Linear combination of random forests for the Relevance Prediction Challenge

Michael Figurnov
michael@figurnov.ru

Alexander Kirillov
arhipisk@gmail.com

Lomonosov Moscow State University,
Faculty of Computational Mathematics and Cybernetics
Moscow, Russia

ABSTRACT

This paper describes our solution of the Relevance Prediction Challenge. Our team ranked second in the contest. The goal of the contest is to predict relevance of URLs for search queries using search log. First we present the features we used. Our features are based on different assumptions about the properties of the dataset. Then we describe our machine learning technique. We trained multiple random forests on samples generated using the pair-wise approach. The resulting solution is a linear blend of the response vectors of the forests. The solution shows good performance and is likely not overfit.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval Models

General Terms

Algorithms, Experimentation

Keywords

Clicks, relevance prediction, pair-wise approach, web search

1. INTRODUCTION

In this paper we describe our solution of the Relevance Prediction Challenge contest [12] held by Yandex. The task of the contest is to predict relevance using data collected on user behavior. Our solution received the 2$^{\text{nd}}$ place award among 85 teams which participated in the Challenge.

Problem statement:

1. Dataset of the problem is a search log. Each line of the log represents a user action in a session, either a search query or a click on an URL. For each action the time passed from the beginning of the session is specified. For a search query the identifier of the query text, the region of the user and a ranked list of 10 URLs shown to the user are also specified. For clicks an identifier of the clicked URL is specified.
2. Dataset is completely anonymous. Sessions, queries and URLs are labeled by numeric identifiers. Time is specified in undisclosed units of time.
3. Dataset is rather large. The search log consists of 340.8 mln user actions. Its size is 16.4 GB.
4. Some of the triples (query, region, URL) are labeled with a binary relevance score: 0 if the document referenced by URL is irrelevant to the query in the region, 1 if it is relevant. The labels were assigned by the company's experts (assessors).
5. We need to solve a 'learning-to-rank' problem. For each of the specified test pairs (query, region) we need to rank all URLs shown by the search engine by the probability of them being relevant in the descending order.
6. The measure of a solution's performance is calculated as follows. From each test pair we delete URLs for which labels are not specified and calculate AUC [7] (Area Under Curve) measure. The measure of the solution's performance, denoted by Q, is the average value of AUC over all test pairs. In this contest Q should be maximized.

In our opinion, the key challenges in this problem are:

- Large size of the dataset.
- Noisiness of the dataset.
- Nonstandard performance measure (average AUC).

2. DESCRIPTION OF FEATURES WE USED

We used the following groups of features:

- Based on URL's presence in a ranked list of 10 (*'show'*).
- Based on clicks.
- Based on time.

We formed two sets of features. In one set the region of a query is taken into account, in another set it is ignored.

Our features are calculated for a pair (q, u), where q is query and region in one set and only query in another set and u is an URL. Most features are calculated using the idea of "score": if an URL satisfies the condition, then the respective score is increased by the specified amount. After calculation, all features are divided by total number of shows of the URL in the query.

2.1 Features based on shows

1. $(11 - i)$ for a show of the URL on ith position.
2. 10 features: the ith feature is the number of shows of the URL on ith position.

2.2 Features based on clicks

This group of features is calculated for an URL u and a query q. We consider all clicks between the query q and the next query or the end of the session as referring to the query q (including clicks on the URLs not present in the list of 10 of the query q[1]).

The main hypothesis behind this group:

- The more clicks on the URL, the more relevant it is.
- Last clicked URL of the query is more relevant than the others, since it is more likely that user have found information he needed after clicking that URL.
- If we assume that user was looking for the same information during the entire session, the last clicked URL in the entire session is likely to be the most relevant one.
- User understands better what he wants to find during the session, therefore clicks closer to the end of query are more relevant.

3. 1 for a click.
4. 1 for the last click in the query.
5. 1 for the last click in the last query of the session.
6. 1 for the last click in the session (possibly not in the last query of the session)
7. 1 for each query with at least one click on u.
8. 1 for each query where u was clicked more than once.
9. Number of clicks in the query before the click on u.
10. 1 for the first click in the session.
11. 1 for the first click in the query.
12. 1 for the first click in the query, if there was more than one click in the query.
13. Total number of clicks in the queries where u was clicked.
14. Total number of clicks in the queries where u was shown.
15. i for the ith click in the query.
16. (number of clicks in the query$-(i-1)$) for the ith click in the query.
17. $\frac{10 \cdot i}{\text{number of clicks in the query}}$ for the ith click in the query.
18. $\left(10 - \frac{10 \cdot (i-1)}{\text{number of clicks in the query}}\right)$ for the ith click in the query.

The next group of features is calculated for sessions with the query q. We consider clicks on URL u in all queries q'. The hypothesis behind this group is that users usually look for the same information during the entire session, so we can assume that all clicks are relevant to all queries of the session.

19. 1 for a click.
20. 1 for a click in the query q and in queries q' before it (in chronological order).
21. 1 for the last click in query.
22. 1 for a click in the last query of the session.

2.3 Features based on time

This group of features is calculated for URL u that was returned for query q. We define the time of click as the time difference between the click and the next action.

Main ideas and problems of this group:

- The time of the last click in the last query of the session is undefined.
- Sometimes users first click all interesting URLs and only then read the documents. In this case the time of the click does not relate to its relevance.
- Users have different speed of browsing. That's why we divide the time of click by the total duration of session.
- In all features of this group we sum the time of click on u divided by the duration of the session. The difference is in the way of accounting for the last clicks in the queries.

23. Time of click on u divided by the time of the session for non-last clicks in the query.
24. Time of click on u divided by the time of the session for all clicks with defined time.
25. Time of click on u divided by the time of the session. For clicks with undefined time we use the average time of all other clicks in the query.
26. Time of click on u divided by the time of the session. For last clicks in query we use the average time of all other clicks in this query.
27. Time of click on u divided by the time of the session. For clicks with undefined time we use the average time of all clicks in the query with defined time.
28. Time of click on u divided by the time of session. For last clicks in query we use the average time of all clicks in this query with defined time.

The next two features are based on the absolute time of the clicks.

[1]There are such clicks in the dataset. We are not sure what they represent, but ignoring them decreases the solution's performance.

29. Time of click on u for non-last clicks in query.

30. Time of click on u for all clicks with defined time.

The last four features are based on the assumption that if the time of the click is greater than the average, the URL is relevant to the query.

31. 1 for non-last click in the query with time greater than the average of all non-last clicks in the log.

32. 1 for non-last click in the query with time greater than the average of all clicks in this query with defined time.

33. Similarly to feature 31, but also add 1 for the last clicks in the query with time greater that the average of all last clicks in the log.

34. Similarly to feature 32, but also add 1 for the last clicks in the query with time greater that the average of all last clicks in this query with defined time.

3. PARTITIONING OF THE TRAINING SET

We have the original training set

$$S = \{(s_q, s_r, s_u, s_{rel})\},$$

where s_q is query, s_r is region, s_u is URL and s_{rel} is the label (0 if the URL is irrelevant to the query in the region, 1 if it is relevant). The label is the target variable.

First, we delete the queries for which all elements have the same labels from S:

$$S' = \left\{ s \in S \,\middle|\, \exists s' \in S\colon\ (s'_q = s_q) \wedge (s'_{rel} = 1 - s_{rel}) \right\}.$$

Then we partition S' into the learning set A and the validation set B using algorithm 1. V is a solution's parameter; rand is a function returning random element of a set.

Properties of the partition:

1. There are no queries in which all URLs have equal relevance.

2. Elements belonging to the same request are all either in the learning set or in the validation set.

3. Size of the validation set is at least V.

Algorithm 1 Partitioning of the training set

```
Require: $S'$, $V \in \mathbb{N}$, $V < |S'|$
Ensure: $A, B\colon S' = A \sqcup B$
 1: $A = S'$
 2: $B = \emptyset$
 3: while $|B| < V$ do
 4:     $\hat{q} = \text{rand}\,\{a_q \mid a \in A\}$
 5:     $X = \{a \in A \mid a_q = \hat{q}\}$
 6:     $A = A \setminus X, B = B \cup X$
 7: end while
```

4. RANDOM FORESTS

We used implementation of regression random forests [1] from `randomForest 4.6-2` package [6] for `R 2.14.0` [9].

Object-features matrix is generated by using pair-wise approach (a similar idea can be found in [14]). For each query and region we enumerate all pairs of URLs. Features of a pair of URLs are all features of both URLs. The target variable is the difference between relevance of the second and the first URLs (-1, 0 or 1). In fact, we set a problem of predicting which of the two URLs is more relevant. We create two matrices: one for the set of features taking the region into account and one for the set of features ignoring the region.

We build 7 subsets of features:

1. **core** = {1, 2.10, 3, 6, 8, 17, 20, 21, 30, 34}
2. **core** ∪ {4, 5, 7}
3. **core** ∪ {2.1, 2.2, . . . , 2.9}
4. **core** ∪ {23, 24, 25, 26}
5. **core** ∪ {19, 22}
6. **core** ∪ {27, 28, 29, 31, 32, 33}
7. **core** ∪ {9, 10, 11, 12, 13, 14, 15, 16, 18}

We will briefly describe the method of deriving such subsets. First, we excluded features that have very small absolute correlation with the target variable. Then we divided the features into 6 groups according to the underlying hypotheses and the principle of calculation. In each group we looked for 1-2 best features according to their performance in linear combination with feature 1 (see algorithm 2). These features compose the **core** subset. Other subsets were produced by uniting the **core** subset with each of the original groups. This method is not well-founded, since performance of a feature in linear combination is not necessarily related to its importance for the whole solution. However, it worked in our case.

After that we create random forests for the learning set A[2]. For each matrix we create 2 random forests for each of the 7 features subsets, a total of 28 random forests. Then each forest predicts relevance of pairs from the validation set B and the test set.

Now we need to convert each random forest's pair-wise relevance scores for validation and test into the URLs relevance scores. We write down all the scores for a fixed query and region in a matrix of dimension $N \times N$, where N is the number of URLs, and (i, j)th elements is the response of the random forest for ith and jth URLs. We assume that the relevance of the ith URL is the sum of the elements in the ith column.

5. LINEAR COMBINATION

By setting a fixed order of triples (query, region, URL) for the validation and the test sets, we obtained the random forests' response vectors: $g_1, \ldots, g_n$ for the validation set and $h_1, \ldots, h_n$ for the test set, $n = 28$. Let $Q(g) \in [0, 1]$ be the problem's measure of performance for the response vector g (i.e. the performance measure of the solution where

[2]Settings of the random forests: `sampsize=10000, ntree=500, mtry=2, forest=TRUE`

we rank the URLs in each query and region by descending of the corresponding elements of g). Algorithm 2 tunes a linear combination of the form $g = c_1g_1 + \dots + c_ng_n, c_i \geq 0$. We used the parameters $\alpha = 17,\ \beta = 10^{-9}$. The response vector for the test set is calculated as $h = c_1h_1 + \dots + c_nh_n$.

Algorithm 2 Tuning of the linear combination

```
Require: $g_1, \dots, g_n,\ Q(g),\ \alpha \in \mathbb{N},\ \beta \in \mathbb{R}^+$
Ensure: $\vec{c} = \{c_1, \dots, c_n\}$
 1: $c_i = 0,\ i = 1, \dots, n$
 2: for $K = \{1, 1/2, \dots 1/2^\alpha\}$ do
 3:   loop
 4:     $Q_0 = Q(c_1g_1 + \dots + c_ng_n)$
 5:     for $i = 1, \dots, n$ do
 6:       $Q_i = Q(c_1g_1 + \dots + (c_i + K)g_i + \dots + c_ng_n)$
 7:     end for
 8:     $k = \arg\max_{i=1,\dots,n} Q_i$
 9:     if $Q_k - Q_0 \geq \beta$ then
10:       $c_k = c_k + K$
11:     else
12:       go to the next $K$
13:     end if
14:   end loop
15: end for
```

The idea of the coordinate descend method with decreasing step is applied. We were inspired to use a linear combination of the response vectors by technology "LENKOR" described in [2]. Another part of the technology is the "deformation" of the combination, i.e. applying non-linear functions such as polynomials and square roots to parts of the combination. We didn't try using it because of the time constraints of the contest, but we think that it might increase the performance.

Advantages of the algorithm:

1. Only $\approx$ 20% of original vectors had non-zero coefficients in the linear combination in our experiments. It means that we really need to calculate the response vectors for the test set only for a portion of the random forests, allowing to significantly reduce the time of building the solution.

2. The algorithm directly optimizes the problem's performance measure.

3. The algorithm is quite fast. Tuning a linear combination of the 28 response vectors takes about a minute[3].

The described solution was built 3 times: for $V = 4000$, 8000, 12000. The construction of each solution took about 1.5 hours (random forests were executed on an Amazon Cluster Compute Eight Extra Large instance). The response vectors for the test set were divided by their maximal values and summed[4].

6. OTHER METHODS

We tried replacing random forests with Gradient Boosting [4] and RankBoost [3] methods. Comparing with random forests, these methods can directly optimize the problem's performance measure. However, their results were much worse. Random forests show excellent performance in other tasks with noisy data, such as credit scoring [11]. In our opinion, the advantage of random forests for the problem of the competition is caused by the noisiness of the dataset.

It is worth noting that features based on Bayesian approach, such as [8] and [13], show relatively poor performance in terms of the problem's performance measure ($Q \approx 0.61$ compared to $Q \approx 0.63$ of the feature #3 and $Q \approx 0.64$ of the feature #30).

7. CONCLUSIONS

The Challenge's test set was split into two subsets. One subset was used for calculating public evaluation score, second subset was used for calculating final results. Our solution showed a result of 0.657424 and 0.659914 on these subsets, respectively. This means that it has good generalization capability and is probably not overfit.

We managed to overcome the large size of the dataset by creating features that can be calculated once in a single pass. Using parallel processing, we reduced the time of the features calculation to 20 minutes on a quad core processor[5].

The pair-wise approach consists of considering pairs of documents and trying to predict which one of them is more relevant. It is often used for solving learning-to-rank problems with SVM [5]. This approach turned out to be the best among the tried ones. Optimizing the mean square error on such sample is almost equivalent to optimizing the performance measure for this Challenge (average AUC). Configuring linear combination allows selecting random forests that perform best in terms of the problem's performance measure.

We use a large feature space based on different assumptions about the properties of the dataset and user behavior models. Using random forests for various subsets of this feature space allows to create a well-performing solution from the provided noisy dataset.

8. ACKNOWLEDGEMENTS

We are very grateful to Alexander D'yakonov for useful discussions and to Alexandre Giglavy for his help in editing the manuscript. We would also like to thank the Relevance Prediction Challenge organizers for hosting an interesting competition. This work is partially supported by Russian Foundation for Basic Research, project no. 12-07-00187-a.

9. REFERENCES

[1] L. Breiman. Random forests. *Mach. Learn.*, 45:5–32, October 2001.

[2] A. D'yakonov. Two recommendation algorithms based on deformed linear combinations. In *Proceedings of the ECML-PKDD 2011 Discovery Challenge Workshop*, pages 21–27, 2011.

[3] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, December 2003.

[4] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[3]We used a computer with quad-core CPU Intel Core i7 and 12 GB RAM.

[4]We also added boosted decision trees from `gbm` package [10] into the solution for $V = 12000$. However, it seems that it did not affect the performance of the solution.

[5]The computer described in the footnote 3 was used. Source search log was previously converted into a binary format.

[5] T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[6] A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2(3):18–22, 2002.

[7] C. X. Ling, J. Huang, and H. Zhang. AUC: a statistically consistent and more discriminating measure than accuracy. In *Proceedings of 18th international conference on artificial intelligence (IJCAI-2003*, pages 519–524, 2003.

[8] C. Liu, F. Guo, and C. Faloutsos. BBM: bayesian browsing model from petabyte-scale data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 537–546, New York, NY, USA, 2009. ACM.

[9] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[10] G. Ridgeway. *gbm: Generalized Boosted Regression Models*, 2010. R package version 1.6-3.1.

[11] D. Sharma. Guide to credit scoring in R. `http://cran.r-project.org/doc/contrib/Sharma-CreditScoring.pdf`, October 2009.

[12] Yandex. Relevance prediction challenge website. `http://imat-relpred.yandex.ru/en/`.

[13] Y. Zhang, D. Wang, G. Wang, W. Chen, Z. Zhang, B. Hu, and L. Zhang. Learning click models via probit bayesian inference. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 439–448, New York, NY, USA, 2010. ACM.

[14] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 287–294, New York, NY, USA, 2007. ACM.