Term Context Models for Information Retrieval

ABSTRACT

At their heart, most if not all information retrieval models utilize some form of term frequency. The notion is that the more often a query term occurs in a document, the more likely it is that document meets an information need. We examine an alternative. We propose a model which assesses the presence of a term in a document not by looking at the actual occurrence of that term, but by a set of non-independent supporting terms, i.e. *context*. This yields a weighting for terms in documents which is different from and complementary to *tf*-based methods, and is beneficial for retrieval.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Maximum entropy, conditional random fields, context-based retrieval

1. INTRODUCTION

At the core of almost every modern ad hoc retrieval algorithm is a reliance both on local properties (statistics from within a document) as well as with global properties (statistics from across a collection). For example, Okapi BM25 [13] uses term frequency (*tf*) and document length (*dl*) as local properties, and inverse document frequency (*idf*) and average document length as global properties. Ponte's language modeling [11] uses *tf* and *dl* for local maximum likelihood estimation, and combines these with a global risk factor $\hat{R}$, a measure of how far the current document's *tf* is from the collection-wide normalized mean. In more recent language modeling work [17], the local maximum likelihood estimate is combined with a global "fallback" model, or $p(t|C)$, the probability of the term appearing in the entire collection.

The motivation for *idf*, $\hat{R}$, and $p(t|C)$, all global properties, comes from the understanding that local properties alone are insufficient. No matter the algorithmic framework, some combination of local and global properties are required for proper retrieval. It is with this understanding that we have developed another useful global property: *term context*. Unlike the aforementioned global properties, which are all based on counts across documents in which a term is present, we create a model that predicts whether or not the term *should* be present in a document *whether or not it is*, based on global contextual usage patterns. It is a measure of how good a fit a term is for a particular document, influenced by how that term is used in other documents in a collection.

For example, suppose a user issues the query "fuel". Furthermore, suppose there are two same length, same "fuel" term frequency documents in the collection. However, one of the documents talks about energy and gas and coal, whereas the other document talks about how various interest rates and tax cuts fuel the economy. Ceteris paribus, *tf*-based methods have no reason to prefer one document over another. However, our models may learn that the document on energy contains a more contextually standard usage of fuel *in a particular collection* than the document on the economy, and thus will be ranked higher when utilizing term context models.

Lest this previous example mislead the reader, this is more than just word sense disambiguation. Take for example the query term "emacs". There is really only one lexical sense of this word. And yet documents containing emacs might encompass a whole range of subjects, everything from keystroke shortcuts to programming environments to email integration. If there is a particular aspect or set of aspects that dominate in a collection, those will be learned by context modeling and documents containing those aspects will be given a boost in the rankings.

We note at the outset that while we are computing global contextual statistics, what we are doing is neither query expansion (via global context analysis) nor pseudo-relevance feedback (via local context analysis) [16]. We do not actually add the terms gas or coal to the fuel query. We do not modify weights on fuel based on the top n documents from an initial retrieval run. This distinction is subtle but important, and will be explained further in later sections.

This paper is structured as follows. In section 2 we dis-

CIKM 2006

cuss related work. Section 3 contains a description of our model, while section 4 shows how we apply the model to ad hoc retrieval. The experimental set up and evaluation is done in section 5. Future work is proposed in section 6 and conclusions are drawn section 7.

2. RELATED WORK

The parametric form that we use for the term context model is the familiar log-linear, also known as exponential or maximum entropy model, and is related to conditional random fields [6]. CRFs can be though of as a structured form of maximum entropy. Conversely, conditionally trained maximum entropy can be thought of as an unstructured, or "0^{th} order", CRF [8]. We prefer this latter perspective, and in section 3 will describe our model in these terms.

Maximum entropy and related models have long been used in information retrieval applications, from early work by [5] to more recent work by [4, 10] and even [9]. However, these models only consider counts and dependencies amongst the terms in the query, and do not make use of the larger statistical language properties of a term.

Other maximum entropy and related models do make use of statistical language but have not been applied to the task of ad hoc retrieval [1, 2, 3]. The model with the most similar form to ours is [14], though again the application of that model is quite different. Another closely related work is [7]. While the application is music information retrieval, notes instead of words, the idea of situating notes in their proper context is analogous to situating terms in their context. Finally, [8] uses maximum entropy to learn manually-annotated text labels for documents (via "social tagging"), using the document as the context for the label. Determining whether a particular term is a good label for a document is similar to our approach. The difference is that the labels we are learning are not external to the content of the documents. This will become clearer in the next section.

3. THE TERM CONTEXT MODEL

Our goal is to assess the presence of a term in a document based *not on the actual observed occurrence of that term*, but on the *evidence of a set of supporting terms, or context, from that document.*

One possible solution is to use term co-occurrence. If a "support" term co-occurs frequently with the "target" (query) term, one may be able to use that support term as a substitute for the target. However, in most cases no one term is a perfect substitute for another, so multiple terms are needed. This presents a problem: When multiple terms are added to a query, precision at low recall can be adversely affected [16]. Furthermore, support terms are not independent. Using straightforward co-occurrence makes the pairwise independence assumption. This is only sometimes, and clearly not necessarily, true. We believe that by modeling the statistical dependencies between support terms we can increase precision at low recall for some queries without adversely affecting precision for other queries. Therefore, we turn to a framework that allows us to selectively model the interactions between the individual support terms.

3.1 Nature of the Field

Suppose we have a lexicon of k terms extracted from some corpus of interest. We create for each term i in the lexicon two binary random variables, t_{x_i} for the observed value of that term, and t_{y_i} for the unobserved (hidden, inferred) value. When given a document d from our corpus, the observed variables $\{t_{x_1} \dots t_{x_k}\}$ are instantiated by assigning a value of "0" when the observed frequency of the term is zero in that document, and a value of "1" when the observed frequency is greater than zero. Now, for each term's hidden binary variable t_{y_i} we define the *context* H_{y_i} as the set of observable variables for all terms in the vocabulary other than the i^{th} term itself:

$$H_{y_i} = \{t_{x_j} : i \neq j\} \tag{1}$$

Terms in H_{y_i} are the only ones that can be examined when we are making the prediction regarding t_{y_i}. In other words, we assume that the probability of term t_{y_i} occurring in d is completely determined by H_{y_i} in our model.

This also means that each term t_{y_i} is conditionally independent of all other terms $t_{y_{j\neq i}}$, given $\{t_{x_1} \dots t_{x_k}\}$; i.e. we propose a "bag of term-context-models" approach. However, it is important to stress that we do not want to assume independence among the conditioning variables; we still allow arbitrary dependencies within the H_{y_i} context.

3.2 Conjunctive Features

A well-known advantage of the random field framework is that it allows arbitrary dependencies between the target t_{y_i} and its context H_{y_i}. Features may be simple or complex, based on everything from term frequencies to the locations of commas. However, for the sake of simplicity we will deliberately restrict allowed dependencies to binary questions of the form: "*does term t_{x_j} occur in this document?*".

We will also allow generalizations where a question is asked about some subset S of the terms in H_{y_i}. The answer to a question of this form will be called the feature function f_S, and S will be referred to as the *support* of f. For a given support $S \in H_{y_i}$, the feature function f_S is defined as the conjunction of answers about the individual terms in $t_{x_j} \in S$:

$$f_S(t_{y_i}, H_{y_i}) = t_{y_i} \prod_{t_{x_j} \in S} t_{x_j} \tag{2}$$

Defined in this manner, our feature functions are always boolean, and equal to 1 if all the terms defined by S occur in the document. A feature function always includes the target term t_{y_i}. This is not a fallacy, since t_{y_i} will never actually be considered a part of its own context. Presence of t_{y_i} in the feature serves only to tie the occurrences of terms in S to the term t_{y_i}. Figure 1 is an example of a term context model for a single term, t_{y_i}.

3.3 Exponential Form

There are a number of different forms we could choose for computing the probabilities $P(t_{y_i}|H_{y_i})$, but it turns out that for random fields there is a natural formulation of the distribution that is given by the maximum-entropy framework. Suppose we are given a set $\mathcal{F}$ of feature functions that define the structure of the field. The maximum-entropy principle states that we should select the parametric form that is: **(i)** consistent with the structure imposed by $\mathcal{F}$ and **(ii)** makes the least amount of unwarranted assumptions — that is the most uniform of all distributions consistent with $\mathcal{F}$. The family of functions that satisfies these two criteria is the exponential (or log-linear) family, expressed as:

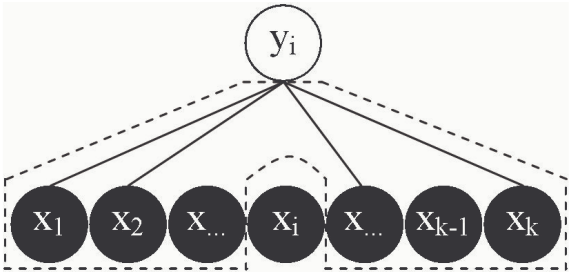

Figure 1: Graphical representation of a Term Context Model. Dark nodes are the observable variables. The light node is the hidden variable for the term t_{y_i} whose context we are modeling. The variables inside the dotted area are the context H_{y_i}. An example set of arcs are shown, where an arc represents a single feature function f_S whose connected nodes are the support S.

$$\hat{P}(t_{y_i}|H_{y_i}) = \frac{1}{Z_i} \exp\left\{\sum_{f\in\mathcal{F}} \lambda_f f(t_{y_i}, H_{y_i})\right\} \quad (3)$$

In equation (3), the set of scalars $\Lambda = \{\lambda_f : f \in \mathcal{F}\}$ are the Lagrange multipliers for the set of structural constraints $\mathcal{F}$. Z_i is the normalization constant that ensures that our distribution sums to unity over all possible values of t_{y_i}:

$$Z_i = \sum_{t_{y_i}} \exp\left\{\sum_{f\in\mathcal{F}} \lambda_f f(t_{y_i}, H_{y_i})\right\} \quad (4)$$

For a general random field, the partition function Z_i is exceptionally hard to compute since it involves summation over all possible configurations of the system, which is exponential. In our case, our assumption of no dependencies between hidden variables $\{t_{y_1} \dots t_{y_k}\}$ makes computation of the partition function extremely simple: Z_i only needs to be computed for $t_{y_i} = 0$ and $t_{y_i} = 1$.

3.4 Objective Function

The ultimate goal of this project is to develop a probability distribution $\hat{P}(t_{y_i}|H_{y_i})$ that will accurately predict the presence of term t_{y_i} in a document. There exist a number of different measures that could indicate the quality of prediction. We choose one of the simplest — log-likelihood of the training data. Given a training set $\mathcal{T}$ of documents d, the log-likelihood is simply the average logarithm of the probability of producing term i in $\mathcal{T}$:

$$\mathcal{L}_{\hat{P}} = \frac{1}{|\mathcal{T}|} \log \prod_{d=1}^{\mathcal{T}} \hat{P}(t_{y_i,d}|H_{y_i,d}) \quad (5)$$

3.5 Feature Induction

If we examine the parametric form in equation (3), we note that there are two things on which the model depends. The first and (in our opinion) foremost is the structure of the field $\mathcal{F}$, represented as a set of constraints or feature functions $f\in\mathcal{F}$. These constraints represent most significant dependencies between the variables of the field. The second thing we learn is the set of weights $\Lambda = \{\lambda_f\}$, one for each feature $f\in\mathcal{F}$. We know that Λ and $\mathcal{F}$ are intimately intertwined and we need to learn them simultaneously, but for the sake of clarity we split the discussion in two sections. This section will describe how we can incrementally induce the structure $\mathcal{F}$ of the field, starting with a very flat, meaningless structure and generalize to more interesting relationships.

The field induction procedure closely follows the algorithm described in [3], the primary difference being that we are dealing with a conditional field, whereas Della Pietra *et al* use a joint model. We start with a field that contains only that term without any dependencies: $\mathcal{F}^0 = \{t_{y_i}\}$. We will incrementally update the structure $\mathcal{F}$ by adding the features g that result in the greatest improvement in the objective function. Suppose $\mathcal{F}^k = \{f_S\}$ is the current field structure. Also assume that the corresponding weights Λ^k are optimized with respect to $\mathcal{F}^k$. We would like to add to $\mathcal{F}^k$ a new feature g that will allow us to further increase the likelihood of the training data. In order to do that we first need to form a set of candidate features $\mathcal{G}$ that could be added. We define $\mathcal{G}$ to be the set of all single term extensions of the current structure $\mathcal{F}$:

$$\mathcal{G} = \{\ f_S \cdot t_{x_j} \ : \ f_S \in \mathcal{F}, j \neq i\ \} \quad (6)$$

In other words, we form new candidate features g taking an existing feature f and attaching a single observable term t_{x_j}. Naturally, we do not include as candidates any features that are already members of $\mathcal{F}$. Now, following the reasoning of Della Pietra, we would like to pick a candidate $g\in\mathcal{G}$ that will result in the maximum improvement in the objective function.

First, let $\tilde{E}[g]$ denote the *empirical* or *target* expected value of g, which is simply how often the feature actually occurs in the training data $\mathcal{T}$:

$$\tilde{E}[g] = \frac{1}{|\mathcal{T}|} \sum_{d=1}^{\mathcal{T}} g(t_{y_i}, H_{y_i})$$

Similarly, our estimate $\hat{P}(t_{y_i}|H_{y_i})$ gives rise to the *predicted* expectation $\hat{E}[g]$ for the function g. Predicted expected value is simply how often our model "thinks" that g should occur in the training set:

$$\hat{E}[g] = \frac{1}{|\mathcal{T}|} \sum_{d=1}^{\mathcal{T}} \sum_{t_{y_i}} \hat{P}(t_{y_i}|H_{y_i}) g(t_{y_i}, H_{y_i})$$

Now, suppose that previous log-likelihood based only on $\mathcal{F}^k$ was $\mathcal{L}_{\hat{P}}$. If we add a feature g weighted by the multiplier α, the new likelihood of the training data would be:

$$\mathcal{L}_{\hat{P}+\{\alpha g\}} = \mathcal{L}_{\hat{P}} + \alpha\tilde{E}[g] - \log \hat{E}[e^{\alpha g}] \quad (7)$$

As our feature functions are binary, the weight α can be determined in closed form by differentiating the new log-likelihood $\mathcal{L}_{\hat{P}+\{\alpha g\}}$ with respect to α and finding the root of the derivative:

$$0 = \frac{\partial \mathcal{L}_{\hat{P}+\{\alpha g\}}}{\partial \alpha} \iff \alpha = \log\left[\frac{\tilde{E}[g](1-\hat{E}[g])}{\hat{E}[g](1-\tilde{E}[g])}\right] \quad (8)$$

Knowing α also allows us to compute the resulting improvement, or *gain*, in log-likelihood in closed form:

$$Gain = \tilde{E}[g] \log \frac{\tilde{E}[g]}{\hat{E}[g]} + (1-\tilde{E}[g]) \log \frac{1-\tilde{E}[g]}{1-\hat{E}[g]} \quad (9)$$

t_{y_i} = hydroelectric		
λ_f	f_S	Co-oc
3.36	dammed	0.031
1.95	cheap	0.006
1.68	governor	0.004
1.57	resources	0.005
1.53	predictions	0.003
1.52	analyst	0.004
1.51	capacity	0.005
1.29	electric	0.008
1.25	populated	0.003
1.25	mountains	0.004
1.22	plants	0.007
1.18	depend	0.003
1.06	ago	0.001
1.05	conserve	0.004
1.01	rivers	0.008
0.81	power	0.004
-0.50	ups	0.001
-0.62	president	0.001
-0.66	people	0.001
-1.14	county	0.001
-1.19	thursdays	0.001
-1.37	companies	0.001
-1.40	city	0.001
-1.83	friday	0.001
-2.07	today	0.001
-2.11	state	0.001
-2.25	los	0.001
-2.33	play	0.000
-2.96	time	0.001
-4.11	<null>	0.001
-4.39	years	0.001

t_{y_i} = projects		
λ_f	f_S	Co-oc
1.06	constructed	0.160
0.73	developed	0.177
0.58	environment	0.097
0.53	acres	0.113
0.51	building	0.167
0.48	fund	0.129
0.45	boulevards	0.076
0.44	work	0.130
0.43	plans	0.157
0.41	director	0.130
0.41	million	0.138
0.36	council	0.127
0.34	complete	0.112
0.34	art	0.088
0.31	proposal	0.141
0.30	program	0.119
0.29	commercial	0.099
0.28	producers	0.092
0.28	design	0.119
0.25	city	0.138
0.24	improvement	0.094
0.23	star	0.056
0.23	research	0.084
0.23	space	0.107
0.21	approved	0.130
0.19	featuring	0.066
-0.19	wins	0.046
-0.20	security	0.055
-0.22	today	0.065
-0.22	team	0.044
-3.82	<null>	0.086

t_{y_i} = railway		
λ_f	f_S	Co-oc
2.61	railroad	0.074
1.91	fe	0.053
1.63	rail	0.053
1.62	visitors	0.014
1.36	locomotive	0.071
0.97	ago	0.004
0.89	labor	0.010
0.86	town	0.010
0.84	training	0.015
0.77	collective	0.008
0.71	bus	0.016
0.64	travel	0.012
0.64	economic	0.008
0.63	passenger	0.024
0.55	centralize	0.008
0.54	historical	0.011
0.53	large	0.006
0.52	stands	0.005
0.45	worker	0.009
-0.40	county	0.003
-0.40	americans	0.003
-0.45	today	0.004
-0.74	tuesday	0.003
-0.75	thursdays	0.002
-0.83	los	0.003
-0.95	ups	0.003
-1.01	wednesday	0.002
-1.46	time	0.003
-1.53	friday	0.002
-2.81	years	0.003
-4.41	<null>	0.002

t_{y_i} = accidents		
λ_f	f_S	Co-oc
1.38	drivers	0.117
1.14	injuries	0.098
0.95	traffic	0.090
0.87	safety	0.076
0.81	occur	0.084
0.79	kill	0.084
0.78	crash	0.098
0.70	fatally	0.070
0.62	alcohol	0.055
0.58	hospitals	0.081
0.56	injured	0.102
0.54	cars	0.094
0.46	crews	0.051
0.45	dies	0.071
0.38	happened	0.052
0.35	involving	0.051
0.33	coasts	0.039
0.30	suffering	0.068
0.30	mile	0.064
0.28	dangerous	0.052
0.25	death	0.062
0.23	industry	0.033
0.19	california	0.049
-0.22	group	0.027
-0.23	years	0.036
-0.24	house	0.028
-0.29	government	0.023
-0.33	arrests	0.035
-0.34	game	0.020
-0.50	shot	0.026
-4.35	<null>	0.030

Figure 2: Context models for the terms "hydroelectric" and "projects" (from TREC query 307) and "railway" and "accidents" (from TREC query 436). We sort the features f_S by their λ_f weights, and draw a line between features with positive and negative weights, for readability. Also shown, for comparison, is the co-occurrence score between each of the support terms and the model target term, i.e. the co-occurrence of "accidents" and "safety" is 0.076.

3.6 Parameter Estimation

In the previous section we described how we can automatically induce the structure of a random field by incrementally adding the most promising candidate feature $g \in \mathcal{G}$. However, since the features $f \in \mathcal{F}$ are not independent of each other, adding a new feature will affect the balance of existing features, and therefore the objective function. We may further improve the objective by re-optimizing the weights.

Assume now that the structure $\mathcal{F}$ contains all the desired features. We adjust the set of weights Λ so that the objective function $\mathcal{L}_{\hat{P}}$ is maximized by computing the partial derivatives of $\mathcal{L}_{\hat{P}}$ with respect to each weight $\lambda_{f'}$, with the intention of driving these derivatives to zero:

$$\frac{\partial \mathcal{L}_{\hat{P}}}{\partial \lambda_{f'}} = \tilde{E}[f'] - \hat{E}[f'] \tag{10}$$

There is no closed-form solution for setting the weights to their optimal values, so we utilize an iterative procedure, a variation of gradient descent:

$$\lambda_f^{k+1} \longleftarrow \lambda_f^k + \beta \frac{\partial \mathcal{L}_{\hat{P}}}{\partial \lambda_f} = \lambda_f^k + \beta \left(\tilde{E}[f'] - \hat{E}[f'] \right) \tag{11}$$

Note that while $\tilde{E}[f]$ is computed only once for each feature f, we have to re-compute the value $\hat{E}[f]$ after every update. This makes the learning procedure expensive. However, learning is guaranteed to converge to the global optimum; $\mathcal{L}_{\hat{P}}$ is $\cap$-convex with respect to the weights λ_f.

3.7 Field Induction Algorithm

We are finally ready to bring together the components of the previous subsections into one algorithm for automatic induction of a context model for term t_{y_i}:

1. **Initialization**
   (a) Let the feature set $\mathcal{F}^0$ be the feature set for the term itself, with no context: $\mathcal{F}^0 = \{t_{y_i}\}$
   (b) Set initial weight $\lambda_f^0 = 1$ for this feature
2. **Weight Update**
   (a) Set $\lambda_f^{k+1} \leftarrow \lambda_f^k + \beta \left(\tilde{E}[f] - \hat{E}[f] \right)$ for each $f \in \mathcal{F}$
   (b) If there is noticeable change in likelihood, repeat step (2a)
3. **Feature Induction**
   (a) Enumerate the set of candidate features
   (b) For every candidate feature $g \in \mathcal{G}$ compute the optimal weight $\alpha_g = \log \left[\frac{\tilde{E}[g](1-\hat{E}[g])}{\hat{E}[g](1-\tilde{E}[g])} \right]$
   (c) For every $g \in \mathcal{G}$ compute expected improvement (gain) from adding g to the structure $\mathcal{F}$
   (d) Pick the candidate g that promises the highest improvement, add it to the structure $\mathcal{F}$, and set $\lambda_g = \alpha_g$
   (e) If there is noticeable change in likelihood, go to step (2), otherwise return $\mathcal{F}$ and Λ as the induced field

3.8 Final Details and Examples

While the general model creation algorithm is described above, in practice we had to impose a few limitations due to the intense computational resources required. The first limitation is on the candidate feature vocabulary $\{t_{x_1} \ldots t_{x_k}\}$. Rather than using the entire vocabulary, we use the 500 terms with the highest document frequency, that also occur at least once with the target term t_{y_i}. This subset is of course different for every t_{y_i}.

We stem and conflate terms using Porter [12], so that there is a single variable t_{x_i} (as well as a single variable t_{y_i}) for all terms that share the same root, i.e. car/cars. We do not want "cars" to be a support feature in our term context model of "car", as this defeats the purpose of learning a good general context model. The second limitation is on iterations. In step (2b), we iterate 12 times, rather than until no noticeable change in likelihood. In step (3e), we iterate 30 times, inducing exactly 30 features, for a total of 31 features including the <null> context seed feature. There is no reason for the these numbers beyond intuition. Our feeling was that 30 felt like a good "handful" of support features, not too many so as to overfit and not to few so as to miss the proper context. We certainly did not optimize these parameters, nor try other parameters. Future work can address this.

The final limitation we impose is on the size of each feature. Recall from equation (2) that a feature may include any number of observable terms t_x. While we have done some exploratory runs allowing two and three support terms, the retrieval results presented in section 5 were done using models limited to features with a single support term. Figure 2 shows models created for four different terms, trained on the ≈131,000 LA Times documents in TREC volume 5.

3.9 Term Context versus Co-occurrence

For an insightful aside, in Figure 2 we show co-occurrence scores across from λ_f feature weights. While we do no want to give the impression that context models and cooccurrence are directly comparable, because cooccurrences assume pairwise independence whereas context models use the entire set of terms, there is one interesting thing to note: The cooccurrence scores and λ_f weights do not yield the same relative ordering. Our approach is taking into account the dependencies between support features, and down-weights features that are already "covered" by previous features.

For example, in the model for "hydroelectric", the term "river" has the 2nd highest co-occurrence score, after "dammed". And yet it is 15th in terms of its λ_f weight. This is because the terms "dammed" and "river" are themselves not independent. When it comes to predicting hydroelectric, occurrences of "dammed" already cover most of the occurrences of "river". Therefore "river", while strongly associated with "hydroelectric" by itself, is not as important within a context that already includes "dammed".

Another important distinction between term-context and co-occurrence is the use of negative features. We do not go into details here, but hint at the usefulness of this in section 6. Further exploration on the relationship between term context models and co-occurrence is desirable, but beyond the scope of this paper.

4. AD HOC RETRIEVAL

Now that we have a framework for creating term context models we wish to use them for ad hoc retrieval. This is only one of many possible uses for these models, but have chosen this task to demonstrate their effectiveness. We begin at index time by creating context models for all terms in the collection vocabulary, independent of any query. Once each term's context model is learned, we iterate once more through the entire collection using the learned models to assess the degree to which each document's context supports the belief in each term (explained in section 4.1). The result of this calculation is a single value or *context score* for every term-document pair in the collection. These values are stored in inverted lists alongside *tf* values.

At retrieval time, this precomputed context score is combined with a standard *tf*-based approach, in our case Okapi BM25 (explained in section 4.2). The mixture between this *tf*-based score and the context model score yields and overall score for a query term in a document (explained in section 4.3). The following sections contain additional details.

4.1 Term Context Model Scoring

At index time, after models are trained, a per term-document context score is is calculated using the entire set of 31 feature functions $\mathcal{F}$ from the context model for term t_i in document d:

$$TCM(t_i,d) = \hat{P}(t_i|H_{i,d}) \tag{12}$$

Note that we do these calculations on the same collection on which we train the model. This is not a fallacy. This split is not necessary as it poses no more problem than "testing" an *idf* weight "trained" on the same corpus one is searching. Our models are not directly intended for prediction of unseen values. Like *idf*, they are meant to capture certain discriminating characteristics of the very corpus they will be used to search. Evaluation is done not on the model's classification accuracy, but on retrieval precision and recall.

Recall that equation (3) is estimated based on expected values. By now going through each document individually, we are determining how each of the documents in the collection deviate from this expected value. There will be occurrences of a term that are irregular, or "out of context", as trained on the collection as a whole. Such occurrences will have a lower score than occurrences that are more contextually standard. The model may assign a low context score even if the *tf* of that term is high. There will also be other documents in which the *tf* for a term is low, but the context score in that document is high. In a manner somewhat unchained from the actual *tf* of a query term, we are determining whether that term is a good fit for that document.

4.2 BM25 Baseline

We use the Okapi BM25 scoring function [13] as our baseline. BM25 has consistently been one of the top scoring functions in TREC style evaluations, and if we can improve upon it then we will show value in our methods. The exact form (for term t_i in document d) is given by equation (13) below, where k_1=2.0 and b=0.75. *tf* is the frequency of t_i in d, *dl* is the length of d, *avgdl* is the average document length across all documents in the collection, N is the total number of documents in the collection, and n is the number of documents in which the query term is found.

$$BM25(t_i,\ d) = \frac{tf}{k_1((1-b)+b\frac{dl}{avgdl})} \log \frac{1}{(n+0.5)/(N-n+0.5)} \quad (13)$$

As mentioned in section 1, this score combines both local features (*tf* and *dl*) with global features (*avgdl*, *N* and *n*) to come up with an overall weighting for a query term.

4.3 BM25 with Term Context Scoring (MIX)

In this section we will bring together the ideas introduced in this paper into a comprehensive whole. From section 4.1 we have a scoring function that increases as the context in a document moves toward the conceptually standard usage across the collection. From section 4.2 we have a scoring function that increases as term frequency increases. What we need is a scoring function that increases both as term frequency *and* contextual centrality increase.

This can be accomplished through data fusion techniques, wherein we use a linear combination of BM25 and the term context model as the score for a document. The idea is that the BM25 approach, which is primarily *tf*-based, is going to give a somewhat different ranking than the TCM approach, which is context-based. The "mistakes" that BM25 makes are hopefully different than TCM, and vice versa. When you fuse the scores the relevant documents should percolate to the top, while the spurious matches should drop out.

Returning to the example on the query term "fuel" from section 1, a document with five occurrences of fuel, but that talks about how interest rates fuel a housing bubble, is going to be lowered in the rankings. At the same time another document that only mentions fuel once, but does so in the context of energy and gas, is going to be boosted in the rankings. A document with five occurrences of fuel, that also talks about it in the context of energy, should be ranked fairly high.

While many fusion techniques are available, we use a simple linear combination of the scores (MIX):

$$MIX(t_i,d) = \gamma \cdot TCM(t_i,d) + (1-\gamma) \cdot BM25(t_i,d)$$

We understand that, at the moment, this is not a formally justified method for combining the two scores, as TCM ranges from 0.0 to 1.0 while BM25 is non-negative (greater than zero). How does a document with a BM25 score of 4.0 and a TCM score of 0.9 compare with another document's 4.3 BM25 score and 0.6 TCM score? While it is not yet clear, we are encouraged by the thought that, for any two documents with a similar BM25 score (i.e. similar *tf* and document length), differing TCM scores provide a differentiating factor. And vice versa.

Future work will address this combination of TCM with other features, but what we show in this paper is simply that there is value in the TCM approach: It works. Just as various combinations *tf* and *idf* took years to develop, so will better methods for utilizing TCM be developed in the future. Furthermore, as detailed in the next section, we found that results are fairly robust with respect to a wide range of γ settings. This indicates that *term context* as an *index-time* globally-informed feature is a good measure of the relevance of a document to a query term.

5. EXPERIMENTS

We use standard TREC datasets [15] for evaluation. Our dataset consists of the ∼131k documents from the LA Times in TREC volume 5 with 143 title-only queries from topics 301 to 450. Actually, there are 150 topics in this set, but seven queries had no relevant documents in the LA Times collection so are omitted from the evaluation. We present two models: BM25 and MIX. For all the terms in the query, we sum the weights over all terms t_i in the query Q given by each of these methods. Documents are sorted in decreasing order by these scores.

5.1 Results

Results are shown in figure 3, with the γ mixing parameter set to 0.5. In general, MIX outperforms BM25. Recall is slightly better, and average precision is about the same, but the highest gains are in the most important area: precision at low recall. These gains are 5-6% and are statistically significant. We feel these results justify the approach. However, in the next section we will provide a more in-depth analysis and see that the results are even better than they first appear.

These results are also quite robust with respect to the mixing parameter γ. We tested mixture weights ranging in stepwise 0.1 intervals from 0.0 to 1.0. Statistically significant improvements in roughly the same amounts and areas were obtained using γ ranging from 0.3 to 0.8, with the best mixture around 0.6 or 0.7. So the results we show are certainly not the best ones possible, but rather than tune the mixing parameter, we chose the "maximum entropy" value of γ=0.5 to demonstrate the robustness of the approach.

5.2 Results: Alternate View

While the results in the previous section clearly demonstrate the utility of our method, we decided to take a closer look to better understand what was happening. Precision at low recall improves 5-6%, but is this the whole story? Due to the nature of information retrieval and the large variability among queries, there is more than one way of getting this statistically significant increase. One potential explanation is that the majority of the queries could do slightly better, but the minority do much worse, for an overall positive average. This is not so beneficial, as a user trades the possibility of only slight improvement for harsh deterioration in retrieval quality. The benefits do not outweigh the risks.

However, we noticed while examining the results by hand that there are a large number of queries in which precision at low recall exhibits no real change, either positive or negative. This intrigued us. Our models use global information to capture the centrality of a term's contextual usage. For a good number of query terms, there is either not a lot of variation in contextual usage or there is already a high correlation between term frequency and contextual centrality. In those cases, term context modeling is not going to provide any benefit over BM25. But on the flip side, it also will likely do no worse, either. How might this affect the results we are seeing?

We decided to further examine this in the following manner: Figure 4 contains the exact same 143 queries as figure 3. However, they are split into two parts. On the left are all the queries (87 total) in which there was exactly 0% difference at interpolated 0.0 recall between BM25 and MIX. On the

All 143 Queries			
	BM25	MIX	
Total documents over all queries			
Retrieved:	143000	143000	
Relevant:	3535	3535	
Rel|ret:	2239	2285	+2.1⋆
Interpolated Recall - Precision			
at 0.00	0.5696	0.6012	+5.6⋆
at 0.10	0.4372	0.4604	+5.3⋆
at 0.20	0.3617	0.3636	+0.5
at 0.30	0.2811	0.2755	-2.0
at 0.40	0.2256	0.2239	-0.7
at 0.50	0.1970	0.1898	-3.6
at 0.60	0.1493	0.1481	-0.8
at 0.70	0.1241	0.1258	+1.4
at 0.80	0.0853	0.0869	+1.8
at 0.90	0.0593	0.0596	+0.6
at 1.00	0.0482	0.0484	+0.5
Average precision (non-interpolated)			
	0.2133	0.2152	+0.9⋆

Figure 3: Overall results: All 143 queries from TREC topics 301-450 (7 topics have no relevant docs)

Subset of 87 Queries			
	BM25	MIX	
Total documents over all queries			
Retrieved:	87000	87000	
Relevant:	2269	2269	
Rel|ret:	1497	1527	+2.0⋆
Interpolated Recall - Precision			
at 0.00	0.7973	0.7973	+0
at 0.10	0.5957	0.6125	+2.8⋆
at 0.20	0.4899	0.4905	+0.2
at 0.30	0.3687	0.3584	-2.8
at 0.40	0.3064	0.2970	-3.1
at 0.50	0.2660	0.2496	-6.2
at 0.60	0.1981	0.1972	-0.4
at 0.70	0.1741	0.1745	+0.2
at 0.80	0.1188	0.1212	+2.0
at 0.90	0.0853	0.0857	+0.4
at 1.00	0.0684	0.0689	+0.8
Average precision (non-interpolated)			
	0.2923	0.2909	-0.5

Subset of 56 Queries			
	BM25	MIX	
Total documents over all queries			
Retrieved:	56000	56000	
Relevant:	1266	1266	
Rel|ret:	742	758	+2.2⋆
Interpolated Recall - Precision			
at 0.00	0.2159	0.2966	+37.4⋆
at 0.10	0.1911	0.2240	+17.2⋆
at 0.20	0.1626	0.1664	+2.3⋆
at 0.30	0.1451	0.1467	+1.1
at 0.40	0.1001	0.1104	+10.3
at 0.50	0.0898	0.0971	+8.1
at 0.60	0.0736	0.0719	-2.4
at 0.70	0.0463	0.0503	+8.6
at 0.80	0.0332	0.0336	+1.2
at 0.90	0.0189	0.0192	+1.5
at 1.00	0.0168	0.0166	-1.4
Average precision (non-interpolated)			
	0.0904	0.0978	+8.1⋆

Figure 4: [LEFT] Subset of the overall results: 87 queries with zero change at 0.0 interpolated precision, [RIGHT] Subset of the overall results: 56 queries with non-zero change at 0.0 interpolated precision

Significance in all cases using a sign test at the 0.05 level is indicated by ⋆

right are all the queries (56 total) in which were was *non*-zero difference at interpolated 0.0 recall. In other words, these are all the queries in which context information either did better *or worse* than BM25 alone.

First, we see from the 87 "no change" queries that our intuition is fairly correct. There are a large number of queries on which context modeling has little to no effect. Rel|ret is slightly better, precision at 0.5 is slightly worse (though still not statistically significant), and at all other levels and on average there is practically no change. On the other hand, for those queries in which there is difference at 0.0 recall, better or worse, the term context approach shows a huge, significant 37% improvement at 0.0 and 17% improvement at 0.1 recall. Average precision improvement is also statistically significant. We saw similar patterns of improvement when we broke down the results from other (γ=0.3 to 0.8) mixture parameters as well.

These are important results. They indicate that when our technique works, it works extremely well. And, equally importantly, when our technique does not work, it does little to no hurt. Risk is minimized, which a desirable feature of any retrieval system.

Finally, please note the difference in absolute values between the two query subsets. The "no change" subset has much higher precision than the "improved" subset. We think this is further evidence of the robustness of our approach: When the original query is already well-performing, context models manage not to monkey with a good thing. On the other hand, when the original query yields poor results, context models manage to improve the query immensely. Context models may even be useful for predicting which queries will and will not perform well.

6. FUTURE WORK

There are a number of possible areas one could use term context models. We have tried to make absolutely clear that the term context modeling done in this paper is not the same as query expansion or pseudo-relevance feedback. However, we acknowledge that one could actually utilize context models in support of some of these techniques. For example, support feature weights are currently trained per term at index time, independent of an actual query. One could easily envision a pseudo-relevance re-training of these weights post-query time, on the top n retrieved documents. That would then alter the context score in the remainder of the collection, and may improve retrieval.

A second possibility involves query expansion. Returning to the example from section 1, suppose we have a model for fuel in which gas and coal are the two highest-weighted support features. We are already using fuel's context score alongside the BM25 score. However, that does not stop us from adding the BM25 scores for the terms gas and coal to the query, i.e. doing expansion. The key point is that we can also mix the context scores for gas and coal into the final score. So we would not just be adding more documents with high *tf* for gas and coal; we would be boosting documents which contained contexts that best supported all three terms: fuel, gas and coal. [16] reports that one of the disadvantages of query expansion is that "individual queries can be significantly degraded by expansion." If we do expansion not just by adding more terms, but by taking into account the centrality of those terms' context, we may be able to improve without risking significant degradation.

Perhaps the most interesting possibility for term context involves richer modeling. Recall from equation (2) that features may be conjuncts of one or more support terms t_{x_j}. We did an experiment in which we induced a model for the term "food" using the LA Times collection. In this model, the feature "drug" has a small, positive λ_f weight. There is clearly a relationship between these two terms, as evidenced by phrases such as "Food and Drug Administration". But both terms are used in enough other contexts that the affinity is small.

At this point, co-occurrence alone has nothing more to say

about these terms other than their slight affinity. However, in our experiment the context model automatically learned a very interesting multiple support term feature: {"drug" AND "police"}. It added this feature to the model with a strong negative weight.

This means that if "drug" is a document by itself there is a slightly higher likelihood of "food" belonging there. However, if both "drug" and "police" are in the document, there is strong supporting evidence against "food". (The term "police" by itself, *without* "drug", carries little evidence one way or the other.) By taking into account the non-independence of support features, the context model implicitly disambiguates between "drug" as a medication and "drug" as an illegal narcotic, and the effect that has on "food". This is something that straight co-occurrence cannot do. It is also something that the context model learned by itself, automatically, with no domain knowledge. We should be able to take advantage of this for retrieval.

We are encouraged by this example to pursue richer models of context. Maximum entropy and random field models are, as has been oft noted, ideal frameworks for the integration of all types of evidence. In the future, we hope to add more useful contextual features to our model. As we have shown, having a good statistical model of a term's context is useful for retrieval.

7. CONCLUSION

Term context models offer a new tool for assessing term presence in a document. By determining whether a document provides support (context) for a term, rather than using the observed frequency of that term, we have created a fundamentally different method for assessing similarity between a query term and a document. Furthermore we have shown that this approach is useful for retrieval, with huge improvements in precision at low recall. Though we are not the first to use maximum entropy or random field models for information retrieval, we are the first to use them in this manner. By explicitly modeling the context of a term we open the doors to many new applications.

8. REFERENCES

[1] D. Beeferman, A. Berger, and J. Lafferty. Text segmentation using exponential models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 35–46. 1997.

[2] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[3] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*, pages 380–393, 1997.

[4] W. Greiff and J. Ponte. The maximum entropy approach and probabilistic ir models. *ACM Transactions on Information Systems*, 18(3):246–287, 2000.

[5] P. Kantor and J. Lee. The maximum entropy principle in information retrieval. In *Proceedings of the ACM SIGIR Conference*, 1986.

[6] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[7] V. Lavrenko and J. Pickens. Polyphonic music modeling with random fields. In *Proceedings of the 11th Annual International ACM Conference on Multimedia*, pages 120–129, November 2003.

[8] A. McCallum and N. Ghamrawi. Collective multi-label text classification. In *Proceedings of CIKM*, pages 195–200, Bremen, Germany, 2005.

[9] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *28th Annual ACM SIGIR Conference*, pages 472–479, Salvador, Brazil, 2005.

[10] R. Nallapati. Discriminative models for information retrieval. In *Proceedings of the ACM SIGIR Conference*, Sheffield, UK, 2004.

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

[12] M. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[13] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *3rd annual Text REtrieval Conference*, NIST - Gaithersburg, MD, 1994.

[14] R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10:187–228, 1996.

[15] E. Voorhees and D. Harman. Overview of the sixth text retrieval conference (trec-6). *Information Processing and Management*, 36(1):3–35, 2000.

[16] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.

[17] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004.