YJRS at the NTCIR-13 OpenLiveQ Task

Tomohiro Manabe Akiomi Nishida Sumio Fujita
Yahoo Japan Corporation
{tomanabe, anishida, sufujita}@yahoo-corp.jp

ABSTRACT

We participated in the NTCIR-13 OpenLiveQ Task in view of improving the baseline method, which is a linear combination of 77 features with different weights. We added three settings of extended BM25F as additional features, replaced the target function of weight optimization and recalculated feature weights using 5-fold cross validation. Our run achieved the best Q-measure score among 10 runs in the offline test. In the online test, our run achieved the second-best total credit among 10 runs with slight difference from the best one due to its robustness. We also checked win-loss counts of our run against other nine runs for each page view. According to the results, our run won all other runs in statistically significantly large number of page views.

Keywords

Probabilistic retrieval model; Linear combination model; QA search and retrieval; Document scoring

Team Name

YJRS

1. INTRODUCTION

In this paper, we report our work at the NTCIR-13 OpenLiveQ Task. For detailed description of the task and the results of other teams, please refer to the overview paper [3].

The task organizers provided us of a tool for feature extraction from the document collection and a short instruction for generating baseline rankings with the tool[1] and RankLib[2]. Our method is based on the baseline method and we applied some modification to it. We, 1) added three settings of extended BM25F as additional features, 2) replaced the target function, and 3) applied 5-fold cross validation.

In addition, we tried to tune extended BM25F parameters by the Coordinate Ascent method [4]. The attempt was not successful in view of improving offline evaluation results.

Moreover, we also tried to replace the baseline learning method, Coordinate Ascent optimization [4] of a linear combination model, by other sophisticated methods, namely Random Forests [1] and LambdaMART [10]. Unfortunately, neither attempt improved the results.

The remainder of the paper is organized as follows. In Section 2, we briefly survey related work. In Section 3, we explain our approaches in temporal order of the attempts. In Section 4 and 5, we explain the offline and online evaluation results of our runs. Finally, Section 6 concludes the paper.

[1]https://github.com/mpkato/openliveq
[2]https://sourceforge.net/p/lemur/wiki/RankLib/

2. RELATED WORK

In this section, we refer to the methods we tried to improve offline evaluation results of our runs.

2.1 Models and Machine Learning Methods

We mainly used linear combination of features as our ranking model. This is a simple weighted summation of multiple numeric features of documents and/or queries.

Only the parameters of the model are feature weights. The Coordinate Ascent method is an approach to optimize such parameters [4]. The method optimizes each parameter one-by-one. After optimizing the last parameter, it reiterates from the first parameter again. Some variations shuffle the parameter list in this time. The optimization finishes if modification of no parameter can improve the value of the objective function. To optimize a parameter, the method checks multiple smaller or bigger values of the current value and greedily updates the current value by a new value that improves the objective function.

We also tried to use more sophisticated models and learning methods, Random Forests [1] and LambdaMART [10], though they found to be not effective for the task.

2.2 Additional Features

The BM25 ranking function [7] is a simple approximation to the 2-Poisson model term weighting where within document term frequencies exhibit either an eliteness distribution, which is a distribution of terms representing the main topic of the document, or a non-elite one, which is of terms used in a general meaning. Term weighting function represents how much influence the number of term occurrences in a document affects the estimation of document relevance. Naturally, longer documents contain more term occurrences due to the large number of topics that a document may mention or to the verbosity in writing. The BM25 is able to calibrate the strength of the document length normalization by b parameter according to the assumptions behind the generation of target document collections.

The BM25F function is an extension of BM25 [7] for structured documents that can score documents containing multiple fields of different weights [8]. A document is structured with typed functional fields such as title, heading, and body text, which are all possible fields of web pages. Assigning heavier weights to title or heading field than body text, BM25F effectively ranks relevant documents higher.

Table 1: Offline evaluation results of our runs.

ID	Description	nDCG@10
5	Test run.	0.34371
10	Naive BM25F.	0.36452
16	Roughly optimized BM25F.	0.33337
25	BM25F, roughly optimized with CA where $n = 3$.	0.33341
28	BM25F, roughly optimized with CA where $n = 3$ and $sf = 0.8$.	0.34316
38	Baseline + naive BM25F.	0.37965
48	Five-fold cross validation.	0.37965
50	Five-fold cross validation (fix).	0.41157
66	Five-fold cross validation (2).	0.40167
71	8foldCV_RandomForest	0.37091
77	Baseline + multiple BM25F features.	0.39637
82	8foldCV_LambdaMART	0.38087
86	Baseline + multiple BM25F features + nDCG@10.	**0.41894**

We extended the original BM25F adopting field-dependent b parameter values. Such an extension is referred to by a paper for example [5]. Denoting a field category by f, the BM25F score of a document D over a query Q (both D and Q are bags of term occurrences) is:

$$\sum_{t\in Q} \frac{w(t,D)}{k_1 + w(t,D)} \log \frac{N - \mathrm{df}(t) + 0.5}{\mathrm{df}(t) + 0.5} \ ,$$

$$w(t,D) = \sum_{f\in D} \frac{\mathrm{tf}(t,f,D)\cdot \mathit{boost}_f}{(1-b_f) + b_f \cdot \mathrm{len}(f,D)/\mathrm{avgLen}(f)} \ ,$$

where N is the number of the all documents in collection, $\mathrm{df}(t)$ the number of documents in the collection in which t occurs, $\mathrm{tf}(t,f,D)$ the number of occurrences of t in the field f of document D, $\mathrm{len}(f,D)$ the number of all term occurrences in the field, and $\mathrm{avgLen}(f)$ the arithmetic mean of $\mathrm{len}(f,D)$ through all documents. The hyper-parameters of the model are k_1, boost_f, and b_f, and of course we can specify boost_f values which are relative weights of each field.

We used scores of extended BM25F as additional features of query-document pairs, where we regard BM25F scores from different parameter settings as different features.

3. OUR APPROACH

On Table 1, we list offline evaluation results of all our runs in nDCG@10 according to the date of submission. In this section, we explain each run in this order.

3.1 Baseline Method

First of all, we reproduced the organizer run according to their instructions (Run 5). This is a linear combination of baseline features whose weights are optimized by the Coordinate Ascent method. We used the RankLib implementation of the method and mean average precision (MAP) as its objective function. A pseudo-code of the implementation is in Algorithm 1. We used MAP instead of nDCG@10, the offline evaluation measure itself, because questions at ranks lower than 10 are also important on the greedy optimization process. The 77 baseline features include six numeric features about questions (namely answer count, page view count, their logarithmic variations, baseline rank, timestamp), three boolean features (namely open to answer or not, open to vote for the best answer or not, and already solved or not), and finally 17 text-based features of query-question pairs (such as variations of TF, IDF, TFIDF, language models, and BM25) across four textual fields of questions (namely, question text, best answer text, search result title and its snippet). The textual features are adopted from 1–15 on Table 3 of the LETOR paper [6], document length, and its logarithmic variation.

Algorithm 1: RankLib variation of Coordinate Ascent

```
Data: $initWeights, initScore$
$bestWeights, bestScore \leftarrow initWeights, initScore$
for $r = 0; r < numRestart; r++$ do
    $weights, score, fail \leftarrow initWeights, initScore, 0$
    while $fail < |weights| - 1$ do
        forall $i \in \mathrm{shuffle}([0, 1, 2, \cdots, |weights| - 1])$ do
            $orig, total, bestTotal, succ \leftarrow weights[i], 0, 0, 0$
            forall $direction \in [-1, 0, 1]$ do
                $step, total \leftarrow 0.001 * direction, 0.001 * direction$
                for $(j = 0; j < 25; j++)$ do
                    $weights[i] \leftarrow orig + total$
                    $s \leftarrow \mathrm{evaluate}(weights)$
                    if $score < s$ then
                        $score, bestTotal, succ \leftarrow s, total, 1$
                    $step, total \leftarrow 2 * step, total + step$
                if $succ = 1$ then
                    break
                $weights[i] \leftarrow orig$
            if $succ = 1$ then
                $weights[i], fail \leftarrow orig + bestTotal, 0$
            else
                $fail, weights[i] \leftarrow fail + 1, orig$
        if $score - initScore < 0.001$ then
            break
    if $bestScore < score$ then
        $bestScore, bestWeights \leftarrow score, weights$
Result: $bestWeights, bestScore$
```

Table 2: Values of parameters *boost* and b in BM25F function. Note that we set 1.2 to remaining k_1.

Field ID	Field/Feature name	*boost*	b
	Textual fields		
0	Search result title	5.0	0.0
1	Search result snippet	1.0	0.75
2	Category	3.0	0.0
3	Question text	1.0	0.75
4	Best answer text	1.0	0.75
	Numeric fields/features		
5	Baseline rank	-0.001	
6	Days passed from post	-0.001	
7	Page view count	0.001	
8	Answer count	0.1	
9	Status	1.0	

3.2 Extended BM25F Features

Since the BM25F function is different from a simple linear combination of BM25 feature of each field, it may be useful to add BM25F features into the baseline model in addition to the existing BM25 features.

Fields and other parameter values are listed on Table 2. We integrated non textual fields on top of textual fields even though they are not the target of query matching in our task.

Table 3: Fields we used in each BM25F setting.

Setting	0	1	2	3	4	5	6	7	8	9
Naive	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SERP	✓	✓	✓			✓	✓	✓	✓	✓
SERP+	✓	✓				✓				

The idea is to optimize not only query-document matching score but also non-textual query-independent features at the same time. Thus extended BM25F taking into considerations of such features as shown in the following formulae, can optimize hyper-parameters just as a learn to rank approach for IR does.

$$\sum_{t \in Q} \frac{w(t,D) + \alpha(D)}{k_1 + w(t,D) + \alpha(D)} \log \frac{N - \mathrm{df}(t) + 0.5}{\mathrm{df}(t) + 0.5} ,$$

$$\alpha(D) = \sum_{f \in D_N} v(f, D) \cdot \mathit{boost}_f$$

where, D_N is a set of numerical fields/features of a document D, $v(f, D)$ is the value of a numerical field/feature f of document D, and boost_f is a weight of f to be learned from the training data.

First, we manually annotated field weights and other parameters of BM25F and ranked the questions based only on the scores of extended BM25F (Run 10). We set 1.2 to k_1.

3.3 Tuning of Extended BM25F Parameters

Next, we tried to optimize extended BM25F parameters with the Coordinate Ascent method. We used our original implementation for this optimization since the RankLib implementation of the method is only for optimizing linear combination as is often the case in many learn to rank approaches for IR; our extended BM25F model can not be represented as a linear combination of features. To optimize the extended BM25F parameters with the Coordinate Ascent, we ordered the parameters into an array, and optimized them one-by-one by fixing the other parameters. We used MAP measures as the objective function of the optimization.

In the first step, we tried to optimize parameters one-by-one by examining a larger and a smaller value from the initial value shown in Table 2, by a predefined step size and update them by a preferred value if any (Run 16).

In the second step, we examined, for each parameter, three smaller and three larger values from the current value at once and update the current value if improved (Run 25).

Finally, we retried the second step but adopting a scaling factor of the step size (Run 28). The scaling factor is set initially to 1, and after each cycle, we scale down the factor by 0.8 . In other words, we scale down the parameter space after each cycle.

Our fine tuning of extended BM25F was not effective for improving the offline evaluation score presumably due to the over-fitting to training data. Note that the training data is based on real users' clicks while the test data is based on judgements by crowd-workers.

3.4 Linear Combination Model with BM25F

After unsuccessful tuning described in the previous subsection, we manually generated two more settings of extended BM25F. Our main idea is that, users can see only search engine result pages (SERPs) for judging whether each question deserves a click. Therefore, we trained extended BM25F settings only based on fields appearing on SERPs in the hope that it is useful for ranking questions to maximize user clicks. Our SERP-preferred setting uses 8 fields on SERPs (SERP in Table 3), and our SERP+ setting uses fields which are prominent on SERPs (SERP+ in Table 3).

We first tried to combine the baseline 77 features with our naive BM25F by the Coordinate Ascent optimization of the linear combination model (Run 38). It improved the nDCG@10 score, which shows that BM25F is still useful on top of some BM25 features. We accidentally re-submitted the same run as the previous one (Run 48).

3.5 Cross Validation

In order to validate generated linear combination models with a small fraction of data, we adopted the k-fold cross validation. We split the training queries into k chunks of the almost same size, then trained k models by using the $k - 1$ chunks each time by alternating excluding one chunk for each model. Finally, we evaluated each model by the corresponding hold out chunk. For these evaluations, we again used the training objective function, MAP measure. First we adopted the 5-fold cross validation and submitted the result of the best model upon these validations (Run 50). Because the RankLib implementation of Coordinate Ascent probabilistically shuffles the parameter order, we repeated the same procedure again (Run 66). With the 5 fold cross validation, we added remaining two settings of extended BM25F, namely SERP and SERP+, into the linear combination model (Run 77). Adding a feature should not harm the results since the optimization assign an adequate weight so that a deteriorative feature might be safely ignored by being assigned a zero weight. Nevertheless, the offline evaluation score degraded, presumably due to the probabilistic optimization method. Therefore, we continued to use the two additional extended BM25F features in the hope of doing well in online test.

3.6 Random Forests and LambdaMART

Linear combination is a simple and robust method to combine multiple features for scoring. On the other hand, more sophisticated methods have been also used for complex modeling of more data.

We replaced Coordinate Ascent with Random Forests [1] (Run 71) and with LambdaMART (Run 82), both of which are implemented in RankLib, where we optimized the training by the 8-fold cross validation. However, both modifications had no positive effect on offline evaluation scores due to small training data and the difference between training and test evaluation methods.

3.7 Objective Function

Finally, we changed the objective function of the Coordinate Ascent to nDCG@10, just same as the offline test evaluation measure (Run 86). This modification further improved our result.

3.8 Feature Weights

In table 9, we list the features' importance by evaluating runs discarding each feature one-by-one. Each row of the table shows the scores of nDCG@10, ERR@10, and Q-measure evaluation measures based on real users' click logs when the indicated feature is discarded. The measures are not compa-

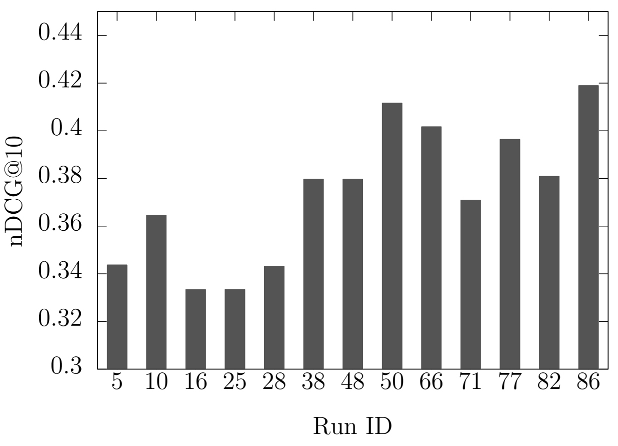

Figure 1: Offline evaluation results of our runs.

rable with the offline evaluation results based on crowdsourcing shown later in Table 5. The top 55 features are shown in ascending order of their corresponding nDCG@10 score, *i.e.* descending order of the feature impacts on nDCG@10.

As shown in this table, *page view count* and *answer count* are the most effective features against nDCG@10, which is intuitively convincing because these features should have strongly indicated the popularity of questions. For measuring the matching between question and query, four textual fields, namely SERP snippet, SERP title, best answer text, and question, found to be more or less important, as these four fields appeared in the top-10 features.

Depending on the evaluation measure, the tendencies slightly change. For example, on ERR@10, *answer count* has only 15th impact, and on Q-measure, question text features are less important as its 17th impact at the best. Our BM25F (SERP) has only 25th impact whereas BM25F (Naive), 33rd, and BM25F (SERP+), 62nd.

4. OFFLINE TEST RESULT

In this section, we explain the offline evaluation results of our best run (Run 86) comparing with the baseline and other teams' runs. As shown in Figure 1, our submission runs consistently outperformed the baseline run.

We carried out statistical significance tests of our runs where we used Student's paired t-test because of its robustness. In the test, each pair consists of nDCG@10 scores of two of our runs over a query. In Table 4, we list the p-values of the run pairs in nDCG@10. Note that we cannot calculate the p-value between the Runs 38 and 48 because the runs are identical.

According to the table, our best run (Run 86) statistically significantly outperformed ($p < 0.05$) our all other runs except Run 50. The difference between Runs 50 and 86 are: 1) the objective functions (MAP and nDCG@10, respectively) of linear optimization and 2) whether excluding the two additional extended BM25F features or not. The MAP function may be a good approximation to the nDCG@10 function and the two extended BM25F features may be not so important (See Table 9).

Table 5 lists all teams and their best runs on nDCG@10. In this table, their evaluation results on nDCG@10, ERR@10, and Q-measure averaged over the queries are also shown. The ERR is one of the most well-known evaluation measures for rankings and is particularly useful for evaluating rankings over navigational queries [2], and the Q-measure is sensitive to lower ranks and known to be highly reliable [9].

As shown on this table, our run achieved the second-best (next to OKSAT) nDCG@10, the third-best (next to OKSAT and cdlab) ERR@10, and the best Q-measure scores.

Table 6 lists the p-values of Student's paired t-test on our best run and other teams' best runs. As shown in this table, the nDCG@10 and ERR@10 measures are not sensitive enough to show statistically significant differences between the runs. On the other hand, it is worth noting that, in Q-measure, our run outperforms all other runs with statistically significant differences ($p < 0.01$).

5. ONLINE TEST RESULT

In this section, we explain the online evaluation results.

Table 7 lists the all teams, their best runs on the offline test, and the amount of credit they obtained during the online test period. The baseline runs include AS-IS, which is the current commercial search result, and N-ANS, a simple descending order ranking based on the number of answers. As shown on the table, our run obtained the second largest (next to Erler) amount of credit among the 10 runs during the online test.

We also list the win-loss page view (PV) counts of our run against all other runs in Table 7. Interestingly, our run consistently achieved more than 50% win-loss ratios against all other runs although it achieved the second-best in total credit. The minimum win-loss ratio was 53.8% (against Erler). This suggests that the Erler run won our run only in less than 50% of page views but with wider margins of credit than our run.

Table 8 lists the p-values of Student's paired t-test for the total amounts of credit and Pearson's chi-square test for win-loss counts between our best run and all other runs. According to the values, the difference in total amount of credit between our best run and Erler's run was not statistically significant. Our best run obtained the amount of credit statistically significantly larger than all other runs except Erler.

Moreover, our best run won all other runs with win page view counts of statistically significantly larger numbers ($p < 0.000005$).

6. CONCLUSIONS

In this paper, we explained our approach to the NTCIR-13 OpenLiveQ task and presented evaluation results.

We generated our runs based on the ranking by linear combinations of basic features and its modifications. Our main ideas behind the modifications include the following: (1) BM25F scores are useful as learning-to-rank features besides linear combination of BM25 scores and (2) document fields appeared on SERPs are more important than other unseen fields. Well-known techniques, such as Coordinate Ascent for learn to rank and optimization by cross validation, found to be useful as well.

Our run achieved the second-best nDCG@10, the third-best ERR@10, and the best Q-measure scores among eight runs in the offline test. And the last but not least, our run obtained the second-largest total amount of credit and consistently won other nine runs in more than a half of PVs in the online test.

Table 4: Resulting p-values of Student's paired t-test among our runs in nDCG@10.

	5	10	16	25	28	38	48	50	66	71	77	82	86
5		0.0641	0.3639	0.4017	0.9679	0.0183	0.0183	0.0000	0.0004	0.1053	0.0011	0.0260	0.0000
10	0.0641		0.0111	0.0103	0.0912	0.2722	0.2722	0.0008	0.0136	0.7190	0.0225	0.3004	0.0002
16	0.3639	0.0111		0.9906	0.1870	0.0004	0.0004	0.0000	0.0000	0.0117	0.0000	0.0003	0.0000
25	0.4017	0.0103	0.9906		0.1228	0.0005	0.0005	0.0000	0.0000	0.0125	0.0000	0.0003	0.0000
28	0.9679	0.0912	0.1870	0.1228		0.0047	0.0047	0.0000	0.0000	0.0768	0.0001	0.0077	0.0000
38	0.0183	0.2722	0.0004	0.0005	0.0047			0.0001	0.0042	0.5323	0.0480	0.9200	0.0001
48	0.0183	0.2722	0.0004	0.0005	0.0047			0.0001	0.0042	0.5323	0.0480	0.9200	0.0001
50	0.0000	0.0008	0.0000	0.0000	0.0000	0.0001	0.0001		0.1417	0.0051	0.0037	0.0181	0.2075
66	0.0004	0.0136	0.0000	0.0000	0.0000	0.0042	0.0042	0.1417		0.0245	0.4406	0.0935	0.0372
71	0.1053	0.7190	0.0117	0.0125	0.0768	0.5323	0.5323	0.0051	0.0245		0.0909	0.2822	0.0013
77	0.0011	0.0225	0.0000	0.0000	0.0001	0.0480	0.0480	0.0037	0.4406	0.0909		0.2534	0.0007
82	0.0260	0.3004	0.0003	0.0003	0.0077	0.9200	0.9200	0.0181	0.0935	0.2822	0.2534		0.0052
86	0.0000	0.0002	0.0000	0.0000	0.0000	0.0001	0.0001	0.2075	0.0372	0.0013	0.0007	0.0052	

Table 5: The best run of all teams in nDCG@10, and their corresponding ERR@10 and Q-measure.

ID	Team	nDCG@10	ERR@10	Q-measure
7	ORG	0.41328	0.24942	0.70247
18	KUIDL	0.35788	0.21967	0.67360
19	TUA1	0.37670	0.23338	0.69432
22	Erler	0.40566	0.24507	0.70657
54	SLOLQ	0.31908	0.19760	0.65563
83	cdlab	0.41800	0.26381	0.69732
86	YJRS	0.41894	0.25391	0.71339
88	OKSAT	0.44471	0.27605	0.69980

Table 6: Resulting p-values of Student's paired t-test between our run and each of other teams' runs.

ID	Team	nDCG@10	ERR@10	Q-measure
7	ORG	0.50645	0.67545	0.00002
18	KUIDL	0.00003	0.07254	0.00000
19	TUA1	0.00052	0.13913	0.00003
22	Erler	0.07422	0.39101	0.00000
54	SLOLQ	0.00000	0.00233	0.00000
83	cdlab	0.94031	0.53139	0.00000
88	OKSAT	0.09991	0.16356	0.00616

Table 7: All runs, their total credit in online test and win-loss counts of our run against each run.

ID	Team	Total credit	Win PV	Loss PV
7	ORG	21301.1	37010	28496
18	KUIDL	16935.8	47498	24552
19	TUA1	17285.2	46083	24772
22	Erler	22345.7	35912	30779
54	SLOLQ	14892.0	50273	20984
83	cdlab	19961.9	40529	31465
86	YJRS	22307.6		
88	OKSAT	16597.7	46958	25169
-	AS-IS	14037.1	52736	19832
-	N-ANS	18917.5	43452	24747

7. REFERENCES

[1] L. Breiman. Random Forests. *Mach. Learn.*, 45(1):5–32, 2001.

Table 8: Resulting p-values of Student's paired t-test for total credit and Pearson's chi-square test for win-loss counts of our best run against other runs.

ID	Team	Total credit	Win-loss PV
7	ORG	0.00000	0.00000
18	KUIDL	0.00000	0.00000
19	TUA1	0.00000	0.00000
22	Erler	0.90975	0.00000
54	SLOLQ	0.00000	0.00000
83	cdlab	0.00000	0.00000
88	OKSAT	0.00000	0.00000
-	AS-IS	0.00000	0.00000
-	N-ANS	0.00000	0.00000

[2] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected Reciprocal Rank for graded relevance. In *CIKM*, pages 621–630, 2009.

[3] M. P. Kato, T. Yamamoto, T. Manabe, A. Nishida, and S. Fujita. Overview of the NTCIR-13 OpenLiveQ task. In *NTCIR*, 2017.

[4] D. Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Inf. Retr.*, 10(3):257–274, 2007.

[5] J. Pérez-Iglesias, J. R. Pérez-Agüera, V. Fresno, and Y. Z. Feinstein. Integrating the probabilistic models BM25/BM25F into Lucene. *CoRR*, abs/0911.5046, 2009.

[6] T. Qin, T.-Y. Liu, J. Xu, and H. Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retr.*, 13, 2010.

[7] S. Robertson, S. Walker, M. M. Hancock-Beaulieu, M. Gatford, and A. Payne. Okapi at TREC–4. In *TREC*, pages 73–96, 1996.

[8] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *CIKM*, pages 42–49, 2004.

[9] T. Sakai. Ranking the NTCIR systems based on multigrade relevance. In *AIRS*, pages 251–262, 2005.

[10] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting Boosting for information retrieval measures. *Inf. Retr.*, 13(3):254–270, 2010.

Table 9: List of features used in our linear combination model. For each feature, we show evaluation scores in case that we set 0.0 to its weight. Features are sorted in ascending order of their corresponding nDCG@10 score (in other words, in descending order of their impacts to nDCG@10 score). We show only top-55 features.

Rank	Feature name	Target field/feature	nDCG@10	ERR@10	Q-measure
1	log_view_num	Page view count	0.19299	0.31318	0.34532
2	log_answer_num	Answer count	0.20867	0.34044	0.35251
3	answer_num	Answer count	0.20910	0.33526	0.35270
4	log_norm_tf_sum	SERP snippet	0.21007	0.32969	0.35342
5	updated_at	Days passed	0.21112	0.33588	0.35340
6	dlen	SERP title	0.21119	0.33213	0.35333
7	lm_jm	Best answer text	0.21192	0.33405	0.35350
8	lm_abs	Best answer text	0.21228	0.33417	0.35346
9	log_dlen	Question text	0.21308	0.33442	0.35383
10	view_num	Page view count	0.21332	0.32962	0.35344
11	tf_sum	SERP title	0.21332	0.32962	0.35344
12	lm_abs	SERP title	0.21344	0.33466	0.35355
13	dlen	Question text	0.21356	0.33538	0.35395
14	lm_dir	SERP snippet	0.21373	0.33468	0.35377
15	log_bm25	SERP snippet	0.21387	0.33513	0.35383
16	rank	Baseline rank	0.21388	0.33474	0.35405
17	idf_sum	Best answer text	0.21393	0.33468	0.35333
18	icf_sum	Question text	0.21407	0.33575	0.35371
19	tfidf_sum	Best answer text	0.21417	0.33516	0.35370
20	log_bm25	SERP title	0.21421	0.33556	0.35388
21	tf_in_idf_sum	SERP title	0.21422	0.33628	0.35385
22	tfidf_sum	Question text	0.21425	0.33644	0.35396
23	log_norm_tf_sum	Best answer text	0.21426	0.33598	0.35385
24	log_tfidf_sum	SERP snippet	0.21427	0.33601	0.35387
25	bm25f	SERP	0.21428	0.33569	0.35387
26	log_bm25	Best answer text	0.21435	0.33546	0.35362
27	bm25	SERP title	0.21437	0.33608	0.35394
28	log_tfidf_sum	Best answer text	0.21439	0.33612	0.35388
29	log_bm25	Question text	0.21441	0.33568	0.35387
30	bm25	SERP snippet	0.21441	0.33646	0.35386
31	log_tf_sum	SERP title	0.21441	0.33655	0.35385
32	tfidf_sum	SERP snippet	0.21442	0.33647	0.35386
33	bm25f	Naive	0.21444	0.33654	0.35383
34	is_solved	Already solved	0.21446	0.33748	0.35419
35	dlen	Best answer text	0.21446	0.33543	0.35384
36	log_idf_sum	Question text	0.21448	0.33654	0.35384
37	log_tfidf_sum	Question text	0.21450	0.33617	0.35388
38	icf_sum	SERP title	0.21455	0.33555	0.35361
39	idf_sum	SERP title	0.21455	0.33555	0.35362
40	lm_dir	Best answer text	0.21456	0.33644	0.35387
41	tf_in_idf_sum	Best answer text	0.21459	0.33638	0.35388
42	idf_sum	SERP snippet	0.21460	0.33646	0.35385
43	log_idf_sum	Best answer text	0.21461	0.33636	0.35386
44	bm25	Best answer text	0.21462	0.33654	0.35388
45	log_idf_sum	SERP snippet	0.21462	0.33653	0.35386
46	lm_jm	SERP snippet	0.21462	0.33653	0.35386
47	lm_jm	SERP title	0.21465	0.33664	0.35388
48	is_vote	Open to vote	0.21465	0.33663	0.35389
49	icf_sum	SERP snippet	0.21468	0.33655	0.35388
50	lm_abs	SERP snippet	0.21468	0.33655	0.35388
51	bm25	Question text	0.21469	0.33663	0.35388
52	lm_dir	SERP title	0.21469	0.33663	0.35388
53	tfidf_sum	SERP title	0.21470	0.33663	0.35388
54	log_tf_sum	Question text	0.21471	0.33664	0.35388
55	log_dlen	SERP title	0.21472	0.33664	0.35388