics 38:963—974.

[8] Greg Ridgeway. Generalized Boosted Models: A guide to the gbm package, 2007

[9] Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. Reitz Lecture, 2001

[10] Tom Fawcett. ROC Graphs: Notes and Practical Considerations for Researchers. 2004.

[11] Krzysztof Dembczy´ nski, Wojciech Kotłowski, Dawid Weiss. Predicting Ads' Click-Through Rate with Decision Rules. 2008.

[12] Cynthia Rudin, Robert E. Schapire. Margin-based Ranking and an Equivalence between AdaBoost and RankBoost. The Journal of Machine Learning Research. p 2193-2232, vol. 10, 2009.

[13] Tecent. http://www.tencent.com/zh-cn/index.shtml. 2012

[14] R. The R Project for Statistical Computing. http://www.r-project.org/, 2012

[15] KDD Cup 2012 evaluation Metric. https://www.kddcup2012.org/c/kddcup2012-track2/details/Evaluation, 2012

[16] Daniel Adler, Christian Gläser, Oleg Nenadic, Jens Oehlschlägel, Walter Zucchini. ff: memory-efficient storage of large data on disk and fast access functions. http://cran.r-project.org/web/packages/ff/index.html, 2012