

# Language Models for Genomics Information Retrieval: UIUC at TREC 2007 Genomics Track

Yue Lu, Jing Jiang, Xu Ling, Xin He, ChengXiang Zhai  
{yuelu2, jiang4, xuling, xinhe2, czhai}@uiuc.edu  
Department of Computer Science  
University of Illinois at Urbana-Champaign  
Urbana, IL 61801

## Abstract

The University of Illinois at Urbana-Champaign (UIUC) participated in TREC 2007 Genomics Track. Our general goal of participation is to apply language model-based approaches to the genomics retrieval task and study how we may extend the standard language models to accommodate two special needs for this year’s genomics retrieval task: (1) gene synonym expansion and (2) conjunctive query interpretation. We also tried user relevance feedback. Preliminary result analysis shows that our synonym expansion method can improve document-level MAP, but generally has little influence on passage-level and aspect measures, while conjunctive scoring is not as effective as the standard KL-Divergence scoring, even though our pre-TREC experiments on a small set of training data showed otherwise. Relevance feedback appears to help. Further experiments and analysis are needed to draw more definitive conclusions.

## 1 Introduction

Language modeling approaches to information retrieval have been successfully applied to many tasks in TREC. We took the opportunity of participating in TREC 2007 Genomics Track to study how effective these approaches are for this year’s genomics retrieval task. Although the task involves returning relevant passages, we simplified the task by fixing the passage size uniformly to a window of three sentences to focus more on studying the ranking accuracy of using language models, with the hope that most relevant passages would be relatively short and our approximation will not significantly hurt the performance. Specifically, in all our experiments, we used a sliding window approach to pre-segment all maximum-length legal spans into passages with up to three sen-

tences, and indexed such passages as retrieval units. This allows us to focus on studying different ranking algorithms. In the end, we post-processed the returned passages to eliminate any possible redundancy that might exist due to the use of a sliding window strategy for generating the passage units.

A straightforward way of applying language modeling approaches to the TREC 2007 genomics task is to treat it as a regular ad hoc retrieval task. However there are two reasons why this strategy may not work well. First, it is well known that biologists express gene names in a non-uniform way and the same gene can be described in multiple ways. Thus using only the gene name(s) given in the query may not have high recall, and performing gene synonym expansion may be beneficial. Second, since the queries usually contain only a few words after stop word removal, intuitively, every remaining query word is important and we would like the returned results to match *all* the query words (i.e., “forcing” a conjunctive interpretation of the query). However, in a regular retrieval function, including a language model function, we generally adopt TF-IDF weighting of terms, so it is possible that a passage matching all query words once may not be ranked as high as one matching only some discriminative query words multiple times.

To address these two issues, we extended a standard language modeling approach to perform gene synonym expansion and to impose a “soft” conjunctive interpretation of the query. Specifically, for gene synonym expansion, we tagged the gene names in the queries using a gene recognizer, and then queried the external resource Entrez Gene to get a list of synonyms for each gene. After that, we constructed a “synonym query” for each synonym and use the original query plus all the synonym queries to do retrieval. Finally, the retrieved

results for each query are pooled together and re-ranked according to a weighting scheme based on the estimated confidence of each gene synonym. To impose a soft conjunctive semantics on the query, we proposed a heuristic generalization of the regular way of estimating the document language model. The idea is to generalize the estimation of document language models so as to flexibly discount the influence of TF and IDF weighting through some extra parameters. As a result, in an extreme case, we can set the parameters appropriately to “turn off” both TF and IDF and essentially score documents with a Boolean conjunctive query. Corresponding to these two ideas, we submitted two automatic runs, **UIUCsyn** and **UIUCconj**.

Besides the two directions discussed above, we also explored the use of user relevance feedback for retrieval. It appears that manual runs have not been very successful for last year’s genomics task [1], so we would like to see whether the language modeling approaches would make a difference. Thus for each topic, we obtained some manual judgments from two domain experts, and used them in our interactive run **UIUCrelfb**.

At the time of writing this report, we have not had sufficient time to run detailed follow-up diagnosis experiments to fully analyze our hypotheses. But some preliminary analysis of our experiment results suggests that (1) our synonym expansion method can improve document-level MAP, but generally has little influence on passage-level and aspect measures; (2) conjunctive scoring is not as effective as standard KL-Divergence scoring, even though our pre-TREC experiments on a small set of training data showed otherwise; and (3) both user relevance feedback and pseudo feedback help increasing the performance.

In the rest of this paper, we will first introduce our basic retrieval method in Section 2. After that we will discuss the new method we proposed for gene synonym expansion in Section 3. In Section 4, we will describe how we modified the KL-divergence retrieval model to impose “soft” conjunctive query semantics. We report our experimental results in Section 5 and conclude in Section 6.

## 2 Basic Retrieval Framework

This year’s task was defined as a passage retrieval task, where legal passages must be within single paragraphs from the full-text articles. Our observation on the training topics is that most answers come in the form of one to three sentences. So to focus on the ranking part of

the task, we took a sliding-window approach and treated every three consecutive sentences in a natural paragraph as a candidate passage for retrieval. The task was further divided into two sub-steps: a retrieval step and a passage post-processing step. In the retrieval step, we used the KL-divergence retrieval method to do an initial retrieval and then the query language model was updated using a robust pseudo relevance feedback method proposed by Tao and Zhai [5]. In the passage post-processing step, we used a heuristic algorithm to eliminate the redundancy in the retrieval results (which may exist because our retrieval units have overlaps) and compress the passages to keep only the most relevant information. In addition, we used Porter stemmer to do stemming and an official list of stop-words from National Library of Medicine <sup>1</sup> to remove stop-words. Most of our experiments are to vary the methods used in the first step, particularly different methods for estimating the document language model and some new gene synonym expansion method.

### 2.1 The KL-Divergence Retrieval Model

The first subtask is candidate passage retrieval, for which we could use any existing document retrieval method. In our experiments, we used the KL-divergence retrieval model [7], which we also used in our genomics experiments in TREC 2006 [2]. In this model, queries and documents are all represented by unigram language models, which are essentially word distributions. Assuming that these language models can be appropriately estimated, KL-divergence retrieval model scores a document  $D$  with respect to a query  $Q$  by computing the Kullback-Leibler divergence between the query language model  $\theta_Q$  and the document language model  $\theta_D$  as follows:

$$D(\theta_Q||\theta_D) = \sum_{w \in V} p(w|\theta_Q) \log \frac{p(w|\theta_Q)}{p(w|\theta_D)}$$

where  $V$  is the set of all words in the vocabulary.

What remains to be solved is how to appropriately estimate the query language model  $\theta_Q$  and the document language model  $\theta_D$ .  $\theta_D$  is estimated from the document  $D$ , and is usually smoothed with a background language model  $\theta_B$ , which is estimated with the whole document collection as follows:

$$p(w|\theta_B) = \frac{\sum_{D \in \mathcal{C}} c(w, D) + 1}{\sum_{w \in V} \sum_{D \in \mathcal{C}} c(w, D) + |V|} \quad (1)$$

<sup>1</sup>available at <http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhlp.html#Stopwords>.

where  $\mathcal{C}$  is the document collection,  $c(w, D)$  is the number of occurrences of word  $w$  in document  $D$ . One of the most effective smoothing methods is the Dirichlet prior smoothing method [6], as shown below:

$$p(w|D) = \frac{c(w, D) + \mu \cdot p(w|\mathcal{C})}{\sum_{w \in \mathcal{D}} c(w, D) + \mu} \quad (2)$$

where  $\mu$  is a parameter that controls the degree of smoothing, and is usually set empirically. In our experiments, we used the Dirichlet prior smoothing method as our baseline method for estimating document language models, and set  $\mu$  to 25 based on our preliminary experiments with the queries and documents from TREC 2006 Genomics Track. In Section 4, we will discuss how we may use some alternative method for estimation to impose a “soft” conjunctive interpretation of query semantics.

The simplest way to estimate the query language model is to use only the query.  $\theta_Q$  can thus be estimated as follows:

$$p(w|\theta_Q) = p(w|Q) = \frac{c(w, Q)}{|Q|}$$

where  $c(w, Q)$  is the number of occurrences of word  $w$  in query  $Q$ , and  $|Q|$  is the length of query  $Q$ . However, since queries are usually very short, they can hardly capture the user’s information need completely. Several methods have been proposed to improve the estimation of the query language model [3, 4, 7].

However the performance of these existing methods is often affected significantly by some parameters, such as the number of feedback documents to use and the relative weight of original query terms; these parameters generally have to be set by trial-and-error without any guidance. In a recent paper [5], a more robust method for feedback based on statistical language models was proposed. The main idea is to integrate the original query with feedback documents in a single probabilistic mixture model and regularize the estimation of the language model parameters in the model so that the information in the feedback documents can be gradually added to the original query. Unlike most existing feedback methods, this new method has no parameter to tune. Our participation in last year’s TREC 2006 Genomics task suggested the effectiveness of this new feedback method in the biomedical domain [2]. So we continued to apply this method in our experiments to incorporate either pseudo relevance feedback or user relevance feedback.

## 2.2 Passage Post-processing

Since we took a sliding-window approach to segment natural paragraphs into candidate passages of up to 3

sentences, it is possible that there is some overlap among the retrieved passages. The goal of the passage post-processing step is to eliminate the redundancy in the retrieval results. Thus in all the experiments, we first retrieve 2000 candidate passages, then go through this post-processing step, and finally keep only the top 1000 ranked passages. Specifically, given the passages returned for each topic, we go through the ranked list and check each passage  $p_i$  against those ranked above it for redundancy. If there is no redundancy, we keep  $p_i$ . (Note that the top-ranked passage is always kept.) If there exists a kept passage  $p_j$  which has overlap with  $p_i$ , then we apply the following two rules in order and drop  $p_i$ .

1. If  $p_i$  and  $p_j$  are both ranked in top  $K$  and they have more than 50% overlap of characters, then we drop  $p_i$  and replace  $p_j$  with a new passage containing the intersection of  $p_i$  and  $p_j$  at the same rank as  $p_j$ . The assumption is that it is the intersection that has caused both messages to be ranked high.

2. If the rank difference of passages  $p_i$  and  $p_j$  is within  $R$  and they have overlap, then simply drop  $p_i$ .

So there are two parameters,  $K$  and  $R$  to tune. We tried  $R = 20, 100, 500$ ,  $K = 10, 20, 30, 50$  with last year’s topics. The best performance came from  $R = 100$   $K = 20$ , which we used in all our TREC 07 experiments. It was observed that the post-processing step consistently improves performance for several different retrieval methods, suggesting that the post-processing algorithm is orthogonal to the retrieval method used.

## 3 Gene Synonym Expansion

Gene synonyms are very common in biomedical literatures, because biologists usually express gene names in a non-uniform way. In order to ensure the recall of the retrieved results, it is important to consider all the synonyms of any gene name in the query. However, there are several challenges: (1) Many genes have different synonyms in different species. Since the query does not explicitly mention the species, it is difficult to decide which synonyms to use for a gene in a query. (2) It is not trivial to assign appropriate weights to the added gene synonyms in the expanded query; under-weighting of synonyms would not bring much benefit, while over-weighting some unreliable synonyms can hurt performance significantly. Our experience in TREC 2006 Genomics track was that without an appropriate weighting scheme, expanding the query with synonyms tends to bias the query and decrease the retrieval performance. So this year, we studied the issue of weighting and ex-

perimented with a new method for synonym expansion, called “regularized synonym expansion.”

To achieve robust synonym expansion, our main idea is to assess the reliability of a candidate synonym (for query expansion) based on how much overlap exists between the original retrieval results and the results from using the candidate synonym to replace the original gene name. Intuitively, since the rest of the query is not changed, the amount of overlap can be an indicator of to what extent the synonym stays in the original query context. For example, if a synonym causes a dramatic draft to a completely different species, we would expect to see little overlap. Thus by applying a threshold to this overlap, we can filter out potentially harmful synonyms in irrelevant species. Moreover, for the synonyms that survive our thresholding, we may also assign each of them a weight proportional to the overlap of its retrieval results with the original results; this would allow us to trust more on a gene that does not drift away, thus achieving more robust synonym expansion. We now describe the regularized synonym expansion algorithm in more detail.

Let  $g$  be a gene name mentioned in query  $Q$  and  $S = \{g_1, \dots, g_n\}$  be a set of candidate synonyms of  $g$ . For each synonym  $g_i$ , we generate a “synonym query”  $Q_i$  by replacing  $g$  in  $Q$  with the synonym  $g_i$ . We then use  $Q$  and  $Q_1, \dots, Q_n$  each to retrieve a ranked list of  $N$  results, denoted as  $R(Q)$  and  $R(Q_1), \dots, R(Q_n)$ , respectively. If all the synonyms were as good as the original gene name, we could have simply merged all these result lists to achieve query expansion. Unfortunately, some synonyms may be distracting and others may not be as reliable as the original gene name, so we propose two heuristics to filter out distracting synonyms and assign appropriate weights to the remaining synonyms. Specifically, we measure the “reliability” of a synonym  $g_i$  based on the overlap of  $R(Q_i)$  and  $R(Q)$  defined as follows:

$$\omega(g_i) = \frac{|R(Q) \cap R(Q_i)|}{N}$$

Clearly,  $\omega(g_i) \in [0, 1]$  and  $\omega(g) = 1$ .

We can thus give each synonym a weight proportional to its overlap value and merge the retrieval results of the original query and all the synonym queries accordingly. Intuitively, as long as a passage is scored well for one of these queries, it would be likely a relevant passage. That is, we can imagine we have a combined disjunctive query formed based on these different versions of query. their retrieval results by ranking all the passages based on their best scores in all the results subject to weighting based on the overlap values. Specifically, let  $p$  be a passage and  $s(p; Q_i)$  be its score w.r.t. query

$Q_i$ . We compute a new score for  $p$  as follows:

$$s(p; Q, S) = \max\{s(p; Q), \lambda \max_{g_i \in S} \omega(g_i) s(p; Q_i)\}$$

where  $\lambda \in [0, 1]$  is a parameter to control the maximum weight a synonym can possibly get so that we can ensure sufficient influence from the original query. In our experiments, we arbitrarily set it to  $\lambda = 0.5$ , which is indeed a quite conservative strategy for expansion because even if  $\omega(g_i) = 1$  for some synonym  $g_i$  (i.e., complete overlap), its scores would only be worth half of those of the original query. The idea of having a weight proportional to  $\omega(g_i)$  is once again to be conservative – synonyms with more overlap would be trusted more. We use the maximum aggregator to implement the desired disjunctive semantics of the combined query.

To further ensure that dramatic drift from the original query context will not happen, we set  $\omega(g_i) = 0$  if  $\omega(g_i)$  is too small (i.e., we really have low confidence on  $g_i$ ); this in effect excludes those potentially distracting gene synonyms when merging results. In our experiments, we used an arbitrary threshold of 0.1 and set  $\omega(g_i) = 0$  if  $\omega(g_i) \leq 0.1$ .

Once we have  $s(p; Q, S)$ , we can simply re-rank the passages using this new score. In our experiments, we first tagged the gene names using a gene recognizer<sup>2</sup>, and then queried the external resource Entrez Gene to get a list of synonyms for each gene. After that, we constructed a synonym query for each synonym and use the original query plus the set of synonym queries to do retrieval. Finally, the retrieved results for each query are put together and re-ranked according to the regularized synonym expansion algorithm described above.

## 4 Conjunctive Query Interpretation

In our preliminary experiments with the 14 training topics, we observed that the baseline KL-divergence retrieval model performed poorly on some topics because some query words were missing in the retrieved results. This is because in the KL-divergence retrieval model, as in many other retrieval models based on the bag-of-words representation, a document can be ranked high through matching some of the highly discriminative query words many times; such a document can be ranked higher than those matching all the query words once. Since the queries of this year’s genomics task usually contain only a few words after stop word removal, we hypothesized that every remaining query word is important so that we would

<sup>2</sup>available at <http://sifaka.cs.uiuc.edu/jiang4/software/GeneRecognition.tgz>

like the returned results to match all the words (i.e., "forcing" a conjunctive interpretation of the query). Of course, we could have used Boolean conjunctive query, but such strict conjunctive semantics may not be robust. For example, if a query contains a word not commonly mentioned in the documents, there may not be such documents that match all the words in the query. The Boolean conjunctive query would fail in this case. Thus we heuristically modified the way we estimate the document model in the KL-divergence retrieval method to impose a "soft" conjunctive semantics on the query.

Specifically, we introduce three more parameters  $\alpha \in [0, 1], \beta \in [0, 1], \gamma \in [1, +\infty)$  in equations 1 and 2 as follows:

$$p(w|\theta_B) = \frac{\sum_{d \in \mathcal{C}} c(w, d)^\alpha + \gamma}{\sum_{w \in V} \sum_{d \in \mathcal{C}} c(w, d)^\alpha + \gamma \cdot |V|} \quad (3)$$

$$p(w|D) = \frac{c(w, D)^\beta + \mu \cdot p(w|\theta_B)}{\sum_{w \in D} c(w, D)^\beta + \mu} \quad (4)$$

Parameter  $\alpha$  is to reduce the difference in the counts of different words, bringing the distribution closer to a uniform distribution, thus reducing the effect of IDF weighting; when  $\alpha = 0$ , there will be no difference between the words seen in the collection. Parameter  $\gamma$  is to further "shrink" the background language model toward a uniform distribution over *all* words, including those not seen in the collection; indeed, when  $\gamma$  approaches infinity,  $p(w|\theta_B)$  would be uniform over all the words. Similar to  $\alpha$ , parameter  $\beta$  is to reduce the influence of TF weighting; when  $\beta = 0$ , all terms would have the same TF (as if each occurred just once).

Clearly, equations 1 and 2 are special cases of equations 3 and 4 when we set  $\alpha = 1, \gamma = 1, \beta = 1$ . To simulate a conjunctive Boolean query, we could set  $\gamma$  to a very large value (e.g., 100,000),  $\alpha = 0$ , and  $\beta = 0$ . This setting would score a document essentially based on the number of query words matched, thus achieving an effect of scoring based on a conjunctive interpretation of the query. By changing these parameters, we can flexibly vary our interpretation of the query.

## 5 Experiment Results

We submitted three official runs, including two automatic runs, UIUCsyn and UIUCconj, and one interactive run, UIUCrelfb, each attempting to test a different hypothesis. UIUCsyn uses the proposed regularized synonym expansion method on top of the standard KL-divergence retrieval model. UIUCconj is KL-divergence

retrieval plus robust pseudo feedback as described in Section 2 but with conjunctive scoring instead of the normal Dirichlet prior smoothing (i.e., setting  $\alpha = \beta = 0$  and  $\gamma = 5000$ ). UIUCrelfb is a relevance feedback run using the standard mixture model feedback in the Lemur toolkit [7]. Table 1 shows the performance of these three official runs.

	DocMAP	PsgMAP	Psg2MAP	AspMAP
UIUCsyn	0.1962	0.0633	0.0391	0.1629
UIUCconj	0.1495	0.0497	0.0296	0.1302
UIUCrelfb	0.1940	0.0811	0.0364	0.1452

Table 1: Results of the three official runs.

To facilitate comparisons, we further tested two baseline runs: Baseline1, which is the standard KL-divergence with Dirichlet smoothing, and Baseline2, which does robust pseudo feedback on top of the standard KL-divergence with Dirichlet smoothing. We now present some preliminary analysis of our experiment results.

### 5.1 Gene Synonym Expansion

To see how effective the proposed regularized synonym expansion method is, we compared UIUCsyn with the corresponding Baseline1 (which is only KL-Divergence retrieval without any query expansion techniques). The results are shown in Table 2. We see that using gene synonym expansion improved over the baseline method in document-level MAP by 10%, while the difference in other measures appears to be little (it slightly improves PsgMAP and AspMAP, but decreases Psg2MAP). Further analysis is needed to understand this behavior.

	DocMAP	PsgMAP	Psg2MAP	AspMAP
UIUCsyn	0.1962	0.0633	0.0391	0.1629
Baseline1	0.1777	0.0628	0.0392	0.1624

Table 2: Effectiveness of regularized synonym expansion

### 5.2 Conjunctive Query Interpretation

In UIUCconj, we meant to replace the document model estimation in KL-Divergence model by equations 3 and 4 by setting  $\alpha = \beta = 0, \gamma = 5000$  in the first round of retrieval, then to perform pseudo relevance feedback using the Robust Feedback model introduced in Section 2, and finally to retrieve using standard KL-Divergence model. But we made a mistake by using conjunctive scoring in both the initial retrieval and the final round of retrieval.

Obviously, it would not make sense to impose a conjunctive semantics on the expanded query model. So we implemented another two runs called UIUCconj1 and UIUCconj2. UIUCconj1 used the conjunctive scoring for the initial retrieval, but did not perform any query expansion. UIUCconj2 did pseudo feedback on the results from UIUCconj1 and used standard KL-Divergence model to do final retrieval.

We describe the theses runs together with Baseline1 and Baseline2 in Table 3 (where conj is short for "conjunctive scoring", fb is short for "robust pseudo feedback", std is short for "standard KL-Divergence scoring") and compare their performance in Table 4. Our preliminary experiments showed that the conjunctive query interpretation improves the retrieval performance on the 14 training topics as well as on last year's TREC Genomics topics. However, we see different results in this year's topics: the general performance of these runs in ascending order is: UIUCconj1 < UIUCconj < UIUCconj2 < Baseline1 < Baseline2. So our tentative conclusions are: (1) Conjunctive scoring is not as effective as standard KL-Divergence scoring in this task. (2) Robust pseudo feedback consistently improves performance.

Some possible explanations of the failure of conjunctive scoring are: (1) The performance depends on different kinds of query topics. The conjunctive semantics hypothesis may not hold for all the query topics. When we further checked the performance of each topic, we found that out of totally 36 topics, UIUCconj (compared with Baseline2) improved 12 in document-level MAP, 7 in PASSAGE2 MAP, and 12 in aspect-based MAP. In some cases, the improvement is substantial, but in some other cases, the decrease is also substantial. (2) We tuned our parameters  $\alpha$ ,  $\beta$ , and  $\gamma$  based on the performance on the limited training data, which might be biased. Indeed, since we only tested the most aggressive conjunctive scoring setting, it is still possible some less aggressive setting could be beneficial.

Run Name	Description
UIUCconj	conj + fb + conj
UIUCconj1	conj
UIUCconj2	conj+ fb + std
Baseline1	std
Baselin2	std + fb + std

Table 3: Description of Runs.

	DocMAP	PsgMAP	Psg2MAP	AspMAP
UIUCconj	0.1495	0.0497	0.0296	0.1302
UIUCconj1	0.1369	0.0444	0.0262	0.1190
UIUCconj2	0.1688	0.0587	0.0351	0.1368
Baseline1	0.1777	0.0628	0.0392	0.1624
Baselin2	0.1918	0.0663	0.0422	0.1504

Table 4: Conjunctive scoring performance.

### 5.3 Relevance Feedback

In addition to the exploration of synonym expansion and conjunctive scoring, we also experimented with relevance feedback in our interactive run UIUCrelfb. To obtain feedback information from users, we first used KL-divergence retrieval method to retrieve top 10 candidate passages for each query. We then asked two domain experts to judge the relevance of these top 10 candidate passages for each query. The candidate passages that were judged to be relevant were then used for feedback with a standard mixture model implemented in the Lemur toolkit [7]. The feedback coefficient was set to be 0.5, so it is a very conservative relevance feedback. Some topics had no true relevant documents among the top-10 retrieved documents; for these topics, we did not use any feedback.

We compared UIUCrelfb (relevance feedback) with both Baseline1 (no feedback) and Baseline2 (pseudo relevance feedback) in Table 5. Both relevance feedback and pseudo relevance feedback improve the performance in document and passage measures, but the aspect-level performance drops. One explanation may be that the expanded query model after feedback is biased towards the aspects in the feedback documents, making it difficult for the final results to cover other aspects. Comparing two different ways of feedback, we can see that UIUCrelfb is better in document-level MAP and Passage MAP, but much worse in PASSAGE2 MAP (which is the main evaluation measure for passage retrieval this year). Comparing UIUCrelfb with Baseline1 (no feedback) also shows that UIUCrelfb increases DocMAP substantially, but at the same time, also decreases PASSAGE2 MAP substantially. Further analysis is needed to better understand these apparently contradictory behavior.

	DocMAP	PsgMAP	Psg2MAP	AspMAP
UIUCrelfb	0.1940	0.0811	0.0364	0.1452
Baseline1	0.1777	0.0628	0.0392	0.1624
Baselin2	0.1918	0.0663	0.0422	0.1504

Table 5: Comparison of relevance feedback, pseudo feedback, and no feedback.

## 5.4 Passage Post-processing

We also evaluated the effectiveness of passage post-processing by comparing the performance of each run with its performance before post-processing. The results are shown in table 6 where runs without post-processing are named with an apostrophe. We can make several interesting observations: (1) post-processing consistently hurts document-level MAP and Passage MAP in all the cases; (2) except for the case of UIUCrelfb, post-processing consistently improves PASSAGE2 MAP and aspect-based MAP. Again, further analysis is needed to understand why it behaves differently for these different measures. It might suggest that there is some bias in some of these measures, while at the same time it also appears that post-processing is an effective strategy to improve the primary measure (i.e., PASSAGE2 MAP) for this year’s task regardless of the retrieval method used.

	DocMAP	PsgMAP	Psg2MAP	AspMAP
UIUCsyn	0.1962	0.0633	0.0391	0.1629
UIUCsyn'	0.1999	0.0776	0.0384	0.1483
UIUCconj	0.1495	0.0497	0.0296	0.1302
UIUCconj'	0.1506	0.0624	0.0277	0.1229
UIUCrelfb	0.1940	0.0811	0.0364	0.1452
UIUCrelfb'	0.1975	0.0822	0.0378	0.1458
baseline1	0.1777	0.0628	0.0392	0.1624
baseline1'	0.1789	0.0736	0.0354	0.1506
baseline2	0.1918	0.0663	0.0422	0.1504
baseline2'	0.1935	0.0776	0.0379	0.1403

Table 6: Demonstration of Post-Processing.

## 6 Summary

In summary, we applied language modeling approaches to this year’s Genomics retrieval task, and proposed some new methods to extend a standard language modeling approach to address two special needs for the task: (1) gene synonym expansion and (2) conjunctive query interpretation. In addition, we also experimented with relevance feedback with language models.

Overall, our experiment results show that the standard KL-divergence retrieval method and the model-based feedback method are robust for this special domain retrieval task. Our new methods tend to have mixed results and often exhibit different behavior in document-level MAP and in PASSAGE2 MAP; they often improve the former but not the latter. This suggests that passage retrieval is not a trivial task and deserves further study, and we need to do more diagnostic experiments and fur-

ther analysis to better understand some of the apparently “contradictory” phenomena and make more confident conclusions.

## 7 Acknowledgments

This material is based in part upon work supported by the National Science Foundation under award number 0425852.

## References

- [1] W. R. Hersh et al. TREC 2006 genomics track overview, proceedings of TREC 2006.
- [2] J. Jiang, X. He, and C. Zhai. Robust pseudo feedback estimation and HMM passage extraction: UIUC at TREC 2006 genomics track. In *Proceedings of TREC 2006*, 2006.
- [3] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, 2001.
- [4] V. Lavrenko and W. B. Croft. Relevance-based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, 2001.
- [5] T. Tao and C. Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*, 2006.
- [6] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*.
- [7] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the 10th ACM International Conference on Information and Knowledge Management (CIKM'01)*, 2001.