That's Not My Question: Learning to Weight Unmatched Terms in CQA Vertical Search

Boaz Petersil
Dep. of Electrical Eng.
Technion, Haifa, Israel
boaz.petersil@gmail.com

Avihai Mejer
Yahoo Research
Haifa, 31905, Israel
amejer@yahoo-inc.com

Idan Szpektor
Yahoo Research
Haifa, 31905, Israel
idan@yahoo-inc.com

Koby Crammer
Dep. of Electrical Eng.
Technion, Haifa, Israel
koby@ee.technion.ac.il

ABSTRACT

A fundamental task in Information Retrieval (IR) is term weighting. Early IR theory considered both the presence or absence of all terms in the lexicon for ranking and needed to weight them all. Yet, as the size of lexicons grew and models became too complex, common weighting models preferred to aggregate only the weights of the query terms that are matched in candidate documents. Thus, unmatched term contribution in these models is only considered indirectly, such as in probability smoothing with corpus distribution, or in weight normalization by document length. In this work we propose a novel term weighting model that directly assesses the weights of unmatched terms, and show its benefits. Specifically, we propose a Learning To Rank framework, in which features corresponding to matched terms are also "mirrored" in similar features that account only for unmatched terms. The relative importance of each feature is learned via a click-through query log. As a test case, we consider vertical search in Community-based Question Answering (CQA) sites from Web queries. Queries that result in viewing CQA content often contain fine grained information needs and benefit more from unmatched term weighting. We assess our model both via manual evaluation and via automatic evaluation over a clickthrough log. Our results show consistent improvement in retrieval when unmatched information is taken into account. This holds both when only identical terms are considered matched, and when related terms are matched via distributional similarity.

Keywords: Unmatched Terms; Document Ranking; Community-based Question Answering

1. INTRODUCTION

One of the fundamental tasks in Information Retrieval (IR) is term weighting, which refers to the assessment of a weight for each term appearing in the document collection, and similarly in the input query. Early Probabilistic IR theories considered the presence or absence of all terms in the lexicon for ranking, both in the query and the documents [29, 26]. However, term weights in these models were found difficult to compute, and the main line of research around weighting models chose to consider mainly the weights of the query terms that are matched in candidate documents. Indeed, weighting schemes such as TF-IDF [31], BM25 [28] and statistical language models [33, 24, 40], as well as Learning To Rank (LTR) methods [20], are primarily based on considering the contribution of the *matched terms*, those query terms that appear also in the document. Though *unmatched terms*, i.e. terms that appear only in the query or only in the document, are not completely ignored, they are considered indirectly in these models, such as by using the document length for weight normalization or via corpus-based smoothing of maximum likelihood estimations.

As suggested by early probabilistic models we argue that analyzing directly unmatched terms may provide additional cues to the relevance of a candidate document to the query. Indeed, while the contribution of stop-words, such as determiners and modals, can be largely ignored, unmatched named entities are strong indicators of semantic differences between the query and the document. For example, for the query "*most deadliest snake*", the document title "*where can I find a list of the deadliest snakes*" is more relevant than "*which is the most deadliest snake in Russia*", though the first title is longer than the second, and second title contains all of the query terms in the right order.

Another intuition regarding direct modeling of unmatched terms refers to the percentage of query terms that are covered in the document. We would like to explicitly indicate that for two queries, a short one and a long one, if both match the same set of terms within a candidate document, this document is likely to be of less relevance for the longer query, which contains more unmatched terms, compared to the shorter one. As an example consider the queries "*most deadliest snake*" and "*most deadliest snake in Russia*" and the candidate document "*where can I find a list of the deadliest snakes*". We would like to explicitly express the lack of relevance of the document to the second query due to unmatched query terms.

We expect the subtleties between different types of unmatched terms to show especially for Web queries with fine-grained information-need. Therefore, we focus in this paper on Web queries with question intent, which constitute ∼10% of the Web queries issued to a search engine [36]. Examples for such Web queries are those resulting in the searcher clicking on a question page belonging to Community-based Question Answering (CQA) sites, such as Yahoo Answers, StackExchange and Quora, and are called here *CQA queries*.

SIGIR '16, July 17-21, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2911451.2911496

Our retrieval scenario is vertical search [25, 2], in which content of a CQA sub-collection should be retrieved on top of general Web search.

In this work, we address this vertical search task by introducing a term weighting model that directly considers the contribution of unmatched terms for ranking. However, instead of a probabilistic framework, we utilize LTR as a ranking framework. To this end, we employ a large set of state-of-the-art features, which capture various attributes of matched terms, both statistical ones (such as variants of TF-IDF) as well as syntactic ones (such as Part-Of-Speech (POS) tags) [20, 9]. We then design "mirror" features that evaluate similar attributes, but for the unmatched terms. These novel mirror features are provided, together with the features that correspond to matched terms, as input to an LTR algorithm, which learns the relative weights of the different features using click-through training data.

Prior work in document ranking noted that, occasionally, different terms in the query and the document actually convey related meanings or even the same meaning (*e.g.* '*guy*' – '*man*', '*drink*' – '*alcohol*') and should be considered as matching for document ranking. One common approach to handle this lexical gap is via translation models, which include some similarity measure between query and document terms as part of term matching [4, 17, 39, 14]. In order to analyze the contribution of our unmatched term modeling under such "soft matching" schemes, we introduce a "soft" variant of all the features we compute for matched terms in our LTR framework, which is based on distributional similarity between terms. We then provide a similar soft variant of our unmatched-term features that complement the soft matched-term features. This should enable the unmatch-based features to better account for only semantically unmatched terms instead of terms that have similar meanings.

We conducted experiments on a vertical search setting that searches a Web query over a large collection of question pages from Yahoo Answers. The contribution of our unmatch-based features for term weighting was evaluated under two setups: a) large-scale automatic evaluation over a click-through query log; b) manual evaluation of the top retrieved documents for a set of tested queries. We compared our model to a state-of-the-art LTR model that utilizes only features that correspond to matched terms. The tested models were assessed both under the *exact* matching modeling, in which only identical terms are considered matched, and the *soft* matching modeling, where terms may be partially matched via distributional similarity. Our novel features provided consistent improvement in document ranking on both scenarios, showing the benefit of directly considering unmatched terms for term weighting.

2. RELATED WORK

Unmatched terms were addressed in prior IR ranking models in different ways, both for general search and for CQA search. We distinguish between two types of unmatched terms: a) terms that appear in the candidate document but not in the query, denoted as *excessive terms*; b) terms that appear in the query but not in the candidate document for ranking, denoted as *missing terms*.

Probabilistic information retrieval theory accounts for presence or absence of all terms in the lexicon, both in the query and in a candidate document for ranking [29]. Similarly, early reformulation of language models for IR (LMIR) [26] considered the query as a set of words, and modeled excessive terms in the document by their ability to generate terms not in the query. However, term weighting computation becomes a difficult problem under these frameworks [30, 33, 40], especially when no relevance feedback is considered. Therefore, recent ranking models, and specifically term weighting models, focus mainly on the matched terms between the query and a candidate document.

The overall ranking score of a document is typically the sum of the weights of the terms in the document that match (to some extent) the query terms. Therefore, document term weights in popular weighting schemes are non-negative and the effect of missing terms in a candidate document is considered indirectly by not contributing their weights to the document ranking score. This is the case in common probabilistic and vector space models, such as Binary Independence Model (BIM) [30], TF-IDF [31], Okapi BM25 [28], divergence from randomness [1], and multinomial language models, which view the query as a sequence of terms [33, 24, 41, 40]. Specifically, language models were extensively explored for CQA retrieval and were extended in different ways to incorporate meta data like categories [7], and the question focus and topic [13]. The same principle of scoring documents by summing matching term weights is also behind different weighting terms at the query side [3, 43].

In another line of ranking research, Learning to rank (LTR) approaches [20, 19] were introduced for learning to combine many features in a supervised way. Various learning algorithms were proposed, such as SVMRank [8] and LambdaMart [38], which may assign negative weights to some features. Still, the features themselves are typically derived for the matched terms, and therefore LTR algorithms learn the relative contribution of each feature with respect to the matched terms. Most derived features are statistical in nature, such as variants of term frequency and document frequency scores [20], and were also utilized in supervised ranking models in CQA [18, 37]. Carmel et al. [9] showed that utilizing features derived from syntactic analysis of the document title improves ranking performance for CQA queries. In a related task of answer sentence ranking within the field of Question Answering, tree kernels that incorporate semantic and syntactic features of the words provide state-of-the-art performance [34]. Still, the overall approach weigh in the number of matched sub-trees but not the unmatched ones.

Missing terms, which appear in the query but not in the document, received considerable attention within attempts to address the lexical gap problem: improving the matching between query and document terms that are not lexically identical but convey similar meaning. One common approach incorporates a translation model as part of term weighting [4, 17]. This approach was found useful also for retrieving CQA content, where translation models were used within LMIR for retrieving related questions [16, 39, 42] as well as for ranking CQA documents for Web queries [37]. Recently, lexical semantic similarity between terms via distributed representations, such as word2vec [23], was found helpful in several IR tasks, including query term weighting [43] and as features in a LTR framework for answer retrieval [10]. Ganguly et al [14] employed similarity between word embedding vectors within a translation model for LMIR as means to overcome the lexical gap between queries and documents, where it outperformed a language model extended with latent topics.

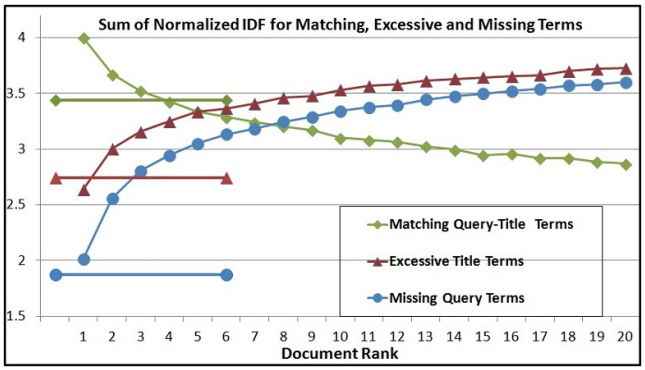

Figure 1: Sum of IDF values (normalized) for the matched, excessive and missing terms, computed separately over all non-clicked documents ranked at positions 1 to 20 (the three plots) and over the clicked documents (three horizontal lines).

Prior work captured the effect of excessive terms (appearing only in the document) on the ranking score mainly by their contribution to overall document context or structure. Many vector space and probabilistic models (*e.g.* TF-IDF, BM25, language models) utilize the document's length as a degrading parameter for term weights, *e.g.* as the denominator of maximum likelihood estimation or in ℓ_2 vector normalization. Other models include all document terms when modeling a global latent representation for each document. One line of works uses latent topics (*e.g.* LDA [5]) as additional smoothing elements within LMIR [35]. This extension was shown useful also in retrieval of related CQA questions [6]. Another approach is to embed the document in a latent space. Latent Semantic Indexing [12] utilizes SVD to represent documents and queries within a reduced dimension space based on the main singular values of a term/document co-occurrence matrix. Lately, deep learning was shown useful for ranking by embedding the query and document texts into a shared latent space, within Web search [15] and within Question Answering [32]. Such approaches were not evaluated under the CQA vertical search setting yet, whose query length distribution and query attributes is quite distinguishable from general Web search and from question/answer datasets [9, 36].

3. YAHOO ANSWERS DATASET

In this work we perform our analysis and experiments on a large document collection taken from Yahoo Answers. Yahoo Answers is a popular CQA website containing questions about diverse topics, such as sports, healthcare, politics, science and many others. Each question page in the site consist of: a) a title, which is typically a short summary of the question, b) a body, containing a detailed description of the question, and c) all the answers provided for this question. We collected 54 million question pages from Yahoo Answers (referred to as our *corpus*) and indexed them using Lucene[1].

We also randomly sampled Web queries that were issued to a popular search engine and resulted in a click on one of the pages in our corpus by analyzing the *search log*. For each sampled query the top 100 results from our corpus were retrieved using Lucene's BM25 ranking function over all fields (title, body, answers). We retained the set of queries for which the clicked page for the query (as extracted from the search log) was found among the top 100 Lucene results. After this process, our click-based query collection consists of 136,000 queries.

[1]lucene.apache.org

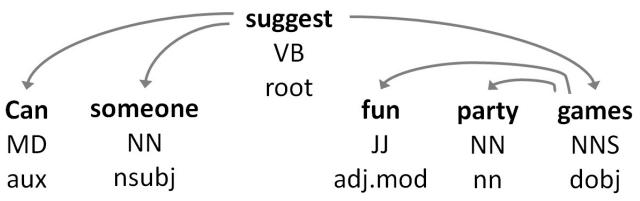

Figure 2: POS tagging and dependency parse tree for the question *Can someone suggest fun party games?*. The upper label of each token is its POS tag and the lower label is its syntactic role.

Since search-engines show mostly the title of a question page on the search result page, Carmel et al. [9] reason that the relevance of the title to the query is one of the main reasons for a user to click on the page. This especially makes sense as the title is often a good summary of the question in the page. Furthermore, both title and query are concise and usually do not contain redundant information. Therefore, we expect that unmatched term weighting would help in retrieval under this scenario. Following them [9], we analyze and model unmatched terms only between the title of a Yahoo Answers question page and the target query.

4. UNMATCHED TERM ANALYSIS

To further motivate our modeling approach we analyze the properties of the unmatched terms: terms that appear in the candidate document but not in the query (*excessive*), and terms that appear in the query but not in the candidate document (*missing*). To this end, we sampled 20,000 queries[2] from our query collection and examined the matched and unmatched terms between each query and the titles of the retrieved documents (using Lucene). We compared the analysis statistics between documents that were clicked by the user who issued the query, denoted as *clicked* documents, and the other top retrieved documents, denoted as *non-clicked* documents.

4.1 IDF Distribution Analysis

First we examined the distribution of IDF values among the matched and unmatched terms. To this end, we computed the sum of IDF values for the matched terms and the excessive terms in each title (normalized by the title length). We also computed the IDF sum for the query's missing terms (normalized by the query length). These three indicators are plotted in Fig. 1, where each point in the plot represents the value averaged over all non-clicked titles ranked at the *i'th* position by the BM25 ranking, for *i=1..20*. The three horizontal lines in Fig. 1 correspond to the values of the three indicators averaged over the clicked documents. Note, higher values intuitively reflect more relevance in the matched term curve but less relevance in excessive and missing term statistics.

[2]Taken from our training set – see Sec. 5.4

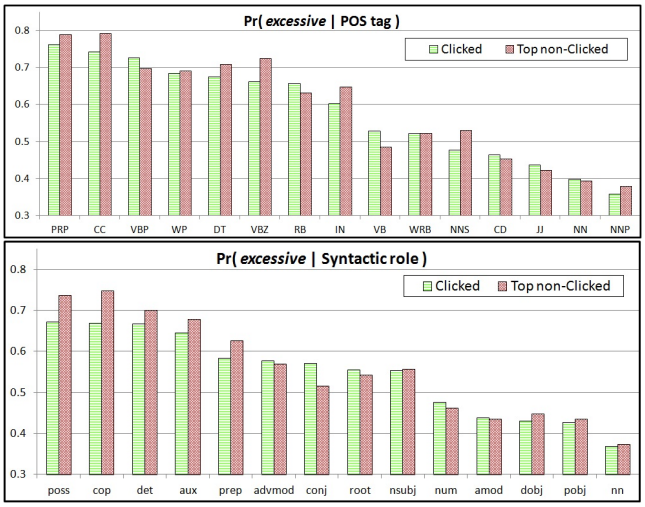

Figure 3: Probability of a term in the title not to match any query term given its POS tag (top chart) or syntactic role (bottom chart)

The matching-term IDF values (the diamond shape points) indicate that the clicked document (the horizontal line) is comparable, on average, only to the document at the $4'th$ position. This means that, with respect to matched terms, top ranked non-clicked documents usually contain as much and even more IDF volume compared to the clicked document. Yet, excessive and missing term analyses reveal complementing phenomena. First, on average, only non-clicked documents that are ranked first have the excessive term indicator lower than the value for clicked documents. This may indicate that while several non-clicked documents contain "important" (high-IDF) query terms in their title, driving them to high ranking positions, they also contain additional "important" terms that do not appear in the query and may change the meaning of the title compared to the query.

Second, the indicator for missing terms in clicked documents stands out even more, as it is lower compared to all non-clicked documents. This could indicate that in CQA queries it takes more than one term to express the gist of the information needed. While some non-clicked documents may include in their title high-IDF query terms, which correspond to "important" terms, they also tend to leave-out more "important" terms compared to the clicked document.

Assuming that click analysis is a useful approximation of relevance analysis, these results suggest that matched term statistics reveal only some aspects of the title's relevance to a searcher's information need. More relevance aspects may be further exposed by explicitly modeling unmatched terms.

4.2 Syntactic Analysis of Excessive Terms

Carmel et al. [9] showed that document terms with different syntactic properties should be weighted differently for retrieval. Hence, we examine similar syntactic properties of excessive terms, namely POS tags and dependency roles.

To this end, all titles in our corpus were syntactically analyzed using the Stanford parser[3] under the "all typed dependencies" setting. Then, for each title term in a retrieved document we extracted its POS tag and syntactic role (the dependency relation in which the term is the dependent). Fig. 2 presents an example for this analysis. Finally, for each syntactic property we counted its total occurrences in each title and its occurrences within excessive terms, and computed their ratio. The ratio of these two counts represents the probability of each syntactic property to be an excessive term. In Fig. 3 we depict two probability families, averaged across all analyzed queries: a) for all clicked documents; and b) for the three highest ranking non-clicked documents (representing the "toughest" competitors to beat for ranking the clicked documents on top of non-clicked ones). For clarity, the charts contain only the results for the 15 most common POS tags and syntactic roles, and they are sorted in decreasing order of the probability value.

Looking at Fig. 3 we observe large differences in excessive term probability between different syntactic tags. For example, in clicked items, this probability for pronouns (PRP - usually a low IDF stopword) is $\sim$0.75, while for proper nouns (NNP) it is only $\sim$0.35. Such large differences echoes previous observation [9] that there is a possible gain in modeling differently terms with different syntactic tags.

Comparing the statistics between clicked titles and the top non-clicked titles, we can see that in quite a few properties there are distinguishable differences between clicked and non-clicked titles. These differences suggest that excessive term weighting may improve if syntactic properties will be considered. As an example we look at verbs, which are important terms in a query (usually capturing the main activity asked about). The syntactic roles that are often associated with verbs in the bottom chart are '*aux*', '*cop*', '*root*' and sometimes '*conj*'. These roles can be partitioned into two groups. The first group contains '*aux*' and '*cop*', whose excessive probability is higher in non-clicked titles. The second group contains '*root*' and '*conj*', which are more likely to be excessive in clicked titles. This shows that syntactic properties can provide more fine-grained distinctions between similar terms or even for the same term when assuming different roles.

As another example, the charts in Fig. 3 also show that nouns (POS tags: NN*, Dep roles: conj, nsubj, dobj, pobj, nn) are more likely to be excessive in non-clicked documents. As nouns typically contain the main participants of a question, it is important to match all (or most) of them to align the exact semantics of the query to that of a title. Therefore, directly assessing both matched and unmatched nouns could improve retrieval. In Fig. 3 the only reverse case is with '*conj*', which is more likely to be excessive in clicked titles. Yet, conjunctive nouns, such as in the example "*good websites that stream movies and tv shows*", may be skipped (and become excessive terms) while maintaining the same semantic gist of the question.

The analysis in this section suggests that modeling statistical properties as well as syntactic properties of unmatched title terms may lead to better assessment of document relevance for CQA queries. We next explicitly construct features for unmatched terms, within a Learning To Rank (LTR) framework, which take these properties into consideration.

5. MODELING UNMATCHED TERMS

The task of our ranking algorithm is to rank a set of candidate documents D given a CQA query q. We follow a standard LTR scheme [19, 20, 9] and define a mapping function

[3]http://nlp.stanford.edu/software/lex-parser.shtml

Feature	Formulation
L_1	$\sum_{t \in q} c(t,d) \times \delta(t \in d)$
L_2	$\sum_{t \in q} \log(c(t,d)+1) \times \delta(t \in d)$
L_3	$\sum_{t \in q} \frac{c(t,d)}{\lvert d \rvert} \times \delta(t \in d)$
L_4	$\sum_{t \in q} \log\left(\frac{c(t,d)}{\lvert d \rvert}+1\right) \times \delta(t \in d)$
L_5	$\sum_{t \in q} \log\left(\frac{\lvert C \rvert}{df(t)}\right) \times \delta(t \in d)$
L_6	$\sum_{t \in q} \log\left(\log(\frac{\lvert C \rvert}{df(t)})\right) \times \delta(t \in d)$
L_7	$\sum_{t \in q} \log\left(\frac{\lvert C \rvert}{c(t,C)}+1\right) \times \delta(t \in d)$
L_8	$\sum_{t \in q} \log\left(\frac{c(t,d)}{\lvert d \rvert} \log(\frac{\lvert C \rvert}{df(t)})+1\right) \times \delta(t \in d)$
L_9	$\sum_{t \in q} c(t,d) \log\left(\frac{\lvert C \rvert}{df(t)}\right) \times \delta(t \in d)$
L_{10}	$\sum_{t \in q} \log\left(\frac{c(t,d)}{\lvert d \rvert} \frac{\lvert C \rvert}{c(t,C)}+1\right) \times \delta(t \in d)$
$G_1(p)$	$\sum_{t \in d \wedge POS(t)=p} c(t,q)$
$G_2(p)$	$\sum_{t \in d \wedge POS(t)=p} idf(t) \times c(t,q)$
$G_3(p)$	$\sum_{t \in d \wedge CPOS(t)=p} c(t,q)$
$G_4(p)$	$\sum_{t \in d \wedge CPOS(t)=p} idf(t) \times c(t,q)$
$G_5(sr)$	$\sum_{t \in d \wedge SR(t)=sr} c(t,q)$
$G_6(sr)$	$\sum_{t \in d \wedge SR(t)=sr} idf(t) \times c(t,q)$
H_1	BM25 score
H_2	log(BM25 score)
H_3	LMIR with Dirichlet smoothing parameter β

Table 1: Matched term features used in [9]; $c(t,X)$ – term frequency of t in X; $df(t)$ – document frequency of t in our corpus C; $idf(t) = log\left(\frac{\lvert C \rvert}{df(t)}\right)$; $\lvert X \rvert$ – total number of terms in X; $POS(t)$, $CPOS(t)$ and $SR(t)$ are the POS tag, coarse-POS tag and syntactic role of t respectively; $\delta()$ is the indicator function.

$\mathbf{\Phi}(q,d) \in \mathbb{R}^n$ from pairs of a query q and a candidate document d to the vector space $\mathbb{R}^n$. Our algorithm uses a weight vector $\boldsymbol{w}$ to compute a score for each $d \in D$ via the inner product $s(q,d) = \boldsymbol{w} \cdot \mathbf{\Phi}(q,d)$. Finally, the candidate documents are ranked according to the value of $s(q,d)$, where the higher the score for some document d, the higher its rank in the retrieved list. The goal of the learning algorithm is to find weights $\boldsymbol{w}$ such that more relevant documents will have high score compared to less relevant ones.

Our work focuses in the design of a new feature mapping $\mathbf{\Phi}(q,d)$ that captures both missing and excessive terms. We build on the work of Carmel et al [9] who proposed only features that consider matched terms and extend their mapping in two ways: (1) taking into account unmatched terms; (2) relaxing the notion of matched/unmatched terms, and allowing soft-matching between terms in the query and the candidate, based on distributional similarity.

We describe in Sec. 5.1 the state-of-the-art features proposed by Carmel et al [9], which are our baseline and starting point. Additionally, we present in Sec. 5.2 an abstraction of these features having in mind our goal to introduce their corresponding new features for unmatched terms. Finally, in Sec. 5.3, we incorporate soft-matching into all features presented until then. We use a previous weight learning scheme [9] (Sec. 5.4) in order to replicate their work as a baseline and have a fair comparison of our new features.

5.1 Matched Term Features

Carmel et al [9] also addressed the task of vertical search for CQA queries within an LTR framework (see Sec. 2). They proposed two types of features that analyze matched terms: standard statistical features [20], such as TF-IDF, and new syntactic-based features that collect matched term statistics for each POS tag and syntactic role separately. All these features are summarized in Tab. 1.

Carmel et al mainly analyzed the performance of the document title for matching the query, arguing that in CQA content, the title is a good summary of the question being answered within the document. Therefore, all statistics are derived only from the document's title, except for the BM25-related features (H_{1-2}) which are computed over the whole document. Under this formulation, which we follow, q and d represent the list of terms in the query and the document's title respectively. Features L_{1-10} and H_{1-3} refer to standard statistical features, while $G_{1-4}(p)$ and $G_{5-6}(sr)$ are feature families that are generated for each POS tag p and syntactic role sr, respectively. For example $G_1(IN)$ is the feature generated for the IN (preposition) POS tag and $G_5(root)$ is the feature generated for the *root* syntactic role.

Each of the features L_{1-10} and G_{1-6} can be viewed more abstractly as (possibly conditional) term summing of a product of two terms: (1) a count (or a function of it) of some event, denoted by f_{F_i}; and (2) a boolean predicate or a numeric value indicating some matching between q and d:

$$L_i(q,d) = \sum_{t \in q} f_{L_i}(t,d) \times \delta(t \in d) \quad (1)$$

$$G_i(q,d) = \sum_{t \in d \wedge cond_{G_i}(t)} f_{G_i}(t,d) \times c(t,q) \quad (2)$$

The matching indication part in L_i is $\delta(t \in d)$ – whether the query term appears in the document. The indication part of G_i is $c(t,q)$ – the occurrence count of the title term in the query, which is 0 for unmatched. Examples for instantiating these abstractions with specific statistics are: a) for L_2, $f_{L_2} := \log(c(t,d)+1)$; and b) for G_1, $f_{G_1} := 1$ and $cond_{G_1} := (POS(t) = p)$.

5.2 Unmatched Term Features

We now introduce our novel features, which induce the f_{F_i} signals in parallel to their counterpart matched term features L_{1-10} and G_{1-6}. Yet, instead for the matched terms, the new feature families do so for the set of excessive terms, denoted by EXL and EXG, and for the set of missing terms, denoted by MIL and MIG. We present the generic representations of these feature families similarly to (1) and (2):

$$EXL_i(q,d) = \sum_{t \in u(d)} f_{L_i}(t,d) \times (1 - \delta(t \in q))$$

$$EXG_i(q,d) = \sum_{t \in d \wedge cond_{G_i}(t)} f_{G_i}(t,d) \times (1 - \delta(t \in q))$$

and,

$$MIL_i(q,d) = \sum_{t \in u(q)} f_{L_i}(t,q) \times (1 - \delta(t \in d))$$

$$MIG_i(q,d) = \sum_{t \in q \wedge cond_{G_i}(t)} f_{G_i}(t,q) \times (1 - \delta(t \in d))$$

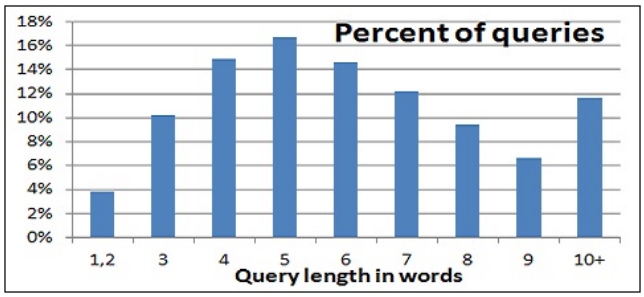

Figure 4: Percentage of queries by length

The differences between these new feature families and their matched term counterparts are in: a) the matching indicators, which turn into unmatching indicators; and b) the term sets over which the summation is performed. For example, the unmatching indicator in the excessive feature family EXL_i is $(1 - \delta(t \in q))$, which is 1 only if the document term is not in the query. In addition, the summation in EXL_i is over $u(d)$, which stands for the set of unique[4] terms in d from which we pick the excessive terms.

We note that for MIG_i, syntactic analysis of the query is required. We used the Stanford parser for query parsing as well, but found that the query dependency trees were of low quality. Therefore, in our experiments we only use the POS tags for queries, and hence only features MIG_{1-4}, leaving the reliable generation of features MIG_{5-6} for future work.

5.3 Soft Matching Formulation

As discussed in Sec. 2, some mismatches in exact matching scheme should actually be accounted as (at least partial) matches, such as in the case of synonyms or related words, *e.g.* '*guitar*' and '*riff*'. We extend our proposed features and present a novel soft matching formulation of all our matched and unmatched features (except BM25-related H_{1-2}). To the best of our knowledge, this is the first formulation in the context of the standard set of LETOR features [20].

We start with a lexical similarity function $sim(t_q, t_d) \in [0, 1]$ between a query term t_q and a document term t_d. The closer the function's value is to 1 the more similar the two terms are. We follow recent successes with word embedding similarity and use in this work:

$$sim(t_q, t_d) := \max(\cos(sg(t_q), sg(d_q)), 0) \ ,$$

where $sg(t)$ is the word embedding vector of term t learned by the SkipGram algorithm [23]. We define $sim(t, t) = 1$ for every word similarity with itself and $sim(t, u) = 0$ if $t \neq u$ and either t or u are not in the lexicon.

To incorporate the similarity score $sim(t_q, t_d)$ into our features we find the best matching counterpart term for each query term and for each document term:

$$\begin{aligned} bm_d(t_q) &= \arg\max_{t_d \in d} sim(t_q, t_d) \\ bm_q(t_d) &= \arg\max_{t_q \in q} sim(t_q, t_d) \\ \delta_s(t, d) &= sim(t, bm_d(t)) \\ \delta_s(q, t) &= sim(bm_q(t), t) \ , \end{aligned}$$

[4]Term repetition is avoided since the number of occurrences of the term t in d is already counted in f_{L_i}.

where $\delta_s(t, d)$ and $\delta_s(q, t)$ are soft indicator functions that capture how well a query (document) term is matched against the document (query) via its similarity score with its best match. We note that if $sim()$ would only return 1 for exact match and 0 otherwise, $\delta_s()$ would become $\delta()$.

Finally, we extend our features using $bm()$ and $\delta_s()$:

$$\begin{aligned} L_i^s(q, d) &= \sum_{t \in q} f_{L_i}(bm_d(t), d) \times \delta_s(t, d) \\ G_i^s(q, d) &= \sum_{t \in d \wedge cond_{G_i}(t)} f_{G_i}(t, d) \times c(bm_q(t), q) \times \delta_s(q, t) \\ EXL_i^s(q, d) &= \sum_{t \in u(d)} f_{L_i}(t, d) \times (1 - \delta_s(q, t)) \\ EXG_i^s(q, d) &= \sum_{t \in d \wedge cond_{G_i}(t)} f_{G_i}(t, d) \times (1 - \delta_s(q, t)) \\ MIL_i^s(q, d) &= \sum_{t \in u(q)} f_{L_i}(t, q) \times (1 - \delta_s(t, d)) \\ MIG_i^s(q, d) &= \sum_{t \in q \wedge cond_{G_i}(t)} f_{G_i}(t, q) \times (1 - \delta_s(t, d)) \ , \end{aligned}$$

where we simply replace (or augment where necessary) the indicator function with the soft indicator variant, and instead of gathering statistics from exact match occurrences, we gather them from the occurrences of the best-match. If a query term appears as-is in the document (exact match), our feature scores are exactly as for exact matching. Yet, when a query term does not exactly appear in the document (or vice versa) instead of returning a matched feature value of zero, we resort to counting with respect to its best soft match instead. We note that a similar formulation using best-matches is utilized by Liu et al [21] for computing similarity between short documents.

5.3.1 *Language Model with Soft Matching*

Prior work showed that extending LMIR with some similarity notion between terms improves retrieval results [39, 14]. We therefore extend our language model feature H_3 in a similar way, following the formalism of Xue et al [39]:

$$\begin{aligned} H_3^s(q, d) &= \sum_{t \in q} \log(P(t|d)) \\ P(t|d) &= \frac{|d|}{|d|+\beta} \left((1 - \alpha) \frac{c(t, d)}{|d|} + \alpha P_{tt}(t|d) \right) + \frac{\beta}{|d|+\beta} P_c(t) \\ P_{tt}(t|d) &= \sum_{t_d \in d} \frac{sim(t, t_d)}{Z(t, d)} \times \frac{c(t_d, d)}{|d|} \quad Z(t, d) = \sum_{t_d \in d} sim(t, t_d) \ , \end{aligned}$$

where q is the query term list; d is the document term list and $|d|$ is its length; $c(t, d)$ is the term-frequency of t in d; $P_c(t)$ is maximum likelihood estimation (MLE) of t in our corpus; Z is a probability normalizer; and α and β are hyper parameters to be tuned.

We note that H_3^s is a variant of Xue et al's language model. Instead of using a translation table as P_{tt}, we followed Ganguly et al [14], who suggested a variant of P_{tt} based on a similarity function $sim()$ (normalized into a probability distribution). Note, when $sim()$ represents exact matching, H_3^s becomes H_3.

5.4 Model Weight Learning

We learned the weights $\boldsymbol{w}$ of our ranking algorithm in a semi-supervised manner based on clickthrough data. The goal of the learning algorithm is to find a vector $\boldsymbol{w}$ such that for each query in the training data, the corresponding clicked document will be ranked as high as possible. We used the online variant of SVMRank [8] with the AROW update [11] as done before [9]. Specifically, for each training query the algorithm first re-ranks the top 100 documents retrieved by Lucene using the currently learned ranker. Then, it selects the top K ranking documents, excluding the clicked document. The algorithm then updates $\boldsymbol{w}$ such as for this query the clicked document would increase its ranking score compared to the selected K documents.

We split our query collection (see Sec. 3) into a 61,000 query training-set, a 14,000 query validation-set and a 61,000 query test-set. The validation-set was used to tune the various hyper-parameters for each tested model *separately*, namely, the number of training rounds, the value of K, and the AROW hyper-parameter r. The only hyper parameters that were tuned once for all models are $\beta = 1$, $\alpha = 0.5$ for the H_3 and H_3^s LMIR features. Specifically, β was tuned on the *LETOR* model and α was then tuned on a soft version of the *LETOR*. See Sec. 6.1 for details on the configuration of each tested model. Finally, to compute term similarity we used publicly available[5] pre-trained word embedding vectors.

6. EXPERIMENTS

We evaluated our proposed models against several baselines via two settings: first, based on a large scale clickthrough data, and second, based on manual judgments.

6.1 Tested Models

We consider six baseline models:

- *LSI*: Latent Semantic Indexing [12], where ranking score is computed as the dot product between the LSI representations of the query and the document title. We utilized the top 200 dimensions of the SVD decomposition of our corpus[6] as the latent space.
- *BIM*: Binary Independence Model [22] with unmatched probability estimation using pseudo relevance, taking the 'Relevant set' as 1, 3 or 5 top ranking documents by BM25.
- *BM25*: Using only the relevance score as provided by Lucene (feature H_1 in Tab. 1).
- *LETOR*: Using only statistical features associated with matched terms (features L_{1-10} and H_{1-3} in Tab. 1).
- *Matched*: Using all the features associated with matched terms (all features in Tab. 1).
- *SoftMatched*: Using soft-matching formulation for the matched features, *i.e.* feature families L_i^s, G_i^s and feature H_3^s (as in Sec. 5.3), and H_{1-2} from Tab. 1.

We compare the baselines to our proposed models:

- *Full*: Combining unmatched and matched features under exact matching formulation, *i.e.* all features in Tab. 1 as well as feature families EXL_i, EXG_i,MIL_i and MIG_i (described in Sec. 5.2).
- *SoftFull*: Combining unmatched and matched features under soft-matching formulation, *i.e.* feature families L_i^s, G_i^s, EXL_i^s, EXG_i^s, MIL_i^s, MIG_i^s and feature H_3^s (described in Sec. 5.3), and features H_{1-2} in Tab. 1.

We trained separately each of the LTR-based models using the algorithm in Sec. 5.4.

6.2 Automatic Evaluation

We conducted a large scale automatic evaluation using our 61,000 query test-set (see Sec. 5.4). For each query we retrieved the top 100 results from the document collection using Lucene, and then re-ranked the top results using each of the tested models. We report Mean Reciprocal Rank (MRR) and Binary Recall at position K (R@K), all derived from the rank position of the clicked document associated with each query. Fig. 4 depicts the query length distribution of our test-set. We remind the reader that CQA queries are usually longer than typical Web queries.

6.3 Manual Evaluation

We randomly sampled 1,000 queries of length 3 or more words from our test-set (shorter queries are scarce in our query collection - see Fig. 4). For each query we collected the top 10 documents as ranked by each of the tested models. Professional editors assessed the relevance of each document in the pool on three Likert-scale levels: (1) non-relevant, (2) partially-relevant, and (3) highly-relevant. Inspecting the evaluations, we found that usually only *highly-relevant* documents refer to relevant content. Hence, we report NDCG with weights of 0 for *non-relevant*, 1 for *partially-relevant* and 10 for *highly-relevant*. We also report Precision at K (P@K) taking only *highly-relevant* documents as relevant.

7. RESULTS

The results for the automatic and manual evaluations are summarized in Tab. 2 and Tab. 3, respectively. All statistical significance figures are computed using t-test. The results in both tables indicate a trend similar to the one reported by Cramel et el [9]. Namely, *LETOR* outperforms *BM25* by a large margin (*e.g.* 10.5% increase in MRR) and adding syntactic features (G_i) on top of statistical features (L_i, H_i) in the *Matched* model consistently provides additional improvement, *e.g.* 1.5% increase in MRR across all query lengths (Fig. 5). We thus refer to *Matched* as our main baseline. We note in passing that the performance of the *LSI* and *BIM* models was significantly lower than the LTR models (*e.g.* MRR of 0.202 for *LSI*, 0.164 for *BIM*) and adding them as additional features did not help either. We therefore excluded their performance report.

We next observe that adding soft term-matching to address the lexical gap between queries and documents (*SoftMatched* model) shows a nice improvement under the clickthrough automatic evaluation. For example, MRR is increased by 2.1% compared to *Matched*. In addition, manual evaluation also shows some improvement using soft matching, specifically at high rank positions. For example, P@3 is increased by 2.9% compared to *Matched*. On the other hand, the results for P@5 and P@10 are comparable to *Matched*. Analyzing our soft matching model, we found quite a few queries where exact matching provided better ranking than soft matching. For example, under the automatic evaluation setting, *SoftMatched* ranked the clicked

[5]https://code.google.com/p/word2vec/

[6]Using RedSVD: http://code.google.com/p/redsvd/

Model	MRR		R@1		R@3		R@5		R@10	
BM25	0.465	(-10.9%)	0.339	(-13.3%)	0.525	(-11.9%)	0.605	(-11.0%)	0.709	(-9.5%)
LETOR	0.514	(-1.5%)	0.386	(-1.3%)	0.581	(-2.5%)	0.663	(-2.5%)	0.763	(-2.6%)
Matched	0.522		0.391		0.596		0.680		0.783	
Full	0.537	(2.9%)	0.405	(3.6%)	0.609	(2.2%)	0.692	(1.8%)	0.792	(1.1%)
SoftMatched	0.533	(2.1%)	0.401	(2.6%)	0.604	(1.3%)	0.690	(1.5%)	0.790	(0.9%)
SoftFull	**0.543**	**(4.0%)**	**0.411**	**(5.1%)**	**0.616**	**(3.4%)**	**0.700**	**(2.9%)**	**0.800**	**(2.2%)**

Table 2: Automatic evaluation results (and percentage of change compared to *Matched* model). All differences are statistically significant at $p < 0.001$.

Model	NDCG		P@1		P@3		P@5		P@10	
BM25	0.685	(-3.9%)	0.522	(-7.9%)	0.396	(-5.5%)	0.336	(-7.9%)	0.268	(-5.6%)
LETOR	0.711	(-0.3%)	0.562	(-0.9%)	0.417	(-0.5%)	0.357	(-2.2%)	0.280	(-1.4%)
Matched	0.713		0.567		0.419		0.365		0.284	
Full	0.714	(0.1%)	0.567	(0.0%)	0.423	(1.0%)	0.366	(0.3%)	0.287*	(1.1%)
SoftMatched	0.716	(0.4%)	0.575	(1.4%)	**0.431**†	**(2.9%)**	0.366	(0.3%)	0.284	(0.0%)
SoftFull	**0.719**	**(0.8%)**	**0.577**	**(1.8%)**	**0.431**†	**(2.9%)**	**0.369**	**(1.1%)**	**0.289**†	**(1.8%)**

Table 3: Manual evaluation results (and percentage of change compared to *Matched* model). Values marked with $\star/\dagger$ indicate differences that are statistically significant at $p < 0.05$ and $p < 0.01$, respectively, compared to the *Matched* model.

query: stick a fork in it
Full: what does the phrase stick a fork in it mean? *Matched*: is it time to stick a fork in **Angels** and **Dodgers**?
query: doctorate vs phd
Full: what is the difference between phd and doctorate? *Matched*: phd in **nutrition** or **naturopathic** doctorate?
query: my family in french
Full: how do you say my family in french *Matched*: **describe** a family **member** in french?

Table 4: Examples where *Full* promoted better content at the top compared to *Matched*

document higher than *Matched* on 25% of the queries but that *Matched* ranked the clicked document higher than *SoftMatched* on 18.3% of the test-set. This may indicate that our current similarity function is noisy and could be improved in future work. While soft matching for retrieval was studied before, this is the first time it is applied in the CQA vertical search scenario. In addition, we are not aware of prior work that directly applies it to a large set of standard LTR features, specifically using similarity between word embedding vectors for lexical semantics (compared to the well studied translation models for this usage).

We now get to our main result, which is split into two parts, corresponding to the exact matching and soft matching settings. Under the exact matching setting, when adding features that directly address unmatched terms (*Full* model) we see a significant improvement in performance in the automatic evaluation compared to only using matched term features (*Matched* model). For example, MRR is increased by 2.9%, and similar trends occur for R@K. In Fig. 5 we plot MRR vs query length from which we observe that the MRR gap is maintained across all query lengths. The gap is slightly decreasing towards longer queries, perhaps because matching many of the terms for longer queries within a question title contains enough information to indicate relevant content. Under the manual evaluation, some improvement is shown. Specifically P@3 and P@10 show an increase of $\sim 1\%$ compared to *Matched* while P@1 and P@5 show comparable results. Tab. 4 shows examples where *Full* promoted better content at the top compared to *Matched*. These examples demonstrate how *Full* downgrades titles containing excessive information that changes the meaning of the title, such as '*Angles*' and '*Dodgers*' in the first example.

Both unmatched term features (*Full*) and soft matched term features (*SoftMatched*) provide a rather similar improvement over exact matching (*Matched*). However, they capture different aspects of query/document ranking, one is addressing the lexical gap between the two and the other is addressing the importance of terms that were not matched from either side. Combining both model approaches together (*SoftFull* model) shows that they convey somewhat complementing elements for ranking. Indeed, *SoftFull* is the best performing model under all metrics. Under automatic evaluation the improvement is rather additive with a nice gap in performance maintained across all query lengths from both *SoftMatched* and *Matched*. Under manual evaluation the improvement is more significant. It seems that under all metrics, but P@3, the combination of soft matching with direct unmatched term assessment is more powerful than each of its parts. This result may indicate that soft matching helps pinpointing the "true" unmatched terms and therefore improves the modeling of their contribution to ranking.

7.1 Excessive vs Missing Features

Our unmatched term features are composed of two types: a) those that address excessive terms, which occur only in the document ($EX\{L, G\}_i$); and b) those that address missing terms, which occur only in the query ($MI\{L, G\}_i$). We evaluated the contribution of each feature type independently by constructing two auxiliary models, both augmenting all matched term features (*Matched*). The first model adds only $EX\{L, G\}_i$ features, denoted *MatchedAndExces-*

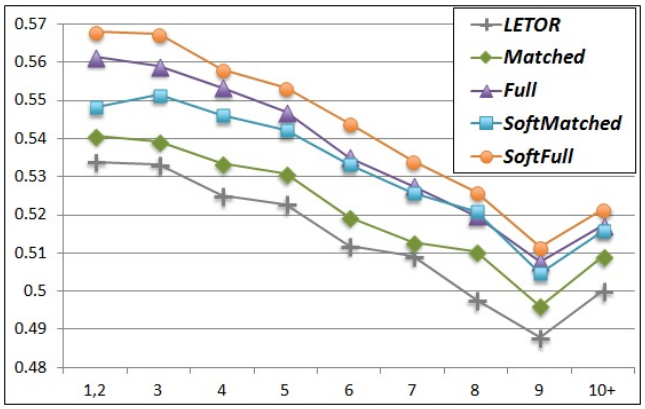

Figure 5: MRR by query length

Query	Document Title
what is candys american dream	the story "Of Mice and Men", how does the excerpt from Candy relate to the American Dream and Garden of Eden?
no virgin birth	No Virgin birth? Was the Virgin Mary really a virgin or was this a mistranslation of the Greek?
92 accord misfire	My '92 Accord misfires and loses power at about 2500 rpm. Possible causes?

Table 5: Examples where *Full* ranked a relevant clicked document low due to many excessive terms

sive, and the second model adds only the $MI\{L,G\}_i$ features, denoted *MatchedAndMissing*.

Fig. 6 presents the performance of the two new models compared to *Matched* and to *Full* on the automatic evaluation setting, measured via the MRR metric. The graph shows that while missing term features are not as strong indicators for irrelevance compared to excessive term features, they still directly improve MRR compared to *Matched*. In addition, while excessive term features contribute more to detecting irrelevant documents, a small improvement in MRR is gained when excessive and missing features are combined in the *Full* model.

Another observation from Fig. 6 is that *MatchedAndExcessive* improves over *Matched* for short queries much more than for long queries. One reason may be that for short queries there are more candidates in the corpus that contain the query terms, but many of them may have a lot of excessive information. As opposed to short web queries, in which the information need is generally wide, CQA queries tend to refer to very specific information needs even in such short queries, *e.g.* "*characteristics of enzymes*". If this hypothesis is true and our algorithm learned that a lot of excessive information indicates an irrelevant candidate, we expect the model to be particularly effective in filtering out such candidates. We note that for short queries of length 1-2 there is no improvement using missing features. This is not surprising, since there are no missing terms when matching queries of length 1, and the amount of missing information in queries of length two is at most a single term.

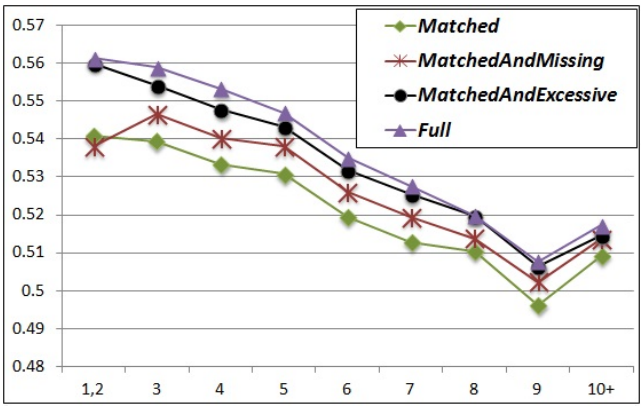

Figure 6: MRR by query length with the addition of Excessive vs. Missing Features

7.2 Error Analysis

To better understand the performance of our features, we conducted error analysis on cases in which *Full*, which considers both matched and unmatched term features, ranks a clicked document significantly lower than *Matched*, which employs only matched term features. To this end, we considered queries from our validation set in which *Matched* ranked the clicked document for the query in one of the top 3 positions while *Full* ranked it far below. We sorted the examples by the rank margin between *Matched* and *Full* and analyzed the 100 queries with the largest rank margin.

We found out that in 35 of the analyzed queries the top document ranked by *Full* was relevant and in 46 queries at least one of the top 3 documents was relevant. This is a known issue when using clickthrough logs as proxy to document relevance, as some unclicked documents may also be relevant to the query. Thus absolute model performance under such evaluation is biased. Yet, comparing ranking algorithms over a large-scale click-based gold labeling is useful for differentiating between their ranking performance [27].

Out of the 65 queries where the top candidate by *Full* was not relevant, we recognized two main reasons for this ranking failure. The most prominent reason, which occurred in 36 cases, is that some terms in the document title were not matched due to the exact-matching scheme used in *Full*, but would have considered matched under proper soft matching. Out of these 36 case, 13 queries had spelling errors and other phenomena, such as unigram/bigram variations (*e.g.* '*countertop*' vs. '*counter top*'). Such lexical variations are not recognized by our current soft term matching which uses word embedding.

The second phenomenon occurred in 16 out of the 65 queries. The clicked document title contains a lot of excessive terms, yet still fulfills the information need behind the query. Tab. 5 presents such examples. While our results show the potential in directly modeling unmatched terms, and specifically excessive terms as negative signals, a large number of such terms may accumulate into an unnecessary downgrading of the ranking score, and further research is required to develop more robust models.

8. CONCLUSIONS

In this work we proposed novel features in a Learning To Rank framework that directly assess the importance of missing and excessive terms within the task of term weighting for vertical search on CQA content. To better model truly unmatched terms we also presented a "soft matching" variant of all our features, basing it on distributional similarity between terms, where similar terms are considered partially both matched, and unmatched. Our experiments show improvement in document retrieval in all settings when unmatched information is taken into account.

In future research we plan to test whether our approach may contribute to other types of Web documents. One direction could be to explore how unmatched terms can be modeled in other parts of the document, which may require different features than the ones used in this paper.

9. ACKNOWLEDGEMENTS

The resrach was supported in part by the Yahoo Faculty Research and Engagement Program.

10. REFERENCES

[1] G. Amati, V. Rijsbergen, and C. Joost. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4), Oct. 2002.

[2] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo. Sources of evidence for vertical selection. In *SIGIR*, 2009.

[3] M. Bendersky, D. Metzler, and W. B. Croft. Learning concept importance using a weighted dependence model. In *WSDM*, 2010.

[4] A. Berger and J. Lafferty. Information retrieval as statistical translation. In *SIGIR*, 1999.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[6] L. Cai, G. Zhou, K. Liu, and J. Zhao. Learning the latent topics for question retrieval in community qa. In *AFNLP*, 2011.

[7] X. Cao, G. Cong, B. Cui, C. S. Jensen, and C. Zhang. The use of categorization information in language models for question retrieval. In *CIKM*, 2009.

[8] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *SIGIR*, 2006.

[9] D. Carmel, A. Mejer, Y. Pinter, and I. Szpektor. Improving term weighting for community question answering search using syntactic analysis. In *CIKM*, 2014.

[10] R.-C. Chen, D. Spina, W. B. Croft, M. Sanderson, and F. Scholer. Harnessing semantics for answer sentence retrieval. In *ESAIR Workshop*, 2015.

[11] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. *MLJ*, 91(2):155–187, 2013.

[12] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407, 1990.

[13] H. Duan, Y. Cao, C.-Y. Lin, and Y. Yu. Searching questions by identifying question topic and question focus. In *ACL*, 2008.

[14] D. Ganguly, D. Roy, M. Mitra, and G. J. Jones. Word embedding based generalized language model for information retrieval. In *SIGIR*, 2015.

[15] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, 2013.

[16] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *CIKM*, 2005.

[17] R. Jin, A. G. Hauptmann, and C. X. Zhai. Language model for information retrieval. In *SIGIR*, 2002.

[18] Q. Liu, E. Agichtein, G. Dror, E. Gabrilovich, Y. Maarek, D. Pelleg, and I. Szpektor. Predicting web searcher satisfaction with existing community-based answers. In *SIGIR*, 2011.

[19] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, Mar. 2009.

[20] T. y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR Workshop on Learning to Rank for Information Retrieval*, 2007.

[21] Y. Liu, C. Sun, L. Lin, Y. Zhao, and X. Wang. Computing semantic text similarity using rich features. In *PACLIC*, 2015.

[22] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*. 2013.

[24] D. R. Miller, T. Leek, and R. M. Schwartz. A hidden markov model information retrieval system. In *SIGIR*, 1999.

[25] V. Murdock and M. Lalmas. Workshop on aggregated search. *SIGIR Forum*, 42(2):80–83, Nov. 2008.

[26] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR*, 1998.

[27] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM*, 2008.

[28] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, Apr. 2009.

[29] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

[30] S. E. Robertson, C. J. van Rijsbergen, and M. F. Porter. Probabilistic models of indexing and searching. In *SIGIR*, 1980.

[31] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, Aug. 1988.

[32] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, 2015.

[33] F. Song and W. B. Croft. A general language model for information retrieval. In *CIKM*, 1999.

[34] K. Tymoshenko and A. Moschitti. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *CIKM*, 2015.

[35] X. Wei and W. B. Croft. Lda-based document models for ad-hoc retrieval. In *SIGIR*, 2006.

[36] R. W. White, M. Richardson, and W.-t. Yih. Questions vs. queries in informational search tasks. In *WWW Companion*, 2015.

[37] H. Wu, W. Wu, M. Zhou, E. Chen, L. Duan, and H.-Y. Shum. Improving search relevance for short queries in community question answering. In *WSDM*, 2014.

[38] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Inf. Retr.*, 13(3):254–270, June 2010.

[39] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *SIGIR*, 2008.

[40] C. Zhai. Statistical language models for information retrieval. *Synthesis Lectures on HLT*, 1(1):1–141, 2008.

[41] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, 2001.

[42] W. Zhang, Z. Ming, Y. Zhang, L. Nie, T. Liu, and T. Chua. The use of dependency relation graph to enhance the term weighting in question retrieval. In *COLING*, 2012.

[43] G. Zheng and J. Callan. Learning to reweight terms with distributed representations. In *SIGIR*, 2015.