Regression Rank: Learning to Meet the Opportunity of Descriptive Queries

Matthew Lease[1], James Allan[2], and Bruce Croft[2]

[1] Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912-1910 USA
mlease@cs.brown.edu

[2] Center for Intelligent Information Retrieval (CIIR)
University of Massachusetts Amherst
140 Governors Drive, Amherst, MA 01003-9264 USA
{allan,croft}@cs.umass.edu

Abstract. We present a new *learning to rank* framework for estimating context-sensitive term weights without use of feedback. Specifically, knowledge of effective term weights on past queries is used to estimate term weights for new queries. This generalization is achieved by introducing secondary features correlated with term weights and applying regression to predict term weights given features. To improve support for more focused retrieval like question answering, we conduct document retrieval experiments with TREC description queries on three document collections. Results show significantly improved retrieval accuracy.

1 Introduction

Term-based approaches to document retrieval are remarkably expressive: a vast number of rankings are possible given different settings of individual term weights. Practical effectiveness, however, depends heavily on how accurately estimated term weights match the user's underlying information need. Standard formulations of vector similarity [21], the probabilistic approach [23], and query-likelihood [18] adopt a fixed weighting scheme that ignores query context and any observational evidence from past queries. We show such information can be leveraged by supervised estimation to significantly improve the accuracy of term-based retrieval. Our framework also allows term-based models to be extended with arbitrary new features, enabling incremental transition from term-based approaches toward richer query and document representations.

Our particular interest is to improve retrieval accuracy for verbose, descriptive queries like "What criticisms have been made of World Bank policies, activities or personnel?" (TREC topic 331). Document retrieval for such queries plays an important role as the foundation and fall back technology for more focused retrieval like question answering. However, in comparison to shorter and less informative keyword queries like "World Bank Criticism", verbose queries actually tend to yield worse retrieval accuracy with current retrieval methods [2, 8]. At the Reliable Information Access workshop [4], detailed failure analysis of verbose queries found that in most cases, significantly improved retrieval could be achieved by simply better estimating term weights (e.g. by emphasizing key terms found in the query and related documents). Historically, the most common approach taken to improve upon context-independent term weighting has been to incorporate feedback: the overall query is used to find related documents, which are then used to re-weight and expand the original query [11, 13, 25]. Unfortunately, explicit feedback requires user interaction, and blind feedback performs inconsistently depending on retrieval accuracy of the original query.

Instead, we pursue supervised estimation as an alternative means of incorporating query context (and improving initial retrieval accuracy for any subsequent blind feedback performed). Imagine we have observed some number of past queries along with implicit or explicit evidence of their relevant documents. Employing such evidence to better estimate term weights for novel queries poses several challenges. For a vocabulary of N words, a context-independent term-based model requires estimating N parameters for each query (terms not present in the query are still typically assigned some non-zero weight). Even if one could accurately estimate this many parameters, say from a commercial search engine log, the learned model would still ignore query context. If one did wish to model context-sensitivity, a straightforward approach would require an exponential number of parameters and so be completely impractical. Consequently, recent work in *learning to rank* (LTR) [7] has backed off from modeling individual words and instead employed aggregate lexical measures alongside additional non-lexical features. While aggregating lexical features certainly simplifies learning, it abandons the expressiveness of modeling individual terms.

Regression Rank presents a middle way between recent LTR and classic approaches, intended to capture the best of each: we can continue to leverage individual terms, predict contextual term weights given past queries, and incrementally add other features. Given a term-based retrieval model (§2.1) and a set of training queries with relevant documents, we begin by estimating effective term weights for each query (§2.2). To reduce the parameter space for effective generalization, we define secondary features correlated with term weights (§2.3). Finally, a regression function is learned to predict term weights for novel queries using secondary features (§2.4). While we restrict attention here to term-based retrieval, the retrieval model can be extended by arbitrary additional features given correlated secondary features for predicting retrieval model feature weights.

To evaluate our approach, we conduct retrieval experiments with TREC description queries on three document collections (§3). Results show both significantly improved retrieval accuracy and a large potential for further improvement.

2 Method

This section describes Regression Rank's four components:

1. A retrieval model (parameterized uniquely for each query)
2. A procedure for estimating retrieval model parameters on a given query
3. A set of secondary features correlated with retrieval model parameters
4. A regression procedure to infer retrieval model parameters from features

2.1 The Retrieval Model

Generally speaking, any parametric retrieval model can be used in our framework. The only real constraint is the need for secondary features which correlate with model parameters and can be practically extracted from queries. In this work, we adopt classic term-based retrieval and use our framework to estimate context-sensitive term weights. We plan to enrich the retrieval model in future work by combining individual terms with other features (§4).

Of the three classic approaches [18,21,23], we adopt query likelihood. Each observed document D is assumed to be generated by an underlying language model parameterized by Θ^D. Given an input query $Q = q_1 \dots q_m$, we infer D's relevance to Q as the probability of observing Q as a random sample drawn from Θ^D. If we further assume bag-of-words modeling, Θ^D specifies a unigram distribution $\{\theta^D_{w_1} \dots \theta^D_{w_N}\}$ over the document collection vocabulary $V = \{w_1 \dots w_N\}$. Given these definitions, query likelihood may be succinctly expressed as $log\, p(Q|\Theta^D) = \sum_{i=1}^m log\, \theta^D_{q_i}$. This formulation is somewhat cumbersome, however, since the relative importance of query terms can only be expressed by their relative frequency. Fortunately, we can arrive at an equivalent and more convenient formulation by explicitly modeling the user's information need [10]. Specifically, we assume the observed Q is merely representative of a latent query model parameterized by $\Theta^Q = \{\theta^Q_{w_1} \dots \theta^Q_{w_V}\}$, consistent with intuition that the underlying information need might be verbalized in other ways than Q. Letting f^Q_w denote the frequency of word w in Q, query likelihood may be re-expressed in terms of Θ^Q's maximum-likelihood (ML) estimate, $\frac{1}{m}\{f^Q_{w_1} \dots f^Q_{w_V}\}$:

$$log\, p(Q|\Theta^D) = \sum_{i=1}^{m} log\, \theta^D_{q_i} = \sum_{w \in V} f^Q_w \; log\, \theta^D_w = m \sum_{w \in V} \hat{\theta^Q_w} \; log\, \theta^D_w \overset{rank}{=} -D(\widehat{\Theta^Q} || \Theta^D)$$

where $\overset{rank}{=}$ denotes rank-equivalence. This derivation shows that inferring document relevance on the basis of Q's likelihood given Θ^D has an alternative explanation of ranking based on minimal KL-divergence between Θ^Q and Θ^D (assuming Θ^Q is estimated by ML). The significance of this for our task is showing query likelihood's implicit ML assumption that all query tokens are equally important to the underlying information need. While this assumption appears fairly benign for keyword queries, it is problematic for verbose queries because natural language terms greatly vary in their degree of correlation with the core information need. Fortunately, we see by this same token a clear opportunity to improve retrieval accuracy by adopting a more effective estimation technique.

While estimation of both Θ^Q and Θ^D impacts retrieval accuracy, our focus in this paper is better estimating the query model underlying verbose queries. Consequently, we adopt standard Dirichlet-smoothed estimation of Θ^D, inferring $\hat{\theta^D_w}$ as a mixture of document D and document collection C ML estimates [26, 14]: $\hat{\theta^D_w} = \lambda \frac{f^D_w}{|D|} + (1-\lambda)\frac{f^C_w}{|C|}$, $\lambda = \frac{|D|}{|D|+\mu}$, where μ specifies a fixed hyper-parameter strength of the prior in smoothing. This reduces parameterization of our query likelihood approach entirely to the query model Θ^Q. Our subsequent estimation goal, therefore, will be accurate prediction of Θ^Q on novel queries.

2.2 Estimating the Query Model

A key idea of our approach is that one can generalize knowledge of successful query models (§2.1) from past queries to predict effective query models for novel queries. In order to do this, we must have query models to generalize from. This requires a method for estimating a query model Θ^Q for each training query given examples of its relevant (and possibly non-relevant) documents. This is akin to explicit feedback [13], only we are performing this feedback on training queries rather than the input query.

We apply the simple yet effective strategy of grid search [17]: sampling retrieval accuracy from a target metric space at regular points corresponding to candidate query models. Estimating the query model based on metric performance rather than likelihood avoids the issue of metric divergence [17] and makes it easy to re-tune the system later according to a different metric if so desired. A few details of our grid search approach merit further detail. First, grid search requires choosing the granularity of assignments to sample. Our choice of granularity reflects a reinterpretation of earlier work in query reduction [8]. This prior work generated all possible reductions (i.e. term subsets) of a verbose query and then explored alternative methods of picking the right subset. In the spirit of the earlier derivation (§2.1) in which query formulation was transformed into query model estimation, we let query reductions define the set of grid points at which to evaluate retrieval accuracy in the metric space. Considering all such reductions provides fairly robust coverage of the query model's effective assignment space. Because previous work showed most optimal query reductions contained six or fewer terms [9], we adopted this as an efficiency expedient, limiting our sampling to query models containing six or fewer non-zero parameters.

The second noteworthy detail concerns how the query model is estimated once samples have been obtained from the metric space. The easiest solution would be to simply pick the query model whose sample achieved maximum score on the target metric. However, it turns out this is not the most effective strategy in the context of our framework. Recall our objective is to enable eventual regression across queries (§2.4). The problem with the easy solution above is that subsequent regression will be based on a single sample that may be drawn from a sharply-peaked local maximum on the metric surface. This would mean that were we to attempt to recover this parameterization via regression, small regression errors could yield a significant drop in metric performance. Instead, we estimate Θ^Q as the *expected* query model $\widehat{\Theta^Q} = \sum_s [\text{ Metric}(\Theta_s)\Theta_s]$, a sum in which each sample query model Θ_s is weighted by the retrieval accuracy it achieved[3]. The intuition here is that this expectation should yield parameter values tending to perform well in general, and so the parameterization will more likely correspond to a smoother portion of the metric surface.

Finally, to provide a more stable basis for regression, we perform a non-linear normalization after which the expected query models fully span the interval $[0, 1]$. On the development set (§3), this yielded a consistent improvement.

[3] Technically this sum should be normalized to yield a proper distribution, but since query-likelihood is a linear model, ranking is invariant to scaling of the parameters.

2.3 Secondary Features

Given examples of past queries and corresponding inferred query models, our next task is to identify secondary features. These features should both correlate with the query model and generalize across queries so that we may predict appropriate query models on future queries. This section describes our current feature set; a complete listing appears in Table 1. While existing features have proven effective, their paucity and simplicity show exploration of the feature space remains an important topic for future work.

Parameter	**Description**
$Q = q_1 \dots q_m$, i	Query Q of length m, indexed by i
C, N	Collection C containing N documents
n, w	Integer scalar & lexical token (parameters)
T	Part-of-speech tag-set

Feature Template	**ID**	**Type**	**Definition**
term frequency: $tf(C,Q,i)$	1	integer	tf_i: raw frequency of q_i in C
	2	real	$tf_i / \max_j^m tf_j$
	3	real	$tf_i / \sum_j^m tf_j$
	4	real	$log(tf_i)$
	5	real	$log(tf_i / \max_j^m tf_j)$
	6	real	$log(tf_i / \sum_j^m tf_j)$
document frequency: $df(C,Q,i)$	7	integer	df_i: # documents in C containing q_i
	8	real	$df_i / \max_j^m df_j$
	9	real	$df_i / \sum_j^m df_j$
	10	real	$log(df_i)$
	11	real	$log(df_i / \max_j^m df_j)$
	12	real	$log(df_i / \sum_j^m df_j)$
residual idf: $ridf(C,q_i)$ [2]	13	real	$log(N/df_i) - log(1/1 - e^{\alpha_i})$, $\alpha_i = tf_i/N$
Google tf: $gtf(q_i)$ [2]	14	integer	raw frequency of q_i in Google 1-grams
stopword: $stop(q_i)$	15	boolean	is q_i a stopword?
q_i's location in Q: $loc(i,m,n)$	16	boolean	does $i = n$? (query initial)
	17	boolean	does $m - i = n$? (query final)
lexical info: $context(Q,i,w)$	18	boolean	does $q_{i-1} = w$?
	19	boolean	does $q_{i+1} = w$?
	20	boolean	is q_i trailed by comma?
part-of-speech: $pos(q_i,T)$	21	boolean	is $tag(q_i) \in T$

Table 1. Secondary features used to predict the query model. We define $log(0) \equiv 0$ and $\frac{anything}{0} \equiv 0$ to account for out-of-vocabulary query terms. Features are parameterized templates, instantiated with various settings to produce multiple feature instances.

Query model parameters can be understood as expressing relative term importance within the context of the overall query. As such, it should not be surprising that the classic statistics of term frequency (tf) and document frequency (df) appear in our feature set (Features 1-12) to model term ubiquity and specificity, respectively. Since we are interested in relative rather than absolute term importance, we also compute these statistics relative to the other query terms (i.e. normalized) as well as in raw form. In addition to these classic statistics, we follow previous work [2] to employ Google 1-gram tf [3] and residual inverse-*df* (*idf*) statistics (Features 13-14). The massive volume of the former is intended to provide another useful estimator of term frequency, particularly in the case of small collections, and the latter assumes important terms can be detected by distributional deviation from Poisson. While Google-based statistics provide a useful measure of term frequency on the Web, we also found it useful to gather the above collection-based statistics (i.e. tf, *idf*, and residual *idf*) from Gigaword [6] in addition to the target retrieval collection. This is reflected in Table 1's notating these feature *templates* as parameterized by a collection argument C to produce different feature *instances* for each collection. Use of out-of-domain data was motivated by previous work's empirical evidence of increased correlation between term importance and *idf* as collection size grows [2], as well as another line of prior work having demonstrated significant retrieval benefit from leveraging external corpora [5, 14]. A final traditionally-inspired feature, $stop(q_i)$ (Feature 15), asks whether or not a given query term appears in the stop list (§3). While we do employ deterministic stopping, we stop before stemming to avoid accidental stemming collisions with the stop list. Nevertheless, stop words produced by stemming often are in fact unimportant to the query, and including a feature comparing stemmed words to the stop list proved useful.

Features 16-17 (*location*) correlate term importance with proximity to the start or end of the query string (experiments in §3 set $n = 5$ as the window size), and we found it beneficial to instantiate this feature for both the user's original query and its normalized version used in retrieval (i.e. after stopword removal, converting hyphenated compounds into separate terms, etc.). Features 18-20 (*context*) correlate term importance with presence of certain surrounding terms or punctuation. All possible terms were considered during feature collection, but few actually survived to instantiation due to feature pruning (see below). Feature 21 asks whether a given term's part-of-speech is a member of a given tag-set, correlating tag-sets with term importance. Given that the only distinction currently employed is distinguishing nouns and verbs from other categories, our implementation admittedly reflects a bit of over-engineering: we fully parse the original query strings with a treebank parser [15] after detecting sentence boundaries [20]. While tags might be more easily obtained, this was done to support future work exploring syntactic features.

Because a given statistic will be more reliably estimated under more frequent observation, we employed feature pruning to discard any instantiated feature that was not observed at least a parameter η times in the training data; we set $\eta = 12$ based on development set tuning (§3). As mentioned earlier, this significantly reduced the number of lexical features and generally helped filter out chance correlations from sparse features. Non-sparse features like tf which occur for every term were unaffected by pruning. Following previous work [7], feature values were normalized to the interval $[0, 1]$.

2.4 Inferring the Query Model via Regression

Given examples of target term weights paired with corresponding secondary features, our final task is to predict the query model given the features. We accomplish this via a standard technique of regularized linear regression.

Given N query terms in the training data, let $Y = \{y_1 \ldots y_N\}$ denote the target term weights and $\mathbf{X} = \{X_1 \ldots X_N\}$ the feature vectors. Next, let d denote the number (i.e. dimensionality) of features and $X_i = \{x_i^0, x_i^1 \ldots x_i^d\}$ the ith feature vector (with $x_j^0 = 1$ by definition for all j). Also, let $W = \{w_0 w_1 \ldots w_d\}$ denote the weight vector with w_0 as the bias term. Assuming $\mathbf{X}$ and Y are drawn from the joint distribution $p(X, y)$, our goal is to minimize expected loss given our prediction $f(X, W)$: $\mathbb{E}_{(X,y)\backsim p}[L(f(X, W), y)]$. Lacking oracle knowledge of $p(X, y)$, we approximate this with the empirical loss $\sum_i^N L(f(X_i, W), Y_i) = \sum_i^N (y_i - \sum_{j=1}^d w_j x_i^d)^2 = (Y - \mathbf{X}W)^T (Y - \mathbf{X}W)$ and minimize to find an optimal weight vector W^*. Conveniently, this *sum of least squares* optimization problem has a closed form solution: $W^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$. However, since this ML solution often overfits, we can alternatively revise the empirical loss formulation as $\sum_i^N L(f(X_i, W), Y_i) = (Y - \mathbf{X}W)^T (Y - \mathbf{X}W) + \beta W^T W$ where β defines a regularization parameter. This L2 (i.e. ridge) regression also has a closed-form solution: $W^* = (\beta I + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$, where I denotes the identity matrix. In addition to ML and L2, we also tried L1 (i.e. lasso) regression, which penalizes the absolute value of W is instead of its square. While lasso regression does not have a closed-form solution, many techniques exist for computing it.

On the development set (§3), experiments measuring squared loss of ML, L1, and L2 methods found L2 consistently performed best, with manual sweep of β finding an optimal setting at $\beta = 1$. Consequently, we adopted L2 with this setting of β in our retrieval experiments.

3 Evaluation

Collection	# Docs	Topics
Robust04	528,155	301-450,601-700
W10g	1,692,096	451-550
GOV2	25,205,179	701-850

Table 2. Collections and topics used. All development was performed on 149 Robust04 topics (301-450 except 342); remaining topics and collections were reserved for blind evaluation. Final results (Table 3) use all available data.

We evaluated Regression Rank on three TREC collections of varying size and content (Table 2). Given our interest in improving support for focused retrieval like question answering, our document retrieval evaluation centers on description queries. Model training used 5-fold cross-validation, and Indri [24] was used for retrieval. Mean-average precision (MAP) and top-5 precision (P@5) are taken from `trec_eval` 8.1[4]. Results marked significant†($p < .05$), highly significant‡($p < .01$), or neither reflect agreement between t-test and randomized test statistics computed by

[4] `http://trec.nist.gov/trec_eval`

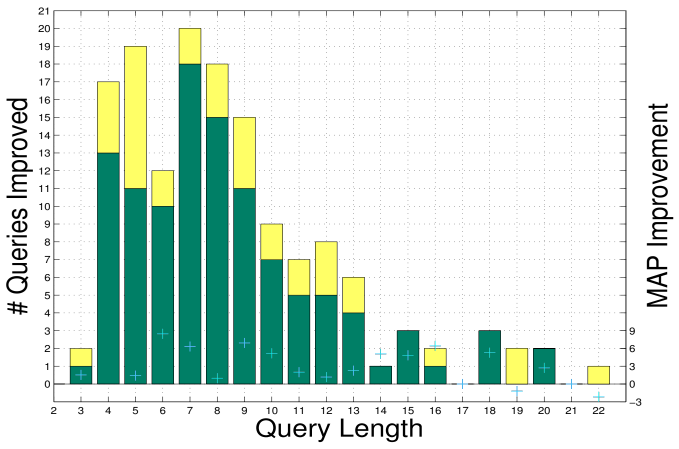

Fig. 1. Retrieval accuracy improvement on the development set as a function of query length. Bars show the number of queries for each query length and ratio improved. MAP improvement achieved at each length is marked by '+'.

`ireval` [22]. Experimental conditions reproduce those of previous work [2] for fair comparison. Queries were stopped at query time using the same 418 word INQUERY stop list [1] and then Porter stemmed [19]. The same Dirichlet parameter $\mu = 1500$ (§2.1) was used. Queries generated by Regression Rank and other experimental data is available online for download[5].

We first present results for the development set (Table 2). For a baseline, we follow standard practice and estimate the query model Θ^Q by maximum-likelihood (ML), assigning uniform weight to each query token. Using ML estimation, title queries achieve $2.83\%^{\ddagger}$ higher MAP (absolute) than the more informative description queries. Additional terms introduced by description queries tend to individually correlate more weakly with the core information need and should generally be assigned lower weight in Θ^Q. ML fails to do this and retrieval accuracy suffers as a result. By better estimation, Regression Rank is able to improve $4.17\%^{\ddagger}$ over ML description accuracy and 1.34% over ML title accuracy.

When comparing between retrieval accuracy of title and description queries, analyzing the effect of verbosity is occasionally complicated by important title words missing from the descriptions[6]. In these cases, title queries may benefit from being more informative in addition to being more focused. To control for this, we identified 122 development set topics for which all title words were contained in the descriptions, and we evaluated this topic subset. The difference in ML estimated title accuracy over description accuracy fell 0.5% to $2.31\%^{\dagger}$ (absolute). Furthermore, Regression Rank showed greater improvement of $4.54\%^{\ddagger}$ over ML estimated description queries and $2.23\%^{\ddagger}$ over ML estimated title queries.

[5] `http://www.cs.brown.edu/people/mlease/ecir09`

[6] Name variations also occur. For example, earlier Key Concepts work [2] noted that title and description queries differed in use of "United States" vs. "U.S." and pre-processed queries to use the latter form exclusively in their published results.

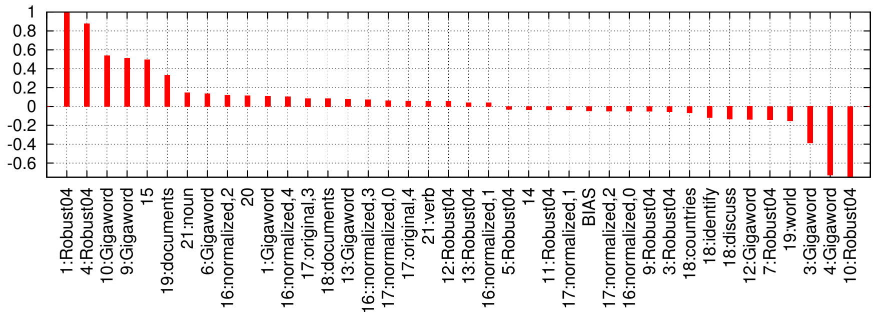

Fig. 2. Learned regression weights for secondary features on the development set. Feature "1:Robust04", Robust04 raw term frequency tf, was assigned weight ≈ 4.1 and is shown here clipped. Features assigned weight $|w| < 0.03$ are not shown. Robust04 document frequency df ("10:Robust04") is seen to have the largest negative weight.

Figures 1 and 2 present additional analysis of development set results. Figure 1 examines change in retrieval accuracy as a function of query length and shows improvement is achieved across lengths (in terms of both number of queries improved and MAP improvement). Figure 2 shows the learned regression weights assigned to secondary features. Features are identified by their ID from Table 1 and the argument to the feature template (e.g. "21:noun" corresponds to Feature 21 where the term's part-of-speech is a kind of noun). Term frequency and document frequency are seen to define the extremes of positive and negative correlation between feature and term weight.

Recall in §2.2 we estimated a query model Θ^Q for each training query and its relevant documents by sampling retrieval accuracy achieved under different candidate models. Since subsequent regression is based on these estimated query models, better estimation should yield more accurate retrieval following regression. To test this, we tried limiting sampling to queries of 15 words or less, which reduced the total number of samples from 502K to 104K. Performing regression based on this smaller set of samples, retrieval accuracy fell 1.07%‡ (absolute). While these results are certainly sensitive to the sampling procedure used, it nonetheless seems clear that strong estimation of training query models has an important effect on downstream retrieval accuracy. This further suggests additional gains might be realized by better estimation.

Our main results (Table 3) use all queries for all three TREC collections (Table 2). In addition to the ML baseline defined earlier, we also compare to Bendersky and Croft's Key Concepts model [2]. Regression Rank achieves highly significant MAP improvement over ML description accuracy for all collections. Compared to title query accuracy, MAP improvement was highly significant for Robust04 and significant for W10g; both Regression Rank and Key Concepts fail to improve over ML title accuracy for GOV2. Regression Rank also achieves 1.4%‡ and 1.6%† absolute MAP improvement over Key Concepts for Robust04 and W10g, with equal MAP achieved on GOV2.

		Robust04		**W10g**		**GOV2**	
Query	**Model**	P@5	MAP	P@5	MAP	P@5	MAP
Title	ML	48.11	25.32	31.20	19.49	56.24	29.61
Description	ML	47.63	24.51	$39.20^{\ddagger}$	18.61	52.21	25.22
	Seq. Depend. [16]	$49.32_{\dagger}$	$25.64_{\ddagger}$	$38.80^{\ddagger}$	19.14	$56.38_{\dagger}$	$27.40_{\ddagger}$
	Key Concepts [2]	47.55	$25.91_{\ddagger}$	$41.40^{\ddagger}_{\dagger}$	$20.40_{\ddagger}$	$57.05_{\ddagger}$	$27.44_{\ddagger}$
	Regression Rank	$52.05^{\dagger}_{\ddagger}$	$27.33^{\ddagger}_{\ddagger}$	$40.60^{\ddagger}$	$22.01^{\dagger}_{\ddagger}$	54.50	$27.35_{\ddagger}$
	Oracle Regression	60.16	32.01	46.60	27.95	62.60	33.43
	Oracle Reduction		35.07		31.75		36.03

Table 3. Retrieval results using all queries and collections (Table 2) compare alternative term weight estimation methods for description queries. A maximum-likelihood (ML) baseline is compared to Regression Rank and Key Concepts [2] models, as well as a non-unigram sequential dependency model [16]. Our evaluation on W10g and GOV2 is blind whereas Key Concepts was developed using all collections. Key Concepts results were generated by Indri queries Michael Bendersky provided and vary slightly from those in [2]. Oracle results for perfect regression and reduction show potential for further improvement. Title query results under ML estimation are also shown. Score^{t}_{d} superscripts and subscripts indicate significance vs. title and description ML baselines.

We also report retrieval accuracy for two oracle conditions: perfect regression and perfect reduction. Perfect regression shows the retrieval accuracy that would be achieved if we could exactly recover the target expected query models (§2.2). This shows a large potential for further improvement by better estimation. Perfect reduction results show even greater accuracy is possible if we could perform accurate regression of optimal reductions rather than expected query models. However, this would present a further challenge to regression since expected query models are more stable against regression error (§2.2). Finally, since oracle results reflect the best query model found while sampling, better estimation can be expected to show greater oracle accuracy as well.

4 Discussion

Regression Rank and Key Concepts [2] both improve verbose query retrieval accuracy by using supervision to estimate a better unigram query model Θ^{Q}. However, each accomplishes this in rather a different way. For supervision, Key Concepts relies on manual annotation to identify a key noun phrase for each query. This is a difficult task for people to perform on complex queries, particularly when relationships are involved, and human intuition can often be mistaken [8]. In contrast, we leverage existing document relevance annotations (i.e. explicit feedback on training queries) and empirically discover term importance based on a target retrieval metric. Both approaches can benefit from additional training data, and while manual annotation is probably easier with the Key Concepts approach, our approach has the capacity to exploit a much larger body of implicit feedback found in search engine logs. Another important difference is what each approach learns to predict. The Key Concepts approach predicts noun phrase weights rather than term weights. This means all terms outside noun phrases are assigned no weight, and all terms within a noun phrase are assigned equal weight (parameter tying). In order to achieve robust retrieval, Key Concepts therefore mixes its predicted Θ^Q with the original ML estimate. Our approach, on the other hand, makes no *a priori* distinction between terms and completely predicts Θ^Q without use of ML mixing.

In addition to use of implicit feedback, various avenues exist for further improvement. Estimation of Θ^Q given feedback could certainly benefit from using a more sophisticated technique than grid search (§2.2). Our set of secondary features (§2.3) could also certainly be improved, possibly by incorporating features from earlier work on verbose queries [2, 8] or by considering richer features like syntax [12]. In terms of the retrieval model (§2.1), we would like to go beyond unigram modeling to incorporate word interactions. While consistent improvement has already been shown by modeling sequential dependencies [16], this work also embodies an implicit ML assumption that all term pairs are equally relevant to the core information need. Just as we have seen this assumption does not hold for individual terms, the same can be said for term interactions, and we expect better estimation should improve retrieval accuracy here as well.

5 Conclusion

This paper presented a novel *learning to rank* framework for estimating traditional term-based retrieval models in the absence of feedback. This was accomplished by introducing secondary features correlated with term weights and applying regression to predict them as a function of features. Empirical validation with description queries on three TREC collections showed significantly improved retrieval accuracy as well as a large potential for further improvement.

Acknowledgments

Support for this work was provided in part by NSF PIRE Grant No OISE-0530118 and the Center for Intelligent Information Retrieval. We thank the anonymous reviewers and our lab fellows for their useful comments. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

1. J. Allan, M. Connell, W.B. Croft, F.F. Feng, D. Fisher, and X. Li. INQUERY and TREC-9. In *Proc. of TREC-9*, pages 551–562, 2000.
2. M. Bendersky and W.B. Croft. Discovering key concepts in verbose queries. In *Proc. of SIGIR*, pages 491–498. ACM New York, NY, USA, 2008.
3. T. Brants and A. Franz. Web 1T 5-gram v1, LDC Catalog No. LDC2006T13, 2006.
4. C. Buckley and D. Harman. Reliable information access final workshop report. *ARDA Northeast Regional Research Center Technical Report*, 2004.

5. F. Diaz and D. Metzler. Improving the estimation of relevance models using large external corpora. In *Proc. of SIGIR*, pages 154–161, 2006.
6. D. Graff, J. Kong, K. Chen, and K. Maeda. English Gigaword. *Linguistic Data Consortium catalog number LDC2005T12*, 2005.
7. Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai. Learning to rank for information retrieval (lr4ir 2007). *SIGIR Forum*, 41(2):58–62, 2007.
8. G. Kumaran and J. Allan. A Case for Shorter Queries, and Helping Users Create Them. In *Proceedings of NAACL HLT*, pages 220–227, 2007.
9. G. Kumaran and J. Allan. Effective and efficient user interaction for long queries. In *Proc. of SIGIR*, pages 11–18, 2008.
10. J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proc. of SIGIR*, pages 111–119, 2001.
11. Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of the 24th ACM SIGIR conference*, pages 120–127, 2001.
12. Matthew Lease. Natural language processing for information retrieval: the time is ripe (again). In *Proceedings of the 1st Ph.D. Workshop at the ACM Conference on Information and Knowledge Management (PIKM)*, 2007. To appear.
13. Matthew Lease. Brown at TREC'08 Relevance Feedback Track. In *Proc. of the 17th Text Retrieval Conference (TREC) Conference*, 2008.
14. Matthew Lease and Eugene Charniak. A Dirichlet-smoothed Bigram Model for Retrieving Spontaneous Speech. In *Proc. of 8th Workshop of the Cross-Language Evaluation Forum (CLEF'07)*, LNCS-5152, pages 687–694. Springer-Verlag, 2008.
15. David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proc. of HLT-NAACL 2006*, pages 152–159, 2006.
16. D. Metzler and W.B. Croft. A Markov random field model for term dependencies. In *Proc. of SIGIR*, pages 472–479, 2005.
17. Donald Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.
18. Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proc. of SIGIR*, pages 275–281, 1998.
19. M. Porter. The Porter Stemming Algorithm. *Accessible at http://www. tartarus. org/martin/PorterStemmer.*
20. Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the fifth conference on Applied natural language processing*, pages 16–19, 1997.
21. Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proc. of SIGIR*, pages 21–29, 1996.
22. Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proc. of CIKM*, pages 623–632, 2007.
23. K. Sparck Jones, S. Walker, and S.E. Robertson. A probabilistic model of information retrieval: development and comparative experiments (parts i and ii). *Information Processing and Management*, 36:779–840, 2000.
24. T. Strohman, D. Metzler, H. Turtle, and W.B. Croft. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligence Analysis*, 2004.
25. C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proc. of CIKM*, pages 403–410, 2001.
26. C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.