Generating query suggestions by exploiting latent semantics in query logs

Saeedeh Momtazi
Hasso Plattner Institute, Germany

Fabian Lindenberg
Hasso Plattner Institute, Germany

Abstract
Search engines assist users in expressing their information needs more accurately by reformulating the issued queries automatically and suggesting the generated formulations to the users. Many approaches to query suggestion draw on the information stored in query logs, recommending recorded queries that are textually similar to the current user’s query or that frequently co-occurred with it in the past. In this paper, we propose an approach that concentrates on deducing the actual information need from the user’s query. The challenge therein lies not only in processing keyword queries, which are often short and possibly ambiguous, but especially in handling the complexity of natural language that allows users to express the same or similar information needs in various differing ways. We expect a higher-level semantic representation of a user’s query to more accurately reflect the information need than the explicit query terms alone can. To this aim, we employ latent Dirichlet allocation as a probabilistic topic model to reveal latent semantics in the query log. Our evaluations show that, whereas purely topic-based query suggestion performs the worst, the interpolation of our proposed topic-based model with the baseline word-based model that generates suggestions based on matching query terms achieves significant improvements in suggestion quality over the already well performing purely word-based approach.

Keywords
Information retrieval; latent Dirichlet allocation; query suggestion; search; topic modelling

1. Introduction

When trying to satisfy an information need, users of common search engines are challenged to formulate a keyword query that best reflects their need. Often, they fail to express their need precisely with the first query they issue and are required to refine their formulation. Typical modifications are specifications (e.g. the query ‘jobs in Hamilton’ is modified to ‘IT jobs in Hamilton’), generalizations (e.g. query ‘walmart newport beach’ is followed by ‘walmart store’), as well as parallel searches (e.g. ‘southwest airlines’ and ‘us airways’) (Rieh and Xie, 2006). Other refinements concern the wording (e.g. ‘auto dealer’ is replaced by ‘car dealer’), the spelling (e.g. ‘niagra falls’ is corrected to ‘niagara falls’), and more. Multiple modifications may be necessary until a query yields results that meet the users’ needs.

In an effort to support users in expressing their information needs more precisely, search engines reformulate the issued queries and either process the reformulations automatically or offer them as suggestions to the users. The ultimate goal is to help the users find (faster) what they are searching for.

Common approaches for generating such reformulations make use of either the preliminary retrieved documents or knowledge about queries issued in the past. Many of these approaches suggest queries that are textually similar to the current user’s query or that are known to co-occur frequently with it. Alternative suggestion techniques retrieve popular phrases from the documents indexed by the search engine that can potentially complete the user’s query, or they extend the user’s query by adding terms drawn from high-ranking documents in the first search result. Pseudo-relevance
feedback is one of the most well-known methods of the latter approach. In this method, the original query first passes to a search engine and the top retrieved documents are selected and assumed to be relevant. The original query is then expanded by top terms from these documents based on their distributions and relevance to the query terms [1].

We, in contrast, focus our efforts on deducing the actual information need from the user’s query and identifying queries of past users with a similar need. This is particularly challenging: not only are keyword queries typically short in nature and, hence, contain little explicit information; they also pose certain difficulties, such as term ambiguity, that result from the complexity of natural language.

We aim at recommending queries as alternative formulations that successfully helped users in the past satisfy the same or a very similar information need. This requires us to (a) deduce the user’s information need from the imprecise queries issued so far, (b) deduce the various information needs from query sessions of past users, (c) find those sessions with a matching need, and (d) identify the queries that have ultimately proven to be successful.

Our emphasis lies on the means to handle the complexity that is characteristic of natural language. This includes, but is not limited to, recognizing the relationship between synonymous query terms or distinguishing between the different meanings of polysemes.

As mentioned, information about past users and their queries, which is provided in form of a query log, is the best data source towards our goal. Although the suggestion technique developed is applicable to different search scenarios, we focus for demonstration purposes on Web search and consequently chose a query log from a commercial Web search engine. Its records include the issued queries along with the times of issue and the ranks of any results the respective user has clicked on. For each clicked result, the domain portion of the URL is provided but no data on the retrieved Web resource is available. Also not available are the URLs of results the user has not clicked on and any information about the user (such as demographic information or location), besides an anonymous user ID.

To this end, we developed a technique for generating query suggestions whose strength lies in abstracting from explicit keyword terms to higher-level semantics. Therefore, the suggestion generation is no longer subject to limitations shared by techniques that rely heavily on query term matching.

We employ latent Dirichlet allocation (LDA), introduced by Blei et al. [2], to learn a probabilistic topic model from the query log that reveals so-called latent semantics. That is, the topic model discovers hidden semantic structures within the query log, especially the thematic relationships of individual terms, queries and entire query sessions. By exploiting this newly gained knowledge, we can examine the topical structures of queries, which we expect to better reflect the actual information needs than their explicit query terms alone can.

The structure of the paper is as follows: in Section 2, we give a brief overview of existing approaches to query suggestion generation. Section 3 presents the processing of the query log, which lays the foundations for the query suggestion techniques developed. In Section 4, we explain how information retrieval models can be adapted to the task of query suggestion. Motivated by the limitations of the presented models, we introduce topic modelling in Section 5. Section 6 covers our proposed topic-based query suggestion and the combined approach that lets us benefit from the advantages of both word- and topic-based suggestion generations. We conduct two different experiments on small and large test datasets in Section 7 to compare our proposed interpolated technique with the state-of-the-art word-based approach. Finally, Section 8 concludes this paper by summarizing all findings.

2. Related work

As already indicated in the introduction, various approaches for query suggestion have been developed. Some approaches aim at generating query suggestions without resorting to the use of query logs. Bhatia et al. [3] argue that for some search scenarios, for example in the enterprise domain or for intranet search, query logs do not exist or the number of past queries is too small to build adequate models. Their probabilistic technique identifies candidate phrases in the document corpus and suggests those phrases that are likely to be completions of the partial query the user is typing. A different approach by Bast and Weber [4] retrieves indexed documents that appear relevant with respect to the user’s partial query and then suggests individual terms from those documents to complete the query. Apart from the fact that neither approach utilizes a query log, both solve a task slightly different from ours: they suggest completions of partial queries. In our view, query completion can cover only a subset of potentially suitable suggestions. Two possible counterexamples are the ones given for generalization and parallel searches at the beginning of Section 1.

Wan et al. [5] proposed a topic-based approach to query suggestion for video search. In fact, their work and ours bear some resemblance in that the authors also employ LDA for the generation of thematically related suggestions. However, there are distinct differences: as with the two previous approaches, their technique offers query completions and not entire reformulations. The second difference is that the suggested terms are drawn from the related topic and not from a query log, as our complete query suggestions are. By consulting the query log, we ensure the suggestion only of queries

Journal of Information Science, 42(4) 2016, pp. 437–448 © The Author(s), DOI: 10.1177/0165551515594723
that have proven to be successful for past users with the same information need, while the approach by Wan et al. lacks this notion. The third difference is that their suggestion generation is purely topic based. According to our own findings (which are presented in detail in Section 7), interpolation of topic-based and word-based query suggestion outperforms both.

Unlike the approaches discussed so far, the works by Baeza-Yates and Tiberi [6] and Boldi et al. [7] are examples of the incorporation of query logs. In the proposed model by Baeza-Yates and Tiberi [6], click-through data is used to provide recommendations. Their method is based on the cover graph model. A cover graph includes both queries and URLs, where a query and an URL are connected if the person who issued the query also clicked on the URL. Suggestions for a query are based on the queries that share more URLs.

In the proposed query suggestion model by Boldi et al. [7], a query flow graph is utilized, which is constructed from their query log (also in Boldi et al. [8]). The graph connects two queries with a directed edge if they are likely to belong to the same search mission [7]. In order to generate suggestions for a given query, the technique performs a short random walk over the query graph, starting from the node that corresponds to the user’s query. Queries are suggested depending on the probability of the random walker being at their corresponding nodes after a certain number of iterations. A major difference between the authors’ and our work is that we take on a document-centric view of the query log, as is discussed next, whereas Boldi et al. created a graph representation, as described. Another important differentiator lies in the calculation of the probability of two queries being connected in the graph. Besides considering statistical features, such as the number of sessions in which the pair of queries appear or their respective issue times, the textual similarity of the two queries is decisive [8]. In contrast, by learning a topic model from the query log, we incorporate higher-level semantics in addition to the calculation of textual similarities.

Broccolo et al. [9] introduce the just-mentioned document-centric view of query logs. They compose so-called virtual documents from those query sessions that are judged to reflect the same information need. Suggestions are, then, extracted from these documents. We find this novel view intriguing, because by constructing a document corpus from the query log, the authors make it possible to adapt well-proven models and algorithms from the field of information retrieval to the task of query suggestion. We thus discuss in Section 4 potential realizations using information retrieval models. While doing so, we identify certain limitations that this approach has and motivate the development of our own suggestion generation technique based thereon. As shown by Broccolo et al., this method outperforms the previous state-of-the-art techniques including cover graph [6] and query flow graph [7]. In this paper, we consider this method as the baseline model for our experimental performance comparison.

3. Query log processing

For the generation of query suggestions, we intend to learn from the search history of past users. Search engines usually collect valuable information in so-called query logs that help reconstruct this history. Such logs list the specific queries that have been issued together with the time of issue and, often, the search results that have been clicked on.

However, the stored data are inevitably incomplete. In particular, they can only include the query as an explicit but uncertain representation of the user’s information need; the actual need might differ from its representation and is not known to the search engine. Thus, before we can benefit from the search history of past users, the challenge for us is to deduce the users’ information needs from the data that is available. This, in turn, requires pre-processing of the query log and the application of adequate heuristics. Both are presented in this section.

3.1. Query session detection

A query session is the subsequently issued queries of a user that pursue the satisfaction of the same information need. For deducing the user’s information need, we find it helpful to consider all queries in a session together. In order to be able to do so, it is necessary to first detect the sessions within a query log, as the search engine has no understanding of which queries belong together.

One of the commonly used heuristics to determine the boundaries of a session is maxBreak [10–12]. This method limits each session based on the length of the breaks between two queries in a list. In other words, the session expires if the user spends a certain amount of time without any interaction with the search engine. The next query after this break will be the start of the next session as a result. The time limit is configurable.

Once all sessions in a query log are detected, we can distinguish between two types of sessions: successful and unsuccessful ones. The former are those that have been able to satisfy the user’s information needs, the latter have failed to do so. Unfortunately, such information is not explicitly available in query logs, unless the search engine collects explicit feedback from its users. Hence, we apply the heuristic suggested by Baraglia et al. [13] and Broccolo et al. [9]: a session
is successful if and only if the user has clicked on at least one search result of the very last query in the session. The assumption is that the visited Web page has satisfied the user’s information need and that the user has consequently ended the session.

Furthermore, we can enforce a minimum session length, which indicates a minimum number of queries per session. By setting the minimum length to 1, all successful sessions are considered.

3.2. The AOL query log

The query suggestion method proposed in this paper is applicable to a range of search scenarios, provided that a query log is available. However, for demonstration purposes, we decided to focus on Web search and accordingly chose a query log from a commercial Web search engine.

We used the query log of the AOL search engine [14], which consists of approximately 21 million queries issued by about 650,000 unique users. The queries have been transformed to lowercase letters and most punctuation has been removed. Of the 21 million normalized queries, about 10 million are unique. The data were collected in the 3 month period from 1 March 2006, to 31 May 2006. Table 1 summarizes detailed statistics.

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click events</td>
<td>21,011,340</td>
</tr>
<tr>
<td>Normalized queries</td>
<td>19,442,629</td>
</tr>
<tr>
<td>Queries without clicks</td>
<td>16,946,938</td>
</tr>
<tr>
<td>Unique queries</td>
<td>10,154,742</td>
</tr>
<tr>
<td>Unique user IDs</td>
<td>657,426</td>
</tr>
</tbody>
</table>

3.3. Construction of a virtual document corpus

Considering our definition for successful session, we assume that two successful sessions pursue the same or at least a very similar information need, if the sessions end with the same exact query. Under this assumption, we can group successful sessions by their final queries. Broccolo et al. [9] propose composing what they call virtual documents. They simply concatenate all successful sessions that share the same final query, thereby creating bags of words of all query terms used in those sessions.

Figure 1 illustrates an example similar to one given by the authors. It shows two successful query sessions that both end in the final query ‘caesars palace’. This last query serves as the title of the newly composed virtual document, whereas all queries up until the respective final query make up the body of the document. By doing this for all unique final queries of successful sessions, we can create a corpus of virtual documents, which we call the virtual corpus. We implement this corpus construction as an additional processing step of the query log. When concatenating queries to compose the body of a document, our implementation can optionally perform the following pre-processing activities: conversion to lowercase letters, word stemming and stop word removal.

4. Leveraging information retrieval techniques for query suggestion

Creating a corpus of virtual documents from a query log, as described in the previous section, lays the groundwork for applying techniques known from the field of information retrieval to the problem of query suggestion. In case of a regular Web search engine, the user’s information need is expressed in the form of a keyword query and the returned documents are those Web pages indexed by the search engine that are deemed to be the closest to the user’s query. When generating query suggestions, we start with the same user query but do not intend to retrieve actual Web pages that satisfy the expressed information need. Rather, we try to support users in expressing their information needs more precisely by reformulating or expanding their original queries.

The approach we are taking to come up with such suggestions is to provide the user with alternative queries that have been issued in the past by users with a similar information need and that helped satisfy this need. Baraglia et al. [13] call this approach the search shortcut problem, and this is where the document corpus created from the query log comes into...
play. As mentioned, the body of each virtual document is composed of keyword queries that presumably express the same (or at least a very similar) information need. The final query serves as the title for the document.

Now, we can use information retrieval techniques to determine the top-\(k\) virtual documents most relevant to the user’s query, that is, documents that express an information need as close to the user’s as possible. We then suggest their titles to the user as alternative formulations of the original query, as these proved to be successful queries in the past.

Creating and utilizing a virtual document corpus in this manner to solve the search shortcut problem has been proposed by Broccolo et al. [9]. This solution is the starting point of our research that can be realized with different information retrieval models. In the present work, we develop our proposed model based on this scenario and within a language model-based framework, which has been proven to outperform tf-idf and BM25-based approaches [15] and has the flexibility of further modifications with advanced models.

To adapt the query likelihood model to the problem of query suggestion, given the user’s current session, which we from now on denote by \(\sigma\), we query our own corpus of virtual documents. To this aim, for every document \(d\), the probability of generating \(\sigma\) given the language model generated from \(d\) and Dirichlet prior smoothing is calculated as follows:

\[
P(\sigma|M_d) = \prod_q \prod_{w \in q} \left( \frac{N_d}{N_d + \mu} P(w|d) + \left( 1 - \frac{N_d}{N_d + \mu} \right) P(w|c) \right)
\]

where \(N_d\) denotes the length of that document and \(\mu\) is the smoothing parameter. The probability of document \(d\) generating word \(w\) is estimated by dividing the number of occurrences in the document by the total length of the document:

\[
P(w|d) = \frac{tf_{w,d}}{N_d}
\]

The probability of generating word \(w\) given the corpus \(c\) is estimated by dividing the number of occurrences in the corpus by the length of the corpus:

\[
P(w|c) = \frac{cf_w}{T}
\]

Then, we rank the virtual documents in descending order of their probabilities and return the titles of the top-\(k\) documents as suggestions.

Using the above model, we can retrieve relevant documents by matching the query terms with terms occurring in the stored documents. The underlying assumption is that the users formulated their queries having the ideal documents that answer their information needs in mind and chose terms that they expected to occur in such documents. While this assumption is reasonable, one should not unconditionally adhere to it under all circumstances. To give just one example, users searching for ‘car’ will most probably find their information needs also satisfied by documents that include the term ‘automobile’ instead of the exact query term.

Strict term-matching, however, is very inflexible and cannot handle the full complexity that is characteristic of natural language. A query term that does not appear in a document does not contribute to the relevance of this document, even
though its meaning might still be related to the content of the document. Such relation in meaning is often given when synonyms, hypernyms, hyponyms or acronyms are used.

At the same time, problems arise, too, when a query term does appear in a document but does not relate to the content of the document. This is often the case for a polyseme, a word that has different but related meanings, and even more so, for a homograph, a word that has several, completely different meanings (which may or may not be distinguishable in pronunciation). In order to avert such problems, we try to veer away from strict term-matching and operate with higher-level semantics instead. To this end, we propose using LDA topic modelling within the above framework to reveal the hidden thematic structure of a document corpus.

5. Revealing latent semantics with topic modelling

Topic models uncover hidden dependencies among the terms within a document as well as across documents in a corpus. These implicit structures are referred to as latent semantics. LDA models documents as a mixture of topics, where each topic, in turn, is a distribution over the terms of a fixed vocabulary [2].

Before we can formally define this process, we introduce the LDA notations:

- $D$ denotes the number of documents in the entire corpus.
- The number of topics, denoted by $T$, is assumed to be known and fixed.
- Each topic $\phi_t$, where $1 \leq t \leq T$, is a distribution over a fixed vocabulary of terms and $\phi_{tw}$ is the term proportion of term $w$ in topic $t$.
- $\theta_d$ is the topic mixture of the $d$th document and $\theta_{dt}$ is the topic proportion of topic $t$ in document $d$.
- $z_d$ are the topic assignments for document $d$, where $z_{dn}$ is the topic assignment for the $n$th term in document $d$.
- $w_d$ are the terms occurring in document $d$, where $w_{dn}$ is the $n$th term in document $d$. All terms are elements of a fixed vocabulary.
- $\beta$ is the Dirichlet prior on the topic-terms distributions.
- $\alpha$ is the Dirichlet prior on the document-topics distributions.

The generative process works as follows (as visualized in Figure 2):

1. For every topic, choose a distribution $\phi_t$ from a Dirichlet distribution with parameter $\beta$. That is, choose $\phi_t \sim \text{Dir}(\beta)$, where $1 \leq t \leq T$.
2. For each document $d$,
   (2.1) Choose $\theta_d \sim \text{Dir}(\alpha)$, and
   (2.2) For each term in document $d$,
       (2.2.1) Randomly choose a topic assignment $z_{dn}$ from the distribution $\theta_d$ for the $n$th term in document $d$.
       (2.2.2) Then, randomly choose a term $w_{dn}$ from the distribution $\phi_{zd,n}$.

![Figure 2. A graphical model of the LDA generative process.](image)
When taking the steps of this process, three implications are particularly noteworthy, also with regard to our own objectives (i.e. the generation of relevant query suggestions). By modelling each topic as a distribution over the terms of the same vocabulary, the ambiguity of terms can be expressed; for example, the term ‘bank’ would appear with a high probability in a topic about finance but also in a topic concerning water streams, maybe with a somewhat smaller probability. At the same time, terms that often appear together in the same context or even share a similar meaning are assigned with high probabilities to the same topic. Consequently, the terms ‘automobile’, ‘car’ and ‘engine’ are all likely terms for a topic about motor vehicles. Documents, in turn, are modelled as mixtures of topics; they all share the same set of topics but differ in the respective topic proportions. This provides LDA with a high expressive power.

6. Generating query suggestions using topic distributions

After extracting topics from a training corpus, LDA has the ability to infer topics from new documents that have not been seen in the training phase. This is a crucial feature in LDA, which helps us to generate query suggestions.

Our proposed topic-based query suggestion method works as follows: we first build a topic model from an external large corpus and then use the trained model to infer topic distributions for the virtual documents. The inferred topics are then used to retrieve those virtual documents that are thematically similar to the query and return their titles as suggestions. The process of generating query suggestions using topic distributions involves several steps, most of which are performed offline. The actual generation of suggestions for a specific query requires online processing, that is, processing at query-time:

*Offline preparation*

1. Split the query log into sessions, as described in Section 3.1.
2. Construct a corpus of virtual documents, as instructed in Section 3.3, from the sessions detected in Step 1.
3. Pre-process the documents of the external corpus.
4. Learn an LDA model from the external corpus.
5. Use the model trained in Step 4 to infer the topic distributions of all virtual documents constructed in Step 2.

*Online processing*

6. Pre-process and concatenate the queries that have been issued in the current session.
7. Retrieve the top-k documents from the virtual corpus that are thematically the most similar and return their titles as query suggestions.

As expressed in Equation 1, in the word-based query suggestion model, \( P(w | d) \) is calculated based on the explicit occurrence of a term in a document. In the topic-based model, instead, we say that a query term is likely to be generated by a document if its associated topic is also present in the document. Analogous to the idea of a document directly generating a query term, we now estimate the probability of a document generating a topic, \( P(\phi_t | d) \) with \( 1 \leq t \leq T \), and the probability of that topic generating the term, \( P(w | \phi_t) \). Then, we can calculate:

\[
P(w|d) = \sum_{t=1}^{T} P(\phi_t|d) P(w|\phi_t)
\]

These two probabilities are learned by the LDA topic model. From Section 5, we know that the topic proportion of topic \( \phi_t \) in document \( d \) is denoted by \( \theta_{dt} \). Hence, \( P(\phi_t|d) = \theta_{dt} \). Every topic \( \phi_t \) is a distribution over the entire vocabulary. Let \( \phi_{tw} \) denote the probability of word \( w \) in topic \( \phi_t \), that is, \( P(w|\phi_t) = \phi_{tw} \). Thus, it follows that:

\[
P(w|d) = \sum_{t=1}^{T} \phi_{tw} \theta_{dt}
\]

Contrary to the word-based model, which requires smoothing, in our proposed topic-based model, there is no need to smooth as it has already been considered when building the LDA model [2]. As a result, we can formulate our language model adaptation for generating topic-based query suggestions as follows:
\[ P(\sigma|M_d) = \prod_{q \in \sigma} \prod_{w \in q} \sum_{t=1}^{T} \phi_{t,w} \cdot \theta_{d,t} \quad (4) \]

The virtual documents are ranked according to their respective probabilities \( P(\sigma | M_d) \) and the titles of the first \( k \) documents are returned to the user as query suggestions.

Admittedly, we do not expect this approach to perform well, since its generated suggestions are solely based on topical associations. By considering the semantic link between a query term and its associated topic, we effectively broaden the search space that comprises possible query suggestions to thereby discover useful suggestions that, although being closely related, would not be found by strict word-matching techniques. However, at the same time, many suggestions that now fall in the search space might be very loosely related to the original query terms. To benefit from the advantages of topic-based approach and avoid its disadvantages, we propose a linear interpolation model, which enables us to combine the two word- and topic-based approaches from Equations 1 and 4:

\[ P(\sigma|M_d) = \prod_{q \in \sigma} \prod_{w \in q} (\lambda \cdot \left[ \frac{N_d}{N_d + \mu} \cdot \frac{tf_{w,d}}{N_d} + \left( 1 - \frac{N_d}{N_d + \mu} \right) \frac{cf_w}{T} \right] + (1 - \lambda) \sum_{t=1}^{T} \phi_{t,w} \cdot \theta_{d,t}) \quad (5) \]

where the parameter \( \lambda \in [0, 1] \) regulates the degree of interpolation. When \( \lambda \) is set to 1, the query suggestions are solely generated by word-based model. In contrast, when \( \lambda \) is set to 0, the query suggestions are solely generated by topic-based model. Choosing \( \lambda \in [0, 1] \) instead lets us benefit from the advantages of both approaches. A larger \( \lambda \) prevents a generalization, and a smaller \( \lambda \) mitigates the drawbacks of strict word-matching.

7. Evaluation

The evaluation of our proposed model consists of two parts: (a) manual assessment of suggestion quality based on a small set of sample queries, the TREC 2010 Web Track Topics; and (b) an automatic assessment employing the so-called search shortcut similarity metric proposed by Baraglia et al. [13]. We briefly present the experimental setup first.

7.1. Experimental setup

As introduced earlier, the dataset that we use is the AOL query log, from which the corpus of virtual documents is constructed. The virtual corpus is created by applying the split strategy \textit{maxBreak} with a time limit of 300 s and a minimum session length of three queries. It includes 736,974 documents.

We use ClueWeb09 Dataset\(^4\) a large Web corpus, as our external corpus for training the LDA model. It consists of about 1 billion Web pages in 10 languages, of which 500 million Web pages are written in English. The dataset has been and continues to be used in several TREC tracks, for example, in the TREC 2009 Web Track [16]. Since the query log we are employing contains, for the most part, English queries, solely the English subset of the ClueWeb09 corpus is of interest. Like any Web corpus, ClueWeb09 corpus is cluttered with HTML tags, scripting language instructions, navigation elements, templates and advertisements, which do not contribute to the actual content. To deal with this issue, we integrated \textit{boilerpipe}\(^5\), a JAVA library written by Kohlschutter et al. [17]. As with the extraction of Web pages from archives, the boilerplate removal is executed on the fly. Owing to both space and time restrictions for running LDA, we reduced the size of the vocabulary effectively. To this end, we decided to discard the most frequent as well as the rarest terms such that the first 500 most frequent terms are removed, then 100,000 subsequent terms are kept as our vocabulary, and the rest are removed. Term stemming and stop word removal are enabled during pre-processing for both corpora, as preliminary experiments have been shown to perform better in this setting.

For LDA topic modelling, we make use of the Machine Learning for Language Toolkit (MALLET), a Java-based suite of algorithms for statistical natural language processing written by McCallum [18]. The LDA algorithm learns \( K = 600 \) topics. The Dirichlet priors \( \alpha \) and \( \beta \) are set to \( \alpha = 50/K \) and \( \beta = 0.1 \), which are the common settings in the literature [19]. Different inference techniques have been applied to LDA successfully. Among them, Gibbs sampling is the most commonly used algorithm for sampling in topic modelling [20]. We also use this sampling inference by running 400 iterations. Since Gibbs sampling is a randomized algorithm to infer the posterior distribution, we use the samples from different Markov chains with different initialization and consider their average to calculate the probabilities.
7.2. Results on the TREC queries

The Text Retrieval Conference (TREC) has provided for the participants in the Web Tracks since 2009 lists of sample queries along with manually assigned structures. We chose the query set of the TREC 2010 Web Track, which includes 50 queries. Whereas the task for the contestants of the Web Track is to return a ranked result set of Web pages from the ClueWeb09 corpus that meet the information need of the respective query, we generate suggestions for every query that serve as alternative formulations of the expressed need. For each suggestion, we manually assess whether or not it is related to the query and is likely to support the user’s search.

After running numerous experiments, our lists contain for the 50 queries 672 relevant and 2141 irrelevant suggestions, a total of 2813 assessments. The combined lists of nearly 3000 assessments are part of the contribution of this paper.

Our suggestion generation technique is made up of two components: the word- and the topic-based probability estimations of $P(\sigma | M_d)$ as well as their interpolation. The interpolation of word- and topic-based estimations needs to be regulated by configuring parameter $\lambda \in [0, 1]$. As mentioned in Section 6, setting $\lambda$ to 1 reduces our suggestion generation technique to the purely word-based approach. In contrast, setting $\lambda$ to 0 generates suggestions exclusively based on topical associations. We expect that, when choosing $\lambda$ larger than 0 but smaller than 1, the respective strengths of the two approaches compensate each other’s weaknesses.

Table 2(a) shows the results of our experiments with varying values for $\lambda$ in which precision-at-10 ($P@10$) is served as evaluation metric. As expected, the best performance is achieved when both word- and topic-based calculations contribute to the estimation of $P(\sigma | M_d)$. Topic-based query suggestion alone ($\lambda = 0$) achieves the worst performance ($P@10$: 0.182). We ascribe this to the limitations discussed in Section 6. The purely word-based approach ($\lambda = 1$) achieves already quite high precision values ($P@10$: 0.47). However, we find that a small incorporation of topic-based estimation improves the results even further ($P@10$ for $\lambda = 0.9$: 0.492). The best performance of 0.51 is achieved for $\lambda = 0.5$; that is, on average, five out of 10 suggestions generated for the TREC 2010 Web Track queries are judged relevant.

As presented in the table, when decreasing the value of $\lambda$ from 0.1 to 0, a sharp decline in performance can be observed. Thus, we conduct additional experiments with values of $\lambda$ in this particular range. The results in Table 2(b) show that the precision values decrease steadily with smaller $\lambda$ until the lowest score is reached. Apparently, for $\lambda < 0.0005$, the influence of the word-based approach is too small to make a difference in suggestion quality.

In addition to $\lambda$, another open parameter in our experiments is $k$, which represents the desired number of suggestions. As mentioned, the above results are based on $P@10$, which means that we are willing to return 10 suggestions for each issued query. In another step of the experiments, we reduce the number of suggestions from 10 to 5. The results are presented in Table 3. As can be seen in the tabulated results, we can improve the suggestion quality even further. This indicates that we not only generate useful suggestions but also are successful in ranking the most relevant suggestions first.

7.3. Results on the search shortcut similarity

The evaluation of the 50 TREC queries has helped demonstrate the workings of our interpolated suggestion technique. We now want to evaluate its generalization capability. For this, we would like to use a dataset that is considerably larger than the Web Track query set. The compilation of a gold standard similar to the one created for the TREC queries...
Table 3. P@10 vs P@5

<table>
<thead>
<tr>
<th>K</th>
<th>P@k</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.552</td>
</tr>
<tr>
<td>10</td>
<td>0.510</td>
</tr>
</tbody>
</table>

would, again, require manual assessment, which is no longer feasible for larger quantities of test data. We thus decided to employ an automatic measurement of suggestion quality instead. To this aim, we follow the search shortcut similarity metric proposed by Baraglia et al. [13].

The idea of this metric is to support a user’s search by suggesting shortcuts, that is, queries that were used successfully in the past by (other) users with a similar search process. By creating virtual documents from successful query sessions and suggesting their titles to the user, we do exactly that. When the queries in the current session of a user are related to the contents of a virtual document, which are the combined successful sessions or search processes of several past users, we propose the document title as a shortcut. A shortcut is supposed to predict the positive ending of the current user’s session given the beginning of that session. It does not necessarily have to anticipate the final, successful query. A shortcut is deemed helpful as soon as it reduces the potential length of the user’s session. However, the more intermediate queries can potentially be skipped, the larger this reduction is and the more important the shortcut. The intuition behind this metric is the following: shortcuts generated for the head of the session that predict the tail of the session contribute positively to the similarity score. Baraglia et al. [13] build on this intuition and propose an evaluation measure based on the similarity between the tail of a session and the suggestions generated for its head as follows:

\[
s(h(\sigma_i), \sigma_f) = \frac{\sum_{q \in h(\sigma_i)} \sum_{m=1}^{n-t} [q = (\sigma_f)_m] f(m)}{|h(\sigma_i)|}
\]

where \( \sigma = <q_1, q_2, \ldots, q_n> \) is a successful session and \( \sigma_f = <q_1, q_2, \ldots, q_t> \) denotes its head, which is the sequence of its first \( t (t \leq n) \) queries. The tail \( \sigma_j \) of a session \( \sigma \) consists accordingly of the last \( n - t \) queries in the session, that is, \( \sigma_j = <q_{t+1}, q_{t+2}, \ldots, q_n> \). \( h(\cdot) \) is a function that generates \( k \) query shortcuts for a given session. Furthermore, \( f(\cdot) \) is a monotonically increasing function. The function \( |q = \sigma_m| \) yields 1 if and only if query \( q \) is equal to query \( \sigma_m \) and 0, otherwise. This strict equality in the function can be relaxed by replacing it with a similarity relation such as Levenshtein distance or Jaccard index.

To perform this evaluation, we randomly split the AOL query log into training (99\%) and test data (1\%). The training data is used, as before, to construct the virtual corpus. The remaining test data includes 9844 query sessions, resulting in nearly 200 times more test cases than for the TREC query set. We vary the value of \( \lambda \) in the search shortcut similarity metric. Since we want to calculate the prediction ability of each session heads toward the tail of that session, we set the length of a session head to be \( n/2 \), where \( n \) is the total length of that session. As the weighting function, we choose the linear function \( f(m) = m \), thereby giving increasing importance to suggestions that predict the end of the tail. We choose the Jaccard index over sets of trigrams with a threshold of 0.7 as the similarity measure.

Table 4 shows the average search shortcut similarity score of the 9844 test sessions. As has been the case in the TREC evaluation, the interpolation of topic-based and word-based suggestion generation significantly outperforms the purely topic-based approach as well as the purely word-based approach.

Interestingly, the degree of interpolation that achieves the best performance is not \( \lambda = 0.5 \), as in Section 7.2, but \( \lambda = 0.2 \). We find three possible explanations for this observation: firstly, the TREC Web Track query set might include too few queries to determine the best degree of interpolation. Secondly, the different nature of the test queries (one set includes hand-selected queries by TREC assessors, the other the raw queries of a commercial search engine) might indicate the necessity to fine-tune parameters to the nature of queries that are expected to be issued in future. Thirdly, the search shortcut similarity metric evaluates different aspects of a suggestion generation technique than the \( P@k \) evaluation that has been conducted for the TREC queries. It rewards not only the suggestion of relevant queries but also the suggestion of queries that already help to shorten the search process.

The interpolation with \( \lambda = 0.2 \) achieves a 10.4\% increase in the average similarity score compared with the non-interpolated baseline with \( \lambda = 1 \). This difference is statistically significant as assessed by one-tailed, paired \( t \)-test \( (p \approx 0.0035 < 0.01) \). Moreover, it is worth mentioning that all other results for different values of \( \lambda \) from 0.1 to 0.9 also significantly outperform the baseline word-based model, which indicates the superiority of our model disregarding any
parameter optimization. Thus, we can conclude that the interpolated suggestion generation technique proposed in this paper generalizes well to unseen test data and significantly outperforms the baseline word-based approach.

8. Concluding remarks

In this paper, we proposed a new model for query suggestion generation that exploits latent semantics in a query log. Following Broccolo et al. [9], we have taken on a document-centric view of the query log, which has allowed us to adapt well-proven models from the field of information retrieval to the task of query suggestion. At the same time, we have identified a severe shortcoming shared by all those models that rely on strict word-matching: their limited capability to handle the full complexity of natural language. This limitation has motivated us to strive to reveal higher-level semantic structures hidden within and across queries and entire query sessions. For this purpose, we have employed LDA and learned a topic model from a large, information-rich Web corpus. The expressive power of the model has enabled us to infer the topical structures of the documents created from the query log as well as the topical structures of queries for which suggestions are to be generated.

With the foundations laid out, we have, then, developed an interpolated query suggestion technique that benefits simultaneously from the strengths of both word- and topic-based suggestion generations. Our adaptation of the language modelling framework has facilitated the seamless interpolation of both approaches. In experiments, we have been able to demonstrate the statistically significant superiority of the interpolated suggestion technique over a purely word-based approach and its capability to generalize to previously unseen data. We can, thus, conclude that exploiting latent semantics in query logs allows us to effectively support search engine users in formulating their information needs more accurately and, thereby, ultimately help them find (faster) what they are searching for.

Acknowledgements

The authors would like to thank Professor Felix Naumann for interesting discussions and valuable feedback during the project.

Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Notes

1. The queries of the AOL log have already been transformed to lowercase letters. To make our implementation general, we nevertheless provide this functionality for other query logs that may have undergone no such pre-processing.
2. This training corpus has to be pre-processed in the exact same manner as the virtual documents previously in Step 2. Depending on the choice of the external corpus, further pre-processing might be necessary as well.
3. The queries available at: http://trec.nist.gov/data/web/10/wt2010-topics.xml
5. https://code.google.com/p/boilerpipe
6. The queries available at: http://trec.nist.gov/data/web/10/wt2010-topics.xml

Table 4. Average search shortcut similarity score for varying values of $\lambda$.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Search shortcut similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4388</td>
</tr>
<tr>
<td>0.1</td>
<td>0.7954</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8249</td>
</tr>
<tr>
<td>0.3</td>
<td>0.8198</td>
</tr>
<tr>
<td>0.4</td>
<td>0.8066</td>
</tr>
<tr>
<td>0.5</td>
<td>0.8076</td>
</tr>
<tr>
<td>0.6</td>
<td>0.7893</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7832</td>
</tr>
<tr>
<td>0.8</td>
<td>0.7781</td>
</tr>
<tr>
<td>0.9</td>
<td>0.7792</td>
</tr>
<tr>
<td>1.0</td>
<td>0.7456</td>
</tr>
</tbody>
</table>
References


