TECHNICAL REPORT

YL-2011-001

BEWARE OF RELATIVELY LARGE BUT MEANINGLESS IMPROVEMENTS

Roi Blanco, Hugo Zaragoza
Yahoo! Research
Diagonal 177, Barcelona, Spain
{roi,hugoz}@yahoo-inc.com

Bangalore • Barcelona • Haifa • Montreal • New York
Santiago • Silicon Valley

Yahoo! Labs Technical Report No. YL-2011-001

Yahoo! Labs Technical Report No. YL-2011-001

BEWARE OF RELATIVELY LARGE BUT MEANINGLESS IMPROVEMENTS

Roi Blanco, Hugo Zaragoza
Yahoo! Research
Diagonal 177, Barcelona, Spain
{roi,hugoz}@yahoo-inc.com

ABSTRACT: When we randomly perturb the scores produced by an ad-hoc retrieval model we observe performance improvements for several measures and collections with unexpectedly high probability. Many of these improvements are relatively large, and statistically significant using today's standard information retrieval validation methodology. These results stress the need for a standard and reliable methodology suitable for IR experimentation and model comparison.

1. Introduction

In information retrieval (IR) it is customary to invent new features in order to enhance document ranking. Typically, these features are incorporated into a retrieval model and performance is optimized over a collection at hand. The objective is to find an improvement over a baseline model, measured using standard metrics (such as mean average precision); since the retrieval problem is very hard, small relative improvements (<5%) in performance are considered interesting (and publishable). However, in practice it is sometimes the case that fixing a bug or changing slightly some pre-processing step over the data produces this sort of improvement. We were interested in determining how likely it is that pure random effects may lead to significant improvements. Results were sufficiently surprising to merit discussion and publication in our opinion.

We tried to construct the simplest possible model of score perturbation. We did this by adding a small random value to each score that the baseline model produces. Formally:

$$s'(q,d) = s(q,d) + \lambda x_d$$

where q and d denote a query and document respectively, $s(q,d)$ and $s'(q,d)$ are the scores assigned by the baseline and perturbed models respectively, x_d is a uniformly distributed random variable between 0 and 1 (assigned to each document in the collection), and λ is a small constant that can be used to adjust the effect of the random perturbation in the final score. If λ is 0 then the perturbed model is identical to the baseline model, and we therefore obtain the same performance. If λ is very large then the perturbed model yields random scores and one would expect that its performance is much worse than the baseline.

We note that under this model each document d gets its own perturbation value x_d independently of the query. In other words, for a given collection $D = (d_1, ..., d_{|D|})$ we generate a *perturbation vector* $X = (x_1, ..., x_{|D|})$. In this sense these perturbations acts as a *document features* associated to each document.

As a first experiment, we investigate how retrieval performance is affected by such a simple perturbation model. We proceed as follows. First, we generate a perturbation vector for the collection. Then we compute, for each query and each document, the baseline score $s(q,d)$ and (for the highest ranking 5000 documents for each query) the perturbed score $s'(q,d)$. This allows us to compute the average performance of the baseline system and the perturbed system for a given set of perturbations X and a given λ (pseudo-code for all these operations is shown in Figure 2 (left column) of Annex I).

We refer to a *perturbation run* to this entire process starting with the generation of perturbations and ending with a performance value for the baseline and the perturbed model. When this process is repeated many times the perturbed performance will vary due to the random factor, whereas the baseline performance obtained will always be the same.

This variation is illustrated in Figure 1, where the x-axis plots the performance distribution for the perturbation runs in terms of MAP (top row) and P@10 (middle row) and MRR (bottom row) with respect to increasing values of λ, when using ten samples (left column) and two hundred samples (right column). Note that the baseline model is the one that has $\lambda = 0$. In this case, the baseline model used is BM25 with parameters ($b = 0.75$, $k_1 = 1.2$)[4].

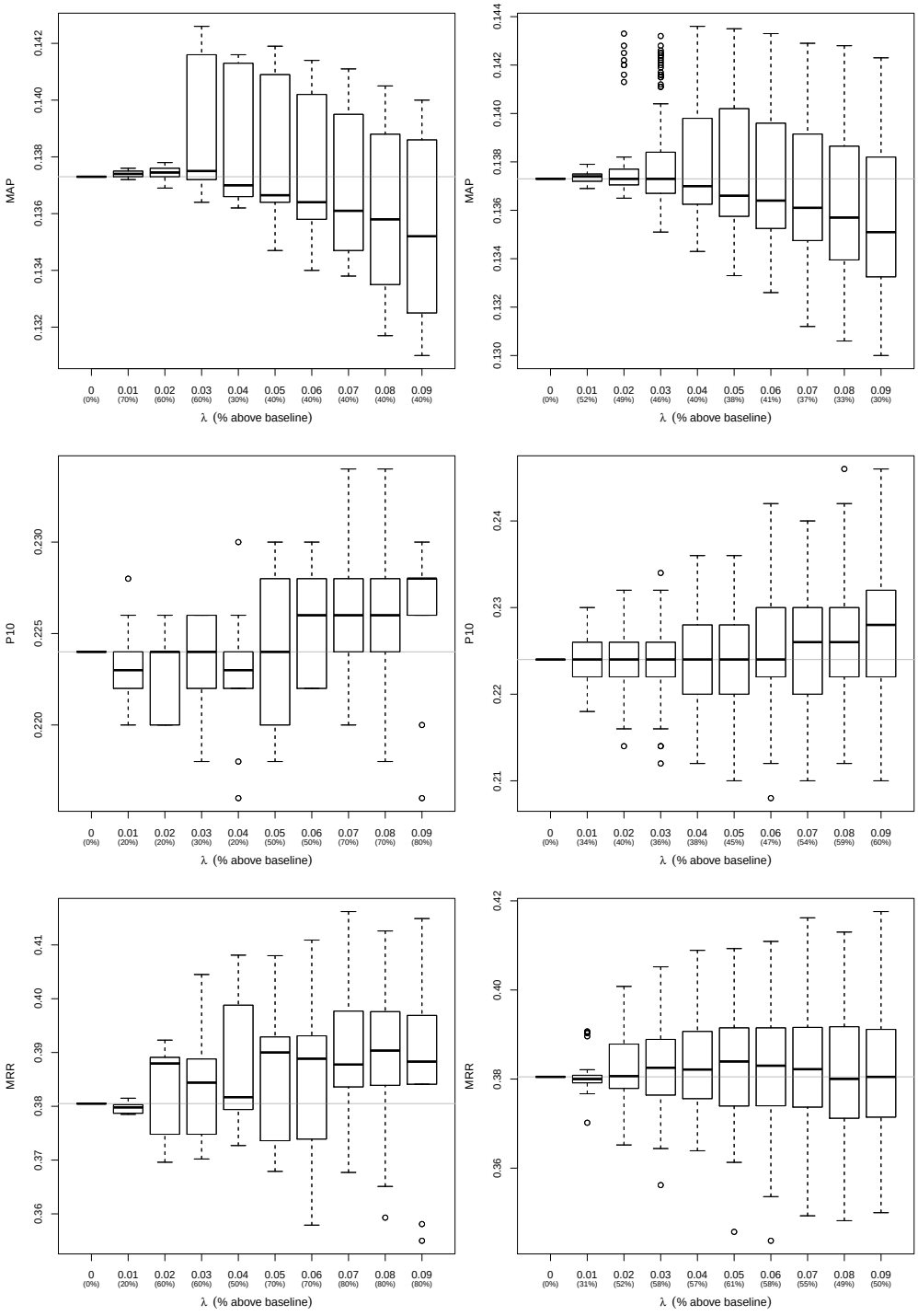

Figure 1: MAP, P@10 and MRR (top, middle and bottom rows respectively) performance distribution of the perturbed model using 10 samples (left column) and 200 samples (right column) for different values of λ.

Surprisingly, in this particular collection, we observe that a large number or runs obtain a performance that is *much higher* than the baseline for some small values of λ, even with as little as ten random runs. This was quite surprising to us. That is, by randomly moving scores slightly up and down we seem to be improving performance quite often and dramatically. Of course, for many runs performance also decreases.

To summarize, we have observed how large improvements in performance can be obtained by a simple random perturbation of the scores. This is not a positive result, since we are not able to predict if random perturbations will improve or degrade performance. On the contrary, it is a negative result: whenever we invent a new feature or model and observe an improvement in performance with respect to a baseline, we must be careful to eliminate the possibility that the gains are just due to some random effect. In the remaining of this paper we pursue this analysis a little further.

Experimental Settings Table 3 contains a description of 16 TREC collections that we employed all through this paper, which they span 14 years of evaluation campaigns. They comprise different domains (blogs, news, general Web, etc) and they differ in size and age. All through the paper we will use the TREC5 and TREC9 as working examples, even though in Annex 1 we report on all 16 collections. We report our findings using MAP, P@10 and MRR since these are the most widely used measures. We experiment with two types of topics: *SHORT* (using the TITLE field of TREC topics) and *LONG* (using the TITLE and DESCRIPTION fields).

We employed Terrier for retrieval [2] and in all the runs we remove stop-words using the default list provided by the system and process the terms using Porter's stemmer algorithm. All baseline runs are generated using the BM25 scoring model [4]. This model has two tunable parameters (namely b and k_1) but it has become commonplace in the community to accept $b = 0.75$ and $k_1 = 1.2$ as *default*, assuming that the configuration would be stable across collections. Indeed in many publications these default parameters are considered a valid and strong baseline for comparison; however performance can be significantly improved by tuning the b parameter. For this reason we will report results with respect to this baseline and with respect to the best possible baseline obtained by tuning b to obtain the highest MAP on each collection (i.e. over-fitting b to the collection)[1].

2. Tuning Noise

In the previous section we observed the effect of perturbations on performance, for different values of λ. In practice, parameters introduced into retrieval models are *optimized*, and typically only the best performance obtained is reported. If no gains are observed, research goes on: the model is further re-though and modified, and tuning is repeated. This cycle is repeated many times (dozens? hundreds?) until sufficient improvements are observed. The process of repeated experimentation and validation is central to scientific research, and it has allowed fields such as IR to improve the performance of crude models with highly sophisticated ones. However, as we saw in the previous

[1]Tuning k_1 does not seem to provide major improvements in performance, see http://barcelona.research.yahoo.net/dokuwiki/doku.php?id=baseline

TREC 5						
Baseline	MAP	Best Pert.	MRR	Best Pert.	P10	Best Pert.
Best Over-fitted Run						
Default	0.1373	+5.0983%	0.3805	+25.4402%*	0.2240	+15.1785%*
Tuned	0.1737	+4.0299%	0.5558	+13.9978%*	0.3280	+7.3170%*
Best Cross-validated Run						
Default	0.1373	+4.6613%	0.3805	+22.9434%*	0.2240	+8.0357%*
Tuned	0.1501	+.5996%	0.4293	+6.2893%	0.2420	+4.1322%

TREC 9						
Baseline	MAP	Best Pert.	MRR	Best Pert.	P10	Best Pert.
Best Over-fitted Run						
Default	0.2296	+2.7003%	0.6409	+12.4824%*	0.3080	+9.0909%*
Tuned	0.2546	+3.1029%*	0.6457	+10.2214%*	0.3280	+7.9268%*
Best Cross-validated Run						
Default	0.2546	+1.3747%	0.6457	+7.5576%	0.3280	+7.3170%
Tuned	0.2296	+1.3937%	0.6409	+9.3774%*	0.3080	+7.7922%

Table 1: MAP, MRR and P@10 of the BM25 baseline model (with Default or Tuned parameters), and relative improvement of the best (Over-fitted or Cross-validated) of 200 random perturbations of the baseline scores. Results here are for the TREC 5 and 9 (Long Queries) ad-hoc task; result for other collections are presented in Appendix I.

section, as one tries more and more things, the probability of observing gains due to a random factor increases. This is precisely what we investigate in this section.

For a given collection D with perturbation values X and a given set of queries Q with judgements, we can test different values of λ to find the one with highest performance[2]. We call this the best *over-fitted* λ, since we are using the same query set to select λ and to report its performance. If we re-generate a perturbation vector X we can repeat the process, obtaining a different optimum value of λ and different performance. Like a researcher looking for a new feature or model, every time we generate X we can optimize for λ and see if we beat the baseline. (Note that in the worse case the optimization procedure will return λ=0 which is equivalent to the baseline; in this sense the optimization procedure can only increase the baseline performance.)

Given that our *features* are random, we would expect that this improvement is negligible and rare, and that very many thousands of experiments are needed to find a lucky candidate. However, Figure 1 hints otherwise: we are likely to find many good values of λ yielding excellent performance even after a few dozen runs.

In Table 1 (*Best Over-fitted Run*) we present the best of 200 runs optimizing performance in the

[2]This can be done for example using *exhaustive search* over a set of λ values such as $(0, 0.1, 0.2, ..., 5)$, as shown in the `overfit_lambda` function in Figure 2. The problem with this approach is that it is hard to choose the correct interval and step size, and may require many trials. Instead we used the greedy line-search algorithm described in [3].

manner described above for the ad-hoc Long Queries tasks of TREC5 and TREC9 (pseudo-code for the experiment is given in Figure 2 (right column) of in Annex I; a more detailed description of the collections used, as well as results for many other collections are presented in Figure 4). We show the MAP, MRR and P@10 of the BM25 baseline model with default and with tuned parameters, and next to this we show the relative improvement obtained by the best perturbed model. Statistical testing symbol * indicates that the difference in performance was found significant using a Wilcoxon signed-rank test (one-sided, $\alpha = 0.05$) (refer to Section 4 for a more thorough discussion on statistical testing).

For example, adding the random perturbation and over-fitting λ over *default* BM25 in the TREC5 collection, yields some remarkable results (the same result holds for both long and short queries). Using the perturbed scores, MAP increases up to a relative 5%, P@10 up to 13% and MRR up to 29%! This type of improvements is much larger than typical improvements reported in literature.

In general, Tables 4 shows that with a relatively small number of trials, it is possible to obtain significant (and even statistically significant) improvements over a baseline. This holds for the three performance measures reported (MAP, MRR and P@10), for default and tuned baselines, for short and long queries, and for many different types of collections (small and large, news-wire and blogs, etc. see Annex I) with different numbers of queries (from 50 to 250). As a broad comment, it seems to be more difficult to improve over the tuned baseline, although this is not always the case (see TREC9 for instance). There is little distinction between long and short queries if the baseline is tuned; however, longer queries are more prone to improve due to random perturbations using the *default* setting.

In our case we know that all these improvements are due to pure random perturbations and therefore *they are meaningless*. However, this tells us that we must be extremely careful when we evaluate new features or algorithms, since purely random effects can yield such apparently high relative improvements.

There are at least two ways in which we are not being careful in these experiments. First, we must establish the statistical significance of improvements, and second we must protect ourselves from the over-fitted λ bias.

3. Cross Validation

One possible critique to our method is that we *over-fit* the λ parameter; in other words: we optimize this parameter on the same query set that we report the performance, and therefore we may be observing unfairly optimistic results. This can be remedied with cross-validation, which is actually a rare practice in IR, partly due to the fact that over-fitting with very few parameters is rare, and partly due to laziness and the extra computational time involved in cross-validation. We were interested in ascertaining how much *protection* from random improvements did cross-validation offer in practice, for our setting. Therefore, we set up the experiment described as follows.

A 2-fold cross-validation run is performed by optimizing λ on one half of the queries and performance is evaluated on the other half. The halves are then swapped and λ is optimized again (see `best_crossval()` in Figure 2). Average performance is obtained by averaging all the query

Long queries																		
	MAP						MRR						P@10					
Collection	W1	W2	T1	T2	S1	S2	W1	W2	T1	T2	S1	S2	W1	W2	T1	T2	S1	S2
Default baseline																		
TREC5	n	n	n	n	n	n	**y**	**y**	**y**	**y**	**y**	**y**	n	**y**	n	**y**	n	n
TREC9	n	n	n	n	n	n	**y**	**y**	**y**	**y**	n	n	**y**	**y**	**y**	**y**	**y**	**y**
Over-fitted baseline																		
TREC5	n	n	n	n	n	n	**y**	**y**	n	n	**y**	**y**	**y**	**y**	**y**	**y**	**y**	**y**
TREC9	n	**y**	n	**y**	n	n	n	**y**	**y**	**y**	n	n	n	**y**	n	**y**	**y**	**y**

Table 2: Statistical Significance (y) or not (n) of the over-fitted runs in Table 1 for Wilcoxon-signed rank test (W), t-test (T), and sign test (S), one (1) and two-sided (2) .

performances obtained. This procedure gives us a less biased estimate of the true performance of our model. Like in the case of over-fitted performance, this procedure can be repeated every time that we generate perturbation values X. As before we repeated this procedure 200 times and report the best model obtained.

Table 1 presents the results for TREC 5 and 9 (long queries); for many other collections refer to Annex I. Indeed, we observe that cross-validation did not help much in order to eliminate the improvements of the random perturbation.

4. Statistical Testing

Another pitfall when comparing two mean performance measures is that their relative difference may not be *statistically significant*. Testing for statistical significance is difficult in practice, because there are many available tests each one relying on different assumptions about the data, none of which hold true in practice. Furthermore statistical testing does not give a binary answer; instead it tells us the probability or accepting or rejecting the null hypothesis at a given significance level α.

Typical practice in IR is to use the *t*-test, the *Wilcoxon signed rank* test and the *sign* test [1] over the query performances of a baseline model and a candidate model. The resulting *test statistic* can be used to compute a *P-value* (for a one-sided test); when the P-value is lower than α the null hypothesis can be rejected

In Table 2 we report results of statistical testing of the improvements that we reported in the previous section (Table 1, *Best Over-fitted Run, Default Baseline*) with Wilcoxon (W), t-test (T) and sign test (S) for one-sided (1) and two-sided (2) tests with $\alpha = 0.05$ (tests for many other collections are reported in Table 6 of Annex I). In the case of 1-sided tests, a *y* indicates that the perturbed performance was statistically significantly better that the baseline. In the case of a 2-sided test, it indicates that one model is better than the other one, without specifying which (it does not follow that the model with the highest average is the best one). It follows that 2-sided tests should be avoided altogether as the null hypothesis is not what we are interested in testing and its results can

be totally misleading[3]. Unfortunately, there still are many works that do not specify which flavor of the statistical test they are using and thus we report on both variants here.

Looking at table 2 we see all the tests indicate no statistical significance for MAP; however all the tests indicate statistical significance for at least one collection and one measure. One-sided (being more strict) does not find significance more often than two-sided; however, the three one-sided tests indicate that some of the improvements are significant. For example, for TREC5 only the sign-test for P@10 (one or two-sided) indicates that there is no statistical significance; but this same measure leads to statistical significance on TREC9.

These results hold when we look at statistical tests for many other collections (in Table 6 of Appendix I). Practically all test for all collections find no significant improvements in MAP, although there are four collections for which significant improvements are found in MAP, and all three tests find one-sided significant improvements for at least one collection! Furthermore note that in this table we only report statistical significance for the *best* run (the same one for which we report performance in Table 1), but there may be other runs with less performance but with better statistically significance (for example, we found runs that pass the three tests with respect to MAP).

There is a simple explanation as to why we appear to be obtaining significant improvements with a random feature: we we have performed multiple experiments (in our case 200 for each λ value) and testing as if we had performed only a single experiment with the winning λ. In statistical testing this is called a *multiple comparisons* setting, and one can adjust for it by dividing the P value by the number of experiments carried out. In our case this would reduce the P value by orders of magnitude, eliminating hopes of statistical significance. However in empirical IR evaluations no one ever uses the multiple comparisons setting. It is not even clear for us how this may be done in practice, since experimentation is often iterative and incremental. Nevertheless our results clearly show that we should be worried about this effect even with small experiments comparing dozens of models. It remains an open question how to protect against it in practice.

To summarize, for all collections and all performance measures we have found some randomly perturbed runs that pass all the significance tests that are common practice today in IR.

5. Conclusions

Under current standard IR methodology, and after a relatively small number of experiments (200), we could show large statistically significant improvements over a number of collections and measures using a random feature. One can argue that a thorough statistical testing of our results would have required adjusting the significance levels to take into account the *multiple comparisons* setting (where many attempts are run and only the best selected), and we agree with this, but we note that this has never been done in the IR literature before, and in fact it is not obvious how one would do it in an incremental setting where new models and features are constantly being developed.

Of the three measures tested, MAP proved to be much more robust than MRR and P@10 in all settings. This validates the well known fact that MAP is more stable as a measure, and that MRR

[3]Sometimes a random perturbation improves the average performance but it decreases the median performance, so the average is not a valid indicator of which model produces higher values if we run a 2-sided test

Name	Collection	Topics	TREC Year
TREC3	Disks 1&2	151-200	1994
TREC4	Disks 1&2	201-250	1995
TREC5	Disks 2&4	251-300	1996
TREC6	Disks 4&5	301-350	1997
TREC7	Disks 4&5	351-400	1998
TREC8A	Disks 4&5	401-450	1999
TREC8B	WT2G	401-450	1999
TREC9	WT10G	451-500	2000
WT10g	WT10G	451-550	200-2001
ROBUST04	Disks 4&5	301-450 + 601-700	2004
TERA04	GOV2	701-750	2004
TERA05	GOV2	701-800	2005
TERA06	GOV2	701-850	2006
BLOGS06	Blogs06	851-900	2006
BLOGS07	Blogs06	901-950	2007
BLOGS0607	Blogs06	851-950	2006+2007

Table 3: Description of TREC Collections and topic sets used.

and P@10 are somewhat dangerous as random perturbations at the top of some queries may lead to apparently large and significant improvements. This is specially important since many recent IR tasks evaluate only top results, and are therefore prone to this type of perturbation.

References

[1] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 2009.

[2] Lioma C. Macdonald C. Plachouras V. Ounis, I. Research directions in terrier. *Novatica/UPGRADE Special Issue on Web Information Access, Ricardo Baeza-Yates et al. (Eds), Invited Paper*, 2007.

[3] S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond, foundations and trends in information retrieval. volume 3, pages 333–389, 2009.

[4] Robertson S. and Walker S. Some simple effective approximations to the 2 poisson model for probabilistic weighted retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241, New York, NY, USA, 1994. ACM/Springer Verlag.

```
function generate_perturbations()
Input:  n
Output:  X =< x_1, ..., x_n >
for i = 1 : n
    x_i ← 𝒳
endfor

function baseline_scoring()
Input:  D =< d_1, ..., d_n >, q
Output:  D_q
for i = 1 : n
   s_i ← BASELINE(d_i, q)
endfor
D_q ←< (d_1, s_1), ..., (d_n, s_n) >
D_q ← SCORESORT(D_q, 5000)

function perturbed_scoring()
Input  D_q, X, λ
Output  D_q
n ← |D_q|
for j = 1 : n
   s'_j ← s_j + λ · X_{d_j}
endfor
D_q ←< (d_1, s'_1), ..., (d_n, s'_n) >
D_q ← SCORESORT(D_q)

function performance()
Input  D, X, λ, Q =< q_1, ..., q_m >
Output  b, p
foreach q ∈ Q
    D_q ← baseline_scoring(D, q)
    D'_q ← perturbed_scoring(D_q, X, λ)
    p_q ← PERFORMANCE(D_q)
    p'_q ← PERFORMANCE(D'_q)
endfor
b ← AVERAGE(p_1...p_m)
p ← AVERAGE(p'_1...p'_m)
```

```
function overfit_lambda()3
Input: D, X, Q
Output:  bestL
bestP ← 0
bestL ← 0
for λ ∈ (λ_1, ..., λ_k)
   (b, p) ← performance(D, X, λ, Q)
   if (p > bestP)
      bestP ← p
      bestL ← λ
   end if
end for

function best_overfit()
Input: D, Q, L = (λ_1, ..., λ_k)
Output:  bestP
bestP ← 0
for e = 1 : 200
   X ←generate_perturbations(|D|)
   l ← overfit_lambda(D, X, Q)
   (b, p) ← performance(D, X, λ, Q)
   if (p > bestP)    bestP ← p
end for

function best_crossval()
Input: D, Q, L = (λ_1, ..., λ_k)
Output:  bestP
Q_1 ← {q_1, ..., q_{|Q|/2}}
Q_2 ← {q_{|Q|/2+1}, ..., d_{|Q|}}
bestP ← 0
for e = 1 : 200
   X ← generate_perturbations(|D|)
   l_1 ← overfit_lambda(D, X, Q_1)
   l_2 ← overfit_lambda(D, X, Q_2)
   (b, p_1) ←performance(D, X, l_2, Q_1)
   (b, p_2) ←performance(D, X, l_1, Q_2)
   p ← 1/2 (p_1 + p_2)
   if (p > bestP)    bestP ← p
end for
```

Figure 2: Experiments Pseudo-code. SCORESORT returns the 5000 highest scoring documents sorted by decreasing score. BASELINE calls a baseline document scoring function, BM25 in our case. PERFORMANCE calls a standard IR retrieval performance function, in our case MAP, MRR and P@10 (we assume that the necessary query-document judgments are available to the function).

Long Queries						
Collection	MAP	Rand	MRR	Rand	P10	Rand
BLOGS06	0.3269	+.06%	0.8290	+9.23%*	0.6440	+5.90%*
BLOGS0607	0.3396	+.06%	0.8205	+7.20%*	0.6650	+3.61%*
BLOGS07	0.3524	+.14%	0.8119	+12.79%*	0.6860	+2.62%*
TERA04	0.2568	+.32%	0.6860	+8.43%	0.4857	+9.68%*
TERA05	0.2861	+.14%	0.7504	+7.70%*	0.5283	+5.36%*
TERA06	0.2880	+.8%	0.7653	+3.99%*	0.5369	+4.24%*
TREC7	0.2067	+.87%	0.6641	+13.48%*	0.4300	+6.05%
TREC8A	0.2563	+.62%	0.6674	+13.47%*	0.4560	+4.83%
TREC8B	0.2852	+1.44%*	0.7155	+7.20%	0.4480	+8.93%*
TREC9	0.2296	+2.70%	0.6409	+12.48%*	0.3080	+9.09%*
WT10g	0.2199	+2.19%	0.6653	+5.08%*	0.3530	+5.10%*
ROBUST04	0.2681	+.26%	0.6925	+4.39%*	0.4466	+1.70%*
TREC4	0.2170	+1.00%	0.7010	+7.36%*	0.4440	+5.41%
TREC5	0.1683	+4.99%	0.5210	+20.29%*	0.3220	+8.70%*
TREC6	0.2317	+2.42%	0.6023	+19.58%*	0.3980	+7.04%*
Short queries						
Collection	MAP	Rand	MRR	Rand	P10	Rand
BLOGS06	0.2903	+.21%*	0.6286	+25.52%*	0.5360	+6.34%*
BLOGS0607	0.2980	+.13%	0.6520	+14.23%*	0.5510	+2.54%
BLOGS07	0.3056	+.20%	0.6754	+16.64%*	0.5660	+3.53%*
TERA04	0.2438	+.25%*	0.6226	+20.53%*	0.4633	+8.81%*
TERA05	0.2736	+.15%*	0.6914	+8.26%*	0.5121	+3.55%*
TERA06	0.2634	+.11%	0.6833	+8.87%*	0.5060	+1.46%
TREC7	0.1808	+1.11%	0.5771	+17.78%*	0.4040	+3.96%*
TREC8A	0.2313	+1.47%	0.5934	+17.39%*	0.4300	+7.44%*
TREC8B	0.2507	+1.48%*	0.6065	+17.82%*	0.4040	+6.93%*
TREC9	0.1803	+2.00%	0.5002	+14.45%*	0.2396	+6.93%*
WT10g	0.1800	+1.78%	0.5425	+8.57%*	0.2888	+5.64%*
ROBUST04	0.2323	+.74%	0.6139	+6.94%*	0.4012	+1.60%*
TREC4	0.2170	+1.01%*	0.7010	+7.36%*	0.4440	+4.05%
TREC5	0.1373	+5.10%	0.3805	+25.44%*	0.2240	+15.18%*
TREC6	0.2341	+1.88%	0.6090	+16.62%*	0.4040	+2.97%*

Long Queries						
MAP	Rand	MRR	Rand	P10	Rand
0.3485	+.06%	0.8440	+9.09%*	0.7140	+2.52%
0.3646	+.11%	0.8351	+4.92%*	0.7130	+2.24%*
0.3873	+.18%	0.8084	+6.24%	0.7180	+1.67%
0.2852	+.25%*	0.7069	+12.85%*	0.5408	+5.66%*
0.3110	+.12%	0.7747	+5.03%	0.5727	+3.18%*
0.3169	+.06%*	0.7788	+4.46%*	0.5893	+1.60%
0.2110	+.85%	0.6810	+18.05%*	0.4380	+5.03%*
0.2581	+.74%	0.6760	+10.71%	0.4520	+6.20%
0.3219	+.56%	0.7243	+10.80%*	0.4780	+4.60%
0.2546	+3.10%*	0.6457	+10.22%*	0.3280	+7.93%*
0.2410	+2.45%	0.6330	+11.61%*	0.3690	+4.07%*
0.2704	+.22%	0.7029	+3.60%	0.4494	+1.16%
0.2184	+.78%	0.6909	+11.43%*	0.4460	+5.83%
0.1737	+4.03%	0.5558	+14.00%*	0.3280	+7.32%*
0.2352	+3.06%	0.6419	+12.20%*	0.3960	+4.55%
Short queries					
MAP	Rand	MRR	Rand	P10	Rand
0.3286	+.12%	0.6608	+24.76%*	0.6440	+4.04%*
0.3401	+.09%	0.7015	+14.19%*	0.6270	+2.08%*
0.3526	+.11%	0.7286	+7.14%*	0.6080	+2.96%*
0.2813	+.39%	0.7396	+7.13%	0.5367	+4.19%*
0.3121	+.10%*	0.7705	+6.33%*	0.5788	+2.61%*
0.3072	+.10%*	0.7630	+4.82%	0.5705	+2.12%
0.1917	+1.1%	0.5772	+17.76%*	0.4240	+1.89%*
0.2505	+1.28%	0.6361	+14.04%*	0.4240	+10.38%*
0.3203	+.59%	0.7137	+9.57%*	0.4580	+4.80%*
0.2181	+2.57%	0.5275	+10.18%*	0.2729	+11.47%*
0.1923	+1.82%	0.5611	+7.70%*	0.2898	+6.70%*
0.2454	+.45%	0.6274	+7.33%*	0.4124	+1.19%
0.2057	+.92%	0.6326	+13.31%*	0.4220	+5.21%
0.1501	+.47%	0.4293	+17.31%*	0.2420	+6.61%
0.2447	+1.39%	0.5908	+16.17%	0.3840	+6.25%*

Table 4: Performance with the baseline model with default (left) and over-fitted (right) parameters and the perturbed model with best λ. See Section 2 for details.

Long Queries						
Collection	MAP	Rand	MRR	Rand	P10	Rand
BLOGS06	0.3269	+.031%	0.8290	+4.62%*	0.6440	+9.32%*
BLOGS0607	0.3396	+.09%*	0.8205	+4.46%	0.6650	+2.11%
BLOGS07	0.3524	+.11%*	0.8119	+8.29%*	0.6860	+2.33%*
TERA04	0.2568	+.16%	0.6860	+8.54%	0.4857	+7.99%*
TERA05	0.2861	+.11%	0.7504	+5.41%*	0.5283	+3.82%*
TERA06	0.2880	+.07%	0.7653	+4.06%	0.5369	+3.74%*
TREC7	0.2067	+.53%	0.6641	+5.77%	0.4300	+5.12%
TREC8A	0.2563	+.74%	0.6674	+8.97%	0.4560	+3.51%
TREC8B	0.2852	+1.54%*	0.7155	+5.01%	0.4480	+7.14%*
TREC9	0.2296	+1.39%	0.6409	+9.38%*	0.3080	+7.79%
WT10g	0.2199	+1.68%	0.6653	+2.57%	0.3530	+3.68%
ROBUST04	0.2681	+.22%	0.6925	+2.87%	0.4466	+1.34%*
TREC4	0.2170	+.92%	0.7010	+5.65%*	0.4440	+5.41%*
TREC5	0.1683	+4.22%	0.5210	+14.70%	0.3220	+4.35%*
TREC6	0.2317	+1.25%	0.6023	+16.50%*	0.3980	+6.03%*
Short Queries						
Collection	MAP	Rand	MRR	Rand	P10	Rand
BLOGS06	0.2903	+.17%*	0.6286	+17.82%*	0.5360	+5.97%
BLOGS0607	0.2980	+.10%	0.6520	+12.79%*	0.5510	+1.63%
BLOGS07	0.3056	+.23%	0.6754	+9.34%	0.5660	+2.83%
TERA04	0.2438	+.29%*	0.6226	+17.28%*	0.4633	+6.15%*
TERA05	0.2736	+.15%	0.6914	+9.69%*	0.5121	+2.58%
TERA06	0.2634	+.08%	0.6833	+6.59%*	0.5060	+2.39%
TREC7	0.1808	+.39%	0.5771	+12.82%	0.4040	+2.48%
TREC8A	0.2313	+.78%	0.5934	+8.12%	0.4300	+6.98%*
TREC8B	0.2507	+1.68%*	0.6065	+14.39%*	0.4040	+4.95%
TREC9	0.1803	+1.22%	0.5002	+12.72%*	0.2396	+6.93%*
WT10g	0.1800	+1.00%	0.5425	+7.48%*	0.2888	+6.72%*
ROBUST04	0.2323	+.52%	0.6139	+5.07%*	0.4012	+2.09%
TREC4	0.2170	+.92%	0.7010	+5.24%	0.4440	+3.60%
TREC5	0.1373	+4.66%	0.3805	+22.94%*	0.2240	+8.04%*
TREC6	0.2341	+1.67%	0.6090	+10.28%	0.4040	+2.97%

Long Queries					
MAP	Rand	MRR	Rand	P10	Rand
0.3485	+.03%	0.8440	+7.25%*	0.7140	+1.40%
0.3646	+.06%	0.8351	+3.31%	0.7130	+1.26%*
0.3873	+.18%	0.8084	+4.82%	0.7180	+1.67%*
0.2852	+.18%	0.7069	+11.89%*	0.5408	+3.77%
0.3110	+.09%*	0.7747	+3.42%	0.5727	+2.48%*
0.3169	+.03%	0.7788	+3.93%	0.5893	+1.36%*
0.2110	+.52%	0.6810	+8.72%	0.4380	+3.65%*
0.2581	+.93%	0.6760	+8.31%*	0.4520	+5.31%*
0.3219	+.56%	0.7243	+6.41%	0.4780	+2.09%
0.2546	+1.38%	0.6457	+7.56%	0.3280	+7.32%
0.2410	+1.37%	0.6330	+8.97%*	0.3690	+2.98%
0.2704	+.11%	0.7029	+3.46%	0.4494	+1.16%
0.2184	+.28%	0.6909	+7.31%*	0.4460	+3.59%
0.1737	+3.57%	0.5558	+10.72%	0.3280	+5.49%*
0.2352	+1.28%	0.6419	+15.56%*	0.3960	+3.03%
Short Queries					
MAP	Rand	MRR	Rand	P10	Rand
0.3286	+.03%	0.6608	+18.54%*	0.6440	+2.79%
0.3401	+.06%	0.7015	+9.11%*	0.6270	+1.75%*
0.3526	+.09%	0.7286	+7.41%	0.6080	+1.97%
0.2813	+.28%	0.7396	+9.53%	0.5367	+3.43%
0.3121	+.06%	0.7705	+3.11%*	0.5788	+2.78%
0.3072	+.10%*	0.7630	+2.77%	0.5705	+2.23%
0.1917	+.73%	0.5772	+17.16%*	0.4240	+0%
0.2505	+1.60%	0.6361	+12.12%*	0.4240	+8.49%*
0.3203	+.19%	0.7137	+6.38%	0.4580	+2.62%
0.2181	+.55%	0.5275	+7.87%*	0.2729	+8.39%*
0.1923	+.42%	0.5611	+4.94%	0.2898	+4.59%
0.2454	+.29%	0.6274	+5.55%*	0.4124	+.70%
0.2057	+.63%*	0.6326	+10.28%*	0.4220	+5.21%
0.1501	+.60%	0.4293	+6.29%	0.2420	+4.13%
0.2447	+1.06%	0.5908	+16.60%*	0.3840	+6.25%*

Table 5: Performance of the baseline model with default (left) and over-fitted (right) parameters and the perturbed model with λ chosen using 2-fold cross-validation. See Section 3 for details.

Short queries																		
Collection	W1	W2	T1	T2	S1	S2	W1	W2	T1	T2	S1	S2	W1	W2	T1	T2	S1	S2
BLOGS06	n	n	n	n	n	n	y	y	y	y	n	y	n	y	y	y	y	y
BLOGS0607	n	n	n	n	n	n	y	y	y	y	y	y	y	y	y	y	y	y
BLOGS07	n	n	n	n	n	n	y	y	y	y	y	y	y	y	y	y	y	y
TERA04	n	n	n	y	n	n	n	n	n	n	n	n	y	y	y	y	y	y
TERA05	n	n	n	n	n	n	y	y	y	y	y	y	y	y	y	y	y	y
TERA06	n	n	n	n	n	n	y	y	y	y	n	y	n	y	y	y	y	y
TREC7	n	n	n	n	n	n	n	y	n	y	n	n	n	n	y	y	y	y
TREC8A	n	n	n	n	n	n	n	y	n	y	n	n	n	n	n	n	n	n
TREC8B	n	y	y	y	n	y	n	n	n	n	n	n	y	y	y	y	y	y
TREC9	n	n	n	n	n	n	y	y	y	y	n	y	y	y	y	y	y	y
WT10g	n	n	n	n	n	n	y	y	y	y	n	n	y	y	y	y	y	y
ROBUST04	n	n	n	n	n	n	n	y	n	y	n	n	y	y	y	y	y	y
TREC4	n	n	n	y	n	n	y	y	y	y	n	n	n	n	n	n	n	n
TREC5	n	n	n	n	n	n	y	y	y	y	y	y	y	y	y	y	y	y
TREC6	n	n	n	n	n	n	y	y	y	y	n	y	y	y	y	y	y	y
Long queries																		
Collection	W1	W2	T1	T2	S1	S2	W1	W2	T1	T2	S1	S2	W1	W2	T1	T2	S1	S2
BLOGS06	y	y	y	y	n	n	y	y	y	y	y	y	n	y	n	y	n	n
BLOGS0607	n	n	n	y	n	n	y	y	y	y	y	y	n	n	n	n	n	y
BLOGS07	n	n	n	n	n	n	y	y	y	y	y	y	y	y	n	n	y	y
TERA04	y	y	n	y	y	y	y	y	y	y	y	y	y	y	y	y	n	y
TERA05	n	y	n	n	n	n	y	y	y	y	y	y	n	y	n	n	n	n
TERA06	n	n	n	n	n	n	y	y	y	y	n	n	n	n	n	y	n	n
TREC7	n	n	n	n	n	n	y	y	y	y	y	y	y	y	n	n	n	n
TREC8A	n	n	n	n	n	n	y	y	y	y	n	y	y	y	n	y	n	y
TREC8B	n	y	y	y	n	n	y	y	y	y	n	n	n	y	y	y	n	y
TREC9	n	n	n	n	n	n	y	y	y	y	n	n	y	y	y	y	y	y
WT10g	n	n	n	n	n	n	y	y	y	y	n	n	y	y	y	y	y	y
ROBUST04	n	n	n	n	n	n	y	y	y	y	n	y	y	y	y	y	y	y
TREC4	y	y	n	y	n	n	y	y	n	y	n	n	n	n	n	n	n	n
TREC5	n	n	n	n	n	n	y	y	y	y	y	y	n	y	n	y	n	n
TREC6	n	n	n	n	n	n	y	y	y	y	n	y	n	y	y	y	n	y

Short queries																	
W1	W2	T1	T2	S1	S2	W1	W2	T1	T2	S1	S2	W1	W2	T1	T2	S1	S2
n	n	n	n	y	y	y	y	y	y	y	y	y	y	y	y	y	y
n	n	n	y	n	n	y	y	y	y	y	y	y	y	y	y	y	y
n	n	n	n	n	n	y	y	n	n	y	y	n	y	y	y	n	y
n	n	n	n	n	n	n	n	n	n	n	n	n	y	n	n	n	n
y	y	y	y	y	y	y	y	y	y	y	y	n	y	n	n	n	n
n	y	y	y	n	n	n	n	y	y	n	n	n	n	n	n	n	y
n	n	n	n	n	n	y	y	y	y	y	y	n	y	n	n	n	n
n	n	n	n	n	n	y	y	y	y	y	y	y	y	y	y	y	y
n	n	n	n	n	n	n	y	y	y	n	n	y	y	y	y	n	y
n	n	n	n	n	n	y	y	y	y	y	y	y	y	y	y	y	y
n	n	n	n	y	n	y	y	y	y	n	n	y	y	y	y	y	y
n	n	n	n	n	n	y	y	y	y	n	n	n	n	y	y	n	y
n	n	y	y	n	n	y	y	y	y	y	y	n	n	n	y	n	n
n	n	n	n	n	n	n	y	n	y	n	y	n	n	n	n	n	n
n	n	n	n	n	n	n	n	y	y	n	n	y	y	y	y	y	y
Long queries																	
W1	W2	T1	T2	S1	S2	W1	W2	T1	T2	S1	S2	W1	W2	T1	T2	S1	S2
n	n	n	n	n	n	y	y	y	y	y	y	n	n	n	n	n	n
n	n	n	n	n	n	y	y	y	y	n	y	y	y	y	y	y	y
n	n	n	n	n	n	n	n	n	y	n	n	n	n	y	y	n	y
y	y	y	y	n	n	y	y	y	y	n	n	y	y	y	y	y	y
n	n	n	n	n	n	n	n	n	n	n	y	y	y	y	y	y	y
y	y	y	y	y	y	y	y	y	y	y	y	n	n	n	y	n	n
n	n	n	n	n	n	y	y	y	y	y	y	y	y	y	y	y	y
n	n	n	n	n	n	n	n	n	n	n	n	n	n	y	y	y	y
n	n	n	n	n	n	y	y	y	y	n	y	n	n	n	n	n	n
n	y	n	y	n	n	n	y	y	y	n	n	n	y	n	y	y	y
y	n	n	n	y	n	y	y	y	y	y	y	n	y	n	y	n	y
n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	y	n	y
n	n	n	n	y	n	y	y	y	y	n	n	n	n	n	y	n	n
n	n	n	n	n	n	y	y	n	n	y	y	y	y	y	y	y	y
n	n	n	n	n	n	y	y	y	y	n	n	n	n	n	n	n	y

Table 6: Statistical Testing results of the random perturbation over the baseline with default (left) and over-fitted parameters (right). See Section 4 for details.