# Improving the robustness of language models – UIUC TREC-2003 Robust and Genomics Experiments

ChengXiang Zhai, Tao Tao, Hui Fang, Zhidi Shang
Department of Computer Science
University of Illinois at Urbana-Champaign

## Abstract

In this paper, we report our experiments in the TREC 2003 Genomics Track and the Robust Track. A common theme that we explored is the robustness of a basic language modeling retrieval approach. We examine several aspects of robustness, including robustness in handling different types of queries, different types of documents, and optimizing performance for difficult topics. Our basic retrieval method is the KL-divergence retrieval model with the two-stage smoothing method plus a mixture model feedback method. In the Genomics IR track, we propose a new method for modeling semi-structured queries using language models, which is shown to be more robust and effective than the regular query model in handling gene queries. In the Robust track, we experimented with two heuristic approaches to improve the robustness in using language models for pseudo feedback.

## 1 Introduction

Recent work on using language models for information retrieval has shown that probabilistic language models have several advantages over the more traditional retrieval models, including being able to optimize retrieval parameters automatically [7] and improve retrieval performance through better language models or estimation methods [5, 2]. Language models have also shown promising empirical results in different TREC tasks (e.g., [3]). In this year's TREC experiments, we focus on the issue of the robustness of language modeling approaches, and examine several aspects of it, including robustness in handling different types of queries, different types of documents, and optimizing performance for difficult topics.

Our basic retrieval method is the KL-divergence retrieval model with the two-stage smoothing method plus a mixture model feedback method [5, 7]. The KL-divergence retrieval model can be regarded as a generalization of the query likelihood method. It has an additional advantage of supporting model-based feedback [5]. The two-stage smoothing was originally proposed in the context of query likelihood, but we can adapt it in a straightforward way to handle a query model generated by a feedback method. We test this basic approach in the Genomics IR track to see how robust such an approach is in handling the Medline abstract documents and the gene description queries. In the Genomics IR track, we propose a new method for modeling semi-structured queries using language models, which is shown to be more robust and effective than the regular query model in handling gene queries. In the Robust Track, we explore how we can automatically set parameters and evaluate how well our approach perform on difficult topics. In the Robust track, we experimented with two heuristic approaches to improve the robustness in using language models for pseudo feedback.

## 2 Retrieval with language models

In this section, we describe our retrieval approaches. The basic retrieval formula we use is the Kullback-Leibler (KL) divergence between the query language model and the document language model [1, 5]. This method is a generalization of the query likelihood retrieval method proposed in [4] and can support feedback more naturally than the query likelihood method.

Suppose that a query $\mathbf{q}$ is "generated" by a unigram language model $p(\mathbf{q} \mid \theta_Q)$ with $\theta_Q$ denoting the parameters, and similarly, a document $\mathbf{d}$ is generated by a unigram model $p(\mathbf{d} \mid \theta_D)$ with $\theta_D$ denoting the parameters. Let $\widehat{\theta}_Q$ and $\widehat{\theta}_D$ be the estimated query and document language models respectively. The KL-divergence retrieval formula scores $d$ with respect to $q$ with the following *negative* KL-divergence function [5]:

$$-D(\widehat{\theta}_Q \,\|\, \widehat{\theta}_D) = \sum_w p(w \mid \widehat{\theta}_Q) \log p(w \mid \widehat{\theta}_D) + (-\sum_w p(w|\widehat{\theta}_Q) log p(w|\widehat{\theta}_Q))$$

Since the second term on the right-hand side of the formula does not depend on $d$, it can be ignored for the purpose of ranking documents. Thus the scoring is essentially based on the first term, i.e., the cross entropy of the document model and the query model. In general, the

computation of this cross entropy involves a sum over all the words that have a non-zero probability according to $p(w|\widehat{\theta}_Q)$. However, when $\widehat{\theta}_D$ is based on the following general smoothing scheme, the computation would only involve a sum over those that both have a non-zero probability according to $p(w|\widehat{\theta}_Q)$ and occur in document $\mathbf{d}$. Such a sum can be computed much more efficiently with an inverted index. See [3] for a detailed explanation of this.

Clearly, the retrieval performance of the KL-divergence would depend on how we estimate the document model $\theta_D$ and the query model $\theta_Q$. Smoothing of $\theta_D$ is very important, and an effective smoothing method is the two-stage smoothing method proposed in [7], where it has been shown to achieve optimal or near optimal performance with completely automatic parameter setting. However, this smoothing method as presented in [7] is based on the query likelihood retrieval method, which is unable to incorporate feedback naturally. Although some techniques for tuning retrieval parameters automatically are proposed in [7], they are based on without pseudo feedback. This is also why the performance reported in [7] is not as good as the best official TREC results on the same data, which are usually obtained based on pseudo feedback. The KL-divergence retrieval formula can be shown to be a generalization of the query likelihood method, and can naturally support feedback as query model updating. Two specific methods for performing feedback using language models are proposed in [5]. In order to automatically tune the retrieval parameters for pseudo feedback, we extend the two-stage smoothing method so that it can work with a query model using KL-divergence. Since maximizing the likelihood is equivalent to minimizing the KL-divergence, this extension is not unnatural. We now explain this extension in detail.

The basic idea of two-stage smoothing is to decouple the two roles of smoothing (i.e., the estimation role and query modeling role [6]) so that we can capture the intrinsic connections between the parameter values and the data. Specifically, the document model $\theta_D$ is smoothed first with Dirichlet prior (to implement the estimation role) and then with a simple linear interpolation with some query background model (to implement the query modeling role). To apply this idea to the KL-divergence retrieval formula, we also use Dirichlet prior as the first-stage smoothing method. That is, our estimated document language model $\theta_D$ is given by Dirichlet prior [6]. According to this method,

$$p(w|\widehat{\theta}_D) = \frac{c(w,d) + \mu p(w|\mathcal{C})}{|d| + \mu}$$

where $c(w,d)$ is the count of word $w$ in $d$, $\mu$ is a query independent smoothing parameter, and $p(w|\mathcal{C})$ is reference language model estimated using the whole document collection. $\mu$ can be set using leave-one-out cross validation on the document collection [7]. To implement the second-stage smoothing, we first interpolate the estimated document model $\widehat{\theta}_D$ with the query background model $p(w|U)$ to obtain a two-stage smoothed document model $\widehat{\theta}'_D$, and then compute the KL-divergence $D(\widehat{\theta}_Q||\widehat{\theta}'_D$. That is,

$$p(w|\widehat{\theta}'_D) = (1-\lambda)p(w|\widehat{\theta}_D) + \lambda p(w|U)$$

where $\lambda$ can be interpreted as the amount of noise that we believe exists in the query, and $p(w|U)$ is the user's query background model. Since no information is available about the noise in the query, we simply approximate $p(w|U)$ by $p(w|C)$. $\lambda$ can also be estimated in a very much similar way as in [7], except that we minimize the KL-divergence between the document mixture model and the query model instead of maximizing the query likelihood given the document mixture model.

The collection language model $p(w|\mathcal{C})$ is typically estimated by $\frac{c(w,\mathcal{C})}{\sum_{w'} c(w',\mathcal{C})}$, or a smoothed version $\frac{c(w,\mathcal{C})+1}{V+\sum_{w'} c(w',\mathcal{C})}$, where $V$ is an estimated vocabulary size (e.g., the total number of distinct words in the collection). One advantage of the smoothed version is that it would never give a zero probability to any term, but in terms of retrieval performance, there will not be any significant difference in these two versions, since $\sum_{w'} c(w',\mathcal{C})$ is often significantly larger than $V$.

Having discussed how we use two-stage smoothing to estimate a document model, let us now look at the query model. The simplest way to estimate $\theta_Q$ is to use the maximum likelihood estimator based on the query text, which gives us essentially the empirical query distribution:

$$p_{ml}(w|\widehat{\theta}_Q) = \frac{c(w,\mathbf{q})}{|q|}$$

Using this estimated value, the KL-divergence scoring formula is essentially the same as the query likelihood retrieval formula, thus we essentially have the two-stage smoothing retrieval method as presented in [7]. This is what we use for the initial round of retrieval. A more interesting way of computing $p(w|\widehat{\theta}_Q)$ is to exploit feedback documents. Specifically, we can interpolate the simple $p_{ml}(w|\widehat{\theta}_Q)$ with a *feedback model* $p(w|\theta_F)$ estimated based on feedback documents. That is,

$$p(w|\widehat{\theta}_Q) = (1-\alpha)p_{ml}(w|\widehat{\theta}_Q) + \alpha p(w|\theta_F) \quad (1)$$

where, $\alpha$ is a parameter that needs to be set empirically.

To compute $\theta_F$, we assume a two component mixture model for the feedback documents, where one component model is $p(w|\theta_F)$ and the other is $p(w|\mathcal{C})$. The likelihood of the feedback documents is thus

$$\log p(\mathcal{F}\,|\,\theta_F) =$$

$$\sum_{i=1}^{k}\sum_{w} c(w;d_i)\log((1-\nu)p(w\,|\,\theta_F)+\nu p(w\,|\,\mathcal{C}))$$

where, $F=\{d_1,...,d_k\}$ is the set of feedback documents, and $\nu$ is yet another parameter that indicates the amount of "background noise" in the feedback documents, and that needs to be set empirically. Given $\nu$, the feedback documents $\mathcal{F}$, and the collection language model $p(w|\mathcal{C})$, we can use the EM algorithm to compute the maximum likelihood estimate of $\theta_F$ [5]. For the sake of efficiency, we truncate the model $\theta_F$ so that we only keep $k$ word with the highest probabilities according to $p(w|\theta_F)$. We thus have four parameters to set in performing pseudo feedback: (1) $\alpha$ controls the influence of feedback documents on the new query model; (2) $\nu$ indicates the amount of noise that we believe exists in the feedback documents; (3) $k$ is the number of terms to select for query model updating; and (4) the number of documents to use for feedback. It is reasonable to assume that the choice of $k$ has a relatively insignificant influence since the words excluded generally have smaller probabilities. But we know that the feedback performance is sensitive to both $\alpha$ and $\nu$ [5], and no doubt, also to the number of top documents to be used for pseudo feedback. This makes the feedback approach less robust.

Thus a major research question we address is how we can make this feedback process more robust. We propose two strategies:

- **Weighted pseudo feedback:** The basic idea of this strategy is to discount the contribution of documents according to their rank in the results. Specifically, we assume the probability of relevance of a document ranked at rank r to be 1/r. Thus the top document has "full" contribution in feedback, whereas the second and third documents are discounted by a factor of $1/2$ and $1/3$ respectively. We hoe this strategy would make the performance less sensitive to the choice of the number of documents used for pseudo feedback because if we use more documents, those documents would just be discounted more.
- **Applying two-stage smoothing after feedback:** The idea of this strategy is to re-estimate the second-stage smoothing parameter $\lambda$ *after* we obtain an updated query model using feedback so that we can "compensate" for any non-optimality in the feedback parameter setting by adjusting $\lambda$ according to the "noise" in the newly updated query model.

Both strategies have shown some positive effect in our preliminary experiments with the past TREC data.

# 3 Genomics IR Track

We participated in the primary task of the genomics IR track, and focused on studying how we may apply language modeling approaches to this new retrieval task.

A critical difference between the genomics IR task and a regular ad hoc task is that a query in the genomics IR task has a structure. More specifically, a gene query has several parts, including an official name, which is usually a long noun phrase, several symbols, each being a unique identifier for the gene, and protein product names. Such a query has a clear disjunctive structure in that matching either a gene symbol or the *complete* gene name would indicate relevance. For example, topic 1 has the following name and symbols:

```
OFFICIAL_GENE_NAME
activating transcription factor 2
OFFICIAL_SYMBOL ATF2
ALIAS_SYMBOL HB16
ALIAS_SYMBOL CREB2
ALIAS_SYMBOL TREB7
ALIAS_SYMBOL CRE-BP1
```

These symbols are *unique* identifiers for the gene, so matching one of the symbols is a sufficiently strong evidence for a document to be relevant, and is as good as matching the whole phrase "activating transcription factor 2". Thus although both "transcription" and "CREB2" are a single word term, "CREB2" should be weighted much higher than "transcription".

Such a semi-structured query clearly violates the basic assumption made in a language modeling approach that the query can be treated as a sample drawn from a language model. As a result, if we use the basic language modeling approach as is, the gene symbols may be underweighted. Indeed, in our preliminary experiments with the training topics, we found that the basic language modeling approaches did not perform as well as well-tuned vector space models.

To address this problem, we propose a new way to estimate our query language model $\theta_Q$ so that it can better model a disjunctive query. Our idea is to first compute the empirical word distribution for each "component query", and then take an average of them. Formally, suppose $q=\{q_1,...,q_k\}$ is a disjunctive semi-structured query, where $q_i$ is a "component query". Our estimated query model is given by

$$p(w|\theta_Q)=\frac{1}{k}\sum_{i=1}^{k}\tilde{p}(w|q_i)$$

where $\tilde{p}(w|q_i)$ is the empirical word distribution of component query $q_i$, and is computed as $\frac{c(w,q_i)}{|q_i|}$ where $c(w,q_i)$ is the count of word $w$ in the component query $q_i$ and $|q_i|$ is the length of $q_i$. Note that each gene symbol is one component query so if $q_i$ is a gene symbol $s$, then we have $\tilde{p}(s|q_i)=1$, whereas if $w$ is a word in the name, then $\tilde{p}(w|q_i)=1/|q_i|$, which is usually smaller than 1 because a name phrase almost always has more than one

| Query | No feedback | | | | Pseudo feedback | | | |
|---|---|---|---|---|---|---|---|---|
| | MAP | Pr@0.1 | Pr@10doc | Rec@1000 | MAP | Pr@0.1 | Pr@10doc | Rec@1000 |
| Official Name | 0.089 | 0.168 | 0.086 | 367/566 | 0.101 | 0.176 | 0.094 | 389/566 |
| Symbols | 0.147 | 0.289 | 0.131 | 390/566 | 0.169 | 0.313 | 0.142 | 466/559 |
| All words | 0.138 | 0.277 | 0.116 | 391/566 | 0.149 | 0.278 | 0.11 | 424/566 |
| All distinct words | 0.160 | 0.310 | 0.140 | 493/566 | 0.171 | 0.321 | 0.134 | 514/566 |
| Name + Symbols (ad hoc weighting) | 0.178 | 0.338 | 0.152 | 519/566 | **0.193** | **0.366** | **0.158** | **524/566** |
| Name + Symbols (model avg. ) | 0.185 | 0.385 | 0.154 | 496/566 | **0.200** | **0.393** | **0.150** | **511/566** |

Table 1: Different query models with/without pseudo feedback. Boldface indicates the results of two official submissions.

word. Thus, in effect, our average query model would favor matching a symbol. Intuitively, this average query model normalizes word counts so that the total contribution from each component query is equal, reflecting the disjunctive semantics of the query. As a baseline method, we also experimented with heuristically duplicating each gene symbol in a query to "boost" its weight.

Our two official submissions represent these two different approaches to model semi-structured queries. In both official submissions, we used symbols and the official name in the gene query description, and used the retrieval approach as described in Section 2 except that we did not apply two-stage smoothing after pseudo feedback. We used top 10 documents for pseudo feedback, set both $\alpha$ and $\nu$ to 0.5, and used top 20 terms for query model updating.

Since many biology names involve digits and hyphens, we used a slightly different tokenization method than what we normally use to handle the special syntax of gene names. Specifically, we retain all the digits as well as any hyphen that connects at least one digit; in other words, we only remove a hyphen when it connects two letters. This method gives a slight improvement in performance based on the training topics. To test the robustness of our method, we deliberately did not remove any stop word, as we believe that this is best handled through appropriate language modeling. We did not apply stemming either.

Table 3 shows the results of our two official submissions along with some other experiments where we use different types of queries and test them with/without feedback. The results of the two official runs are highlighted in boldface.

We can make the following observations:

1. Using gene name alone is least effective, significantly worse than using any other versions of the query. This is probably because the words in a name are too general and thus not discriminative. Indexing the whole name as a phrase may help improve the performance.

2. Using symbols is more effective than using all the words, which means that we treat the whole gene description as one long text query. This suggests that the symbols alone are the most important information in the query, and there are noise terms in other part of the query (mostly the gene name and protein names).

3. Using all words but ignoring word frequency (i.e., "All distinct words" in the table) improves performance significantly. Since symbols always have a frequency of 1, this suggests that when pooling all words together, we are overweighting the regular words in those names. The fact that "all distinct words" also performs significantly better than using the symbols alone indicates that the gene name is also very useful.

4. The two official runs all use names combined with symbols and both put more weight on a symbol term. We see that both perform better than other runs, indicating that treating such a disjunctive query as a regular text query is non-optimal and we need to increase the weight for short component queries (i.e,. symbols).

5. Comparing the two different methods for modeling disjunctive queries (i.e., ad hoc weighting by duplication and language model averaging), we see that the average query model approach performs slightly better in mean average precision (MAP) and much better in precision at recall level of 0.1 (i.e., the front end precision). However, it has a lower recall at 1,000 documents and a lower precision at 10 documents in feedback runs (i.e., our official runs).

# 4 Robust Track

Our goal for the robust track is to evaluate and improve the robustness of the language model based retrieval approach described in Section 2. This approach is a combination of the two-stage smoothing method which has been shown to

| Topics | Query Type | MAP | Pr@0.1 | Pr@10doc | Rec@1000 | percent(pr@10=0) | MAP area (worst 12 topics) |
|---|---|---|---|---|---|---|---|
| Old | title | 0.108 | 0.275 | 0.268 | 2034/4416 | 18% | 0.0057 |
| | description | 0.113 | 0.293 | 0.266 | 1827/4416 | 26% | 0.0031 |
| | title+desc | 0.140 | 0.380 | 0.326 | 2178/4416 | 12% | 0.0080 |
| New | title | 0.303 | 0.608 | 0.430 | 1392/1658 | 6.0% | 0.024 |
| | description | 0.372 | 0.684 | 0.494 | 1404/1658 | 12% | 0.0207 |
| | title+desc | 0.392 | 0.741 | 0.528 | 1476/1658 | 10% | 0.0466 |

Table 2: Different types of queries on old and new topics.

| Topics | Method | MAP | Pr@0.1 | Pr@10doc | Rec@1000 | percent (pr@10=0) | MAP area (worst 12 topics) |
|---|---|---|---|---|---|---|---|
| Old | no re-est (20 terms) | 0.113 | 0.293 | 0.266 | 1827/4416 | 26% | 0.0031 |
| | 2s re-est (20 terms) | 0.1178 | 0.311 | 0.280 | 1892/4416 | 24% | 0.0036 |
| | 2s re-est (50 terms) | 0.1250 | 0.326 | 0.282 | 1933/4416 | 26% | 0.0034 |
| New | no re-est (20 terms) | 0.372 | 0.684 | 0.494 | 1404/1658 | 12% | 0.0207 |
| | 2s re-est ( 20 terms) | 0.364 | 0.680 | 0.486 | 1402/1658 | 16% | 0.0193 |
| | 2s re-est (50 terms) | 0.375 | 0.675 | 0.498 | 1409/1658 | 16% | 0.0189 |

Table 3: Effect of two-stage smoothing after feedback on old and new topics.

be quite robust w.r.t., parameter setting [7], and the mixture model feedback approach [5], which is effective but sensitive to several parameters.

As discussed in Section 2, we propose two strategies to reduce the sensitivity of our approach to the setting of parameters: (1) Estimate the probability of relevance for each document to be used for pseudo feedback so as to discount the contribution of documents based on their ranks. (2) Apply the two-stage smoothing method to re-estimate the smoothing parameters after query model updating. In some sense, this method is to "bypass" the problem of choosing optimal values for the some of the feedback parameters. In our preliminary experiments with the old topics, these methods perform better than several baseline approaches

We did minimum preprocessing (only stemming, no stop word removal) to test the robustness of our method, and submitted five runs with one run using title of the query (UIUC03Rt1, three runs using the description part of the query (UIUC03Rd1, UIUC03Rd2, UIUC03Rd3), and one run using both the title and the description (UIUC03Rtd1). In all our submissions we applied the first strategy (i.e., discounting documents based on ranks), and in UIUC03Rd2 and UIUC03Rd3, we also applied the second strategy. UIUC03Rd2 and UIUC03Rd3 differ in the number of top terms selected to update the query model (20 and 50 respectively). In all cases, we used top 10 documents for pseudo feedback, and set $\alpha$ and $\nu$ both to 0.5 as in all our experiments.

In Table 4, we compare the performance of different versions of the queries on both old topics and new topics. It is clear that, in all cases, using the regular precision measures over all the topics, "title+description" performs much better than "description only", which performs better than "title only", though in the case of old topics, the recall of "description only" is worse than that of "title only". However, it is very interesting to see that for both old and new topics, "title only" has the least number of topics with no relevant document in top 10 documents, and its MAP area for the worst 12 topics is also better than the description. "title+description" has the largest MAP area for the worst 12 topics, but, compared with the "title only" run, it actually has more topics which have no relevant document in top 10 documents. These results clearly show that using titles appear to help improve the performance on difficult topics. This may be because the words in the query title are more accurate and when we use more words from other parts they just bring up many non-relevant documents perhaps because the relevant documents do not have a good match in vocabulary with the words in these other parts. Another possibility is that our approach may not be discriminative enough to give title words more weight and due to the fact that we do not remove stop words, the description part just adds too many noisy words. It would thus be interesting to apply our disjunctive query model to model the title and description parts to see if it can improve the performance, which we will explore in the future.

In Table 4, we compare the three "description only" runs. One (baseline) run applies two-stage smoothing only for the initial run and after the feedback the same smoothing parameters are used. The other two runs fur-

ther apply two-stage smoothing to *re-estimate* the smoothing parameters after we obtain an updated query model. These two runs differ in the number of top terms to be selected for query model updating. Here we can make the following observations:

1. Looking at all runs in which we use 20 terms for feedback, we see that for old topics, the re-estimation strategy helps improve performance by all the regular measures averaged on all the topics, but it does not improve the performance on difficult topics. Indeed, it actually hurts the performance for difficult topics. For new topics, the re-estimation strategy not only hurts the performance for difficult topics, it is worse by all measures.

2. Comparing all the runs involving re-estimation, we see that for old topics, using 50 terms for feedback is better than using 20 terms when averaging over all the topics, but is worse on difficult topics. For new topics, it appears to perform similarly to using 20 terms.

Thus our re-estimation strategy does not seem to help improve the robustness. We have not yet had more experiments to examine whether the first strategy – discounting feedback documents based on their ranks – is useful.

# 5 Conclusions

In this paper, we report our experiments in the TREC 2003 Genomics Track and the Robust Track. A common theme that we explored is the robustness of a basic language modeling retrieval approach. We examine several aspects of robustness, including robustness in handling different types of queries, different types of documents, and optimizing performance for difficult topics. Our basic retrieval method is the KL-divergence retrieval model with the two-stage smoothing method plus a mixture model feedback method. In the Genomics IR track, we propose a new method for modeling semi-structured queries using language models, which is shown to be more robust and effective than the regular query model in handling gene queries. In the Robust track, we experimented with two heuristic approaches to improve the robustness in using language models for pseudo feedback. One is the discount feedback documents based on their ranks and the other is to apply two-stage smoothing after feedback to re-estimate the smoothing parameters. Preliminary result analysis appears to suggest that the re-estimation of smoothing parameters does not really help. But we need to do more experiments and analysis in order to make reliable conclusions.

# References


[1] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'01*, pages 111–119, Sept 2001.

[2] V. Lavrenko and B. Croft. Relevance-based language models. In *Proceedings of SIGIR'01*, pages 120–127, Sept 2001.

[3] P. Ogilvie and J. Callan. Experiments using the lemur toolkit. In *Proceedings of the 2001 TREC conference*, 2002.

[4] J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR'98*, pages 275–281, 1998.

[5] C. Zhai and J. Lafferty. Model-based feedback in the KL-divergence retrieval model. In *Tenth International Conference on Information and Knowledge Management (CIKM 2001)*, pages 403–410, 2001.

[6] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of ACM SIGIR'01*, pages 334–342, Sept 2001.

[7] C. Zhai and J. Lafferty. Two-stage language models for information retrieval. In *Proceedings of ACM SIGIR'02*, pages 49–56, Aug 2002.