Query Suggestion and Data Fusion in Contextual Disambiguation

Milad Shokouhi
Microsoft
milads@microsoft.com

Marc Sloan*
University College London
marc.sloan.10@ucl.ac.uk

Paul N. Bennett
Microsoft
paul.n.bennett@microsoft.com

Kevyn Collins-Thompson*
University of Michigan
kevynct@umich.edu

Siranush Sarkizova*
Harvard University
ssarkizova@g.harvard.edu

ABSTRACT

Queries issued to a search engine are often under-specified or ambiguous. The user's search context or background may provide information that disambiguates their information need in order to automatically predict and issue a more effective query. The disambiguation can take place at different stages of the retrieval process. For instance, *contextual query suggestions* may be computed and recommended to users on the result page when appropriate, an approach that does not require modifying the original query's results. Alternatively, the search engine can attempt to provide efficient access to new relevant documents by injecting these documents directly into search results based on the user's context.

In this paper, we explore these complementary approaches and how they might be combined. We first develop a general framework for mining context-sensitive query reformulations for query suggestion. We evaluate our context-sensitive suggestions against a state-of-the-art baseline using a click-based metric. The resulting query suggestions generated by our approach outperform the baseline by 13% overall and by 16% on an ambiguous query subset.

While the query suggestions generated by our approach have higher quality than the existing baselines, we demonstrate that using them naïvely for injecting new documents into search results can lead to inferior rankings. To remedy this issue, we develop a classifier that decides when to inject new search results using features based on suggestion quality and user context. We show that our context-sensitive result fusion approach (Corfu) improves retrieval quality for ambiguous queries by up to 2.92%. Our approaches can efficiently scale to massive search logs, enabling a data-driven strategy that benefits from observing how users issue and reformulate queries in different contexts.

1. INTRODUCTION

An ambiguous query may be issued to a search engine when a user has an inexplicit information need or when the user enters only a few search terms in lieu of a longer, more complex query. A comprehensive study of two search logs found that between 7% and 23% of queries could be considered ambiguous, depending on how ambiguity is defined, and that such queries are typically short, averaging 1.02 to 1.23 terms [31]. In such cases, user context may help disambiguate the query and personalize search results [23]. For instance, a user searching for `[ eagles ]` with a recent history of sports-related queries may prefer the 'Philadelphia Eagles' American football team website over that of the rock band 'The Eagles'.

In this paper, we build on existing query disambiguation work by introducing user context. User context can encompass multiple aspects such as age, gender, topic, or location and can be short-term [9] or long-term [23]. In this work, we use categories from the Open Directory Project (ODP)[1] to represent the short-term topical context of a user's information need. In particular, by combining click information with categorized URLs retrieved for previous queries in the session, we can interpret the most probable category as the user's short-term *session context* [4]. We then use this inferred context together with observed query reformulations across query log sessions to mine contextual reformulations for query suggestion. Some examples of session context, original query, and the top ranked contextual suggestion are shown in Table 1.

Table 1: Example contextual query suggestions

Session Context	User Query	Contextual Suggestion
Sports	`eagles`	`philadelphia eagles`
Music	`eagles`	`eagles band`
Physics	`pascal`	`pascal unit`
Computer Science	`pascal`	`pascal programming`
Video Games	`forge`	`minecraft forge`
Crafts	`forge`	`metal forge`

We generate contextual query suggestion candidates using a similar utility-function based methodology to that employed by Ozertem et al. [27]. In this framework, candidates are found by extracting consecutive pairs of queries from sessions in search logs. These are then ranked according to the difference in utility value, where utility is a measure of the query's usefulness calculated using clicked documents. We also find that adding contextual features leads to significant gains in query suggestion quality particularly for ambiguous queries.

Moreover, we explore disambiguation approaches that go beyond simply offering contextual suggestions: we aim to improve the user's efficiency in a session by eliminating the need for a separate query reformulation altogether. Instead, we introduce methods for predicting *when* and *how* it is appropriate to incorporate

*Work performed while at Microsoft

[1] `http://www.dmoz.org/`

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741646.

context-driven results directly into the initial ranking. This goal is similar in motivation to previous work on whole-session relevance that merged in results from a user's predicted future queries for a task [28]. Toward this goal, we first learn to predict which queries are likely to benefit from this blending; then, given such a query, we use a data fusion technique [3, 32] to incorporate relevant results from the top ranking contextual suggestion with those of the original query. Finally, we show that this fusion approach significantly improves search results for ambiguous queries.

In particular, the problem of differentiating when a candidate contextual query reformulation is best left as a suggestion, versus when it is a good candidate for data fusion, is a key challenge unaddressed by previous work. Good suggestion candidates may not necessarily lead to a good result fusion. For example, the query suggestion `[ kmart ]` for the original query `[ target ]` is relevant and may help the user explore, but blending the results of the former into the Search Engine Results Page (SERP) of the latter would be confusing, as the *results* of the latter are not directly relevant to the original query. To mitigate this, we trained a novel classifier using session, context and query-based features to identify instances in which fusing results is beneficial over simply suggesting the query. Our experiments demonstrate that incorporating this fusion classifier by selectively choosing when to blend yields significant improvements in effectiveness.

In sum, the contributions of this work are as follows:

- The addition of contextual categorization (Section 2) to utility-based query suggestion generation, resulting in improved contextual query suggestions (Section 3).
- Investigating and demonstrating the need for selective blending by analyzing the failure cases of blindly fusing candidate suggestions with the original query in all cases (Section 4).
- The definition and evaluation of a fusion classifier that predicts whether to blend search results or suggest queries for contextual disambiguation (Section 5).

2. CONTEXTUAL QUERY DISAMBIGUATION

The aim of contextual query disambiguation is to use context to improve the search experience of a user who has issued an ambiguous query. Query suggestions provide the user with easily accessible reformulation options if they decide to issue a new query. As such, the quality and ranking of suggestions is important. Suggestions are used by users not only to disambiguate their original queries, but also in cases where the user wishes to broaden or narrow the scope of their initial query or otherwise adjust their search intent. Given this variability in usage, along with a relatively low engagement rate ($< 5\%$ on the Bing search engine), an attractive alternative to offering query suggestions as a separate interaction step is to directly blend their results with those for the initial query. This blending approach has at least two advantages. First, because blending can incorporate search results from multiple suggestions, the pool of potentially relevant documents is increased. Second, directly satisfying the user's query with a blended result decreases the total time required for a user to satisfy their information need. This not only saves time, but also potential frustration and abandonment.

Given a user query q, we seek a suitable contextual reformulation candidate q_c for suggestion or blending, where the context $o \in \Theta$ belongs to a finite, discrete set of contextual labels (such as topic, gender or location). Candidates are found by mining sessions in query logs for consecutive pairs of queries q_a, q_b. Sessions are defined as a series of queries issued by a single user over a period

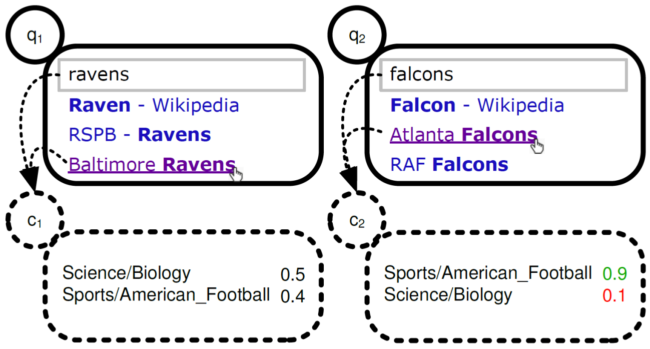

Figure 1: Session context (o^*) example. The queries and clicks observed throughout a session are used to update the context model (Θ). Suppose that the user had already clicked on a document related to Science/Biology in the previous query (q_0). Based on the clicks recorded on q_1, the context model is updated to c_1 which will become available at the time q_2 is submitted. Similarly, the context model is updated for the next query based on the clicks recorded on q_2.

of time, where we adopt the widely-used practice of demarcating session boundaries by 30 minutes of user inactivity [41].

We use the ODP taxonomy as our classification labels O as they are high quality, discrete and general enough for use on search log data. Nonetheless, it is worth noting that our technique is applicable to any classification space where meaningful labels can be attached to queries. In this study, we limit ourselves to the top two levels of the ODP hierarchy, giving us 17 topics (e.g. *Arts*, *Sport*) and 185 subtopics (e.g. *Arts/Music*, *Sports/Baseball*). This set of categories gives us enough granularity to distinguish the different contexts in which reformulations typically occur, while being broad enough to represent the topical meaning of most content, and general enough so that repeated similar behavior is pooled into statistically robust sets of observations.

To create the *context model* in a session, we directly implement the query classification technique used by Bennett et al. [4]. Their method uses a conventional topic classifier that assigns labels to individual URLs based on a given taxonomy. The classifier is applied to the clicked URLs from search results in a search log, where click information, dwell time and a number of query-URL features are combined to calculate a probability distribution over the 202 ODP categories described above. The session *context model* is then formed at each point by aggregating the ODP probability distributions of documents clicked for previous queries in the session.

We define *session context* (o^*) as the ODP category with highest probability (confidence) in this context model (Θ). Session context takes into account all URLs and interactions of the preceding queries meaning that it becomes more accurate and representative of the user's search intent as the session progresses.

Figure 1 gives an example of how session context is determined: suppose that the user has already clicked on a document related to *Science/Biology* in the previous query of the session. Once the user clicks on the third document for q_1 the context model is updated to also capture *Sports/American_Football* and in this case gives similar weights to the two subtopics (the weights assigned to the other 200 categories are negligible). The observed click on next query (q_2) increases the confidence score for *Sports/American_Football* subtopic in Θ for the future queries.

A drawback to using session context is that we cannot classify (and thus disambiguate) a user's query if it is the first in a session: in this case we simply return its usual results without modification. One potential way to alleviate this would be to classify queries in the user's longer-term search history in order to find personalization labels. A balance of short and long-term labels could be used to provide context from the start of a session, a topic explored more completely by Bennett et al. [5]. We leave this long-term scenario for future work.

3. QUERY SUGGESTION UTILITY

In this section we focus on query suggestion as a method for contextual query disambiguation. Inspired by previous work [27] we introduce a new utility function for measuring the effectiveness of query suggestion candidates for a given query. We then demonstrate how this function can be used to mine contextual (and non-contextual) query suggestion candidates from past search logs.

It is often the case that search logs are used to find and score query suggestions. The query-flow graph is one such example; its nodes are queries, its edges are query reformulations and their weights are the frequency of occurrence [7]. Variants include the click-bipartite graph which links queries with clicked URLs, and in both cases suggestion candidates can be found using random walks, clustering and other graph-based techniques [1, 11].

A simple approach for extracting query suggestions from logs is to mine query co-occurrences in search sessions [20]. For a given query q_a, suggestion candidates can then be ranked according to their maximum likelihood computed as

$$P(q_b|q_a) = \frac{P(q_b, q_a)}{P(q_a)} \tag{1}$$

where q_b is a query reformulation and $P(q_b, q_a)$ represents its co-occurrence probability with q_a based on past searches. We can extend this to include context by also conditioning on the contextual category, giving us

$$P(q_b|q_a, o) = \frac{P(q_b, q_a, o^*)}{P(q_a, o^*)}. \tag{2}$$

Here the session context (discussed earlier in Section 2) is denoted by o^*. At run-time the query and its session context are matched against this probability distribution for ranking candidates.

3.1 Reformulation Utility

Ranking query suggestions by maximum likelihood encounters two main issues. First, a suggestion candidate could be unrelated and biased towards popular navigational queries (topic drift). Second, the results for a suggestion may be low quality. Ozertem et al. [27] addressed the problem of reformulation quality by defining a relative utility function based on the click differences found in both the original query (q_a) and its reformulation (q_b). Suppose q_a and q_b have co-occurred consecutively[2] in a session. Using clicks from their impressions, they compute the utility of q_b as a query suggestion candidate for q_a using

$$\Delta = \sum_{d \in C(q_b)} \left(\frac{1}{\log r(q_b, d)} - \frac{1}{\log r(q_a, d)} \right). \tag{3}$$

[2]The authors showed that relaxing this constraint does not improve the results. Ozertem et al. also described an extended version of their technique enhanced with task boundary detection and supervised utility inference. Both of these extensions are orthogonal to our contributions and neither was reported to substantially improve the quality of the top suggestion which is what we are mostly concerned with in this work.

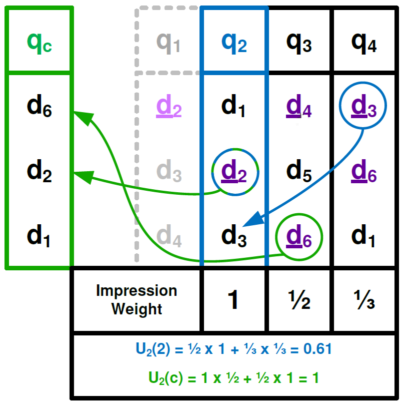

Figure 2: Example illustrating the utility function given by Equation 4. In this case, $q_a = q_2$ and its interactions are given in blue, whilst the candidate's q_c are given in green. Underlined documents represent SAT clicks.

Here $C(q_b)$ represents the clicked documents on the results returned for q_b in that session and the $r(q, d)$ function returns the position of document d in the results returned for query q. A candidate is found useful whenever it returns a clicked document that either did not exist in the ranked results for the original query, or did exist but at a relatively lower position. The authors then extended Equation 1 and ranked suggestions based on the probability $P(q_b|q_a, \Delta > 0)$.

In this work, we use their method as our baseline because (1) it is state-of-the-art and has been reported to significantly outperform other alternatives, (2) it applies a retrieval-based utility function which fits well with fusion scenarios that we discuss later in this work, and (3) we use a similar retrieval-based utility function that relies on clicks for computing suggestion utilities. However, we define a different relative utility function that unlike their approach does not ignore the clicks recorded on the original query. To alleviate data sparsity issues that arise when taking into account context we also use clicks that occur later in the session. We follow the common practice of considering clicks with longer dwell time than 30 seconds as satisfied clicks [4, 28]. Fox et al. [16] showed such clicked documents are highly likely to be relevant. In this paper, clicks *always* mean SAT clicks.

Our utility function measures the quality of queries and suggestion candidates based on the click-statistics of their results in the session and applies two reciprocal rank discounts, we thus refer to it as cumulative reciprocal rank (*CRR*). Suppose that the user has issued a query q_a, the I_ath query in the search session. Let us denote D_c as the documents in the ranking for query suggestion candidate q_c. We define the *CRR* of q_c with respect to q_a as

$$\mathit{CRR}_a(q_c) = \sum_{d \in D_c} \underbrace{\frac{1}{r(q_c, d)}}_{\text{Result Reciprocal Rank}} \times \underbrace{\frac{1}{(i(d, I_a) - I_a + 1)}}_{\text{Impression Reciprocal Rank}} \tag{4}$$

where $r(q, d)$ is as described above and I_a is the impression position of q_a in the session. The function $i(d, I_a)$ is the position in the session of the first query including or after q_a where d was clicked. If no such click exists in the session then i is set to $+\infty$.

In summary, the utility of a candidate query suggestion is a combined function of the relevance of a document (measured by whether it receives a click in the session at query q_a or after) and the position of the document in the ranking q_c (weighted by reciprocal rank).

We follow Sakai and Dou [30] and degrade the pseudo-relevance labels assigned to these documents by applying an impression reciprocal rank discount for clicks that occur later in the session, emphasizing the current information need. The relative utility between q_a and q_c is then simply

$$\Delta \mathit{CRR}(q_c, q_a) = \mathit{CRR}_a(q_c) - \mathit{CRR}_a(q_a). \tag{5}$$

This function is illustrated in Figure 2, where q_2 is the user's current query (so $I_a = 2$ in this case) in the session. The session context model at this point is formed based on the documents clicked for the previous queries in the session (q_1). The next two queries in the session (q_3, and q_4) are obviously unobserved at the time q_2 is submitted, but we use information about their clicked documents to generate pseudo-relevance labels in order to measure the utility of suggestion candidates. In this particular scenario, the underscore indicates that d_2 is the only document we have observed a *satisfied* click on for q_2. There are three more documents clicked later in this session (d_3, d_4 and d_6). While these documents can be regarded as pseudo-relevant under their respective queries (q_3 and q_4) their association with q_2 is weaker due to the potential topic drift.

In the example above, applying the impression discount allows us to consider d_3 as a relevant document for q_2 – albeit with discounted weight – although it has not been clicked under q_2 directly. We perform the same procedure for calculating *CRR* of the current query (q_2) and all suggestion candidates such as q_c. We compare suggestion candidates in terms of how they perform against the original query with respect to their $\Delta \mathit{CRR}$ (in this case $\Delta = +0.39$)[3]. Note that in this example, even though the user query ranks clicked document d_2 in the same position as q_c, q_c ranks d_6 (which is clicked in the next impression) very highly, which contributes to its improved score. By leveraging and weighting all of the clicks that occur in the remainder of the session we gain a broader view of the documents that are relevant to the query and can reward those candidates that find relevant documents that weren't returned for the original query (which may often lack clicks due to the query being ambiguous).

3.2 Ranking suggestions

For each given query (q_a, q_c) pair we compute the score as

$$P(q_c|q_a, \Delta \mathit{CRR}(q_c, q_a) > 0)$$

The probabilities are computed over historical logs and at run-time are used to rank suggestions accordingly. The contextual extension of our model considers the session category ($o*$) of each historical impression when computing these probabilities:

$$P(q_c|q_a, o^*, \Delta \mathit{CRR}(q_c, q_a) > 0)$$

At run-time, we match both query and its context against this probability distribution to rank candidates. We denote our approach *CosQus* (short for **Co**ntext **S**ensitive **Qu**ery **S**uggestion). We refer to the context-free version of our method that only computes probabilities over query-candidate *pairs* as *CosQus*(P), and to the contextual version that computes probabilities over query-category-candidate *triples* as *CosQus*(T). Note that we are only interested in the top-ranked suggestion in our experiments. We also propose a *hybrid* approach *CosQus*(H) that computes both of these probabilities and at run-time selects the candidate with the highest probability across the two.

[3]The top-ranked results for each candidate can be obtained by issuing that query, or from historical logs. In our work, we use the most recent SERP logged for a candidate in our logs.

3.3 Data

Our experiments are conducted over the query logs of the Bing search engine. We use log data for mining query suggestions, computing their utility values, evaluating the performance of different query suggestion approaches and later in the paper for training supervised result fusion classifiers. Therefore, it is important to carefully sample the logs at each stage to avoid the risk of leakage and overfitting. Figure 3 depicts the time periods of the search logs used in our work. For the query suggestion experiments in this part of the paper, we are only using the first three segments, and the purple and green boxes can be ignored.

Candidate generation. To generate the potential query suggestions for a given query, we mine all consecutive query reformulations submitted by Bing users between December 1st, 2012 and December 31st, 2013. Note that all our experiments are performed on the query logs of the *English-US* market. We reduced the space of query-candidates to a subset of up to 50 common reformulations by only including the top contextual and non-contextual suggestions based on Equations 1 and 2. For this, we selected the top 25 most probable reformulations without any context for the query (Equation 1). We also included the top 25 contextual reformulations for each query (Equation 2). It is worth noting that the same reformulation could appear for multiple contexts. Doing this allowed us to generate candidates at different topic granularities, capture instances where a non-contextual suggestion was ideal, and allowed us to try out different candidate selection strategies in our experiments. This pool of up to 50 unique candidates for a query effectively provides us with a framework for comparing different query suggestion rankings on the same dataset.

Utility computation. The blue box in Figure 3 specifies the period over which candidate utilities are computed for *CosQus* and our experimental baselines. We sampled 11.4 million search sessions from Bing logs between Feb. 1st, 2014 and Feb. 28th, 2014. In total there are roughly 23.5 million impressions in this set. We use the logs in this period to compute the utility values (e.g. $P(q_c|q_a, \Delta \mathit{CRR}(q_c, q_a) > 0)$ for *CosQus*(P)) over all reformulations that were extracted in the previous step.

Query suggestion evaluation. To compare the effectiveness of query suggestion techniques we sample another set of 5.7 million sessions (11.8 million impressions) from the query logs between March 1st, 2014 – March 14th, 2014. For each impression during this period we use the SAT-clicks in the session as previously described (Section 2) to assign pseudo-relevance labels to documents. These labels are later used to compute the *CRR* values for the query and all its candidates.

Ambiguous queries. We also targeted a subset of Wikipedia-derived ambiguous queries as ideal candidates for contextual query disambiguation. We first consider the title of all Wikipedia disambiguation pages[4] as ambiguous queries. We then used the Wikilinks dataset[5] containing millions of links and anchor text to Wikipedia articles extracted from webpages. We found instances where either different anchor text was used to link to the same Wikipedia article, or where the same anchor text was used to link to different Wikipedia articles. Combining all three approaches gave us a set of 1,319,309 ambiguous queries, of which around 1/3 matched queries in our candidate generation period. On our evaluation dataset,

[4]en.wikipedia.org/wiki/Wikipedia:Disambiguation
[5]code.google.com/p/wiki-link/

Table 2: **The effectiveness of different query suggestion approaches according to cumulative reciprocal rank (*CRR*) on two experimental query sets ("Ambiguous" and "All"). The *suggestion utility* approach of [27] is used as a baseline. For reference we also provide the average *CRR* values for the original query $CRR_a(q_a)$ on the two datasets. All pairwise differences are statistically significant according to the t-test ($p < 0.01$).**

Method	Queries	$CRR_a(q_a)$	$CRR_a(q_c)$	(% ▲/▼)
Baseline [27]			0.336	
Baseline+			0.318	(▼ 5.4%)
CosQus(P)	All	0.530	0.313	(▼ 6.8%)
CosQus(T)			0.358	(▲ 6.5%)
CosQus(H)			0.378	(▲13.0%)
Baseline [27]			0.259	
Baseline+			0.245	(▼ 5.4%)
CosQus(P)	Ambiguous	0.458	0.244	(▼ 5.8%)
CosQus(T)			0.278	(▲ 7.3%)
CosQus(H)			0.300	(▲16.0%)

we report the results on the subset of impressions that match these queries separately, as we find these queries to be most suitable for disambiguation by definition.

3.4 Experimental results

As stated earlier, the first contribution in our work is a new approach for ranking query suggestions. We compare our approach with the non-contextual utility function from Ozertem et al. [27] as our baseline. We also tested against a variant baseline (Baseline+) which incorporated the propagation of session clicks in a similar way to our *CRR* function. This was to make sure our comparison was fair and that both techniques had access to the same information for candidate ranking.

Our results in Table 2 show that the use of context (*CosQus*(T)) is clearly beneficial in helping to choose query suggestions, outperforming the three non-contextual methods. We can also observe significant gains by using the hybrid approach (*CosQus*(H)) and switching between contextual and non-contextual candidates based on their probabilities. Some additional conclusions we can draw from our results: (1) Using clicks from later in the session does not improve performance on non-contextual suggestions (Baseline+ and *CosQus*(P)), possibly because there is greater topic drift when not conditioning on context; (2) The biggest gains were seen with ambiguous queries, reinforcing our aim of contextual disambiguation, but we still made significant gains on all queries; and (3) Overall the *CRR* values were lower for ambiguous queries, most likely due to there being less high ranking clicks to learn from. Of note is the fact that results from the query suggestions were generally poorer than the original query. This will factor into our data fusion experiments and is discussed further in the next section.

4. DATA FUSION

While improving contextual query suggestions significantly is helpful, the level of user engagement with such suggestions is typically low. Therefore, we seek a way to translate these gains in suggestion quality to direct gains in result relevance and/or a reduction in user effort searching. In particular, users have been found to be more likely to click on suggestions when their query is rare or a single term, after they have clicked on several URLs, or when the suggestions themselves are unambiguous or a correction of the original query [21]. Instead of suggesting queries, we can directly perform contextual disambiguation by altering the search results of the user's query using data fusion.

In data fusion, we blend the search results from multiple rankings into one list. In our setting, these rankings result from multiple queries. This can range from injecting a single result into the original ranking to completely replacing the original ranking. The benefits are that we can find relevant documents from a number of sources and combine them into one list, but it is also possible to remove relevant documents in the process, so results blending must be done carefully. Search rankings can come from a number of sources. For instance, in federated search the same query is issued on multiple search engines and the results merged using a utility function [35]. LambdaMerge [33] uses a two-layer neural network to learn a ranking function for merging results lists for query reformulations based on query, document and reformulation features. Their focus is to optimize over a retrieval quality metric to produce effective merged lists whereas ours is on predicting when contextual blending is going to be effective; thus we adopt a simpler merging technique.

4.1 Contextual CombSUM

Our data fusion for contextual disambiguation method involves blending the results returned for a query with those returned for a related query suggestion. Running many queries in parallel is prohibitively costly in practice. Hence, we restrict ourselves to running only one additional suggestion candidate per query, and use the top candidate from *CosQus*(H) given its superior performance based on the results reported in the previous section.

CombSUM is a simple merging technique that sums normalized URL rank scores across queries, producing a new ranking in decreasing order of CombSUM scores [32]. The result is that highly ranked URLs which appear in multiple lists are more likely to appear in the merged list. For a given q_a, a query suggestion q_c, and a document d selected from one of their respective results, the CombSUM scores are computed as

$$\text{CombSUM}(d) = \frac{1+\epsilon}{r(q_a,d)} + \frac{1}{r(q_c,d)}. \qquad (6)$$

The reciprocal rank takes the place of the rank score and ϵ is introduced to give a priority to the URLs that appear in the original ranking. We set a low weight for ϵ (0.01) primarily to serve as a tie-breaking mechanism in favor of the original ranking. After ranking the merged list by CombSUM, we keep the same number of results as the original ranking.

4.2 Naïve Fusion

We applied CombSUM data fusion to the rankings of every user query and *CosQus*(H) contextual query suggestion in our dataset (over the *memorization period*). Overall, we found the performance of these blended lists to be inferior to that of the original query with no blending. On average the *CRR* value dropped by about 7% on both datasets ("Ambiguous", and "All"). This is expected as we reported earlier, the *CRR* values for top query suggestions are substantially worse than the original queries (see Table 2); consequently the blended list will suffer.

To investigate whether the fusion between the results of a query and its suggestion candidate can improve search results we measured $\Delta CRR(q_c, q_a)$, and $\Delta CRR(f, q_a)$, where f signifies the CombSUM fusion ranking between query q_a and candidate q_c. We plotted ΔCRR for every suggestion and blended ranking in our scatterplot in Figure 4. Here, we observe a clear trend that shows that suggestions with good rankings (x-axis) lead to better blended rankings (y-axis), and bad suggestions hurt blending performance. There

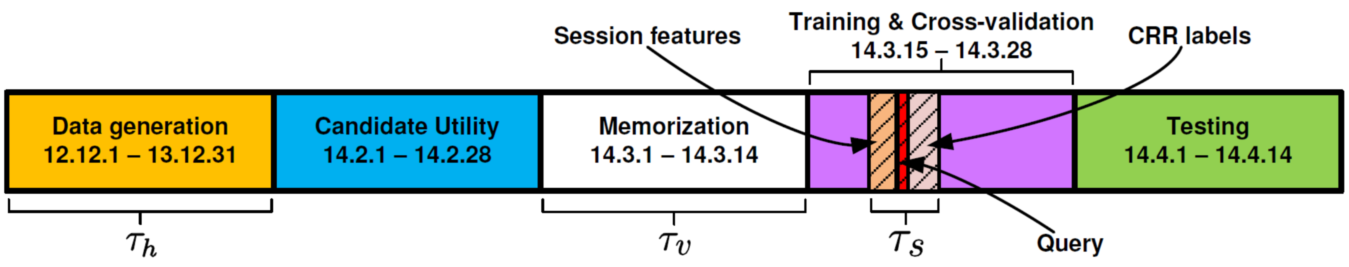

Figure 3: The timeline for the logs used in our feature generation and evaluation pipelines. Historical frequency features are computed between December 2012 and December 2013 (τ_h). Query suggestion utilities are computed based on the logs sampled in February 2014. Query suggestion techniques are evaluated against the logs sampled between March 1st, and March 14th, 2014. The same period is used for *memorizing* the effectiveness of blended result lists that are then used in our fusion baselines and as classification features (τ_v). The logs between March 15th, March 28th, 2014 are used for training (cross-validation) Corfu. The fusion models are compared based on their performance on our test data (first two weeks of April 2014). The red line shows an example query in our training period sampled from a session (τ_s) (represented by the shaded area). The data from the session before the query are used for computing the contextual session features, and the clicks observed for the query and later in the session are used for generating the pseudo-relevance labels in *CRR*.

are also more candidates with poor rankings than those with good (observable from histograms along the upper and right sides), which supports our finding in Table 2 that suggestion rankings on average did not perform as well as the original ranking.

The conclusions that can be drawn from this experiment are that the rankings for query suggestions can vary in quality with regard to the original query. In many cases (such as the case with our `[ kmart ]` example), a query suggestion may suffice as a form of contextual disambiguation and data fusion should not be applied naïvely across all such queries. The results are consistent with previous work [33] in suggesting that a more sophisticated data fusion technique may not necessarily lead to improvements due to low candidate quality.

5. FUSION CLASSIFIER

Based on the above findings, we developed a classifier to identify when a query suggestion's results should be fused with those of the original query. As we did in the previous section, we find candidate suggestions using *CosQus*(H). If our classifier indicates that it is better to fuse, we use CombSUM to merge the results of the suggestion with the original query: otherwise, we keep the original ranking We refer to our approach as **Co**ntextual **R**esult **Fu**sion (Corfu).

To generate training data for our classifier, we sampled 6.2 million sessions (13.2 million impressions) from Bing search logs between March 15th, 2014 and March 28th, 2014. This period is distinguished in a purple box in Figure 3. For each of these impressions we computed the ΔCRR value between the query and the top-ranked candidate suggested by *CosQus*(H). For training the classifier, instances with $\Delta > 0$ values are considered as positive examples, and those with $\Delta < 0$ are considered as negative examples. We trained both logistic regression and gradient boosted decision-tree models for classification (GBDT) [17]. Since GBDT consistently outperformed logistic regression, we focus here on the GBDT results only. We did a parameter sweep over the number of trees $\{20, 100, 500\}$, number of leaves $\{2, 4, 8, \cdots, 128\}$, minimum number of instances per leaf $\{1, 10, 50\}$, and learning rate $\{0.02, 0.05, 0.1, 0.4\}$ and picked the best model by 5-fold cross validation. We evaluate the performance of Corfu classifiers on test data comprising another mutually exclusive set of logs. Our test data is comprised of 13.7 million impressions sampled from 6.4 million Bing search log sessions between April 1st, 2014 and April 14th, 2014 (green box in Figure 3). We discard all impressions with zero gain or loss from our test data, leaving 2.2 million impressions, on which we report the final numbers.

5.1 Features

We trained our classifier using a range of query, context and session features: the full list is given in Table 3. Many of the features are self-explanatory and we give further insight into our choices below. Different subsets of the features were collected from different time periods in the Bing search logs owing to the different phases of candidate generation, utility scoring, training and validation. A breakdown of the timeline of the log, including when features were extracted, is given in Figure 3 and also discussed below.

Memorization. The *memorization* features are computed over the candidate utility computation period τ_v (white box in Figure 3). We averaged and recorded the blended $\Delta CRR_\mu(f, q_a)$ scores for each query pair (q_a, q_c) and contextual triple (q_a, q_c, o) (the values shown in Figure 4). From this we were able to create a white-list of queries where we found positive ΔCRR_μ. We use these white-lists as *static* fusion baselines against our Corfu classifier and respectively refer to them as Memorized(P) and Memorized(T). The experimental results summarized later in this section confirm that our classifier generalizes beyond what simple white-listing attains.

Session and context. Our session features are a mix of lexical and context-based attributes obtained from sessions τ_s containing q_a during the training and validation period (an example segment is given by the shaded area within the purple box in Figure 3). The lexical features measure the similarity of q_a and q_c with the queries preceding q_a in the session, an indicator of whether topic drift is occurring in the session. The context features revolve around the context model (Θ) at query time and the top category in this model (o^*) which represents the *session context*.

Historical. We derived our historical features over the candidate generation period τ_h captured over 13 months of query logs. These statistics capture the long-term frequency of q_a and q_c independently and together along with the typical overlap of their rankings.

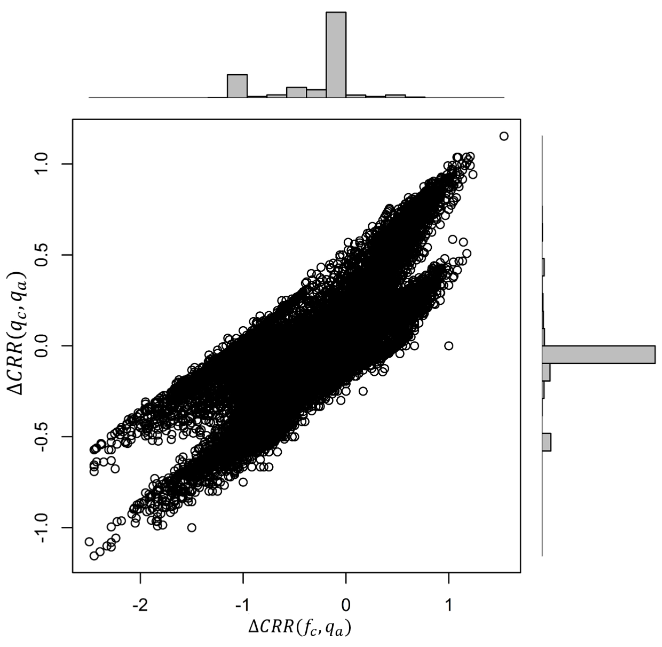

Figure 4: Scatterplot of the gain in *CRR* between the original query ranking and its suggestion against the gains between the original and the blended ranking.

Cross source. We also recorded the typical context model for q_c over τ_h and compared its similarity to that of Θ in τ_s, a measure of the topic alignment of q_c with q_a.

Next we show how a classifier trained based on these features can effectively decide when to blend the search results of a query with those returned for a related query suggestion.

5.2 Evaluation

Our evaluation results for Corfu and the Memorized variants are given in Table 4. We define *wins* and *losses* only on cases where fusion improves or hurts performance. That is, the correct cases of backing off to the original ranking do not contribute to the wins. Similarly, missing out on good fusion candidates does not count as a loss.[6] Using Corfu we were able to make significant gains over the original ranking, clearly alleviating the problems caused by naïvely using fusion on all query candidates. Our gains also apply across both ambiguous and general queries, although consistent with our previous experiments they are higher on the former set. This confirms our initial hypothesis that ambiguous queries should benefit more from contextual disambiguation. Simply memorizing good blendings was shown to not be an effective alternative. memorization generally leads to negative changes in all cases except a very slight gain in the case of context-independent memorization for ambiguous queries.

It is worth noting that the numbers in Table 4 are averaged over all impressions in our evaluation set. However, Corfu triggers fusion only in 5.68% of all impressions, and 9.52% over the ambiguous subset. This means that in a great majority of the cases, Corfu does back off to the ranking of the original query. If we average Δ*CRR* only over impressions that Corfu recommends for fusion,

[6]We ignore the costs associated with running an extra query here. In practice, zero gain might be considered as a slight loss.

Table 3: The features used for training the contextual blending classifier. Here, s is the search session containing the query q_a; $\mathcal{Q}_a$ represents all queries that appear in the session up until impression I_a; $\Theta(s,a)$ denotes the context model of s before receiving q_a which consists of pairs of ODP categories o and their associated confidence value $p(o)$. The category with the highest confidence value in Θ is distinguished by o^*, and the last query in $\mathcal{Q}_a$ is denoted by q_l.

Name	Description
Memorization (τ_v)	
$\Delta CRR_\mu(f, q_a)$	Average fusion utility of (q_a, q_c).
$\Delta CRR_\mu(f, q_a \vert o^*)$	Average fusion utility of (q_a, q_c, o^*).
Session and Context (τ_s)	
I_a	Position of query q_a in the session.
$p_\mu(o)$	Average category confidence values in Θ.
$p(o^*)$	Confidence of top category in Θ.
$H(\Theta)$	Entropy of category confidence values in Θ.
$Sim(q_a, q_l)$	Lexical similarity between q_a and q_l.
$Sim(q_c, q_l)$	Lexical similarity between q_c and q_l.
$Sim_\mu(q_a, \mathcal{Q}_a)$	Avg. text similarity of q_a and $\mathcal{Q}_a$ queries.
$Sim_\mu(q_c, \mathcal{Q}_a)$	Avg. text similarity of q_c and $\mathcal{Q}_a$ queries.
$Sim_{\max}(q_a, \mathcal{Q}_a)$	Max. text similarity of q_a and $\mathcal{Q}_a$ queries.
$Sim_{\max}(q_c, \mathcal{Q}_a)$	Max. text similarity of q_c and $\mathcal{Q}_a$ queries.
$\mathcal{I}(q_a \in \mathcal{Q}_a)$	Has q_a appeared previously in $\mathcal{Q}_a$?
$\mathcal{I}(q_c \in \mathcal{Q}_a)$	Has q_c appeared previously in $\mathcal{Q}_a$?
$\mathcal{I}(c^*, L1)$	Is c^* a 1st-level ODP category?
$\mathcal{I}(c^*, L2)$	Is c^* a 2nd-level ODP category?
$\mathcal{I}(q_a == q_l)$	Is q_a the same as q_l?
Historical (τ_h)	
$F(q_a)$	Historical query frequency for q_a.
$F(q_c)$	Historical query frequency for q_c.
$\Omega(q_a, q_c, 1)$	No. common docs in top-1 of q_a and q_c.
$\Omega(q_a, q_c, 5)$	No. common docs in top-5 of q_a and q_c.
$\Omega(q_a, q_c, 10)$	No. common docs in top-10 of q_a and q_c.
$F(q_a, q_c)$	Number of times q_a is followed by q_c.
$\delta F(q_a, q_c)$	$F(q_a) - F(q_c)$.
$\rho F(q_a, q_c)$	$F(q_a)/F(q_c)$.
Cross source ($\tau_{s,h}$)	
$\cos(\Theta, \Theta_h(q_c))$	Cosine similarity between Θ and $\Theta_h(q_c)$

we get a better picture of *f-segment* (fusion-segment) gains. On the *Ambiguous queries* dataset, Corfu achieves $\Delta CRR = 0.141$, improving over the results returned for the original query by 30.72%. Similarly, the f-segment Δ*CRR* on the *All-queries* dataset is 0.126 representing a 23.87% improvement over the no-fusion baseline.

Our results indicate that the causes of query ambiguity and fusion effectiveness are complex and it is not enough to simply memorize which contextual candidates work and which do not. Ranking the features from Corfu over ambiguous queries gives us insight into which are important when learning when to perform data fusion (Table 5). The ranking is based on the normalized reduction in residual squared error and the weights are normalized with respect to the top feature. The strongest feature is the Memorized utility score for query pairs, which is not surprising given the positive results for Memorized(P) on ambiguous queries. This demonstrates that Corfu is able to build on its success with the introduction of the other features. Of note is the fact that 3 of the top 5 features relate to the distribution of category confidence scores. The success of Corfu is strongly tied to the quality of the context model and a clear sign that context plays a part in determining whether data fusion will be successful during query disambiguation. Finally, the 2nd highest ranked feature is the frequency of q_c in τ_h, potentially a sign of confidence in q_c as a fusion candidate.

Table 4: The performance of different fusion methods on the testing dataset. The numerical columns respectively represent number of wins (#W), number of losses (#L), accuracy (Acc.) and Δ*CRR*. All numbers are computed with respect to the results of the original query (no fusion). All differences are statistically significant according to the paired t-test ($p < 0.01$). For reference, the *CRR* of a naïve system that always blends is about 7% worse than the no fusion baseline on both testbeds.

Method	#W	#L	Acc.	Δ*CRR*	(% ▲/▼)
All queries					
Memorized(P)	128K	144K	0.776	-0.007	(▼1.32%)
Memorized(T)	154K	199K	0.762	-0.021	(▼3.96%)
Corfu	50K	32K	0.803	0.007	(▲1.35%)
Ambiguous queries					
Memorized(P)	55K	53K	0.778	0.001	(▲0.21%)
Memorized(T)	69K	76K	0.768	-0.009	(▼1.96%)
Corfu	50K	26K	0.804	0.034	(▲2.92%)

Table 5: The top five most important features according to the Corfu model trained on *Ambiguous* queries. Similar trends can be found for the other dataset (not presented here). All feature weights are normalized with respect to the feature with the highest weight.

Feature	Weight	Description
$\Delta CRR_{\mu}(f, q_a)$	1.000	Memorized fusion utility of (q_a, q_c).
$F(q_c)$	0.572	Historical frequency of the candidate.
$p_{\mu}(o)$	0.433	Mean of confidence values in Θ.
$H(\Theta)$	0.285	Entropy of confidence values in Θ.
$p(o^*)$	0.200	Confidence of the top category in Θ.

Beyond CRR. The results presented so far confirm the effectiveness of Corfu in deciding when to apply contextual fusion, and demonstrate the improvements in retrieval quality as measured by *CRR*. However, evaluation metrics do have their shortcomings [29]. Given that our models are optimized for *CRR*, it would be reassuring to re-evaluate our results using alternative metrics. For this purpose, we define two metrics based on the first[7] and the last satisfied clicks in the session. We refer to the document that receives the last satisfied click in the session as $d_{\mathcal{L}}$. Likewise, the document that receives the first satisfied click is denoted by $d_{\mathcal{F}}$. Our first metric ($RR_{\mathcal{L}}$) measures the reciprocal rank of $d_{\mathcal{L}}$ in the original and blended rankings for a given query. This is inspired by observations in prior work that considered the last click of the session as a proxy for the user's intended destination [14, 42]. Our second metric ($RR_{\mathcal{F}}$) is similarly defined based on $d_{\mathcal{F}}$. Here, the idea is that boosting the position of these documents would allow users to satisfy their information needs more quickly.

The results are consistent with our previous findings. Corfu improves $RR_{\mathcal{L}}$ from $0.340 \rightarrow 0.351$ (▲2.3%) across all queries, and from $0.326 \rightarrow 0.338$ (▲3.6%) on the ambiguous subset. The f-segment gains are more substantial, as expected: $0.214 \rightarrow 0.351$ (▲64.2%) on all queries, and $0.232 \rightarrow 0.357$ (▲53.7%) on the ambiguous subset. The $RR_{\mathcal{F}}$ trends are consistent: Corfu improves $RR_{\mathcal{F}}$ from $0.535 \rightarrow 0.544$ (▲1.5%) across all queries, and from $0.541 \rightarrow 0.556$ (▲2.8%) on the ambiguous subset. The f-segment gains are: $0.477 \rightarrow 0.622$ (▲30.2%) on all queries, and $0.462 \rightarrow 0.621$ (▲34.3%) on the ambiguous subset. Overall, the $RR_{\mathcal{F}}$ numbers confirm that Corfu successfully places the first satisfied clicks higher in the ranking, allowing users to access high quality documents faster. The improved performance on $RR_{\mathcal{L}}$ also suggests that the blended rankings selected by Corfu are able to place these *target* documents higher in the ranking compared to the no-fusion baselines.

[7]That is the first satisfied click after issuing the current query. Clicks for previous queries in the session are excluded.

6. RELATED WORK

We group prior research by the three key problems we address in this paper: query disambiguation, finding effective query reformulations, and selectively blending results from multiple queries.

Contextual query disambiguation. Word disambiguation is an ongoing topic of research in Natural Language Processing and information retrieval (IR). In IR, disambiguation is usually applied to focus search results for queries that are either polysemous (have multiple senses) or that belong to a number of subtopic categories [36]. For instance, queries that are abbreviations and acronyms can be matched to their expanded forms by mining word associations in anchor text from a corpus and query reformulations from query logs [40]. Semantic graphs are commonly used to model word senses and are usually built using thesauri or lexical databases such as WordNet[8]. In these cases, supervised and unsupervised approaches such as PageRank, HITS or node similarity can be used to find alternative queries [24, 38]. While our approach is also unsupervised, we do not require an external ontology or corpus-learned language model. Similar work by Mihalkova and Mooney [25] makes use of prior queries within a session to perform disambiguation, although in their case at least 5 clicks on URLs with distinct hostnames are required before disambiguation can be achieved. They use a Markov logic network to re-rank for the user's ambiguous query, whereas in our fusion approach we can both re-rank and inject relevant URLs into the user's ranking. We note that intentionally ambiguous queries can be issued by users in a session when exploring an intrinsically diverse topic [28]: in these cases it would be beneficial to use session context to assist the user in narrowing or adjusting their information intent.

As well as disambiguation, web search ranking can be improved by incorporating user context. This context can include explicit features based on demographic identifiers such as age, gender, and location, or implicit features that are inferred from search behavior and history. For example, the predicted reading ability of a user, combined with an estimation of the technical difficulty of documents, can be used to improve search results, especially for educational documents [10]. Long-term user search histories can personalize search results [37] and even identify cases in which users are behaving atypically [15]. All of these categorical contextual markers can be used in our technique, both long-term and short-term. We focus our study on short-term, session subtopic categories as these are easy to interpret and useful in identifying the ambiguous queries from search logs that our method specifically targets.

Query reformulation. The prevalence of session search in query logs [19] has resulted in a wealth of research on the interpretation of query reformulations. While it has been found that as many as 10-20% of reformulations are simply spelling corrections [12], our underlying retrieval system helps correct for this with a robust spelling-correction system, meaning the observed reformulations are useful as observations of user progress and intent. Huang and Efthimiadis [18] performed an in-depth analysis of term-based query changes during a session in order to create a reformulation taxonomy. This was taken further by Kato et al. [22] who used a

[8]http://wordnet.princeton.edu/

questionnaire and search log analysis to define six cognitive search intents that users exhibit when reformulating queries, which they incorporated into a machine learning-based query expansion algorithm. In our work, we do not attempt to differentiate between reformulation types: we record effectiveness scores for all query-context-reformulation triples.

Early work on query reformulation involved query expansion, a classic IR technique that automatically replaces, re-weights or adds terms to queries to improve ranking and retrieval. Early methods searched for term co-occurrences within the document collection as candidates for expansion [26]. Modern techniques find term associations in thesauri, semantic graphs and query logs to create better expanded queries [13]. Typically, these expansion approaches replace the initial search results entirely with those of the expanded query, whereas our method can selectively fuse ranked results from multiple query variations.

A less intrusive method than query expansion, and one widely deployed for Web search, is to *suggest* new queries to the user. Two main avenues for providing new contextual queries to the user have been personalized query auto-completions [2, 34], which are estimated using previous queries from the same user, or query suggestions. Query suggestions occupy space on the SERP and are found by mining sessions in search logs [27], traversing query/click graphs [7, 11] or extracting phrases from corpora [6].

The query reformulation research most similar to ours is by Cao et al. [9], who proposed an unsupervised approach to cluster search log queries into unlabeled concepts and demonstrated their method's scalability by evaluating over a large search log. While their goal was to perform query suggestion, our goal is to find good reformulation candidates for contextual blending. In turn, our use of blending gives our method more fine-grained control over the nature of the final search results at the level of individual documents, whose rankings are influenced by the query disambiguation results. Our addition of a taxonomy to label concepts also makes interpreting query results easier. Finally, our technique has an even smaller computational footprint and we demonstrate its scalability in our evaluation over an even larger search log.

In later work, Cao et al. [8] use a taxonomy combined with click information, similar to our use of the ODP and clicks to create session context. While they also use other query and term features to train a prediction method offline to categorize search queries, their focus was on predicting faceted vs. vertical search, not performing selective blending of query suggestion results. Also, our method is trained offline but can be dynamically updated as searches occur in order to find new reformulations and reinforce established ones.

Other related reformulation work includes Wang and Zhai [39] that mined a search log for reformulations and then determined whether the user's original query under-specified (the user did not know which terms to use) or mis-specified (the user used the wrong terms) their search intent by mining term co-occurrences in a corpus. Ozertem et al. [27] followed a similar approach, but with query co-occurrences in a machine learning framework. Like these approaches, our method mines search logs for reformulations as a means for disambiguation, but does not operate on a term-by-term basis. In addition, the method in Ozertem et al. is mostly concerned with identifying effective individual terms and posits term suggestion as a potential use, whereas we directly learn reformulations to improve rankings for the user's original query.

Blending results from multiple queries. Data fusion techniques merge relevant results from related or expanded queries so as to improve the original ranking [32, 33]. While effective, these methods have not taken user context into account. Instead, state of the art research has investigated the blending of results from expanded queries; for instance, LambdaMerge is a supervised learning method that learns how to optimally blend results from query reformulations found using a click graph [33], whilst Xu et al. [43] implemented a kernel-based method for learning blending weights for results from query reformulations found using click-through data. Optimal blending strategies are an active area of research and not the focus of this paper, so we adopt a fixed blending strategy that allows us to evaluate the reformulation technique more effectively. Our novel contribution related to blending is the development of the contextual fusion classifier that uses session, context, historical and other features to learn when to selectively apply blending. To the best of our knowledge, ours is the first work to develop a context-based classifier of this type.

7. CONCLUSIONS

In this paper we investigated methods for improving the search experience for ambiguous queries by using user context. We defined user context as the topical categorization of the user's session and query. We began by focusing on contextual query suggestion. Unlike other methods which find suggestion candidates based on query/click graphs, we mined search logs for commonly co-occurring queries in sessions. We defined a utility function that accounted for the sparsity of clicks on ambiguous queries by including and weighting those that occur later in the session. Conditioning this utility on contextual categories and aggregating over search logs allowed us to rank and select a contextual query suggestion candidate. In our evaluation we found that a hybrid of both contextual/non-contextual queries (the $CosQus(H)$ method) gave significant gains over a state-of-the-art baseline.

Next, we looked at data fusion as a means of directly improving search results for ambiguous queries. We blended results from our contextual query suggestions with those of the user's query and observed that naïvely blending on all suggestions led to poor overall performance. To resolve this, we proposed a novel fusion classifier (Corfu) that could distinguish when to perform data fusion and when to suggest a query instead. We trained it using session, context and query features extracted over a large-scale search log from `bing.com`. Our approach showed significant improvements over the rankings of the original query, for both ambiguous and general queries, and also made gains over simply memorizing which queries to blend on. An analysis of our feature set highlighted the importance that context plays in classifying when to use data fusion for contextual query disambiguation.

This work could be extended by exploring different types of contextual information for disambiguation. For instance, location would provide a suitable classification space and would lead to a different set of candidate queries and feature set for the classifier. Additionally, user history (rather than session context) could be used as a means for results personalization and a way to use context in the first query of a session. Also, more complex data fusion methods could be explored, potentially by taking context into account during the merging itself, though our approach of contextual query suggestion combined with a fusion classifier demonstrates that a simple fusion technique is sufficient.

References

[1] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. EDBT'04, pages 588–596. Springer-Verlag, 2004.

[2] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. WWW '11, pages 107–116. ACM, 2011.

[3] N. Belkin, P. Kantor, E. Fox, and J. Shaw. Combining the evidence of multiple query representations for information retrieval. *Information Processing and Management*, 31:431–448, 1995.

[4] P. N. Bennett, K. Svore, and S. T. Dumais. Classification-enhanced ranking. WWW '10, pages 111–120. ACM, 2010.

[5] P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short- and long-term behavior on search personalization. SIGIR '12, pages 185–194. ACM, 2012.

[6] S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. SIGIR '11, pages 795–804. ACM, 2011.

[7] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: Model and applications. CIKM '08, pages 609–618. ACM, 2008.

[8] H. Cao, D. H. Hu, D. Shen, D. Jiang, J.-T. Sun, E. Chen, and Q. Yang. Context-aware query classification. SIGIR '09, pages 3–10. ACM, 2009.

[9] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. KDD '08, pages 875–883. ACM, 2008.

[10] K. Collins-Thompson, P. N. Bennett, R. W. White, S. de la Chica, and D. Sontag. Personalizing web search results by reading level. In *Proceedings of CIKM 2011*, pages 403–412. ACM, 2011.

[11] N. Craswell and M. Szummer. Random walks on the click graph. SIGIR '07, pages 239–246. ACM, 2007.

[12] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In D. Lin and D. Wu, editors, *EMNLP 2004*, pages 293–300, Barcelona, Spain, 2004. Association for Computational Linguistics.

[13] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Query expansion by mining user logs. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):829–839, July 2003.

[14] D. Downey, S. Dumais, D. Liebling, and E. Horvitz. Understanding the relationship between searchers' queries and information goals. In *CIKM '08*, pages 449–458, Napa Valley, CA, 2008.

[15] C. Eickhoff, K. Collins-Thompson, P. N. Bennett, and S. Dumais. Personalizing atypical web search sessions. WSDM '13, pages 285–294. ACM, 2013.

[16] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst.*, 23(2):147–168, Apr. 2005.

[17] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[18] J. Huang and E. N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. CIKM '09, pages 77–86. ACM, 2009.

[19] B. J. Jansen, A. Spink, and J. O. Pedersen. A Temporal Comparison of AltaVista Web Searching. *Journal of The American Society for Information Science and Technology*, 56:559–570, 2005.

[20] E. C. Jensen, S. M. Beitzel, A. Chowdhury, and O. Frieder. Query phrase suggestion from topically tagged session logs. FQAS'06, pages 185–196. Springer-Verlag, 2006.

[21] M. P. Kato, T. Sakai, and K. Tanaka. When do people use query suggestion? a query suggestion log analysis. *Inf. Retr.*, 16(6):725–746, Dec. 2013.

[22] M. P. Kato, T. Yamamoto, H. Ohshima, and K. Tanaka. Investigating users' query formulations for cognitive search intents. SIGIR '14, pages 577–586. ACM, 2014.

[23] F. Liu, C. Yu, and W. Meng. Personalized web search by mapping user queries to categories. CIKM '02, pages 558–565. ACM, 2002.

[24] C. Makris, Y. Plegas, and S. Stamou. Web query disambiguation using pagerank. *J. Am. Soc. Inf. Sci. Technol.*, 63(8):1581–1592, Aug. 2012.

[25] L. Mihalkova and R. Mooney. Search query disambiguation from short sessions. In *Beyond Search: Computational Intelligence for the Web Workshop at NIPS*, 2008.

[26] J. Minker. An evaluation of query expansion by the addition of clustered terms for a document retrieval system. *Information Storage and Retrieval*, 8(6):329–348, Dec. 1972.

[27] U. Ozertem, O. Chapelle, P. Donmez, and E. Velipasaoglu. Learning to suggest: A machine learning framework for ranking query suggestions. SIGIR '12, pages 25–34, Portland, OR, 2012.

[28] K. Raman, P. N. Bennett, and K. Collins-Thompson. Toward whole-session relevance: Exploring intrinsic diversity in web search. SIGIR '13, New York, NY, USA, 2013. ACM.

[29] T. Sakai. On the reliability of information retrieval metrics based on graded relevance. *Inf. Process. Manage.*, 43(2):531–548, Mar. 2007.

[30] T. Sakai and Z. Dou. Summaries, ranked retrieval and sessions: A unified framework for information access evaluation. SIGIR '13, pages 473–482, Dublin, Ireland, 2013.

[31] M. Sanderson. Ambiguous queries: Test collections need more sense. SIGIR '08, pages 499–506. ACM, 2008.

[32] J. A. Shaw and E. A. Fox. Combination of Multiple Searches. In *Text REtrieval Conference*, 1994.

[33] D. Sheldon, M. Shokouhi, M. Szummer, and N. Craswell. Lambdamerge: Merging the results of query reformulations. WSDM '11, New York, NY, USA, 2011. ACM.

[34] M. Shokouhi. Learning to personalize query auto-completion. SIGIR '13, New York, NY, USA, 2013. ACM.

[35] M. Shokouhi and L. Si. Federated search. *Foundations and Trends in Information Retrieval*, 5(1):1–102, 2011.

[36] R. Song, Z. Luo, J.-R. Wen, Y. Yu, and H.-W. Hon. Identifying ambiguous queries in web search. WWW '07, pages 1169–1170. ACM, 2007.

[37] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. SIGIR '05, pages 449–456. ACM, 2005.

[38] G. Tsatsaronis, I. Varlamis, and K. Nørvåg. An experimental study on unsupervised graph-based word sense disambiguation. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 6008 of *Lecture Notes in Computer Science*, pages 184–198. Springer Berlin Heidelberg, 2010.

[39] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. CIKM '08, pages 479–488. ACM, 2008.

[40] X. Wei, F. Peng, and B. Dumoulin. Analyzing web text association to disambiguate abbreviation in queries. SIGIR '08, pages 751–752. ACM, 2008.

[41] R. White and S. Drucker. Investigating behavioral variability in web search. In *WWW '07*, 2007.

[42] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR '07*, pages 159–166, Amsterdam, The Netherlands, 2007.

[43] J. Xu, W. Wu, H. Li, and G. Xu. A kernel approach to addressing term mismatch. WWW '11, pages 153–154. ACM, 2011.