

# Term Necessity Prediction

Le Zhao and Jamie Callan  
Language Technologies Institute  
School of Computer Science  
Carnegie Mellon University  
5000 Forbes Ave., Pittsburgh, PA 15213, USA

## ABSTRACT

The probability that a term appears in relevant documents ( $P(t | R)$ ) is a fundamental quantity in several probabilistic retrieval models, however it is difficult to estimate without relevance judgments or a relevance model. We call this value *term necessity* because it measures the percentage of relevant documents retrieved by the term – how necessary a term’s occurrence is to document relevance. Prior research typically either set this probability to a constant, or estimated it based on the term’s inverse document frequency, neither of which was very effective.

This paper identifies several factors that affect term necessity, for example, a term’s topic centrality, synonymy and abstractness. It develops term- and query-dependent features for each factor that enable supervised learning of a predictive model of term necessity from training data. Experiments with two popular retrieval models and 6 standard datasets demonstrate that using predicted term necessity estimates as user term weights of the original query terms leads to significant improvements in retrieval accuracy.

## Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: *Retrieval Models*

## General Terms

Theory, Experimentation, Measurement

## Keywords

Term weighting, Necessity, Mismatch, Ad-hoc retrieval models

## 1. INTRODUCTION

*Term necessity* is defined as the probability of a term  $t$  occurring in documents relevant to a given query, i.e.  $P(t | R)$ , where  $R$  is the set of relevant documents for the query. It is the proportion of relevant documents that match  $t$ , thus, equivalently, it measures *term recall*, or  $1 - \text{term mismatch rate}$ .

Term necessity is not a new idea. It plays an important role in the theory of probabilistic information retrieval, similar to inverse document frequency (idf) [1]. Prior research did not make much progress in predicting necessity, for example setting it to a constant [7], or used simple methods to predict it [2, 14]. And these efforts were not especially successful in understanding necessity. For example, idf – a query independent statistic – was used as the only feature to predict necessity, which is a query dependent probability.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM’10, October 26–30, 2010, Toronto, Ontario, Canada.

Copyright 2010 ACM 978-1-4503-0099-5/10/10...\$10.00.

It has been known since the 1970s that more accurate necessity probabilities offer the possibility of significant improvements in ad-hoc retrieval [1].

This paper revisits the problem of predicting term necessity. It identifies several factors that might cause a term to have low necessity (or equivalently, to mismatch relevant documents). Such factors include a query term not being central to the information need, a query term having synonyms, and a query term being too abstract. This work further develops a small set of easily-computed term-dependent and query-dependent features related to each factor, enabling each query term to be represented as a feature vector. Training data from past queries enables a predictive model to be learned, which can predict the necessity of new terms in previously unseen queries. This approach to predicting term necessity produces significant improvements in ad-hoc retrieval accuracy. It is also easy to extend as new hypotheses are developed about factors that contribute to term necessity.

Term necessity prediction has been a difficult problem for more than 30 years. Our contribution to understanding this problem is a reconsideration of the factors that affect necessity, the expression of these factors as easily-computed features, and the framing of the problem as one of learning a predictive model from data.

The following section provides background for understanding term necessity and how it is instantiated in two well-known probabilistic retrieval models. Section 3 considers term necessity to be a prediction problem. It discusses the factors that might affect term necessity, and how they might be expressed as features. Section 4 presents experiments that investigate the effectiveness of this approach to estimating  $P(t | R)$  in well-known retrieval models. Section 5 discusses related work that is not covered elsewhere. Section 6 concludes.

## 2. BACKGROUND

### 2.1 Roots in Probabilistic Retrieval Models

The Probability Ranking Principle states that the best retrieval effectiveness is achieved when documents are ranked in decreasing probability of relevance to the user that submitted the request, where probabilities are estimated using all available evidence [1]. The Probability Ranking Principle is usually instantiated as ranking by the Odds ratio,

$$\frac{P(R | d)}{P(\bar{R} | d)} \propto \prod_{q_i \in q \cap d} \left( \frac{P(q_i | R)}{1 - P(q_i | R)} \times \frac{1 - P(q_i | \bar{R})}{P(q_i | \bar{R})} \right) \quad (1)$$

where  $R$  is the set of relevant documents,  $\bar{R}$  is the set of non-relevant documents,  $d$  is a document to be ranked,  $q$  is the query, and  $q_i$  is a query term in  $q$ . This is known as the Binary Independence Model (BIM) [1]. And broken down to each query term, the BIM yields the well known RSJ term weight for that term.

Two probabilities determine the final term relevance score:  $P(t | R)$  and  $P(t | \bar{R})$ .  $P(t | \bar{R})$  is closely associated with the document frequency of the term divided by the collection size, as demonstrated in the predictive version of BIM [1] and also con-

firmed by the data analysis by Greiff [2]; it leads to the well-known inverse document frequency (idf) metric.  $P(t | R)$  is  $t$ 's necessity to relevance: the probability that a relevant document contains  $t$ . Historically  $P(t | R)$  has been more difficult to predict.

The BIM itself is not particularly effective, but the later Okapi BM25 [3] uses this term weight as part of the retrieval model. Thus, if there is a way to predict term necessity, the predictions can be directly plugged into BIM, and in turn BM25, without any modification to the retrieval models.

## 2.2 Basing on the Relevance Model

The original generative language model [4] directly scores a document by how likely the document model generates the query, and does not model relevance. The relevance model [17] provides a way to model relevance in the language model framework. Given a search topic, the relevance model is an ideal term distribution  $P_m(t | R)$  for all  $t$ , estimated from true relevant documents if complete relevance judgements are given. When there are no relevance judgements for the current topic, the relevance model can be estimated in an unsupervised manner from top documents of an initial retrieval, thus, instantiated as a query expansion and pseudo-relevance feedback method. The final retrieval status value of a result document is given by the KL-divergence between the relevance model and the document model, or equivalently, weighting each term by its relevance model probability:

$RSV(q, D) = \sum_{t \in V} P_m(t | R) \times \log P(t | D)$ , where  $P_m(t | R)$  is the relevance model, a multinomial distribution over all terms in the vocabulary, i.e.  $\sum_{t \in V} P_m(t | R) = 1$ .

In theory, term necessity is a Bernoulli distribution of whether a term appears or does not appear in a relevant document, i.e.  $P(t | R) + P(\bar{t} | R) = 1$ . Nevertheless, it can be normalized into a multinomial term distribution and fitted into the relevance model:

$$RSV(q, D) = \sum_{q_i \in q} \hat{P}(q_i | R) \times \log P(q_i | D) \quad (2)$$

where  $\hat{P}(q_i | R)$  is the term necessity. (In this work, we only consider the necessity of the query terms, but in theory there is no such restriction.)

It is possible to fit the necessity probabilities into the relevance model because conceptually the relevance model provides more of a representation that allows one to plug in relevance based term weights into the language model retrieval framework, rather than a particular way of estimating these weights.

In fact, the necessity probability based term weights can be shown as a particular way of estimating the relevance model for a given query. Suppose term occurrences are binary, (tf is either 0 or 1), suppose each term occurrence in the relevant documents is an independent observation from the relevance model, and let  $L$  denote the total length of all the  $|R|$  relevant documents for the query (sum of numbers of unique terms), then, the relevance model probability  $P_m(t | R)$  is only a constant factor off its necessity:

$$\begin{aligned} P_m(t | R) &= \text{Count}(t | R) / L = |R| / L * \text{Count}(t | R) / |R| \\ &= |R| / L * P(t | R) \end{aligned} \quad (3)$$

Despite the connection, this work focuses on the supervised prediction of the necessity probabilities, which provides a novel and effective way of estimating the relevance model, and can be used to improve the traditional unsupervised estimation. And at the same time, term necessity, being independent of the average length of the relevant documents, is a more direct and easier objective for prediction than the near 0 multinomial term probabilities of the relevance model.

Empirically, the Bernoulli necessity estimation based on binary term occurrences also leads to better retrieval performance, as shown in Section 4.5.1. This treatment of using a multinomial

language model for query generation, but using instead a Bernoulli model to estimate term necessity creates an inconsistency in the overall model, and demands a theoretically cleaner solution.

## 2.3 Example Necessity Values

Necessity values for 5 example terms from 5 TREC queries are shown below, idf values  $-\log(N/df)$  – are listed for reference:

**Table 1 Necessity values of 5 example terms on sample queries**

Query (TREC 3 Ad-hoc retrieval track)	Term	$P(t   R)$	idf
<i>Oil Spills</i>	<i>spill</i>	0.9914	5.201
<i>Term limitations for US Congress members</i>	<i>term</i>	0.9831	2.010
<i>Insurance Coverage which pays for Long</i>	<i>term</i>	0.6885	2.010
<i>Term Care</i>			
<i>School Choice Voucher System and its effects on the US educational program</i>	<i>effect</i>	0.2821	1.647
<i>Vitamin the cure or cause of human ailments</i>	<i>ail</i>	0.1071	6.405

Table 1 shows that necessity is not uniform for all query terms, nor is it constant for a specific term. For example, ‘*term*’ has two different necessity values in the two different queries. There is also no simple correlation between necessity and idf.

## 2.4 What’s in the Name?

No prior work named this probability. The RSJ term weight [1] – the product term in Equation (1) – has been called ‘‘term relevance’’ by Salton and ‘‘relevance weight’’ by Spärck Jones [21], but not the necessity probability itself (which is effectively the only part of the weight about relevance). We argue that *necessity* is the right name: it shows what this probability is, where it comes from and suggests how it can be used.

*First*, generally  $P(a | b)$  measures in probability how likely event  $a$  occurring is a necessary condition for  $b$  [23], (whether event  $a$  must occur in order for  $b$  to be true.)  $P(t | R) = 1$  means term  $t$  must appear in the document in order for it to be relevant, while, low  $P(t | R)$  means  $t$  is unlikely necessary for relevance.

*Second*, it urges one to reason why a term becomes low necessary. For example, low necessity does not imply that a term is irrelevant to the information need. Synonyms, appearing in place of the query term in relevant documents, cause mismatch and lower necessity. Thinking about factors that affect necessity helps in designing features to predict it for unseen terms in new queries.

*Third*, not only for weighting terms, necessity can also help query reduction and expansion. Intuitively, terms unnecessary for relevance can be safely removed or replaced, while low necessary terms should be expanded, as found in [26].

## 2.5 Prior Work on Necessity Prediction

Prior research had difficulty modeling  $P(t | R)$ , so it focused on  $P(t | \bar{R})$  (idf) and probabilistic indexing (tf). Robertson and Spärck Jones [1] recognized that term necessity could be accurately estimated from relevance judgments, which is the retrospective case of BIM. For a term  $t$ , if it appears in  $r$  number of totally  $|R|$  relevant documents, then an unbiased estimate of necessity is,

$$P(t | R) = r / |R| \quad (4)$$

Without relevance judgments, the predictive case of BIM assumes necessity to be always 0.5 [1]. Croft and Harper treated necessity as a tuned constant (the Croft/ Harper Combination Match [7]). Greiff [2] used a 3-piece linear function of idf to predict the overall term weighting which includes necessity and idf. More recently Metzler [14] predicted necessity as a function of df. But because idf was the only feature used in all of the above work, necessity prediction was inaccurate, and improvements were only shown over simplistic baselines.

### 3. NECESSITY PREDICTION

Our goal is to predict the necessity of each term in a query, without using relevance judgments of the current query. This section presents a framework for using term- and query-dependent statistics to predict necessity. It includes a discussion of factors that affect necessity, related features and the prediction model.

#### 3.1 Problem Definition

We cast the necessity prediction problem into a standard regression prediction problem, where each query term together with its true necessity value is treated as a sample, for training or testing. The query term is represented as a set of features so that necessity values of query terms in the training set can be generalized to predict necessity for test terms previously unseen, and the objective is to minimize the prediction error of term necessity values. More formally, for a set of topics  $Q$  together with the corresponding document collections, a training sample consists of a query term  $q_i$  and its necessity  $P(q_i | R_q)$  estimated from the set of judged relevant documents  $R_q$  (as described in Section 3.2). Each term is represented as a set of features  $f_{j=1..k}(q_i, q)$  depending on the corpus, the term  $q_i$  and the query  $q$ . A regression model  $M$  predicts necessity as a function of the features:

$$\hat{P}(q_i | R_q) = M(f_1(q_i, q), \dots, f_k(q_i, q)) \quad (5)$$

Prediction error  $L_Q(M)$  can be measured in average L1 loss as

$$L_Q(M) = \sum_{q \in Q} \sum_{q_i \in q} |\hat{P}(q_i | R_q) - P(q_i | R_q)| / \#terms \quad (6)$$

Here, using the necessity probability to measure prediction loss is intuitive, but also arbitrary. One may train over odds probability (or log odds), and then translate the predicted odds back into probability. Odds maps  $[0, 1)$  into  $[0, +\infty)$ , stressing the higher probability region. In our retrieval experiments, training in odds probability is slightly more stable than probability or log odds.

#### 3.2 Estimating Ground Truth Term Necessity

To both train and evaluate necessity predictors, ground truth necessity values need to be estimated from relevance judgments.

In practice, the unbiased estimate of term necessity,  $r/|R|$ , can be unstable for topics with a small number of judged relevant documents  $|R|$ , thus, some sort of smoothing is usually applied. We use the standard Laplace smoothing in the form of  $(r + 1)/(|R| + 2)$  to get more reliable estimates of  $P(t | R)$ .

Pooling is a technique used widely in IR evaluation to save judgment efforts, where only top ranked documents from a set of retrieval runs are manually assessed for relevance. Using pooled judgments might bias the estimation of term necessity. Our results on multiple TREC datasets show that sparser judgments still give similar amount of improvement, which suggests that necessity can be effectively estimated on pooled judgments.

#### 3.3 Factors and Features

The main contribution of this work is pointing out some of the factors that might affect term necessity and designing features for them. Listed below are the factors that we discovered.

1. If a term is not **central** to a topic, it is unlikely to be necessary. In practice, we use the occurrences of a term in top returned documents from an initial retrieval to estimate centrality, assuming that terms occurring consistently in top documents are central. For the query “*Vitamin – the cure of or cause for human ailments*”, “*ailments*” appears less consistently in top documents than *vitamin*, and is in fact much less necessary.

2. **Synonyms** may replace the original query term in relevant documents, lowering the necessity of the query term. In “*US educational system*”, if relevant documents say *America* or *USA*

instead of *US*, it lowers the necessity of *US*. For terms with multiple senses, only the synonyms of the queried sense should matter.

3. If a term does not often co-occur with its synonyms, then it is often **replaceable** by the synonyms, and the term is less necessary. Lu et al [13] used a similar measure, synonym novelty, to measure how many new documents a synonym can match.

4. Many **abstract** but **rare** terms are unnecessary, because they are at a different abstraction level than the relevant documents. Examples are *maulings* in “*dog maulings*”, *fundamentalism* in “*Christian fundamentalism*” and *ailments* in the vitamin query. Often, hyponyms of abstract terms appear in relevant documents.

It is important to note that term necessity is different from concept necessity. “*maulings*” is a necessary concept for the “*dog maulings*” query, but the term “*maulings*” is not necessary, because of searchonyms such as *attack* or *bite*.

Our research explores a set of features designed to capture factors 1, 2 and 3. There is no direct way to capture abstractness, so we resort to using features that correlate with factor 4.

In generating the features, we used minimal external information by restricting ourselves on only the corpus and the queries, avoiding sources such as WordNet or query log data, which have the out of vocabulary problem and may bring in their own biases.

Features related to topic centrality, synonymy, and replaceability can be computed based on term similarity in a lower-dimensional space formed by Singular Value Decomposition.

Synonyms may not co-occur in the same document, but they typically occur in similar contexts (i.e. a higher order co-occurrence). As [20] pointed out, Singular Value Decomposition (SVD) identifies these higher order co-occurrences.<sup>1</sup> As input to SVD, entries of the term-document matrix are tf\*idf weighted.

Here SVD is used only to provide features for predicting term weights, no expansion terms are used in the final queries.

SVD can be applied globally to a corpus, or locally to top-ranked documents returned for a given query. Local SVD is more efficient and may improve the quality of synonyms. This is because a local SVD improves the quality of the identified synonyms by focusing on the sense being queried. For words with multiple senses, mutual disambiguation among query terms causes the top-ranked documents to be about the senses intended by the user, thus the synonyms identified are not about an arbitrary sense of a query term, but the sense being queried. Schütze et al [19] used a similar approach, which they called local LSI.

Denote  $S(t, w_1)$  to be the inner product (similarity) of terms  $t$  and  $w_1$  in the SVD concept space. Also assume terms  $w_i$  are in descending order of similarity to the query term  $t$ , i.e.  $S(t, w_1) \geq S(t, w_2) \geq \dots$ , so that, higher in ranking, more likely synonym.

A **topic centrality feature** is defined as

$$f_1(t, q) = S(t, w_1) \quad (7)$$

Term  $w_1$  with maximal similarity to  $t$  is usually  $t$  itself, thus this feature indicates how much weight of the term is preserved after SVD. (When  $t$  does not equal  $w_1$ , we still use  $S(t, w_1)$  as the feature value.) This residual weight is a measure of how close the term is to the space spanned by the top documents. For example, if a term appear frequently in many top documents, then most

<sup>1</sup> Other methods of finding higher order co-occurrences should also work, for example topic models like pLSI or Latent Dirichlet Allocation which may also be used to compute term similarity in a concept space.

of its weight will be kept. Tf\*idf weighting is used in SVD to prevent stopwords from having the highest centrality.

A **synonymy feature** is defined as

$$f_2(t, q) = \sum_{i=2}^{c+1} S(t, w_i) / c \quad (8)$$

Similarity in the concept space indicates the likelihood that two terms are synonyms, so this feature simply takes an average of the next  $c$  highest similarities. We fixed  $c$  to be 5 from a pilot study. As evident in Table 2, this feature can not only capture synonyms, but also antonyms, hyponyms or even misspellings that “must be considered equivalent (to the query term) for search purposes”, which is called **searchonyms** by Richard P.C. Hayden [22].

A **replaceability feature** is defined as

$$f_3(t, q) = \sum_{i=1..6}^{w_i \neq t} \frac{df_i - C(t \& w_i)}{df_i} \times \frac{S(t, w_i)}{S(t, t)} \quad (9)$$

where  $C(t \& w_i)$  is the number of documents in the collection matching both  $t$  and  $w_i$ , and  $df_i$  is the document frequency of  $w_i$ . This is a modified version of the synonymy feature and measures how likely the original query term is replaced by its searchonyms in the documents.  $[df_i - C(t, w_i)] / df_i = P(\bar{t} | w_i)$ , and measures the likelihood that searchonym  $w_i$  matches additional documents that  $t$  does not match. Normalizing by  $S(t, t)$  removes the effect of the topic centrality of term  $t$ . Overall, this feature has a negative correlation (about -0.2) with necessity.

Table 2 shows top similar terms and their similarities to the query term, for one term in each of the 4 queries.

**Table 2. Query terms and their top 5 similar terms using SVD**

<i>Oil spills</i>	<i>Insurance coverage which pays for long term care</i>	<i>Term limitations for US Congress members</i>	<i>Vitamin the cure of or cause for human ailments</i>				
<i>oil</i>	<i>term</i>	<i>term</i>	<i>ail</i>				
<i>spill</i>	0.5828	<i>term</i>	0.3310	<i>term</i>	0.3339	<i>ail</i>	0.4415
<i>oil</i>	0.4210	<i>long</i>	0.2173	<i>limit</i>	0.1696	<i>health</i>	0.0825
<i>tank</i>	0.0986	<i>nurse</i>	0.2114	<i>ballot</i>	0.1115	<i>disease</i>	0.0720
<i>crude</i>	0.0972	<i>care</i>	0.1694	<i>elect</i>	0.1042	<i>basler</i>	0.0718
<i>water</i>	0.0830	<i>home</i>	0.1268	<i>care</i>	0.0997	<i>dr</i>	0.0695

It’s easy to see from Table 2 that searchonyms extracted with SVD are query dependent, the same term ‘term’ results in two different sets of synonyms. But, mutual disambiguation in local SVD is not perfect. For ‘term’ in the “term limitation” query, the word ‘care’ which co-occurs with the ‘long term care’ sense of ‘term’ shows up as similar (0.0997 similarity) to the query term ‘term’. The extracted searchonyms and similarities are not perfect. For example, one query term may become another query term’s searchonym, (‘spill’ is identified as a searchonym for ‘oil’), simply because they co-occur. External resources, such as thesauri or Wikipedia, or better searchonym extraction techniques may be used to improve these features. Nevertheless, our experiments show that the current features based on SVD term similarities can still be used to effectively predict necessity and improve retrieval.

Overall, the SVD features depend on 2 meta-parameters, 1) the number of top retrieved documents to perform SVD and 2) the number of latent dimensions to keep. These parameters are tuned on a development set in experiments using 5-fold cross validation.

A term **rareness feature** is simply defined as the inverse document frequency (idf) of a term. It was often used in prior research e.g. [2, 14] to estimate term weights. Idf is a real-valued term-

specific statistic. We use the specific form of  $idf(t) = \log(N/df)$ , where  $df$  is the document frequency of  $t$  in a corpus of totally  $N$  documents. Together with the abstractness feature below, we use it to capture factor 4 which affects necessity.

The modified terms of TREC queries are usually abstract. For example, in the query “US educational system”, *system* is the head noun being modified by the other two terms, and it is abstract. Since the head is an internal node in a dependency parse tree, while the modifiers are leaves, a binary **abstractness feature** can be defined as whether a term is a leaf node in the dependency tree of the query. We used version 1.6.1 of the Stanford parser (<http://nlp.stanford.edu/software/lex-parser.shtml>), with the output format “typedDependenciesCollapsed” – so that the output can be conveniently transformed into a dependency tree. Our pilot study shows a 0.2 correlation with necessity.

Several other features have been tested, but do not perform as well as the combination of the 5 above listed features.

A first order co-occurrence statistic between two terms was used to measure term similarity. But when combined with the SVD features, this feature does not improve performance.

Term clarity [10] is a characteristic of the term itself, and initially we thought that it would correlate well with necessity. Intuitively, terms with low clarity, such as *effect* or *deficit* will have low necessity. It does perform reasonably well without the SVD features, but when combined with the SVD features, clarity hurts performance. This is expectable because terms with low clarity would not appear consistently in top-ranked documents, thus topic centrality would be low, making clarity more or less redundant.

### 3.4 Regression Model

Regression is typically used to predict continuous variables such as a probability, and in our case, the term necessity probability.

#### 3.4.1 Non-linearity

Features like idf do not have a linear relation with necessity, thus a non-linear model being able to fit more complex prediction curves is a better choice. However, a non-linear model requires more training data, or a smaller number of features to avoid data sparsity. In this work, the model trains on over 400 samples and only 5 features, justifying the use of a non-linear model. More complex or more features may force the model to be linear to achieve high accuracy, such as in Regression Rank [6].

A pilot study confirms that for the current set of features, non-linear kernel regression model outperforms linear regression.

#### 3.4.2 Kernel regression

Kernel regression is an instance based regression method similar to nearest neighbor regression. It weights the necessity probabilities of the nearby training samples according to their similarities to the test term to produce a prediction. A training term closer to the test term shares a higher weight. The rationale for using kernel regression is that when the features are discriminative enough, we can expect terms close to each other in the feature space to have similar necessity. In our pilot study we tested kernel regression with linear, polynomial and RBF kernels using SVM-light version 6.02 (<http://svmlight.joachims.org>). Results show that RBF kernel regression performs the best.

Except simple scaling to make each feature roughly range from 0 to 1, the final regression model treats all features the same. A  $\gamma$  parameter controls the RBF kernel width. A larger  $\gamma$  gives more weights to nearby training instances.  $\gamma$  is the third and last meta-parameter in this work. We tune the meta-parameters on development sets, and report results of 5-fold cross validation.

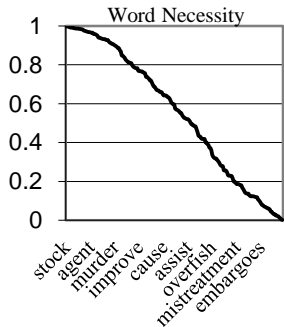


Figure 1. TREC 3 query term necessity in descending order

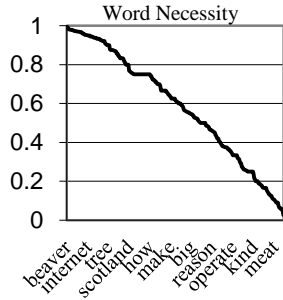


Figure 2. TREC 9 query term necessity in descending order

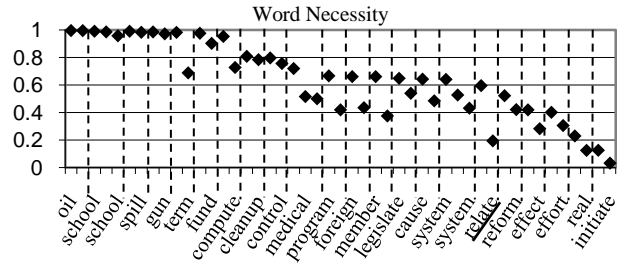


Figure 3. TREC 3 grouped necessity probabilities of the same term recurring in different queries. The term *relate* has the largest difference across its two occurrences, with 0.1935 in topic 172 “The Effectiveness of Medical Products and Related Programs Utilized in the Cessation of Smoking” and 0.5957 in topic 183 “Asbestos Related Lawsuits”.

## 4. EXPERIMENTS

This section examines necessity prediction and its application in retrieval, showing the effectiveness of the current necessity prediction model, its positive impact on retrieval and the potential of using more accurate necessity predictions to improve retrieval.

### 4.1 Datasets

We test necessity prediction on 12 standard TREC ad-hoc retrieval datasets, 6 of which are used as training sets. We looked at only training set topics and relevant documents for data analyses. Smaller datasets with more complete judgments include ad-hoc retrieval tracks of TREC 3 to 8. Larger datasets are Web tracks of TREC 9 and 10, Topic Distillation tasks of TREC 11/2002, 12/2003 and Terabyte tracks of TREC 13/2004 and 14/2005. The larger datasets also have sparser judgments.

#### 4.1.1 Text representation

Stopwords in the standard INQUERY stopword list [16] (except the word “us” which was used extensively in the queries to mean United States) were removed from the term necessity statistics. Stopwords are easy targets for necessity prediction. They are generally highly necessary given their frequent occurrence in relevant documents. Furthermore, their low idf and topic centrality (again, due to low idf weight) make them very distinct from the other terms. To make the term necessity statistics more revealing for the content terms, stopwords were removed. (For retrieval, stopwords were left in, which did not affect performance much).

Meta-language terms in the descriptions, e.g. *instances*, *discuss*, *relevant* etc., are generally unnecessary. We used simple rules to remove those phrases that the descriptions begin with, e.g. “(find / identify / provide) (documents / reports / information) (on / about)”. Removing these improves baseline ad-hoc retrieval performance, but has little effect on necessity weighting based runs.

Web documents include anchor texts from in-links as part of the content, which improves term matching and increases necessity.

Krovetz stemmer was used for parsing queries and documents. Without stemming, the un-stemmed query term matches fewer relevant documents, and leads to lower necessity.

### 4.2 Significance Tests

Smucker et al. [15] recommended the two tailed randomization test for measuring the statistical significance of a difference between mean performance measures, say MAP. The sign test exhibits very different characteristics from the randomization test, and may make false alarms when the randomization test does not, thus they recommended to abandon sign test in comparing MAP.

We draw different conclusions from [15]. The different characteristics of the two tests allow us to view test results from two

different perspectives. The randomization test takes the magnitudes of the differences into account, but because of that, may favor runs with outliers far from the baseline. The sign test does not take into account the magnitude of the difference, thus is robust to outliers, but at the same time may favor runs which improve slightly on many topics, but fail wildly on a small set of the topics. Thus, in order to avoid false alarms and draw safer conclusions, in this work, both tests were used. Bold faced results mean significant by both tests.

### 4.3 Term Necessity Landscape

Our research is based on the hypothesis that  $P(t | R)$  varies by term and query. We begin by examining this assumption.

#### 4.3.1 Necessity distribution across collections

Figures 1 and 2 show that true necessity varies by term for TREC 3 titles and TREC 9 description queries (excluding stopwords). There are a total of 245 terms for TREC 3. Necessity values distribute quite evenly from 0 to 1, averaged at 0.55, and are definitely not constant as prior work assumed. This also suggests that it might be worthwhile to learn and predict term necessity.

In Figure 2, a very similar distribution is seen on the WT10G corpus for descriptions of TREC 9, totally 314 query terms. Except, there is a slight increase in the average necessity to 0.59, and there are much fewer terms with extremely low necessity (lower than 0.05). This is likely because, given a larger collection, judgment pools are shallower, and judged relevant documents are more biased toward those containing the query terms.

#### 4.3.2 Necessity variation for recurring terms

This section explores the question “How much does necessity of the same term vary across topics?” In the 50 TREC 3 title queries, 22 unique terms appear in more than one topic, accumulating 47 individual occurrences whose true necessity values are plotted in Figure 3. The values for a single term are grouped within vertical dashed lines. The variation is less than 0.1 for 2/3 of the term occurrences, when comparing an occurrence to the previous occurrence of the same term. The other 1/3 occurrences have larger variation, typically because of *term sense difference* (“bear” the animal in one query has higher necessity than “bear” the verb in another query), or *term context difference* (“disorder” appearing in a medical phrase or proper noun has a much higher necessity than as a general noun). The same 1/3 vs. 2/3 divide is observed on a larger query set consisting of 250 queries from TREC 3 to 7 description queries. This suggests that query-specific necessity prediction is important, while at the same time, for many term occurrences, the computationally more expensive feature based prediction may be avoided, and a historic necessity value of the

**Table 3. Term necessity prediction performance, training on TREC 3 titles and testing on TREC 4 descriptions.** (TREC4 queries as provided by TREC do not include titles.)

Prediction method	Avg L1 loss	Change
Baseline: Average (constant)	0.2936	0%
IDF	0.3078	+4.837%
IDF+Leaf+Clarity	0.2539	-13.52%
IDF+Leaf+SVD features 1-3	0.1945	-33.75%
Tuning meta-parameters	0.1400	-52.32%
TREC 3 previous occurrences	0.1341	N/A

same term and the same sense (or use) being queried may be sufficient to provide an accurate prediction.

#### 4.3.3 Query characteristics across topic sets

TREC 3 titles have a similar verbosity level as TREC 9 descriptions, with an average of 5-6 terms per query. TREC 4-8 descriptions are more verbose, averaging 8 to 9 terms per query. Reflected in necessity values, TREC 4-8 descriptions have an average term necessity of 0.38 to 0.43, while TREC 3 titles and TREC 9 descriptions have an average necessity of 0.54 to 0.59.

Despite the change of characteristics of TREC queries from year to year, we show below that training data from one TREC dataset can be used to predict necessity for other datasets.

### 4.4 Necessity Prediction Accuracy

This section measures the accuracy of several term necessity prediction methods in per term average absolute prediction error (L1 loss) as defined in Section 3.1. TREC 3 dataset with relevance judgments is used as training data and TREC 4 as test data.

Table 3 shows the performance of several prediction methods, including a baseline that always predicts 0.55 (the average training set necessity), kernel regression with different sets of features, and predicting from the true necessity of a previous occurrence of the same term. With idf as the only feature, prediction is slightly worse than the baseline, indicating that idf alone as used in [2, 14] does not predict necessity effectively. After adding dependency parse leaf and clarity features, error drops by 13%. The SVD features clearly outperform clarity, where meta-parameters are defaulted to 1000 feedback documents, 100 latent dimensions and  $\gamma$  equals 1.5. Cross validation finds the meta-parameters to be 180, 150 and 1.5 which further decreases prediction error to half of the baseline. Removing any of the 5 features hurts performance.

The last row of Table 3 uses the true necessity of a previous occurrence of a term to predict the necessity of another occurrence of the term in a different query. Different from previous methods, this was tested on recurring terms of the TREC 3 training set. Given that the necessity distribution of recurring terms (Figure 3) is quite similar to that of all query terms, the prediction of necessity on just recurring terms should not be much easier than on all terms. And this low prediction error of 0.1341 shows promise in this history based necessity prediction method. It also indicates that for many occurrences of the same term, because a majority sense or majority use dominates, necessity varies little across those topics, and a term dependent prediction method may perform well enough for most terms. But we should bear in mind that 0.1341 is an optimistic estimate, being obtained from a single TREC 3 dataset and tested on a small subset of possibly biased terms – recurring terms. Since the average L1 losses of the other methods were all evaluated on TREC 4 description terms, the 0.1341 loss is not comparable to the other numbers in the table. To really make a comparable evaluation would require coverage of all query terms in TREC 4, which requires relevance judgments

for so many topics that it becomes unpractical for TREC scale judgments. The requirement of many training topics is a limitation of the history based method, on academic scale topic sets.

### 4.5 Necessity Weighting for Ad-hoc Retrieval

The necessity enhanced retrieval models used here are those described in Section 2.1 and 2.2. Baseline models include Language Modeling (LM) with Dirichlet smoothing and Okapi BM25. From a pilot study, we fixed the baseline Dirichlet prior  $\mu$  at 900 for TREC 3-8, 11-12, and 1500 on other sets. BM25  $k1=1.2$ ,  $b=0.75$ ,  $k3=7$ . After adding necessity term weighting, less smoothing is needed, because without necessity weighting, a higher level of smoothing is needed to explain/generate the low necessity query terms from relevant documents. But we did not fine tune the smoothing parameter, and leave the relationship between necessity weighting and smoothing as future work.

Retrieval accuracies are reported in Mean Average Precision (MAP), which is used as the objective for tuning the meta-parameters on the development sets. In Top 10 and 20 Precision, 10%-20% improvements were observed for the experiments in Table 5, with less significance. All experiments and significance tests were run using the Lemur/Indri toolkit version 4.10. Small modifications were made to allow for setting term necessity weights into the RSJ weight for the BM25 baseline, and for running local SVD to obtain term similarity values.

#### 4.5.1 Potential of improvement using true necessity

This subsection evaluates the case where relevance judgments are used to estimate term necessity (as per Section 3.2). True necessity values are used to weight query terms, and retrieval is evaluated on the same set of relevance judgments that provided the necessity term weighting. This is only intended to show the potential of using necessity predictions as term weights, similar to the retrospective case of [1]. To our knowledge, this is the first work to report performance of applying true necessity weights on retrieval models other than BIM. Improvements over state-of-the-art models underscore the potential of applying necessity prediction.

Table 4 shows that on different datasets, using true necessity term weights gives a consistent 30%-80% improvement over the baseline model on description queries. Bold faced results are significantly better than the baseline by both significance tests. Baselines include language modeling with Dirichlet smoothing, and Okapi BM25. Title queries (denoted as *title* in the tables) perform better than descriptions (*desc* in tables). Applying true necessity weights on title queries give steady but smaller improvements over title baselines. True necessity weighted description queries outperform true necessity weighted titles, simply because there are more terms to tune weights on. For the same reason, predicted necessity applied to title queries does not lead to a significant change in retrieval performance, (these results are omitted from Table 5).

As Section 2.2 mentions, the relevance model if assuming multinomial term occurrences suggests to estimate relevance term weights from the relevant documents as a multinomial term distribution. The main difference with multiple Bernoulli necessity estimation lies in the fact that multiple term occurrences in the same document will be rewarded. Applying the multinomial estimates as term weights gives the “Multinomial RM” row in Table 4. It is less stable and leads to a retrieval performance much worse than the Bernoulli necessity estimates in all test collections. Contrary to this result, a prior finding showed that the document language model is best estimated as a multinomial term distribution instead of multiple Bernoulli [27]. We leave to future work a more unified model for document and relevance model estimation.

**Table 4. Retrieval performance with true necessity weighted query terms, in Mean Average Precision. Bold face means significant by both randomization and sign tests with significance level  $p < 0.05$ . Queries generated from topic titles are denoted as *title*, and description queries are denoted as *desc*.**

TREC	4	6	8	9	10	12	14
Document collection	disk 2,3	disk 4,5	d4,5 w/o cr	WT10g		.GOV	.GOV2
Topic numbers	201-250	301-350	401-450	451-500	501-550	TD1-50	751-800
LM <i>desc</i> – Baseline	0.1789	0.1586	0.1923	0.2145	0.1627	0.0239	0.1789
LM <i>desc</i> – Necessity	<b>0.2703</b>	<b>0.2808</b>	<b>0.3057</b>	<b>0.2770</b>	<b>0.2216</b>	<b>0.0868</b>	<b>0.2674</b>
Improvement	<b>51.09%</b>	<b>77.05%</b>	<b>58.97%</b>	<b>29.14%</b>	<b>36.20%</b>	<b>261.7%</b>	<b>49.47%</b>
$p$ - randomization	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001
$p$ - sign test	0.0000	0.0000	0.0000	0.0005	0.0000	0.0000	0.0002
Multinomial RM	<b>0.1988</b>	<b>0.2088</b>	<b>0.2345</b>	0.2239	0.1653	<b>0.0645</b>	<b>0.2150</b>
Okapi <i>desc</i> – Baseline	0.2055	0.1773	0.2183	0.1944	0.1591	0.0449	0.2058
Okapi <i>desc</i> – Necessity	<b>0.2679</b>	<b>0.2786</b>	<b>0.2894</b>	<b>0.2387</b>	<b>0.2003</b>	<b>0.0776</b>	<b>0.2403</b>
LM <i>title</i> – Baseline	N/A	0.2362	0.2518	0.1890	0.1577	0.0964	0.2511
LM <i>title</i> – Necessity	N/A	<b>0.2514</b>	0.2606	<b>0.2058</b>	<b>0.2137</b>	<b>0.1042</b>	<b>0.2674</b>

**Table 5. Retrieval performance (MAP) with predicted necessity. Bold face means significant by both significance tests with  $p < 0.05$ . All these runs are obtained using topic descriptions as queries.**

TREC train sets	3	3-5	3-7	7	3-9	9	11	13	
Test/x-validation	4	6	8	8	10	10	12	14	
LM <i>desc</i> – Baseline	0.1789	0.1586	0.1923	0.1923	0.1627	0.1627	0.0239	0.1789	
LM <i>desc</i> – Necessity	<b>0.2261</b>	<b>0.1959</b>	<b>0.2314</b>	<b>0.2333</b>	0.1813	<b>0.1810</b>	<b>0.0597</b>	<b>0.2233</b>	
Improvement	<b>26.38%</b>	<b>23.52%</b>	<b>20.33%</b>	<b>21.32%</b>	11.43%	<b>11.25%</b>	<b>149.8%</b>	<b>24.82%</b>	
$p$ - randomization	0.0000	0.0023	0.0043	0.0003	0.0050	0.0012	0.0012	0.0001	
$p$ - sign test	0.0000	0.0002	0.0013	0.0077	0.0595	0.0005	0.0000	0.0325	
P@10	Baseline	0.4160	0.2980	0.3860	0.3860	0.3180	0.3180	0.0200	0.4720
	Necessity	<b>0.4940</b>	0.3420	0.4220	0.4380	0.3280	0.3400	<b>0.0467</b>	0.5360
P@20	Baseline	0.3450	0.2440	0.3310	0.3310	0.2400	0.2400	0.0211	0.4460
	Necessity	<b>0.4180</b>	<b>0.2900</b>	0.3540	0.3610	<b>0.2790</b>	<b>0.2810</b>	<b>0.0411</b>	0.5030
IDF only	Predicted	0.1776	0.1691	<b>0.2325</b>	<b>0.2383</b>	0.1696	0.1687	<b>0.0576</b>	0.2227
Clarity only	Necessity	0.1900	<b>0.2028</b>	<b>0.2309</b>	<b>0.2318</b>	0.1399	0.1516	<b>0.0473</b>	0.2057
Relevance Model <i>desc</i>	<b>0.2423</b>	0.1799	<b>0.2352</b>	<b>0.2352</b>	<b>0.1888</b>	<b>0.1888</b>	0.0221	0.1774	
Relevance Model <i>title</i>	N/A	0.2605	<b>0.2790</b>	<b>0.2790</b>	0.1710	0.1710	0.0896	<b>0.2782</b>	
RM reweight-Only <i>desc</i>	<b>0.2215</b>	0.1705	<b>0.2435</b>	<b>0.2435</b>	0.1700	0.1700	<b>0.0692</b>	0.1945	
RM reweight-Trained <i>desc</i>	<b>0.2330</b>	0.1921	<b>0.2542</b>	<b>0.2563</b>	0.1809	<b>0.1793</b>	<b>0.0534</b>	<b>0.2258</b>	

We note that the performance gain using true necessity term weights is similar to the gain reported on the WT10g (TREC 9, 10) collection by the best sampled term weights [6]. The Oracle case of [6] directly tuned term weights using MAP as objective, thus, this close performance means true necessity weights are close to the best sampled term weights of [6] in retrieval effectiveness.

#### 4.5.2 Predicted necessity term weighting

This subsection tests whether term necessity can be effectively predicted to improve retrieval. In Table 5, we present results using *predicted* necessity values (Section 3) as user term weights. Models were trained on TREC topics from previous year(s), and were tested using 5-fold cross validation on the 50 TREC topics of the next year. Here, the RBF kernel regression model was always trained on the 50 training topics (if training set includes only one TREC dataset). 50 test topics were split into 5 folds, 4 of which were used as development set to tune meta-parameters, 1 fold was used for testing. However, the learning model does not require that much development data. 2-fold cross validation uses fewer (only 25) development topics, and still yields the same performance and optimal parameter values as 5-fold cross validation does. Meta-parameters  $m$  – the number of latent dimensions to keep after SVD and  $\gamma$  which controls the kernel width were fixed at 150 and 1.5, because on all datasets, cross validation found the two parameters to be within 130~170, and 1.2~1.6.

Consistent improvements of 10% to 25% are observed in MAP across all 6 ad-hoc retrieval datasets, statistically significant by both sign and randomization tests. Most runs have strong significance levels of  $p < 0.005$ . This shows that models trained on one dataset can reliably predict term necessity on a similar dataset. It also shows that the prediction model and the features do adapt to collections of different scales, with different levels of judgment depths, and do predict necessity reliably.

The number of feedback documents –  $n$  plays an important role in adapting the SVD features to different collections. During cross validation, when tuning  $n$  on development set, the optimal  $n$  is found to be 180 on smaller TREC 3-8 datasets. For the larger WT10g of TREC 9-10,  $n$  is 200, the even larger .GOV TREC 11-12 increases  $n$  to 500. And for .GOV2 of TREC 13-14,  $n$  is 600. The larger collections probably contain a larger number of helpful or even relevant documents, thus expectedly, the optimal number of feedback documents increases as collection size does.  $n$  might also depend on how many relevant documents there are for the query, and whether the majority sense of a query term is used.

Comparing training on one collection vs. training on multiple collections, it seems that a set of 50 topics is enough to train the model to optimize MAP. More training topics from dissimilar collections neither improve retrieval, nor hurt performance much.

Necessity weighting also yield consistent improvements in top precision, as shown in Table 5. The improvements are even statistically significant on multiple collections. At first, it may seem counterintuitive that a better estimate of term recall actually improves top precision. However, *better estimates of term recall in fact corrects the idf-based retrieval models' bias to the matching of rare but possibly unnecessary terms, thus reduces top ranked false positives and boosts up partially matched relevant results.*

Since *idf* has been used by prior research (as the only feature) to predict necessity, Table 5 also includes a baseline of using *idf* as the only feature to predict necessity. Results show that even though *idf* does not predict necessity well (Table 3), using *idf* predicted necessity does lead to a perceivable gain in retrieval, though not stable across collections. This is encouraging because without computing more complex term features based on an initial retrieval, *idf* based prediction can still provide some gain. Clarity has been shown to correlate well with query performance, however, using term clarity as the only feature to predict necessity does not lead to a stable retrieval performance.

#### 4.5.2.1 Relevance model variations

The relevance model theory [17] suggests to use term relevance probabilities as user term weights, which we follow, as shown in Section 2.2. Relevance model also provides a way to estimate the relevance based term weights from top documents of an initial retrieval. This estimation procedure, though different in goal, is similar to how we created the SVD features in terms of the sources of information being used. Given the similarities, we also include the relevance model as another baseline.

The relevance model (RM3) interpolates the weighted expansion query with the original query. The same development sets were used to tune the free parameters (number of feedback documents, feedback terms, and mixing weight with the original query). Same cross validated results were reported.

As shown in Table 5, despite a more expressive feedback query (weighted expansion terms + interpolation with original query), trained on the same datasets, the *Relevance Model* method is unstable compared to predicted necessity weighting. It is significantly better than the baseline on 3 collections, but insignificant on 3 others. It hurts performance for the low initial retrieval run of TREC 12. This performance is expectable because even though relevance model estimates relevance based term weights, it estimates term weights in an unsupervised fashion, while our supervised framework uses relevance judgements from training topics to guide the prediction. Another difference is that the relevance model tends to use only a few top documents (around 5-20), while our local SVD performs optimally with hundreds of documents. The SVD based features can leverage term appearance information from more documents, achieving a more stable performance.

The *RM reweight-Only* method uses the term weights estimated by the relevance model from top-ranked documents to only reweight the original query terms. It is a variation of the original relevance model approach, and is more similar to our approach of using top-ranked documents to compute user term weights for the original query terms. Results in Table 5 show this baseline to be unstable. On some collections, it is significantly better than the Language Model baseline, but on as many other collections, the improvement is insignificant by both tests. This is quite expectable, as relevance models can be seen as directly using topic centrality as term weights, thus, it shall not outperform the prediction based on the whole set of features. In some cases, *RM reweight-Only* outperforms necessity prediction. This shows promise in favoring higher ranked documents according to their relevance

**Table 6. Correlation values between features and true necessity, tested on TREC 4. Here, predicted necessity based on all 5 features yields a high correlation of 0.7989. (RMw uses the term weights estimated by the Relevance Model.)**

IDF	DepLeaf	Centr	Syn	Repl	RMw
-0.1339	0.1278	0.3719	0.3758	-0.1872	0.6296

**Table 7. Effects of features on TREC 4. Bold face means significance over LM baseline by both tests ( $p < 0.005$ ).**

Features used	MAP	Features used	MAP
IDF only	0.1776	All 5 features	<b>0.2261</b>
IDF + Centrality	<b>0.2076</b>	All but Centrality	<b>0.2235</b>
IDF + Synonymy	<b>0.2129</b>	All but Synonymy	<b>0.2066</b>
IDF + Replaceable	0.1699	All but Replaceable	<b>0.2211</b>
IDF + DepLeaf	0.1900	All but DepLeaf	<b>0.2226</b>

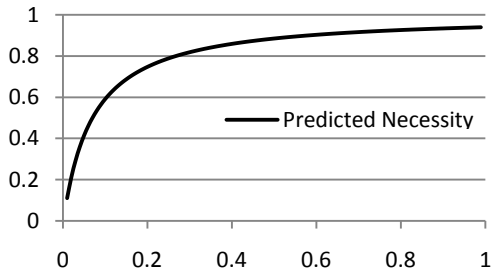
scores as the relevance model does, while our current local SVD treats all top documents the same.

*RM reweight-Only* performs just slightly lower than the RM baseline which includes expansion terms. Thus, it is either the case that expansion terms contribute little to retrieval, for description queries of about 5-10 words long, or expanding terms as a weighted combination of feedback terms, as in the RM, is not effective, other forms of expansion may do better.

Overall, these results suggest that for description queries, *most of the performance gain from pseudo relevance feedback (PRF) methods is due to the fact that term weights computed from PRF methods tend to correlate well with true necessity.* The reason is that *terms that occur consistently in relevant documents also tend to appear consistently in top ranked documents.* In fact, a 0.63 correlation is observed on TREC 4 test set, while our best predicted necessity have a higher 0.80 correlation with necessity, explaining the better performance. Details are listed in Table 6. On title queries, however, because of the more accurate user query, better initial retrieval performance and the introduction of weighted expansion terms, Relevance Model (Table 5) outperforms true necessity weighted title queries on 4 out of the 6 collections, and in most cases outperforms predicted necessity weighted description queries (bold faced means significance over title query baseline). True necessity weighted description queries still outperform Relevance Model consistently, suggesting to apply necessity prediction on the Relevance Model expansion terms. Since the focus of this work is on reweighting query terms, we leave necessity prediction for expansion terms to the future.

Relevance model uses unsupervised estimation of term weights, while this work adds supervised learning of necessity into the picture. We can combine the two by using RM term weights as a feature to predict necessity, reported in Table 5 as the *RM reweight-Trained* runs. The *RM reweight-Trained* runs used relevance model term weights as the only feature and trained a necessity prediction model to predict term necessity. A simple per query scaling is used to map the near 0 RM weights to [0, 1], so that the maximum query term weight is always 1 for a given query.

*RM reweight-Trained* performed consistently better than the base *reweight-Only* method, except on TREC 12. This consistent improvement empirically differentiates our framework from the relevance model, and shows that supervised learning of term necessity improves the unsupervised relevance model estimates. The *reweight-Trained* method performs slightly better than with the SVD features on average, but also slightly less stable, in terms of significance levels over the Language Model baseline.



**Figure 4. The learnt function using Relevance Model term weights (x-axis) alone to predict term necessity (y-axis).**

The resulting prediction model is a function that maps the RM term weights into necessity, and the final form of the function, trained on the TREC 3 dataset, is plotted in Figure 4. It basically boosts terms with lower middle range RM weights (0-0.6) to larger values. This simple mapping works because *the Relevance Model tends to underestimate the term necessity values*.

Including Relevance Model weight (RMw) as the 6th feature further improves MAP by about 5%, but the meta-parameters become less stable during cross validation. Given that the focus of the paper is to establish the framework of supervised necessity learning and prediction, we omit these further results.

#### 4.5.2.2 Feature ablation

To see the effects of individual features, in Table 7, we present retrieval performance for different feature combinations using TREC 3 as training and TREC 4 as test set. Results show that all the features contribute to effective retrieval. Among them, the Synonymy feature is most effective and results in more stable improvements, but the other features can still significantly improve retrieval without Synonymy. Similar trends are observed on the other test sets. The only exception is the TREC 10 test set, where topic centrality outperforms synonymy, and the improvement from synonymy alone is not significant by the sign test.

Overall, the evaluation using L1-loss of necessity prediction (Table 3) and feature correlation (Table 6) show consistent trends as retrieval effectiveness (Table 7). IDF alone is not effective in predicting necessity. All the 5 designed features contribute to necessity prediction accuracy, and further, retrieval effectiveness. For the current set of features, better necessity prediction leads to better retrieval performance, which is also consistent with the Oracle performance of Table 4.

#### 4.5.2.3 Efficiency

SVD and dependency parsing take most of the processing time. Per query, SVD on 300 documents with 20,000 terms takes ~1 second. Further speedup is possible. Since the goal of SVD is just to find possible searchonyms of query terms, SVD does not need to converge; a smaller number of iterations may suffice. Dependency parsing takes ~3 seconds per query. Speedup is also possible, as the feature only indicates whether a term is a leaf node in the parse tree, and does not require knowledge of the whole parse tree. Simpler but more direct dependency leaf identification models would suffice. Alternative term abstractness measures can also help avoid parsing the queries.

## 5. RELATED WORK

Research on predicting  $P(t | R)$  can be viewed as one specific line of research within a broad body of research intended to predict effective term weights. Earlier work includes the Binary Independence Indexing [24] and the Berkeley regression [25]. More

recently, Lease et al applied regression rank [6] to optimize retrieval performance (MAP) by directly tuning term weights. All of these approaches used regression to predict term weights and most of this prior work used only features based on simple tf- or idf-based statistics. Regression rank used more features, including part-of-speech and a term’s location in a query. It also tuned user term weights to directly optimize MAP. Without a relevance probability prediction in the middle, its formulation is more direct than ours, but also harder, e.g. optimal term weights are difficult to find, and not unique. Regression rank normalized term weights within each query to make the problem tractable, which produces weights that cannot be compared easily across different queries. This is not generally viewed as a serious flaw, however our data analysis suggests that term necessity probabilities estimated for one query are often effective for the same term in other queries. Moreover, our necessity prediction framework has a simple and well defined objective – term necessity, it uses fewer well justified and interpretable features, and achieves comparable performance. Understanding necessity leads to a better understanding of why basic retrieval models and PRF methods work, which can guide query formulation, and shows the generality of our work.

Query reduction for long queries [11, 12] is another relevant problem. Essentially, unnecessary terms in long queries are to be reduced. In fact,  $p(c_i | q)$  in equation (2) of [11] can be interpreted as a necessity measure of a query concept  $c_i$ . [11] even hinted at concept *necessity* in the introduction, “concepts ... must be ... in a retrieved document in order for it to be relevant”, though concept keyness was actually pursued in the experiments.

## 6. CONCLUSIONS

Term necessity, the necessity of a term’s occurrence to the relevance of a document  $P(t | R)$ , is of central importance for probabilistic retrieval models, sharing a similar role as idf. The necessity to non-relevance is a precision measure easily predicted by idf [2], while the necessity to relevance is a *term recall* measure and is harder to predict. Most previous work predicted term weights as a whole, consisting of the idf and the necessity components [2], or as a way to improve idf weighting [14]. We attempt to predict the necessity probability itself, through understanding the probability and identifying factors that may affect it. These factors include a term’s centrality to the topic, term synonymy, replaceability by synonyms, rareness and abstractness. Using 5 features designed to capture the factors, term necessity is effectively predicted.

Predicted necessity used as user query term weights significantly improves ad-hoc retrieval of verbose queries. On 6 standard TREC ad-hoc retrieval and web search collections of different sizes and judgment depths, predicted necessity brings a consistent and significant 10% to 25% improvement on verbose queries. Results also show that weighting terms by their ground truth necessity estimated from relevance judgments gives a significant 30% to 80% improvement over state-of-the-art ad-hoc retrieval models.

Necessity improves our understanding of why pseudo relevance feedback works, and can be used to improve PRF methods by supervised necessity learning. Experiments show that most of the performance gain from pseudo relevance feedback comes from the ability to reweight query terms and the weights correlate well with necessity, even though estimated in an unsupervised fashion.

To follow up on this work, a more complete understanding of the causes of low necessity or mismatch is needed. Creating better features for these causes and factors will also motivate the design of better NLP tools to attack problems such as measuring term abstractness and identifying synonyms or searchonyms.

For *ad-hoc retrieval*, a better understanding and theoretical justification of applying necessity is needed, especially what form of weighting should be used in cases of prediction inaccuracies.

Necessity prediction can also be applied to (structured) query formulation in general. For example, in dealing with long queries [6, 11, 12], necessity can help decide which terms to include.

Among the factors that affect necessity, the relation between necessity and synonymy is especially important. It derives effective features for predicting necessity, and also suggests that necessity can be applied to guide query expansion toward the terms that actually need expansion. Expansion can be applied to a specific query term, instead of the whole query, for example using the synonym (#syn) operator in the Indri query language. This search strategy gives more control over where and how much to expand, and is favored by expert searchers in library, biomedical or legal domains, but is less explored by general retrieval systems. Query formulation and modification here can be totally automatic or through suggestions and user interactions.

## 7. ACKNOWLEDGEMENTS

We thank Andrea Bastoni and Lorenzo Clemente for maintaining and making the SVD code available for the Lemur toolkit. This work is supported by National Science Foundation grant IIS-0707801 and IIS-0534345. The views and conclusions are the authors', and do not reflect those of the sponsor.

## 8. REFERENCES

- [1] S. E. Robertson and K. Spärck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129-146, 1976.
- [2] W. Greiff. A theory of term weighting based on exploratory data analysis. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 11-19, 1998.
- [3] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC 1994)*. 109-126. Gaithersburg, USA, November 1994.
- [4] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 275-281, 1998.
- [5] INDRI - Language modeling meets inference networks. <http://www.lemurproject.org/indri/>. Retrieved Oct 1, 2009.
- [6] M. Lease, J. Allan and W. B. Croft. Regression rank: Learning to meet the opportunity of descriptive queries. In *Proceedings of the 31st European Conference on Information Retrieval (ECIR)*. 90-101, 2009.
- [7] W. B. Croft and D. J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35(4):285-295, December 1979.
- [8] C. T. Yu, K. Lam, and G. Salton. Term weighting in information retrieval using the term precision model. *Journal of the ACM*, 29(1):152-170, January 1982
- [9] S.E. Robertson. On relevance weight estimation and query expansion. *Journal of Documentation*, 42(3): 182-188, 1986
- [10] S. Cronen-Townsend, Y. Zhou and W. B. Croft. Predicting query performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 299-306, 2002.
- [11] M. Bendersky, W. B. Croft. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 491-498, 2008.
- [12] G. Kumaran and V. Carvalho. Reducing long queries using query quality predictors. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 564-571, 2009.
- [13] Y. Lu, H. Fang and C. Zhai. An empirical study of gene synonym query expansion in biomedical information retrieval. *Information Retrieval*, 12(1): 51-68, 2009.
- [14] D. Metzler. Generalized inverse document frequency. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. 399-408, 2008.
- [15] M. D. Smucker, J. Allan and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*. 623-632, 2007.
- [16] J. Allan, M. Connell, W. B. Croft, F. Feng, D. Fisher and X. Li. INQUERY and TREC-9. In *Proceedings of the Ninth Text REtrieval Conference (TREC 2002)*. 551-600, 2000.
- [17] 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*, 120-127, 2001.
- [18] L. Zhao and J. Callan. Effective and efficient structured retrieval (poster description). In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. 1573-1576, 2009.
- [19] H. Schütze, D.A. Hull and J.O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 229-237, 1995.
- [20] A. Kontostathis and W. M. Pottenger. Detecting patterns in the LSI term-term matrix. *IEEE ICDM02 Workshop Proceedings, The Foundation of Data Mining and Knowledge Discovery (FDM)*. 2002.
- [21] C.J. van Rijsbergen. *Information Retrieval (2nd Edition)*, chapter 6. Butterworths. London 1979.
- [22] R. Lawlor. Information technology and the law. *Advances in Computers*, 3: 299-346, 1962.
- [23] G. Goertz and H. Starr (eds.) *Necessary conditions: theory, methodology, and applications*. Lanham, Md.: Rowman & Littlefield 2002. page 10.
- [24] N. Fuhr and C. Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems* 9(3):223-248. 1991.
- [25] W. Cooper, A. Chen and F. Gey. Full text retrieval based on probabilistic equations with coefficients fitted by logistic regression. *NIST Special Publication 500-215: The Second Text REtrieval Conference (TREC-2)*. 57-66, 1993.
- [26] V. Dang and W. B. Croft. Query reformulation using anchor text. In *Proceedings of the third ACM International Conference on Web Search and Data Mining*. 41-50, 2010.
- [27] D. Metzler, V. Lavrenko and W. B. Croft. Formal Multiple-Bernoulli Models for Language Modeling. *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 540-541, 2004.