Chapter 8
Query Suggestion

Zhen Liao, Yang Song, and Dengyong Zhou

Abstract Query suggestion is one of the few fundamental problems in Web search. It assists users to refine queries in order to satisfy their information needs. Many query suggestion techniques have been proposed in the past decades. The mainstream idea is to leverage query logs which contain the search behaviors of users to generate useful query suggestions. In this chapter, we introduce several log-based query suggestion techniques. These methods fall into four categories: (1) query co-occurrence; (2) query-URL bipartite graph; (3) query transition graph; and (4) short-term search context. We also briefly discuss other related work in this field and point out several future directions.

8.1 Introduction

8.1.1 An Overview of Query Suggestion Approaches

How effectively users are able to retrieve information from the Web largely depends on whether they can formulate input queries properly to express their information needs. However, formulating effective queries is never meant to be an easy task. On the one hand, given the same query, different search engines may return different results. This means that it is unlikely to define a single standard to guide query formulation across different search engines. On the other hand, queries are typically expressed in just a few words [11, 20, 22], which potentially increases the difficulty for search engines to understand query intents.

Z. Liao
Facebook Inc., Menlo Park, CA, USA
e-mail: zhangzliao@fb.com

Y. Song (✉) · D. Zhou
Google Research, Seattle, WA, USA
e-mail: ys@sonyis.me; dennyzhou@google.com

Y. Chang, H. Deng (eds.), *Query Understanding for Search Engines*,
The Information Retrieval Series 46, https://doi.org/10.1007/978-3-030-58334-7_8

Most commercial search engines, including Google,[1] Yahoo![2], and Bing[3] provide query suggestions on their search result pages to help user formulating queries. A recent study [28] shows that query suggestions are particularly useful in the following scenarios: (1) the original query is a rare query; (2) the original query consists of only one word; (3) the suggested queries are unambiguous; and (4) the suggested queries are generalizations or error corrections of the original query. Based on the study in [15], around 30% of searches in commercial search engines are generated from query suggestions.

Studies on query suggestions can be traced back to the early years of this century [4, 20, 46]. Since then, many techniques [2, 3, 8, 10, 18, 21, 23, 26, 29–32, 36, 40–42, 47] have been proposed to improve the quality of query suggestions. Roughly speaking, query suggestions have the following major objectives [35]: (1) when a user's information need is not satisfied, the search results from the suggestions should provide more relevant results or (2) when a user's information need is satisfied but the user wants to explore more, the suggestions can provide useful guideline to obtain related information.

Search engine logs contain information on how users refine their queries as well as how users click on suggested queries, which can help address both of the aforementioned objectives. As a result, most query suggestion techniques leverage search logs as a useful source of information.

Formally, given a query q, query suggestion aims at optimizing a scoring function $f(q, q') \in \mathbb{R}$ that can be used to rank suggestion candidates q'. To include short-term search context in query suggestion, the relevance function $f(q, q')$ can also be extended as $f(q_{1,\ldots,i}, q')$, where $q_{1,\ldots,i} = \{q_1, \ldots, q_i\}$ $(i \geq 1)$ represents the previous search sequence.

8.1.2 Examples of Query Suggestion Approaches

As we mentioned above, one of the most important and effective query suggestion techniques leverages query logs [3, 8, 10, 18, 21, 29–32, 36, 40–42, 46]. Query logs record user interactions with search engines. A typical query log entry contains *timestamp*, *query*, *clicked URL* as well as other information (e.g., anonymous user ID, search platform, etc.). In contrast, suggestion methods that do not use query logs [5] often generate candidates from external data sources. Those approaches do not consider the fact that the text used in formulating search queries is usually quite different from text in external sources (e.g., typos, acronym, no grammar, etc.). Thus, they are less effective in practice.

[1] http://www.google.com.

[2] http://www.yahoo.com.

[3] http://www.bing.com.

From the perspective of modeling and organizing search logs, query suggestion techniques can be categorized into four classes: (1) query co-occurrence; (2) query-URL bipartite graph; (3) query transition graph; and (4) short-term search context methods.

Query co-occurrence methods utilize the query co-occurrence information to provide suggestions. Co-occurrence is often computed from search sessions [20] or tasks [30], where the relevance functions range from simple raw counts to statistical methods like log likelihood ratio (LLR) [25].

Query-URL bipartite graph methods leverage clicks on URLs. These methods often represent queries and URLs into bipartite graphs with the edges indicating the click information. Graph traversal methods like random walks are often employed to estimate the similarities between queries. Examples include random walk with restart [40, 44], forward and backward random walks [4, 10], hitting time [36], etc.

Query transition graph methods model the query refinement process in the search sequence by constructing query transition graphs where edges on the graph indicate the reformulation relationships between queries. Examples in this category include query flow graph (QFG) [7], term transition graph (TTG) [42], etc.

Short-term search context methods focus on leveraging immediate previous queries as contextual information to model and disambiguate the current input query. Typical methods in this category are based on decay factors [7, 20], query clustering [8, 29], Markov models [9, 18, 31], etc.

Besides the classical query suggestion methods which mainly rely on a single data source, other studies proposed to combine different data sources for generating suggestion candidates through various strategies (e.g., machine learning for query suggestion candidates ranking [38, 42], query suggestion diversification [34, 41], query suggestions personalization [24], etc.). There are also approaches to build better visualization [48] or user interface [27] for query suggestions.

8.1.3 Evaluation Metrics for Query Suggestion

The evaluation metrics for query suggestions can be categorized into offline (e.g., precision, recall) and online (e.g., click-through rate) approaches.

For offline evaluation, previous work often leverage a small number of case studies [4, 46], while recent methods focus more on leveraging human assessors [8, 29, 40, 42]. Examples of metrics in this category include Precision [8], Mean Average Precision (MAP) [40], Normalized Discounted Cumulative Gain (NDCG) [42], and Mean Reciprocal Rank (MRR) [1].

Formally, given a binary label $r(i) \in \{0, 1\}$ indicating whether a suggestion ranked at position-i is relevant (1) or not (0), precision at position K is defined as:

$$Precision@K = \frac{\sum_{i=1}^{K} r(i)}{K}. \tag{8.1}$$

Similarly, we can also define the recall at position K as:

$$Recall@K = \frac{\sum_{i=1}^{K} r(i)}{M}, \tag{8.2}$$

where M is the total number of relevant suggestions. Comparing to precision, recall is rarely used since it is nearly impossible to get all relevant suggestions for a query.

Instead of computing recall, coverage is often used as an alternative [29]:

$$Coverage = \frac{\text{\# of testing queries with suggestions}}{\text{\# of testing queries}}. \tag{8.3}$$

MAP is defined as the mean of the average precision (AP) of all suggested queries:

$$AP = \frac{1}{M} \cdot \sum_{i=1} \frac{Precision@i \cdot r(i)}{i}. \tag{8.4}$$

NDCG at position K is defined based on $DCG@K = \sum_{i=1}^{K} \frac{2^{r(i)}-1}{log_2(i+1)}$:

$$NDCG@K = \frac{1}{Z_K} \sum_{i=1}^{K} DCG@K, \tag{8.5}$$

where Z_K is the normalized factor of $DCG@K$ which corresponds to the ideal ranking results.

MRR is defined as:

$$MRR = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{rank_q}, \tag{8.6}$$

where Q is the number of testing queries in the evaluation dataset and $rank_q$ is the rank of first relevant query in the suggestion list for a testing query q.

For online evaluation, click-through rate (CTR) is widely used, which is defined as [38]:

$$CTR@K = \frac{\text{\# of clicks at top-}K\text{ suggestions}}{\text{\# of impression with at least }K\text{ suggestions}}. \tag{8.7}$$

Due to the difficulty of reproducing all methods on a standard evaluation dataset for comparable results, in this chapter we do not emphasize on the evaluation metrics comparison among different methods. In addition, it is hard to compare different query suggestion techniques while they are proposed in different scenarios (e.g., for Web documents search, image search, or sponsored search) or optimizing different

metrics (e.g., coverage, diversity, etc.). Therefore, we focus on the motivation and mathematical formulation of these methods. For effectiveness comparison, we provide illustrative examples to show the differences. Readers can refer to the original publications if they are interested in the detailed comparison of metrics.

8.1.4 Notation Used in This Chapter

Table 8.1 lists notations with detailed meanings in this chapter.

8.1.5 Structure of This Chapter

In the rest of this chapter, we introduce several query suggestion techniques in Sects. 8.2–8.5 which correspond to co-occurrence, query-URL bipartite graph,

Table 8.1 Notations used in this chapter

Meaning	Notation
Query	q, q_i, q_j
Search sequence with last query as q_i	$q_{1,\ldots,i}$
Query suggestion candidate	q'
URL	u, u_j, u_x
Set of queries, URLs	Q, U
Number of queries, URLs	$\|Q\|$ or N_q, $\|U\|$ or N_u,
Count/frequency	$Cnt(\cdot)$
Frequency of query in sessions	f_i, f_j
Query co-occurrence matrix	$\mathbf{C}$
Co-occurrence between q_i and q_j	$\mathbf{C}_{ij}$
Query-URL click matrix	$\mathbf{B}$
Click frequency of q_i on u_j	$\mathbf{B}_{ij}$
Query transition probability matrix	$\mathbf{A}$
Transition probability from q_i to q_j	$\mathbf{A}_{ij}$
One-hot vector of query q_i	$\mathbf{v}_i^0$
Final optimized suggestion results	$\mathbf{v}^*, \mathbf{v}_i^*, h_i^*$ (with *)
Number of iterations	t
Re-start probability for forward random walk	α
Self-transition probability for backward random walk	s
Terms in query	w, w_i
Search topic of a query	T
Decay factor for short-term search context	β
Hidden search state for a search sequence	z, z_{i+1}

query transition graph, and short-term search context methods, respectively. After that, we summarize other related suggestion techniques as well as evaluation studies in Sect. 8.6. In Sect. 8.7 we conclude this chapter with discussions and future directions.

8.2 Query Co-occurrence Methods

In this section, we introduce several widely used methods that compute the similarity between queries by leveraging their co-occurrence information from search logs. Given a sequence of queries $\{q_1, \ldots, q_n\}$, traditional approaches [8, 18, 29, 39] defined search sessions to segment search logs. Specifically, consecutive events are segmented into different sessions if the time interval between them exceeds a certain threshold (e.g., 30 min). Within each session, different similarity functions can be defined to find similar queries [25, 30, 33].

8.2.1 Similarity Functions

Let $\mathbf{C}$ denote a co-occurrence matrix where $\mathbf{C}_{ij}$ indicates the co-occurrence count between query q_i and q_j. Let $f_i = \sum_j \mathbf{C}_{ij}$ denote the total number of sessions that contain query q_i. Depending on the scenarios, $\mathbf{C}$ can be either symmetric [30] or asymmetric [16, 18], where the symmetric way ignores the issuing order of queries, while the asymmetric way considers the issuing order of queries. Specifically, asymmetric $\mathbf{C}$ defines $\mathbf{C}_{ij} = Cnt(q_i \rightarrow q_j)$ and $q_i \rightarrow q_j$ denotes q_j occurring after q_i.

Below are some examples of co-occurrence methods proposed in [20]:

$$Jaccard(q_i, q_j) = \frac{\mathbf{C}_{ij}}{f_i + f_j - \mathbf{C}_{ij}} \tag{8.8}$$

$$Dependence(q_i, q_j) = \frac{\mathbf{C}_{ij}}{\min(f_i, f_j)} \tag{8.9}$$

$$Cosine(q_i, q_j) = \frac{\sum_k \mathbf{C}_{ik} \cdot \mathbf{C}_{jk}}{\sqrt{\sum_k \mathbf{C}_{ik}^2} \cdot \sqrt{\sum_k \mathbf{C}_{jk}^2}}. \tag{8.10}$$

Both Jaccard and Dependence functions define the relative co-occurrence between q_i and q_j, which tends to favor popular queries. The Cosine function tries to address this bias by adding an L_2 normalization on query frequencies.

From probabilistic perspective, we can define the probability of issuing q_j after q_i as:

$$P(q_j|q_i) = \frac{\mathbf{C}_{ij}}{f_i} \propto \mathbf{C}_{ij}. \tag{8.11}$$

Since the denominator f_i is independent of q_j, $P(q_j|q_i)$ is in favor of popular queries. To address this issue, we can leverage the pointwise mutual information (PMI) and mutual information (MI) [23, 38]:

$$PMI(q_i, q_j) = \log \frac{P(q_i, q_j)}{P(q_i) \cdot P(q_j)} \propto \frac{\mathbf{C}_{ij}}{f_i \cdot f_j}, \tag{8.12}$$

$$\begin{aligned} MI(q_i, q_j) = \; & P(q_i, q_j) \cdot PMI(q_i, q_j) + P(q_i, \overline{q_j}) \cdot PMI(q_i, \overline{q_j}) \\ & + P(\overline{q_i}, q_j) \cdot PMI(\overline{q_i}, q_j) + P(\overline{q_i}, \overline{q_j}) \cdot PMI(\overline{q_i}, \overline{q_j}). \end{aligned} \tag{8.13}$$

Here $\overline{q_i}$ denotes all queries in the search logs except q_i, and $P(q_i, q_j) = \frac{\mathbf{C}_{ij}}{\sum_{ij} \mathbf{C}_{ij}}$.

Jones et al. [26] leveraged LLR [13] to measure the degree of correlation between queries q_i and q_j. Their method makes the null hypothesis that $H_1 : P(q_j|q_i) = P(q_j|\overline{q_i})$ and the alternative hypothesis that $H_2 : P(q_j|q_i) \neq P(q_j|\overline{q_i})$. The LLR function is defined as the log ratio of the likelihood between H_1 and H_2:

$$LLR(q_i, q_j) = -2 \cdot \log \lambda = -2 \cdot \log \frac{L(H_1)}{L(H_2)}, \tag{8.14}$$

where a higher LLR score indicates a stronger correlation between q_i and q_j. Using the notation above, $LLR(q_i, q_j)$ is defined as:

$$LLR(q_i, q_j) = -2 \cdot \{L_{h1}(k_1, n_1) + L_{h1}(k_2, n_2) - L_{h2}(k_1, n_1) - L_{h2}(k_2, n_2)\}, \tag{8.15}$$

where $L_{h1}(k, n) = \log\{k \cdot \log \frac{k_1+k_2}{n_1+n_2} + (n-k) \cdot \log(1 - \frac{k_1+k_2}{n_1+n_2})\}$, $L_{h2}(k, n) = \log\{k \cdot \log \frac{k}{n} + (n-k) \cdot \log(1 - \frac{k}{n})\}$, and $k_1 = \mathbf{C}_{ij}, k_2 = \mathbf{C}_{\overline{i},j}, n_1 = \sum_j \mathbf{C}_{ij}, n_2 = \sum_j \mathbf{C}_{\overline{i}j}$.

In [38], the authors have shown that MI and LLR are mathematically similar in evaluating query correlations.

8.2.2 *Extracting Tasks from Sessions*

Using the co-occurrence information to define the query similarity function highly relies on the segmentation of query sequences. As we described before, *session* has been widely used to extract co-occurrence in existing work. However, time-based

segmentation can possibly lose the inner correlation among queries that span longer period of time than a single session. Therefore, a concept of *task* is proposed [25, 30, 33, 45] to address this issue. Below we introduce the approach in [30] to extract tasks from sessions. In the common definition, task is defined as an atomic user information need [25, 30, 33].

The motivation of task extraction can be illustrated from the example shown in Table 8.2, which is a real user search session from search engine Bing. The user began this session with query "facebook" and finished the session with several attempts to search for lyrics of a song. From the table, we can see that one session may contain multiple or interleaved tasks. The reasons behind that are: (1) web search logs are ordered chronologically; (2) users often perform multiple tasks at the same time. On the one hand, treating the entire session as an atomic unit may not accurately capture the multi-tasking behavior. As shown in Table 8.2, query "gmail log in" seems to have no correlation with its adjacent queries. Besides, failing in searching for lyrics of a song does not mean that the user did not find useful information for query "facebook." On the other hand, dividing sessions at query level may lose information of reformulation by users. For example, in Table 8.2, even if the user had no click on query "amazon", he still managed to find relevant information by reformulating "amazon" into "amazon kindle books" and made a click. From the study of [30], about 30% of sessions contain multiple tasks and about 5% of sessions contain interleaved tasks.

To extract tasks from sessions, Liao et al. [30] proposed the following approach. First, the similarity between queries is learnt from a binary classifier; Second, queries within a session are grouped into tasks using a clustering algorithm. This approach is motivated by [25, 33], where Jones and Klinkner [25] proposed to classify queries into tasks using a binary classification approach, and Lucchese et

Table 8.2 An example of session in web search logs from [30]

Time	Event type	Detailed entry information	User ID	Session ID	Task ID
09:03:26 AM	Query	Facebook	U1	S1	T1
09:03:39 AM	Click	www.facebook.com	U1	S1	T1
09:06:34 AM	Query	Amazon	U1	S1	T2
09:07:48 AM	Query	faecbook.com	U1	S1	T1
09:08:02 AM	Click	facebook.com/login.php	U1	S1	T1
09:10:23 AM	Query	Amazon kindle	U1	S1	T2
09:10:31 AM	Click	kindle.amazon.com	U1	S1	T2
09:13:13 AM	Query	Gmail log in	U1	S1	T3
09:13:19 AM	Click	mail.google.com/mail	U1	S1	T3
09:15:39 AM	Query	Amazon kindle books	U1	S1	T2
09:15:47 AM	Click	amazon.com/Kindle-eBooks?b=.....	U1	S1	T2
09:17:51 AM	Query	i'm picking up stones	U1	S1	T4
09:18:54 AM	Query	i'm picking up stones lyrics	U1	S1	T4
09:19:28 AM	Query	pickin' up stones lyrics	U1	S1	T4

Table 8.3 Basic statistics of browse and search logs reported in [30]

Statistics	Browse logs	Search logs
Avg. # of queries in sessions	5.81	2.54
Avg. # of queries in tasks	2.06	1.60
Avg. # of tasks in sessions	2.82	1.58
% of single-task sessions	53.29	70.72
% of multi-task sessions	46.71	29.28
% of interleaved task sessions	15.25	4.78
% of single-query tasks	48.75	71.86
% of multi-query tasks	51.24	28.13

Table 8.4 Query refinement pattern within tasks from browse and search logs in [30]

Reformulation patterns	Browse logs	Search logs
% of identical	66.37	50.45
% of shorter	12.48	16.77
% of longer	21.45	32.76

al. [33] proposed to cluster queries into tasks based on empirically designed distance functions.

Specifically, a similarity function between queries $sim(q, q')$ can be learnt through features from submitting time, textual similarity (e.g., edit distance, word similarity), result set (e.g., similarity between search engine result pages (SERPs) of q and q'), etc. Next, a graph can be constructed with queries as nodes and $sim(q, q')$ as the weight of an edge. With the constructed graph, graph cutting methods can be used to group queries into tasks. In [30], the authors applied an SVM classifier to learn $sim(q, q')$ and proposed a heuristic based query task clustering (QTC) algorithm to group queries into tasks.

Table 8.3 shows the statistics regarding query distribution as in tasks and sessions reported in [30]. From the table we can observe that multi-tasking behavior is quite common in users' searches. For consecutive queries within a task, Table 8.4 presents their length distribution from the previous query to its next query. More than half adjacent query pairs are identical, where about 90% of identical pairs are from refreshing search result pages or clicking the back button, and about 10% of identical patterns are from pagination. Besides, we can see that longer reformulation pattern occurs twice more often than shorter reformulation pattern. These statistics indicate that it is more effective to recommend longer and more specific queries than queries that are more general and have fewer words.

8.2.3 Method Analysis and Comparison

To better understand the difference between co-occurrence and LLR methods, a few examples are shown in Table 8.5. As we can see, using purely frequency-based method (e.g., Jaccard in Eq. (8.8) and $P(q_j|q_i)$ in Eq. (8.11)), those most popular

Table 8.5 Suggestions of session-based models

	Methods	
Test cases	Session co-occur	Session LLR
Amazon	Facebook	eBay
	eBay	Walmart
	Google	Target
	Youtube	Best buy
	Yahoo	Barnes and nobel
Cell phone	Facebook	Cheap cell phones
	Verizon wireless	Phone
	Sprint	All cell phone companies
	Verizon	Verizon cell phones
	Google	Sprint

queries like "facebook" and "google" are always recommended. As a comparison, LLR addresses the bias systematically.

Table 8.6 presents several queries from high, medium, and low frequency categories with their suggestions. From suggestions generated by different methods, we have the following observations. (1) Session-based models often generate related queries in a broad range such as providing "verizon" as a suggestion to query "att." (2) For low-frequency queries, task-based and session-based methods generate nearly same suggestions. (3) Task-based methods often generate more specific queries for further narrowing down user information need, which are different from session-based approach. As a result, suggestions provided by task-based methods can be treated as complementary to results from session-based approaches.

8.2.4 Summary

In this section, we described co-occurrence based query suggestion methods. Simple co-occurrence based approaches have a frequency bias towards popular queries. We saw that methods like MI or LLR can help address the issue systematically. In general, the quality of query suggestions based on LLR tends to be better than other co-occurrence based approaches.

The essential point of co-occurrence based method is to define query similarity based on co-occurrence. Most existing works are session-based, where sessions are segmented based on the timestamp between consecutive queries. Due to the nature of multi-tasking searching behavior by search engine users, extracting tasks from session is useful to generate related queries from the same search task. As illustrated in Table 8.6, task-based methods tend to be complementary to session-based methods. Feild and Allan [14] also studied the task-aware query recommendation problem and show that queries from the same search task are useful as context for

Table 8.6 Example of query suggestions provided by different methods [30]. Superscripts h, m, l are notations for high, medium, and low frequency queries

	Methods	
Test case	Session LLR	Task LLR
Amazon^h	eBay	Amazon books
	Walmart	Amazon kindle
	Target	Amazon electronics
	Best buy	Amazon music
	Barnes and nobel	Amazon DVD movies
ATT^h	AT&T my account	AT&T my account
	Verizon	ATT wireless
	Sprint	AT&T email
	Tmobile	AT&T bill pay
	ATT wireless	AT&T customer service
Exchange^m	Military exchange	Military exchange
	Exchange rate	Exchange rates
	Easyfreexbox360	Navy exchange
	Tennis	Microsoft exchange
	Aafes	Base exchange
Harry Truman^m	Winston Churchill	Harry Truman quotes
	Robert Byrd	Bess Truman
	Nelson Mandela	Harry Truman facts
	Neil Armstrong	Harry S Truman
	Teddy Roosevelt	
"Popular Irish baby names"^{l}	Top Irish baby names	Unique Irish baby names
	Unique Irish baby names	Irish baby names
	Irish baby boy names	Irish baby boy names
	Irish baby names	Top Irish baby names
	"Traditional Irish baby names"	Top 100 baby names

query suggestion. There is also task extraction across multi-sessions [45], which can be used to generate cross session query suggestions.

The co-occurrence based approaches usually work well for high and medium frequency queries and perform poorly in low-frequency queries. To help generating good suggestion for low-frequency queries, graph-based approaches are preferred. The idea of graph-based methods is to construct a graph with nodes as queries and edges as similarities between queries and leverage the entire graph to help finding relevant queries. Sections 8.3 and 8.4 describe graph-based methods using query-URL bipartite graph and query transition graph, respectively.

8.3 Query-URL Bipartite Graph Methods

Although the click information on SERP URLs are often noisy, aggregating clicks from a large number of users tends to reflect the relevance between queries and URLs. Such rich query-URL relevance information can be used for generating high quality query suggestions. As an example, the co-occurrence based method may fail to generate suggestion for a tail (typo) query "faecboek." If we can leverage the top clicked URLs on the SERP of the query, it is likely to generate relevant suggestions. In practice, such approach can help address the issues on tail queries that lack enough co-occurrence information.

Typically, query-URL bipartite graph-based methods use clicks from queries on URLs as signals. They usually work as follows. First, a probabilistic matrix is constructed using click counts. Next, a starting node (i.e., a test query) is chosen. Third, a random walk (RW) is performed on the graph using the probabilistic matrix. Forth, final suggestion is generated using RW results.

Let $\mathbf{B}$ denote the matrix derived from the query-URL click-through bipartite graph, where

$$\mathbf{B}_{ij} = Cnt(q_i, u_j). \tag{8.16}$$

Here $Cnt(q_i, u_j)$ represents the click count of query q_i on URL u_j. An alternative method using inverse query frequency (IQF) to initialize $\mathbf{B}_{ij}$ was proposed by Deng et al. [12]:

$$\mathbf{B}_{ij} = Cnt(q_i, u_j) \cdot IQF(u_j), \tag{8.17}$$

where $IQF(u_j) = \log \frac{|Q|}{n(u_j)}$ and $n(u_j)$ is the number of distinct queries clicking on u_j. It is suggested in [12] to apply the IQF to re-weight click counts, which decreases the weight of frequently clicked URLs and increases the weight of less frequent but more relevant URLs.

By normalizing the rows of $\mathbf{B}$, we can get the transition probability from query q_i into url u_j using $P(u_j|q_i) = \frac{\mathbf{B}_{ij}}{\sum_k \mathbf{B}_{ik}}$. Similarly, we can derive the transition probability from url u_j to query q_i using $P(q_i|u_j) = \frac{\mathbf{B}_{ij}}{\sum_k \mathbf{B}_{kj}}$. Based on these probabilities, we can derive the transition probability from query q_i to q_j as:

$$P^u(q_j|q_i) = \sum_u P(q_j|u) \cdot P(u|q_i). \tag{8.18}$$

8.3.1 Forward and Backward Random Walks

Let matrix $\mathbf{A}$ represent the transition matrix derived from the Query-URL click graph, where $\mathbf{A}_{ij} = P^u(q_j|q_i)$. The forward random walk with restart approach (RWR) is formulated as [30, 40]:

$$\mathbf{v}_i^{t+1} = (1-\alpha) \cdot (\mathbf{v}^t)^T \cdot \mathbf{A} + \alpha \cdot \mathbf{v}_i^0, \tag{8.19}$$

where α is the restarting probability. $\mathbf{v}_i^0$ is the initialized one-hot vector for query at index-i. t is the number of iteration.

If we set p to be 0, the process of iteration can be viewed as a Markov chain through the probabilistic matrix $\mathbf{A}$. According the Markov chain theory [37], if a Markov chain is irreducible and aperiodic, there exists a unique stationary distribution $\boldsymbol{\pi}$. Additionally, in this case $\mathbf{A}^k$ converges to a rank-one matrix in which each row is the stationary distribution $\boldsymbol{\pi}$, that is:

$$\lim_{k\to\infty} \mathbf{A}^k = \mathbf{1} \cdot \boldsymbol{\pi}, \tag{8.20}$$

which produces vector $\boldsymbol{\pi}$, where $\boldsymbol{\pi}_i$ can be interpreted as the popularity of query q_i.

Compared to forward propagation defined in Eq. (8.19), Craswell et al. [10] proposed a back propagation method, which leverages a back propagation matrix $\mathbf{A}^b$ defined as:

$$\mathbf{A}_{ij}^b = \begin{cases} (1-s) \cdot \mathbf{A}_{ij}, & \text{if i} \neq \text{j} \\ s, & \text{if i = j.} \end{cases} \tag{8.21}$$

Here s is a self-transition probability to keep the propagation stay on the current query.

Based on the matrix $\mathbf{A}^b$, the backward RW is computed by multiplying $\mathbf{A}^b$ with $\mathbf{v}_i(t)$, which is formulated as:

$$\mathbf{v}_i^{t+1} = norm(\mathbf{A}^b \cdot \mathbf{v}_i^t). \tag{8.22}$$

Here $norm(\cdot)$ denotes the normalization to make $\sum_k \mathbf{v}_i[k] = 1$.The basic idea of backward propagation is that given a query q_i at time t, we aim at finding the probability of starting from q_j at step 0 by using $P_{0|t}(q_j|q_i) = [(\mathbf{A}^b)^t \cdot \mathbf{Z}^{-1}]_{ij}$. Here $\mathbf{Z}$ is a diagonal matrix and $\mathbf{Z}_{jj} = \sum_i [(\mathbf{A}^b)^t]_{ij}$ is used for a row normalization purpose. To set up the parameters of s and t, from the experiment results shown in [10], a self-transition s=0.9 with step t=101 can result in a good performance in the application of image retrieval.

Equations (8.19) and (8.22) look similar but are different in nature. For example, let $\mathbf{v}_2(0)$=[0, 1]^T denote the starting vector and $\mathbf{A} = \mathbf{A}^b$ denote the transition matrix

between q_1 and q_2,

$$\mathbf{A} = \mathbf{A}^b = \begin{bmatrix} 0.7 & 0.3 \\ 0.6 & 0.4 \end{bmatrix}. \tag{8.23}$$

Then one step backward propagation gets $\mathbf{A} \cdot \mathbf{v}_2(0)=[0.3, 0.4]^T$, and one step forward propagation gets $\mathbf{v}_2(0)^T \cdot \mathbf{A}=[0.6, 0.4]^T$.

8.3.2 Hitting Time Approach

Both forward and backward propagations need to tune the parameters (e.g., restart probability α or self-transition probability s). Mei et al. [36] proposed a parameter-free method using hitting time. The hitting time $h_i[q_j]$ is defined as the expectation of arriving at q_j while starting at q_i. To compute the hitting time, Mei et al. [36] proposed an iterative process:

$$h_i^{t+1} = \sum_{j \neq s} P^u(q_j|q_i) \cdot h_j^t + 1, \tag{8.24}$$

where s denotes the index of a test query and $h_i(0)=0$. Here $P^u(q_j|q_i)$ is the same as in Eq. (8.18). After certain steps of iterations, the final h_i^{t+1} is used for the suggestion. Note that hitting time represents the expected arriving steps from a suggested query to the test query; therefore, a smaller value indicates a higher relevance. The iteration can stop with a given maximum number of step (e.g., 1000), or when the difference of $h_i^{t+1} - h_i^t$ becomes insignificant (e.g., less than 10^{-3}).

8.3.3 Combining Click and Skip Graphs

It has been shown that click graph can benefit popular queries which have enough user click feedbacks. However, using only click graph tends to ignore the relevant information presented on SERP which causes potential issues particularly for tail queries. For rare queries with very few clicks, click graph is unable to capture the underlying relationship among queries. Comparing with click graph, a skip graph which contains information of (query, skipped URL) pairs can enrich the information for tail queries with fewer clicks. Here a URL is skipped if it was viewed by the user without being clicked. For instance, if a user only clicked the 3rd-ranked URL after issuing the query, the 1st and 2nd ranked URLs are skipped.

Figure 8.1 presents an example which shows the motivation of the combination of click and skip graphs approach. The left figure (a) shows the click graph for three queries and five URLs that returned as top SERP results. Ideally, *audi parts*

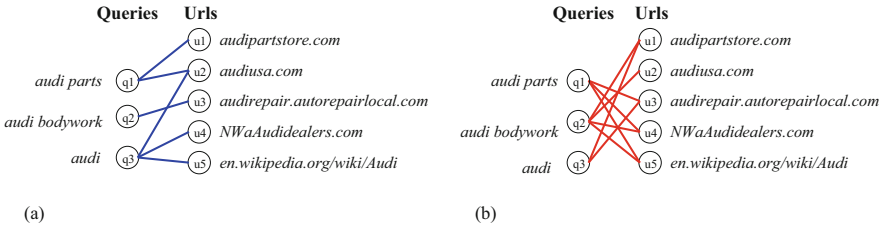

Fig. 8.1 An illustrative example of query-URL click graph (**a**) and skip graph (**b**). Query *audi parts* and *audi bodywork* are not correlated if only performs random walk on the click graph, but will be highly correlation if random walk is performed on the skip graph. More details on the text

should be a good suggested query for *audi bodywork* (and vice versa). However, after performing a random walk on the click graph, only the query *audi* can be suggested to *audi parts* because there is no commonly clicked URLs between *audi parts* and *audi bodywork* so that their correlation is zero. However, if we leverage the top-skipped URLs for *audi parts* and *audi bodywork* as shown in Fig. 8.1b, it can be clearly observed that both queries skipped their top-returned two URLs: *NwaAudidealers.com* and *en.wikipedia.org/wiki/Audi*. As a result, a random walk on the skip graph assigns a high correlation score to these two queries.

To show that skip graph contains rich information for tail queries, Fig. 8.2 shows user session statistics from a dataset with 40 million unique queries. The figure compares the query frequency (x-axis) against the number of clicked and skipped URLs (y-axis). It can be observed that when the query frequency is low, more URLs are skipped than clicked during the same user session. However, with the increase of query popularity, the click patterns become more stable. Generally, users tend to click more often on top-returned results for popular queries. While for rare queries, click distribution is more random.

For the quality of skipped URLs for rare queries, Song et al. [40] selected 6000 queries which have been issued less than 20 times within a week. They asked human raters to judge the relevance of clicked and skipped URLs on a 1–5 scale (5 means the best). Figure 8.3 demonstrates the comparative ratings. Overall, skipped URLs indicate slightly less relevance than clicked URLs. On average, clicked URLs have a rating of 3.78, while skipped URLs have 3.65. This observation further supports our claim that skipped URLs should be leveraged for rare queries in the context of relevance measurement.

Following the same notation as used previously, we define the query-to-query click transition matrix $\mathbf{A}^+$ using Eq. (8.16) or (8.17). Similarly, we can also define a query-to-query skip transition matrix $\mathbf{A}^-$ by replacing the click count as the skip count. Hence, we can conduct the random walk on both click graph with $\mathbf{A}^+$ and skip graph with $\mathbf{A}^-$ using Eq. (8.19) and generate the final suggestion vectors $\mathbf{v}_i^+$ and $\mathbf{v}_i^-$ for each graph. After that, we can combine both vector for final suggestions as:

$$\tilde{\mathbf{v}}_i^* = \alpha' \cdot \mathbf{v}_i^+ + (1 - \alpha') \cdot \mathbf{v}_i^-. \tag{8.25}$$

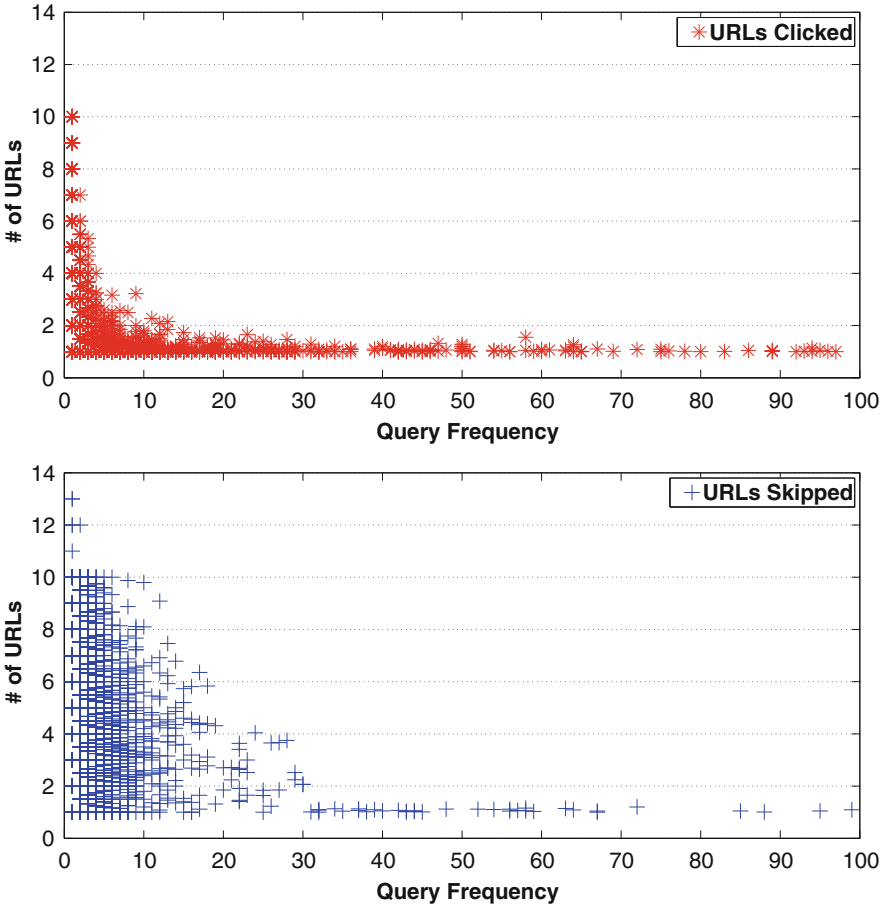

Fig. 8.2 Number of URLs clicked vs. number of URLs skipped in the same user sessions from 1 week search log. There are more URLs skipped than clicked for queries with lower frequencies

Together with restarting probability α, this approach has two parameters: α and α'. Cross validation can be used to tune the parameters to achieve the best results on held-out datasets.

We can construct a matrix $\mathbf{Q}^* = [\mathbf{v}_1^*, \ldots, \mathbf{v}_{|Q|}^*]$ as query similarity matrix where $|Q|$ is the total number of queries. A similar approach (e.g., compute URL transition probability $P(u_j|u_i) = \sum_q P(u_j|q) \cdot P(q|u_i)$ and conduct random walk on URL nodes) can be performed to get a URL similarity matrix $\mathbf{U}^* = [\mathbf{v}_1^*, \ldots, \mathbf{v}_{|U|}^*]$, where $|U|$ is the total number of URLs. Due to the difficulty of obtaining ground-truth for $\mathbf{Q}^*$ and $\mathbf{U}^*$, Song et al. [40] proposed to tune the parameters by minimizing the difference between URL correlation matrix $\mathbf{U}$ and $\mathbf{U}^*$ and apply same parameters for query suggestion.

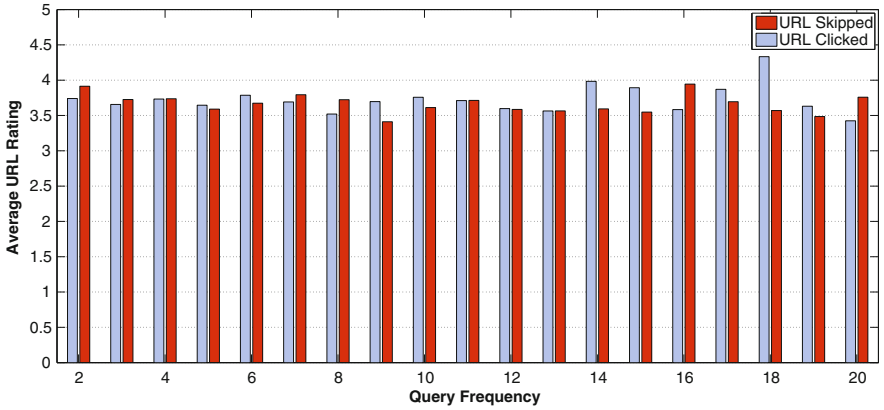

Fig. 8.3 Human judger ratings [40] in terms of relevance for clicked and skipped URLs in query logs. Break down accordingly to query frequency. Clicked URLs and skipped URLs have almost the same ratings for rare queries (queries with frequency less than 20)

8.3.4 Method Analysis and Comparison

To illustrate the differences among query-URL graph approaches, we show a few examples from [40]. The compared methods are defined as follows:

- **RW-F** The basic random walk on click graph using Eq. (8.19).
- **RW-B** The random walk approach with backward propagation using Eq. (8.22).
- **RW-P** Random walk based on pseudo relevance feedback where top-10 URLs of the testing queries are used to conduct random walk propagation on the click graph instead of the clicked URLs.
- **RW-C** This is to combine the random walk of click and skip graph using Eq. (8.25).

Table 8.7 illustrates the results with a few queries. From the table we can observe some interesting results: (1) RW-F and RW-B provide slightly different results. RW-F is more likely to suggest popular queries than RW-B since the propagation assigns larger probabilities to queries with more clicks. For example, for the query "nfl teams with 5 super bowl wins," RW-F recommends "super bowl champions" as the top suggestion and RW-B suggests "super bowl champs" on the 4th position. (2) RW-P has better suggestion quality than RW-F and RW-B, especially for tail and ambiguous queries. For example, for query "single ladies" the relevant query "single ladies by beyonce" is recommended as top candidate by RW-P. For tail queries with less clicks, non-clicked URLs on SERP becomes important for query suggestions. Therefore, RW-P performs better on tail queries than RW-F and RW-B. (3) RW-C works better than RW-P with more labeled relevant queries. The reason is that RW-P treats both clicked and skipped URLs equally, while RW-C utilizes the click and skip counts.

Table 8.7 Examples of query suggestions by different RW methods in [40]. Bold queries are judged as relevant

Query	RW-F	RW-B	RW-P	RW-C
Valentine one (shopping and car)	Valentines day	Valentine activities	Valentine gifts	**Best radar detector**
	Valentine activities	Valentine gifts	**Valentine one review**	**Escort radar**
	Valentine gifts	Valentines day	**Radar detector**	**Radar detector**
	Anniversary gifts	Free valentines crafts	Valentine one eBay	**Valentine one review**
	Free valentines crafts	Anniversary gifts	Valentine activities	Valentine one eBay
Single ladies (music)	Dating ladies	Single women	**Single ladies by beyonce**	**Beyonce single ladies**
	Beyonce single ladies	Dating single ladies	Single ladies mp3	**Single ladies by beyonce**
	Single women	Dating ladies	Dating single ladies	Single ladies lyrics
	Single women myspace	Dating ladies myspace	Single ladies myspace	Single ladies mp3
	Eharmony	Single moms	**Beyonce single ladies**	**Single ladies download**
NFL teams with 5 super bowl wins (sports and long)	**Super bowl champions**	Super bowl 2009	Super bowl history	**List super bowl winners**
	Super bowl	Super bowl 2008	**Super bowl winners**	**Super bowl winners**
	Super bowl 2009	**Super bowl champs**	**Past NFL super bowl winner**	Super bowl steelers
	Super bowl 2008	List of superbowl	Super bowl 2009	**Past NFL super bowl winner**
DC ups (ambiguous)	D-cups	DC power	**DC power supply**	DC power
	D cup	D-cups	DC ups power	**DC ups power**
	DC ups systems	**DC ups systems**	D-cups	**DC postal service**
	D-cup	**DC power system**	**DC ups systems**	**DC power supply**
	DC control	**Universal power supply**	DC USA	**DC ups systems**

8.3.5 Summary and Discussion

In this section we introduced several well-known query suggestion approaches using query-URL bipartite graph, namely forward random walk with restart, backward random walk, hitting time, and combining click and skip graphs. Existing study in [40] showed that random walk tends to get into data sparsity issue for tail queries with few clicks. Utilizing search result information (e.g., skipped URLs) can help improving both coverage and quality. Basically, the more relevant URLs we can obtain for a given query, the better suggestion results we are able to provide.

8.4 Query Transition Graph Methods

In this section, we introduce the query transition graph methods. Particularly, QFG and TTG approaches are described.

8.4.1 Query Flow Graph (QFG)

One typical approach to model the query refinement process is the QFG proposed by Boldi et al. [7]. The idea of QFG is to consider the whole search sequence as a flow of queries and model it in a probabilistic way.

Specifically, a transition probability between query q_i to q_j is defined as:

$$P^s(q_j|q_i) = \frac{Cnt(q_i \rightarrow q_j)}{Cnt(q_i)}, \tag{8.26}$$

which leverages the adjacent information between queries in the search sequence. Let matrix $\mathbf{A}$ denote the transition on the whole graph with $\mathbf{A}_{ij} = P^s(q_j|q_i)$. One can use either session or task described in Sect. 8.2 to organize the query sequence. The authors in [7] defined query chains to help identifying queries for the same information need, which is very similar to task defined in Sect. 8.2.2.

Second, similar to RWR in Eq. (8.19), an iteration process on QFG is defined as:

$$\mathbf{v}_i^{t+1} = \alpha \cdot (\mathbf{v}^t)^T \cdot \mathbf{A} + (1-\alpha)\mathbf{v}_i^0. \tag{8.27}$$

Here α is restart probability of query node to itself.

The QFG approach was generalized as the query template flow graph (QTFG) in [43], where the phrases in a query were generalized by WordNet hierarchy. For example, "chocolate cookie recipe" can be generalized as "<food> cookie recipe."

For a test query q and a suggesting candidate q', suppose they can be generalized into templates x and x', respectively. A template-based similarity between q and q'

can be defined as:

$$f(q, q') = (1 - \tau) \cdot P(q'|q) + \tau \cdot \sum_{x,x'} P(q|x') \cdot P(x'|x) \cdot P(x|q), \tag{8.28}$$

where τ is a parameter to combine query transition probability on QFG and the query transition probability on QTFG. One can define different types of template generalization for a query. In [43] the authors proposed to utilize the WordNet hierarchy to generalize each possible phrase in query as a template and compute all transition probabilities using the query co-occurrences in sessions. For example, following the definition in [43]: τ is set as 0.5, the term $P(q'|q)$ is the $\mathbf{v}_q^*[q']$ from Eq. (8.27) above, $P(q|x')$ is set as 1 if q can be generalized as x' and 0 otherwise, $P(x|q)$ is defined based on the WordNet hierarch distance of q to x. The term $P(x'|x)$ is defined as transition probability between all queries q to q' falling into template x and x', respectively.

8.4.2 Term Transition Graph (TTG)

Based on the observations that most of the time only the last term of the query is modified when users refine their queries for the same search tasks, Song et al. [42] proposed a TFG approach for query suggestion. Three types of actions, namely *Modification*, *Expansion*, and *Deletion* were proposed for a query refinement, where some examples are shown in Table 8.8.

Given a word vocabulary $W = \{\epsilon, w_1, \ldots w_n\}$ where ϵ is used to denote the empty string, three cases of user query refinements are formulated as [42]:

- *Modification*: user modifies the last term of the query, e.g., "single ladies song" $\rightarrow$ "single ladies lyrics." Denote as: $\{w_1, \ldots, w_m\} \rightarrow \{w_1, \ldots, w'_m\}$.
- *Expansion*: user adds one term to the end of the query, e.g., "sports illustrated" $\rightarrow$ "sports illustrated 2010." Denote as: $\{w_1, \ldots, w_m\} \rightarrow \{w_1, \ldots, w_m, w_{m+1}\}$.

Table 8.8 Three types of user refinement examples

Type	User activity	Pattern
Modification	1. q:{single ladies song}	song→lyrics
	2. q:{single ladies lyrics}	
	3. URL click	
Expansion	1. q:{sports illustrated}	$\epsilon \rightarrow$2010
	2. q:{sports illustrated 2010}	
	3. URL click	
Deletion	1. q:{eBay auction}	auction $\rightarrow \epsilon$
	2. q:{eBay}	
	3. URL click	

- *Deletion*: user removes the last term of the query, e.g., "eBay auction" → "eBay." Denote as: $\{w_1, \ldots, w_{m-1}, w_m\} \rightarrow \{w_1, \ldots, w_{m-1}, \epsilon\}$.

Here the original query $q = \{w_1, \ldots, w_m\}$ and the refined query can be noted as q'.

A nature way to estimate the probability for *Modification* and *Deletion* can be formulated as:

$$P_{modify}(q'|q) = P(w_1, \ldots, w_{m-1}, w'_m|w_1, \ldots, w_m), \tag{8.29}$$

where w'_m can be the empty string ϵ or other words in W.

Similarly, *Expansion* can be formulated as:

$$P_{expan}(q'|q) = P(w_1, \ldots, w_m, w_{m+1}|w_1, \ldots, w_m). \tag{8.30}$$

However, this simple approach tends to fall back into the co-occurrence (or adjacency) based approach, where most frequent queries followed q are selected as suggestions. Hence it has issue to provide good quality suggestions for low-frequency queries. To address the issue, Song et al. [42] introduced a topic based method to generalize the words.

With topic T, the *Modification* and *Deletion* can be formulated as:

$$P_{modify}(q'|q) = \sum_T P(w_m \rightarrow w'_m|T) \cdot P(T|w_1, \ldots., w_m), \tag{8.31}$$

while *Expansion* can be formulated as:

$$P_{expan}(q'|q) = \sum_T P(w_{m+1}|T) \cdot P(T|w_1, \ldots, w_m). \tag{8.32}$$

Here $P(w_m \rightarrow w'_m|T) = P(w'_m|w_m, T)$ is the term transition probability under topic T, and $P(w_{m+1}|T)$ can be viewed as a popularity of w_{m+1} under topic T. The term $P(T|w_1, \ldots, w_m)$ is the probability of topic T for the given query. Note that Eq. (8.31) can be applied to any q' which has one word modified from q.

The topic T can be a predefined taxonomy (e.g., ODP as used in [42]), or an automatically learned topic distribution through approaches like LDA [6], pLSI [19], etc. Since the probability of $P(w_m \rightarrow w'_m|T)$ and $P(w_{m+1}|T)$ is not on the same magnitude, in [42] the authors proposed to multiply a $P(w_m)$ on $P_{modify}(q'|q)$ to make the final score comparable, based on the assumption that $P(w_{m+1}|T) \approx P(w_{m+1}|w_m, T) \cdot P(w_m)$.

Therefore, the final suggestion model can be formulated as:

$$P_{final}(q'|q) = \begin{cases} P_{modify}(q'|q) \cdot P(w_m), & \text{using Eq. (8.31)} \\ P_{expan}(q'|q), & \text{using Eq. (8.32)}. \end{cases} \tag{8.33}$$

Table 8.9 Examples of query suggestions reported from [7, 42] (Q: test query, S: suggestion)

	Query flow graph		Term transition graph	
Q	Music	Evening dress	Battlefield bad company 2	Dante's inferno xbox 360
S	Music	Evening dress	Battlefield bad company 1	Dente's inferno xbox 260 wiki
	Yahoo music	Formal evening dress	Battlefield bad company 2 Ringtones	Dante's inferno ps3
	Music video	Red evening dress	Battlefield bad company 2 slots	Dante's inferno xbox 360 cheats
	Music download	Myevening dress	Battlefield bad company 2 realms	Dante's inferno xbox 360 walkthrough
	Free music	Prom 008 dress	Battlefield bad company 2 games	Dante's inferno

8.4.3 Analysis of Query Transition Methods

Table 8.9 shows query suggestions by QFG and TTG, respectively. From the table, we can observe that: (1) Suggestions from QFG for frequent queries are usually more specialized, which is in accordance to the query reformulation pattern statistics in Table 8.4. (2) TFG can provide relevant suggestions for long queries which are more likely tail queries. This is in accordance with the QTFG approach [43] which leverages term/phrase information in queries.

8.4.4 Summary

In this section, we introduced methods using query transition graph information. Both QFG and TTF approaches are introduced with a few examples to show their effectiveness. Query transition graph may have issue for low-frequency queries with less follow-up queries. To address the problem, we can either generalize the query into template or utilize the term transition information extracted from query refinements.

8.5 Short-Term Search Context Methods

In this section, we introduce the short-term search context methods. Short-term search context usually refers to queries and clicks issued shortly before the current

one. One straightforward way of getting context-aware suggestions is to leverage the search history sequence $q_{1,\ldots,i} = \{q_1, \ldots, q_i\}$ and predict the next query q_{i+1} based on the frequency of $(q_{1,\ldots,i,i+1})$ in search logs. However, such approach suffers from the data sparsity problem due to the exponential growth of space of query sequences. Next, we introduce different types of methods to ease the data sparsity problem.

8.5.1 Decay Factor Based Approaches

Huang et al. [20] proposed a cosine based context-aware method, which is formulated as:

$$f(q_{1,\ldots,i}, q') = \sum_{k=1}^{i} \beta^{i-k} \cdot Cosine(q_k, q'). \tag{8.34}$$

Here $\beta \in [0, 1]$ is a decay factor to control the quality of suggestion.

Following [7, 36], we can initialize vectors $\mathbf{v}_k(0) = \beta^{i-k}$ for $k = 1, \ldots, i$ and conduct random walks. To assign higher weights to more recent queries, β^{i-k} is used as a decay factor. Specifically, we can formulate the suggestion method as:

$$\mathbf{v}^*_{context} = \sum_{k=1}^{i} \mathbf{v}^*_k, \text{ and } \mathbf{v}^{t+1}_k = (1-\alpha) \cdot (\mathbf{v}^t_k)^T \cdot \mathbf{A} + \alpha \cdot \mathbf{v}^0_k, \tag{8.35}$$

where $\mathbf{v}^*_k$ is the final result of $\mathbf{v}^{t+1}_k$ and $\mathbf{A}$ is the query transition probability matrix.

8.5.2 Sequence Mining Approaches

Next, we introduce sequence mining approaches: concept mining [8, 29], mixture variable Markov Model [18], and variable length Hidden Markov Model [8, 31].

Cao et al. [8, 29] proposed to mine the concept sequence instead of query sequence for suggestion. The idea is straightforward, i.e., instead of matching query sequence $q_1, q_2, \ldots, q_i$ with search history for query suggestions, we can first map each query into a concept (e.g., a cluster of queries) and utilize the concept sequence $c_1, c_2, \ldots, c_i$ for query suggestions. The suggestion method based on concept sequence proposed in [8, 29] can be formulated as:

$$f(q_{1,\ldots,i}, q') = \text{Cnt}(\text{Back\_Off}(c_{1,\ldots,i}), c'). \tag{8.36}$$

Here $c_{1,\ldots,i}$ represents the concept sequence for $q_{1,\ldots,i}$, and c' is the concept of q'. Back_Off$(\cdot)$ is a function to get the longest pattern $(c_{a,\ldots,i}, c')$ $(1 < a <= i)$ which

exists in the model. For example, if we have an input concept sequence (c_1, c_2, c_3) but our model mined from search logs can provide suggestion for (c_2), (c_3), and (c_2, c_3), the Back_Off function returns longest found sequence (c_2, c_3) and ignores (c_1, c_2, c_3) since it is not found in the model.

Similarly, He et al. [18] proposed a mixture Variable Markov Model (MVMM) to model the search sequence for query suggestions. Using MVMM, the query suggestion of a given query sequence $q_{1,\ldots,i}$ can be formulated as:

$$f(q_{1,\ldots,i}, q') = \sum_{k=1}^{i} w(q_{k,\ldots,i}, q') \cdot Cnt(q_{k,\ldots,i}, q'), \tag{8.37}$$

where $w(q_{k,\ldots,i}, q')$ is the weight for the sequence. The weight function is dynamically changing for different query sequences. If we set all $w(\cdot)$ to be 1, the model falls back as a combination of suggestions from different length matching of query sequences. In [18], the optimal weight parameters are learnt to maximize the generalization probability for next queries in the search logs. An alternative way of learning the weight is to build two separate datasets, where the first one is to estimate the frequency of search sequences, and the second one is used to optimize the weights of the sequences for better generalization ability.

A more generic extension of the search context modeling is proposed in [9, 31], namely the variable length Hidden Markov Model (vlHMM), where each hidden state in the model represents a hidden concept for each query. Using vlHMM, the query suggestion function can be formulated as:

$$f(q_{1,\ldots,i}, q') = P(q'|z_{i+1}) \cdot P(z_{i+1}|q_{1,\ldots,i}), \tag{8.38}$$

where z_{i+1} is the predicted hidden state at time $i+1$, $P(q'|z_{i+1})$ is the probability of generating q' from state z_{i+1}, and $P(z_{i+1}|q_{1,\ldots,i})$ is the probability of generating the next search state z_{i+1} given the sequence $q_{1,\ldots,i}$. Similar to optimizing hidden Markov models, the parameters of $P(z_{i+1}|q_{1,\ldots,i})$ and $P(q'|z_{i+1})$ are learned to maximize the probability of predicting the next query. By initializing the state of concept using clustering methods [8, 29], the EM (Expectation–Maximization) learning process can be greatly accelerated to converge within 10 iterations, which makes this approach scalable to large-scale datasets.

8.5.2.1 Concept Mining Using Clustering Algorithm

Concept mining has been shown to be useful for query suggestion, which is capable of alleviating the data sparsity problem by grouping queries into concepts. In this section, we introduce a fast clustering algorithm, namely Query Stream Clustering (QSC) for concept mining proposed in [8, 29].

Generally, the process of the QSC algorithm can be summarized into the following steps. First, each query q is represented as a feature vector using its

clicked URLs. Second, q is compared to existing clusters to find a closest match, where the distance between q and a cluster is given by their URL feature vectors. Finally, if the diameter of a cluster after adding q is smaller than a predefined threshold, q is added into the cluster. Otherwise, a new cluster with only q is created.

Due to the fact that the average number of clicked URLs of a query is small, QSC algorithm can be very efficient in practice since it scans the dataset only once. For each query q, the number of clusters to be accessed is at most number of queries shared at least one clicked URL with q. Therefore, the computation cost for each query is near constant, which leads to the complexity of the whole algorithm to be $O(|Q|)$, where $|Q|$ is the number of queries in the dataset.

The QSC algorithm is memory-intensive since it needs to hold all data structure in the memory to conduct fast clustering. Therefore, once the data becomes large and cannot be stored on a single machine, the algorithm fails. To address such limitation, Liao et al. [29] proposed two extensions of QSC: (1) for datasets with small size, an efficient iterative clustering method is proposed and (2) for large datasets, a distributed master–slave framework is proposed for clustering. Interested readers can find more details in [29].

8.5.3 Method Analysis and Comparison

We presented experiment results from [29] to illustrate the difference of query suggestion methods, where the input is a query sequence $q_{1,\ldots,i} = q_1, \ldots, q_i$:

- **Adjacency**. It ranks queries by their frequencies immediately following the last query q_i in the training sessions and output top queries as suggestions.
- **N-Gram**. It ranks queries by their frequencies of immediately following the entire query sequence in training sessions and output top queries as suggestions.
- **Cosine**. It ranks queries by their cosine similarities with every query in the sequence $q_{1,\ldots,i}$ as in Eq. (8.34).
- **CACB**. Short for context-aware concept-based method which uses the concept sequence to provide suggestions as formulated in Eq. (8.36).

Table 8.10 shows a few queries with suggestions from above methods. We can find that N-Gram method fails when the input query sequence is not frequently occurring in the search logs (e.g., providing no suggestion for query sequence"www.chevrolet.com ⇒ www.gmc.com"). Similarly, the results from Adjacency indicates that the method ignores the context information (e.g., suggesting repeated query "www.chevrolet.com" for query sequence"www.chevrolet.com ⇒ www.gmc.com"). Conversely, CACB provides better suggestions and avoids the duplications by other methods (e.g., "msnnews" provided by Cosine to query "msn news").

Table 8.11 shows a few ambiguous queries with or without context information, where the suggestions are provided by CACB. We can see that utilizing the context can help disambiguate the query intent and yield more relevant suggestions.

Table 8.10 Examples of query suggestions provided by different methods [29]

	Methods			
Test case	Adjacency	N-Gram	Cosine	CACB
www.at&t.com	AT&T	AT&T	ATT wireless	ATT wireless
	www.att.com	www.att.com	Cingular	Cingular
	Cingular	Cingular	ATT net	Bellsouth
	www.cingular.com	www.cingular.com	Bellsouth	Verizon
	ATT net	ATT net	AT&T	Tilt phone
msn news	CNN news	CNN news	CNN news	CNN news
	Fox news	Fox news	msnnews	Fox news
	CNN	CNN	MSNBC news	ABC news
	msn	msn	KSL news	CBS news
			Yahoo news	BBC news
www.chevrolet.com	www.chevy.com	<null>	www.chevy.com	Ford
⇒ www.gmc.com	www.chevrolet.com		www.dodge.com	Toyota
	www.dodge.com		www.pontiac.com	Dodge
	www.pontiac.com			Pontiac
Circuit city	Circuit city	Walmart	Walmart	Radio shack
⇒ best buy	Walmart	Target	Staples	Walmart
	Target	Sears	Office depot	Target
	Best buy stores	Office depot	Dell	Sears
	Sears		Amazon	Staples

8.5.4 Summary

In this section we introduced query suggestion methods based on short-term search context. We have shown that utilizing queries in the short-term search context can effectively improve query suggestion. Directly mining frequent query sequences from search logs suffered from the data sparsity problem, and decay factor and sequence mining based approach can alleviate this issue. A general way to address the data sparsity problem is to group queries into concepts using clustering approaches [8, 29], which can provide suggestions with both good precision and high coverage.

8.6 Other Query Suggestion Related Work

In this section, we briefly discuss other related work of query suggestion that are relevant but did not cover in this chapter.

Some early studies of query suggestion proposed to group queries into clusters and provide queries within same cluster as suggestions. Some examples are:

Table 8.11 Examples of query suggestions for ambiguous or multi-intent queries when context information is available and absent [29]

No context available	Context available	
Comcast	*eBay* ⇒ *Comcast*	*Cable* ⇒ *Comcast*
Myspace	Myspace	Verizon
eBay	AOL	AT&T
AOL	Comcast email login	Dish network
Comcast email login	Craiglist	Quest
Craigslit		T-mobile
MQ	*Games* ⇒ *MQ*	*Websphere* ⇒ *MQ*
Games	Dragonfable	MQ client
Dragonfable	Adventure quest	MQ document
Miniclip	Runescape	MQ training
Runescape	Miniclip	
Adventure quest	Tribal wars	
Webster	*Online dictionary* ⇒ *Webster*	*Citibank* ⇒ *Webster*
Dictionary	Encarta	Bank of America
Encarta	Thesaurus	American Express
Thesaurus	Free dictionary	Peoples Bank
Free dictionary	Oxford dictionary	Citizens
Bank of America	Spanish dictionary	Chase
CTC	*Tenax* ⇒ *CTC*	*Child tax* ⇒ *CTC*
Central Texas College Transcript	Central Texas College Transcript	Child tax benefit
Child tax benefit	GoArmyEd	Tax rebate
Tarleton State University	Tarleton State University	Working tax credit
GoArmyEd	University of Maryland	Tax credits
Tax rebate	Temple college	IRS

- **Agglomerative** Beeferman and Berfer [4] proposed an agglomerative clustering method to iteratively group queries and URLs into clusters.
- **DBScan** Wen et al. [46] used DBScan clustering algorithm to cluster queries based on both textual and click information.
- **K-means** Yates et al. [3] proposed to cluster queries using K-means algorithm and compute query similarity using the click-through information.

As pointed out by [8, 29], the aforementioned clustering algorithms have high time complexity (e.g., $O(N_q^2)$ for N_q queries). Therefore, the QSC algorithm [8, 29] was proposed as an alternative to efficiently generate the clusters in $O(N_q)$ time.

Different from traditional work which utilize query-URL click-through information to compute query similarities, some recent work [5, 15, 26] proposed to model query as bag-of-words or phrases and generate suggestions by considering the phrase similarity.

- **Phrase substitution method** Jones et al. [26] proposed to segment queries into phrases using pointwise mutual information and find related phrase substitution

through Log Likelihood Ratio (LLR). To rank suggestion candidates, they further applied a machine learning framework to classify whether a suggestion is more generic, specific, or irrelevant based on textual and LLR features.

- **SERP-based method** Feuer et al. [15] proposed a generalization/specification approach for suggesting phrases from the top ranked search results. They proposed to generate query suggestions using proximal sub-phrases and unordered super phrase based on the phrase frequency in top search documents.
- **External corpus based method** Bhatia et al. [5] proposed to mine a phrase set from documents in external corpus (e.g., news article dataset used in TREC). Their approach splits an input query q into the completed phrase Q_c and typing phrase Q_t and finds a suggesting candidate p_i which can both optimize the probability of $P(Q_c|p_i)$ and $P(p_i|Q_t)$. Here $P(Q_c|p_i)$ is estimated by the probability of document containing phrase Q_c while p_i presents, and $P(p_i|Q_t)$ is computed using normalized frequency of p_i containing a complete word c starting with Q_t.

Besides improving the quality and coverage of suggestions, the diversity of the suggestion results was studied in [24, 34, 41].

- **Diversifying Suggestion** Ma et al. [34] proposed a hitting time based iterative algorithm to add diversified suggestion candidates one by one. To generate all suggestions, they conducted the following steps: (1) given a test query q, get a top-1 query suggestion q' and add it into a set HS; (2) perform a hitting time algorithm to get next query q', add q' to HS; (3) repeat step 2 until obtaining enough suggestion results. The essential idea in [34] is that in step (2), the hitting time approach computes the hitting time of q starting from q' without visiting any nodes in HS. Therefore, all nodes already in HS are skipped for a diversification purpose.
- **Diversifying Search Results** Song et al. [41] proposed a machine learning framework to systematically optimize the relevance and diversity for query suggestion. The proposed learning framework utilized result set features to compute the similarity between queries. Note that diversification in [41] is to provide different SERP comparing with testing query q, where diversity in [34] is to diversify the queries in the suggestion results.
- **Diversifying and Personalization** Jiang et al. [24] proposed to address the diversity and personalization problem together through combining multiple bipartite graphs (e.g., query-URL graph, query-session graph, query-term graph) for query representation and diversification and utilizing offline user profile for personalization. Their diversification algorithm is similar to method in [34] described above. After getting all suggestions, they personalized the results by computing a similarity between suggesting queries and user profiles.

Several studies are proposed to improve the user interface for better utility and experience [27, 48]:

- **Text + Images** Zha et al. [48] proposed a suggestion UI where a picture along with the suggesting query is presented to users in the scenario of image search.

Such UI design is integrated into the commercial search engine nowadays where some suggestions show both textual and image information.

- **Structured UI** Kato et al. [27] studied structured suggestion style with specialization and parallel movements where suggesting queries are grouped as clusters with text labels. Based on the success rate on predefined search tasks, the new UI with grouping and tags outperforms the traditional UI with a plain list of results.

Machine learning approaches were applied to query suggestion to better combine different features. Jain et al. [21] proposed to synthesize query suggestion based on a CRF model to drop less important terms and combine click-through and session information to get good suggestions within a learning framework. Similarly, Ozerterm et al. [38] proposed to learn the suggestion function through both lexicon and result set features using Gradient Boosting Decision Tree (GBDT) method. They validated the importance of aboutness feature which measures the similarity between SERP of a suggesting query and the test query, which is accordance with findings in [41].

To evaluate the suggestion quality, several metrics were proposed in [1, 8, 27]:

- **Human Label** This is the most common evaluation strategy [8, 29–31, 40–42]. Given a test query q with a suggesting q', the annotator is presented with both queries with some necessary information (e.g., the search context of the query, the search results of q and q') to label whether q' is relevant or not.
- **Task Accomplishment** Kato et al. [27] proposed to evaluate the effectiveness of different query suggestion UI by the success rate of predefined search tasks.
- **SERP Annotation** Ma et al. [35] proposed to annotate the relevance of the suggestion by considering the result set information of whether a suggestion q' provides a better results or not comparing with testing query q.
- **User Behavior Prediction** He et al. [18] utilized search logs for automatic evaluation of their query suggestion methods. Part of users' search sequences were given to query suggestion methods to predict the next submitted queries. Albakour et al. [1] used daily search logs to measure the suggestion results in a similar manner. They leveraged MRR (Mean Reciprocal Rank) as the evaluation metric.

It has been shown that query suggestion techniques are useful for other applications as well. For example, Jones et al. [26] applied query suggestion techniques (e.g., LLR) for sponsored search and illustrated improvement of sponsored suggestion. Hasan et al. [17] proposed to leverage query suggestion techniques for e-commerce websites (e.g., eBay) and evaluate the effectiveness (e.g., CTR) for product search.

8.7 Discussions and Future Directions

In this chapter, we summarized several types of query suggestion methods: (1) Co-occurrence; (2) Query-URL bipartite graph; (3) Query transition graph; and (4) Short-term search context.

Co-occurrence methods [16, 20, 26, 30] use co-occurrence of query pairs in sessions or tasks. This type of method is usually straight-forward to understand and compute. One problem of such approach is that it usually can provide good suggestions for high-frequency queries and may not be able to provide suggestion to tail queries with few or no co-occurred queries.

Query-URL bipartite graph methods [10, 36, 40] use clicked URLs of a query to find similar queries. This type of method usually conducts random walk on the click graph to propagate the similarities. For tail queries with less or no clicks, one can leverage the post-web information (e.g., skipped URLs on the SERP [40]) to enrich the pseudo relevant URLs of a query. If the search engine performs bad on a query, it is hard to provide good query suggestions by using the click or post-web information.

Query transition graph methods [7, 42, 43] use the query refinement information in search logs to find next possible queries in the search process. This type of method usually constructs a query transition graph and performs random walk on the graph starting from testing queries. For tail query with less or no refinement information, one can leverage the query string information to generate the query as template [43] or construct term-level transition graph [42]. At the meantime, one needs to carefully design the approach for generalizing queries as templates or constructing term-level transition graph to achieve a good relevance.

Short-term search context methods [8, 18, 20, 29, 31] use search sequence information (e.g., queries within current session) to improve the relevance of suggestions. Sequence mining approaches [8, 18, 29] are usually applied to predict next possible queries given current search sequence. To address the data sparsity problem of search sequence, clustering algorithms are proposed in [8, 29] to group similar queries as clusters and mine cluster level search sequences.

Moving forward, tail queries with few click information or irrelevant search results need to draw more attention for better suggestion algorithms. Although graph and SERP based approaches are able to help certain types of tail queries, the coverage remains as a critical issue for most of the existing works.

References

1. M-Dyaa Albakour, Udo Kruschwitz, Nikolaos Nanas, Yunhyong Kim, Dawei Song, Maria Fasli, and Anne N. De Roeck. AutoEval: An evaluation methodology for evaluating query suggestions using query logs. In *European Conference on Information Retrieval*, pages 605–610, 2011.

2. Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, and Aristides Gionis. An optimization framework for query recommendation. In *Proceedings of the Third International Conference on Web Search and Data Mining*, pages 161–170, 2010.
3. Ricardo A. Baeza-Yates, Carlos A. Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *International conference on extending database technology*, pages 588–596, 2004.
4. Doug Beeferman and Adam L. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416, 2000.
5. Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. Query suggestions in the absence of query logs. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 795–804, 2011.
6. David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *Journal of machine Learning research*, 3 (Jan): 993–1022, 2003.
7. Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. The query-flow graph: model and applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 609–618, 2008.
8. Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 875–883, 2008.
9. Huanhuan Cao, Daxin Jiang, Jian Pei, Enhong Chen, and Hang Li. Towards context-aware search by learning a very large variable length hidden Markov model from search logs. In *Proceedings of the 18th International Conference on World Wide Web*, pages 191–200, 2009.
10. Nick Craswell and Martin Szummer. Random walks on the click graph. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 239–246, 2007.
11. Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. Probabilistic query expansion using query logs. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 325–332, 2002.
12. Hongbo Deng, Irwin King, and Michael R. Lyu. Entropy-biased models for query representation on the click graph. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 339–346, 2009.
13. Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19 (1): 61–74, 1993.
14. Henry Allen Feild and James Allan. Task-aware query recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 83–92, 2013.
15. Alan Feuer, Stefan Savev, and Javed A. Aslam. Evaluation of phrasal query suggestions. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, pages 841–848, 2007.
16. Bruno M. Fonseca, Paulo Braz Golgher, Bruno Pôssas, Berthier A. Ribeiro-Neto, and Nivio Ziviani. Concept-based interactive query expansion. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 696–703, 2005.
17. Mohammad Al Hasan, Nish Parikh, Gyanit Singh, and Neel Sundaresan. Query suggestion for e-commerce sites. In *Proceedings of the Forth International Conference on Web Search and Data Mining*, pages 765–774, 2011.
18. Qi He, Daxin Jiang, Zhen Liao, Steven C. H. Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. Web query recommendation via sequential query prediction. In *Proceedings of the 25th International Conference on Data Engineering*, pages 1443–1454, 2009.
19. Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

20. Chien-Kang Huang, Lee-Feng Chien, and Yen-Jen Oyang. Relevant term suggestion in interactive web search based on contextual information in query session logs. *J. Assoc. Inf. Sci. Technol.*, 54 (7): 638–649, 2003.
21. Alpa Jain, Umut Ozertem, and Emre Velipasaoglu. Synthesizing high utility suggestions for rare web search queries. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 805–814, 2011.
22. Bernard J. Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. Real life information retrieval: A study of user queries on the web. *SIGIR Forum*, 32 (1): 5–17, 1998.
23. Eric C. Jensen, Steven M. Beitzel, Abdur Chowdhury, and Ophir Frieder. Query phrase suggestion from topically tagged session logs. In *International Conference on Flexible Query Answering Systems*, pages 185–196, 2006.
24. Di Jiang, Kenneth Wai-Ting Leung, Lingxiao Yang, and Wilfred Ng. Query suggestion with diversification and personalization. *Knowledge-Based Systems*, 89: 553–568, 2015.
25. Rosie Jones and Kristina Lisa Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 699–708, 2008.
26. Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, pages 387–396, 2006.
27. Makoto P. Kato, Tetsuya Sakai, and Katsumi Tanaka. Structured query suggestion for specialization and parallel movement: effect on search behaviors. In *Proceedings of the 21st World Wide Web Conference*, pages 389–398, 2012.
28. Makoto P. Kato, Tetsuya Sakai, and Katsumi Tanaka. When do people use query suggestion? A query suggestion log analysis. *Information retrieval*, 16 (6): 725–746, 2013.
29. Zhen Liao, Daxin Jiang, Enhong Chen, Jian Pei, Huanhuan Cao, and Hang Li. Mining concept sequences from large-scale search logs for context-aware query suggestion. *ACM Transactions on Intelligent Systems and Technology*, 3 (1): 17:1–17:40, 2011.
30. Zhen Liao, Yang Song, Li-wei He, and Yalou Huang. Evaluating the effectiveness of search task trails. In *Proceedings of the 21st World Wide Web Conference*, pages 489–498, 2012.
31. Zhen Liao, Daxin Jiang, Jian Pei, Yalou Huang, Enhong Chen, Huanhuan Cao, and Hang Li. A vlHMM approach to context-aware search. *ACM Transactions on the Web*, 7 (4): 22:1–22:38, 2013.
32. Zhen Liao, Yang Song, Yalou Huang, Li-wei He, and Qi He. Task trail: An effective segmentation of user search behavior. *IEEE Transactions on Knowledge and Data Engineering*, 26 (12): 3090–3102, 2014.
33. Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Gabriele Tolomei. Identifying task-based sessions in search engine query logs. In *Proceedings of the Forth International Conference on Web Search and Data Mining*, pages 277–286, 2011.
34. Hao Ma, Michael R. Lyu, and Irwin King. Diversifying query suggestion results. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1399–1404, 2010.
35. Zhongrui Ma, Yu Chen, Ruihua Song, Tetsuya Sakai, Jiaheng Lu, and Ji-Rong Wen. New assessment criteria for query suggestion. In *Proceedings of the 35th International ACM SIGIR conference on research and development in Information Retrieval*, pages 1109–1110, 2012.
36. Qiaozhu Mei, Dengyong Zhou, and Kenneth Ward Church. Query suggestion using hitting time. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 469–478, 2008.
37. James R Norris and James Robert Norris. *Markov chains*. Number 2. Cambridge university press, 1998.
38. Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. Learning to suggest: a machine learning framework for ranking query suggestions. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 25–34, 2012.
39. Craig Silverstein, Monika Rauch Henzinger, Hannes Marais, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33 (1): 6–12, 1999.

40. Yang Song and Li-wei He. Optimal rare query suggestion with implicit user feedback. In *Proceedings of the 19th International Conference on World Wide Web*, pages 901–910, 2010.
41. Yang Song, Dengyong Zhou, and Li-wei He. Post-ranking query suggestion by diversifying search results. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 815–824, 2011.
42. Yang Song, Dengyong Zhou, and Li-wei He. Query suggestion by constructing term-transition graphs. In *Proceedings of the Fifth International Conference on Web Search and Data Mining*, pages 353–362, 2012.
43. Idan Szpektor, Aristides Gionis, and Yoelle Maarek. Improving recommendation for long-tail queries via templates. In *Proceedings of the 20th International Conference on World Wide Web*, pages 47–56, 2011.
44. Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Random walk with restart: fast solutions and applications. *Knowledge and Information Systems*, 14 (3): 327–346, 2008.
45. Hongning Wang, Yang Song, Ming-Wei Chang, Xiaodong He, Ryen W. White, and Wei Chu. Learning to extract cross-session search tasks. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1353–1364, 2013.
46. Ji-Rong Wen, Jian-Yun Nie, and HongJiang Zhang. Clustering user queries of a search engine. In *Proceedings of the Tenth International World Wide Web Conference*, pages 162–168, 2001.
47. Ryen W. White, Mikhail Bilenko, and Silviu Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 159–166, 2007.
48. Zheng-Jun Zha, Linjun Yang, Tao Mei, Meng Wang, and Zengfu Wang. Visual query suggestion. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 15–24, 2009.