DEIM Forum 2016 C4-6

Query Suggestion for Helping Struggling Search

Zebang CHEN†, Takehiro YAMAMOTO†, and Katsumi TANAKA†

† Graduate School of Informatics, Kyoto University
36-1 Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501 Japan
E-mail: †{chenzb,tyamamot,tanaka}@dl.kuis.kyoto-u.ac.jp

Abstract Search engine users often struggle to find the relevant information for their information need by reformulating their queries many times. Since many queries during struggling search are failed and sparse, the existing query suggestion methods often fail to generate effective query suggestions for struggling search. In this paper, we propose a method to generate effective query suggestions aiming to help struggling search. The core is identifying struggling component of on-going struggling session and mining the effective representations of it. The struggling component is the semantic unit of information need for which the user struggled to find an effective representation during the struggling session. The proposed method first identifies the struggling component of given on-going struggling session and mines the sessions containing the identified struggling component from a query log to build a struggling flow graph. The struggling flow graph records the struggling reformulation behaviors of multiple users for the given struggling component. The experimental results demonstrate that the proposed method outperforms the existing algorithm in terms of MRR@10 and nDCG@10.

Key words query suggestion, information retrieval, struggling search

1. Introduction

Search engines have been widely used to find information and solutions to problems. However, many users often struggle to locate relevant information to their problem [1]. This search process is referred to as *struggling search*. In struggling search, the user struggles to reformulate query many times for the information. For example, Table 1(a) shows a struggling session in which a user attempted to find information about *what sports came from Australia*. Unfortunately, although the user attempted to reformulate his/her query many times, he/she could not locate the required information and gave up. Hassan *et al.* reported that, among long search sessions, 36% are struggling sessions and many users fail to locate the required information [1]. Therefore, it is important for search engines to help users in struggling search effectively.

Query suggestion is an effective way to help a user formulate query. Many query suggestion methods have been proposed and proven very useful. However, existing query suggestion methods have limitations when addressing struggling search. One problem is that most queries in struggling sessions are failed queries with few clicks, which means that existing methods that rely on query terms and clicked documents are likely to generate ineffective suggestions and may lead to another failed trial. Another problem is that the information needs of many struggling sessions are long-tail information need, which means that no user or few users have searched for these information previously, thus, there are no effective related queries in the query log.

In this paper, we propose a query suggestion method aiming to help struggling search. The core idea of our method is identifying the *struggling component* of an on-going struggling session and mining effective representations of the struggling component to generate query suggestions. The *struggling component* is a semantic component of the information need in which a user needs to struggle to find its effective representation (See Section 3). For example, there are three semantic components in the information need of the session shown in Table 1(a): *sport*, *came from*, and *Australia*. Among these components, user struggled to find effective representations for *came from*. The same struggling component may occur in search sessions for different information needs. As shown in Table 1(b), another user attempted to find out *where did donut come from*. This user also struggled to find an effective representation for *came from*, similar to the user in Table 1(a). However, this user finally obtained an effective representation for the struggling component: *origin*, which can also enable the user in Table 1(a) to locate relevant information using query: *"sport origin Australia"*.

Table 1: Examples of struggling sessions (terms in italics are the struggling phrase of the query)

Query
Sport Australia *history* <failed>
Sport *of* Australia <failed>
Sport *from* Australia <failed>
Sport *came from* Australia <failed>

(a)

Query
the donut *came from* where <failed>
history *of* the donut <failed>
origin of donut <successful>

(b)

Given an on-going struggling session, we first identify its struggling component and retrieve all sessions with the same struggling component from a query log. We then build a *struggling flow graph* by aggregating the struggling reformulation behaviors in the retrieved sessions. Further we can mine effective representations of the struggling component to generate effective query suggestions for the on-going struggling session.

The contributions of this paper can be summarized as follows.

- We propose a query suggestion method to help struggling search. To the best of our knowledge, this is the first query suggestion algorithm designed to help struggling search.
- We introduce the concept and a method to identify the struggling component of a struggling session, which enables us to understand which part that a user is struggling at.
- We introduce the *struggling flow graph*, which is a graph that records multiple users' struggling reformulation behaviors for the same struggling component.

The remainder of this paper is organized as follows. In Section 2, we explain related work including query suggestion, struggling search, and long-tailed search. In Section 3, we introduce and define the key concepts used in this paper. In Sections 4 and 5, we describe how we identify the struggling component and generate effective query suggestions, respectively. In Section 6, we explain our experiment and analyze the results. The paper is concluded in Section 7.

2. Related Work

Query Suggestion. One main approach to query suggestion is mining click-through data. Beeferman and Berger [2] built a bipartite graph based on click-through data and clustered queries with the same clicked documents because they are likely to have the same search intent. Baeza-Yates *et al.* [3] clustered queries based on their term-weight vector calculated from queries as well as their clicked documents. The queries in the same cluster are ranked and considered possible query suggestions. Mei *et al.* [4] ranked queries using the hitting time on the bipartite graph to ensure semantic consistency between the original query and query suggestions. Cao *et al.* [5] summarized queries and mapped sessions to concepts based on click-through data to build a concept sequence suffix tree, which is used to obtain the context of the current search in order to generate context-aware query suggestions. Another main approach to query suggestion is mining query reformulations of search sessions. Boldi *et al.* [6] used the query flow graph [7] to generate query suggestions. Each node in a query flow graph represents a query and each edge between two nodes means that they are consecutive in at least one session. The short random walk is then applied to the graph to generate query suggestions. Jones *et al.* [8] proposed a model to generate a set of query substitutions by replacing the whole query or parts of the query with related phrases.

Struggling Search. Struggling search was first formally introduced by Hassan *et al.* [1]. They analyzed the characteristics of struggling session and proposed a model to distinguish struggling sessions and exploratory sessions [9]. Task difficulty is a main factor that leads to struggling search. Carmel *et al.* [10] investigated what makes a query difficult, by capturing and analyzing the relationships among the main components of a topic: the textual expression (the query or queries), the set of documents relevant to the topic, and the entire collection of documents. Aula *et al.* [11] studied behavioral signals when a user has difficulty with a search task. They found that, when finding relevant information is difficult, users attempt to formulate more diverse queries, use advanced operators more frequently, and spend more time on viewing search results pages. Liu *et al.* [12] explored the effect of task difficulty on search behavior for users with different levels of domain knowledge. Although many studies have been done on understanding task difficulty and how users behave when experiencing difficulty, few studies have attempted to find ways to assist struggling search. Liu *et al.* [13] used a learning-to-rank approach to rank query suggestions generated using a previously proposed method [5] for difficult search, that depends on effective click-through data. Odijk *et al.* [14] studied the characteristics of search tasks where users struggle and proposed ways to predict future actions and their anticipated impact on search outcomes.

Long-tail Search. Yao *et al.* [15] conducted an empirical study of user behavior with rare queries using a large-scale query log and proved significant differences among many features, including query length and search results. To generate query suggestions for long-tail queries, Bonchi *et al.* [16] [17] built a center-piece subgraph containing term nodes and query nodes with two types of connections: term-query and query-query connections. A random walker starts with the terms in a given query to generate query suggestions even if the query has not occurred previously. To extend the reach of query suggestion for long-tail queries, Szpektor *et al.* [18] introduced a query template and an enhanced query flow graph [7] with query templates, through which suggestions are available even for the long-tail queries that do not have succeeding query.

3. Preliminaries

In this section, we first explain the conceptes of query log, session, and struggling session used in this paper. We then introduce the key concepts for modeling struggling search in order to clarify our approach to generating query suggestions.

Query log: A *query log* is a log that records user search queries and document clicks with their timestamps. A typical format for a record in a query log is *<user_id, query, clicked document, timestamp>*. In this paper, we use the AOL query log, which contains the search behaviors of approximately 650,000 users over 3 months, as our primary dataset. Note that the proposed method is not specific to the AOL query log, but is applicable to other query logs.

Session: A *session* represents a topically coherent session during which a user's information need does not change. One well-known way to extract sessions from a query log is to use a predefined time threshold, *e.g.*, 30 min of inactivity [19], which may contain search queries and document clicks for different information needs in the same session [20]. To ensure that sessions are topically coherent, we used the settings in [1] with a small adjustment: two consecutive queries are topically coherent if they are no longer than 10 min apart and they must share at least one non-stopword term. In the rest parts of this paper, we simply use *session* to represent a *topically-coherent session*.

Struggling session: A *struggling session* is a session in which a user experiences difficulty locating information that is relevant to their information need. Typical user search behaviors in a struggling session are reformulating query many times and spending significant time on the search process [1]. Several examples are shown in Table 1.

Here, we introduce key concepts for modeling struggling search.

Information need: An *information need* is the need to locate information in order to satisfy a user's requirement. Example information needs are as follows: *what sports came from Australia*, *where did donut come from*, and *gas mileage of 2000 4runner*.

Component: A *component* is a semantic unit that corresponds to things in the real world, such as concepts, objects, actions, and relations. In this work, we model that an information need comprises a set of components. For example, the information need *what sports came from Australia* comprises three components, *sport*, *came from*, and *Australia*, and the information need *gas mileage of 2000 4runner* comprises *gas mileage* and *2000 4runner*.

Each component has a set of representations, which are used by users to describe the component in their queries. For example, to describe the component *2000 4runner*, users may use *2000 4runner, 4runner, toyota 4runner, toyota 2000 4runner, etc*. More examples are shown in Table 2.

During a session, users choose a representation for each component of their information need based on their own knowledge and experience to formulate a query, which means that each term in the query is an input that describes the corresponding component of their information need. If the returned results do not satisfy the information need, users may reformulate the query by changing, adding/or removing term(s) in order to locate relevant information, as shown in Table 3.

Struggling component: In a struggling session, we define a *struggling component* as a component for which user has difficulty to find an effective representation. For example, for the session shown in Table 1(a), the information need of the session is *what sports came from Australia*, which has three components, *sport*, *came from*, and *Australia*. During the session, the user frequently reformulated the representation of the *came from* component, which may mean that the user had difficulty finding an effective representation of this component. Thus, *came from* is the struggling component in this session.

Struggling phrase: Given a query in a struggling session, the *struggling phrase* of the query is defined as the set of terms in the query that represent the struggling component of the session. Note that the struggling phrase of a query may be empty. Table 1 shows examples of struggling phrases in a struggling session. For example, the struggling phrases of the four queries in Table 1(a) are {history}, {of}, {from}, and {came, from}.

Table 2: Example components.

Component	Representations
gas mileage	gas mileage, mpg, fuel mileage, miles per gallon, kilometer per liter, *etc*.
come from	of, from, come from, originate from, history, *etc*.
amount	how many, number, count, amount, statistics, *etc*.

Table 3: Example struggling session for information need *where does donut come from,* with components: *donut, came from*. Terms between *()* are input for representing *donut*, terms between *[]* are input for representing *came from*.

Query	Division based on components
the donut came from where	(the donut) [came from where]
history of the donut	[history of] (the donut)
origin of donut	[origin of] (donut)

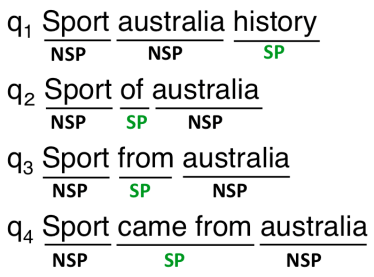

Figure 1: Example of struggling phrase classifier.

4. Identifying Struggling Phrase

In this section, we explain how we identify the struggling phrases of a struggling session.

4.1 Problem statement

Given a struggling session with a sequence of queries, our objective is to distinguish the struggling phrase in each query. To this end, we take a machine learning approach and build a struggling phrase classifier. Specifically, for each term in a query of a struggling session, we classify it into one of two classes, *struggling phrase* (SP) and *non-struggling phrase* (NSP), as shown in Figure 1.

4.2 Features

To build the struggling phrase classifier, we designed features for identifying struggling phrases in a struggling session, as shown in Table 4. There are two types of features, *i.e.*, term-level features and session-level features.

- **Term-level features:** These features describe the characteristics of a term. A term with many alternatives has higher probability to be classified as a struggling phrase because a user may struggle to find an effective representation from these alternatives. For example, a term that refers to an entity typically has fewer alternatives. In addition, the part-of-speech of a term may indicate whether it has many (or few) alternatives to a certain degree.
- **Session-level features:** These features describe the relationships of a term and the session. A term that the user gives up on quickly has high probability of being classified as a struggling phrase, and a term that the user keeps in all queries in

Table 4: Struggling phrase classifier features

Name	Description
Term level features	
isEntity	whether the term refers to an entity
POS	part-of-speech of the term
Session level features	
occurenceRatio	ratio of queries in which the term occurred in the session
firstOccurence	order number of the query in which the term first occurred

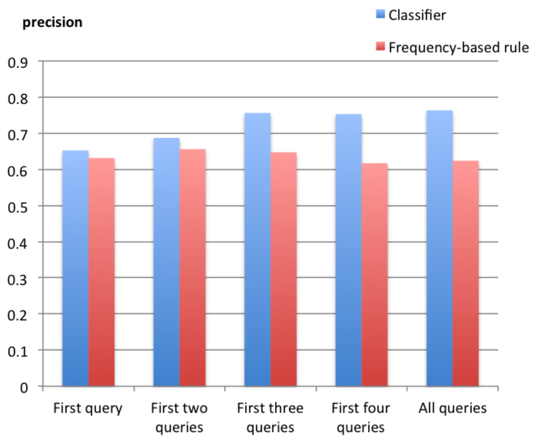

Figure 2: Precision of struggling phrase classifier.

the session has high probability of being classified as a non-struggling phrase.

4.3 Experimental setup

Data. We manually extracted 146 struggling sessions from the AOL query log, which resulted in 540 queries. For each query, we manually labeled terms as a struggling phrase or a non-struggling phrase by checking the query, the search results, and the context of the session, assuming we were the searcher with an information need estimated by the search behavior. We trained the classifier with these labeled data to distinguish struggling phrases and non-struggling phrases of a query for a struggling session.

Baseline. For the baseline, we used the frequency-based method to identify the struggling phrase of a query. This method classifies a term as a non-struggling phrase only when all queries in the session contain the term; otherwise, the term is was classified as a struggling phrase.

Setting. We used the support vector machine with a linear kernel to train the classifier. 10-fold cross-validation was used for this experiment. Note that the performance of the proposed method depends on the number of queries in a session. To investigate the effect of the number of queries available for the classifier on classification performance, we experimented with several cases by changing the number of queries in a session (e.g., the first query, the first two queries, the first three queries . . ., all queries).

4.4 Experimental results

Figure 2 shows the precision of the classification obtained by the proposed method and the frequency-based method. The x-axis is the number of queries used by each method to classify a term.

Table 5: Example struggling sessions.

Query
2007 lincoln mkz *mileage* <failed>
2007 lincoln mkz *fuel* <failed>
2007 mkz *fuel per mile* <failed>
lincoln mkz *mpg* <successful>

(a)

Query
Honda civic *fuel efficiency* <failed>
Honda civic *mileage* <failed>
Honda civic *gas mileage* <successful>

(b)

As can be seem, the classifier outperformed the frequency-based method in terms of precision. The precision of the classifier increased with an increased number of queries. The precision reached 0.76 when the proposed method used all queries in a session. In contrast, the precision of the frequency-based method did not improve with an increased number of queries. One reason for this is that a user slightly modified the terms of non-struggling phrase later in the session.

5. Generating Effective Query Suggestion

In this section, we first explain how we extract sessions that have the same struggling component as the on-going struggling session. We then introduce the struggling flow graph, how to build the graph, and mine effective representations for the struggling component in order to generate effective query suggestions.

5.1 Mining sessions with the same struggling component

As introduced in Section 3, the struggling component is a component of an information need for which user experiences difficulty to find an effective representation of the component. The same struggling component may occur in sessions with different information needs. We have hypothesized that sessions with the same struggling component have similar struggling phrases, and share the same effective representations for the struggling component. Two examples have been shown in Table 1(a) and Table 1(b), in which the first user searched for *what sport came from Australia* and the second user searched for *where does donut came from*, wherein both struggled to represent the struggling component *came from* effectively. Further examples are shown in Table 5(a) and Table 5(b), where the first user searched for *the gas mileage of 2007 lincoln mkz* and the second user searched for *the gas mileage of Honda civic*, and both struggled to represent the struggling component *gas mileage* effectively.

Figure 3 illustrates how we mine sessions with the same struggling component from the query log. Given an on-going struggling session, we hope to obtain all sessions with the same struggling component as the given session from the query log. Specifically, given an on-going struggling session, we first identify its struggling phrases and retrieve the struggling sessions with the same struggling phrases in their queries from the query log. Then, we again extract the struggling phrases of each retrieved session to obtain more sessions with same struggling phrases in a breadth-first-search manner.

5.2 Struggling flow graph

Having obtained sessions with possibly the same struggling com-

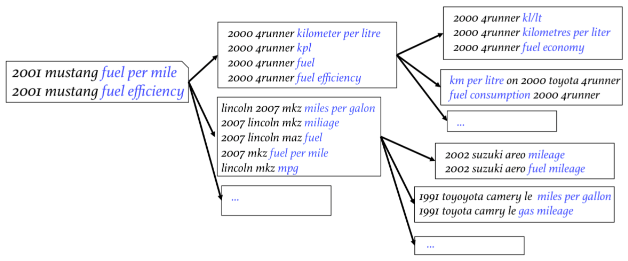
2001 mustang fuel per mile
2001 mustang fuel efficiency
2000 4runner kilometer per litre
2000 4runner kpl
2000 4runner fuel
2000 4runner fuel efficiency
lincoln 2007 mkz miles per galon
2007 lincoln mkz miliage
2007 lincoln maz fuel
2007 mkz fuel per mile
lincoln mkz mpg
...
2000 4runner kl/lt
2000 4runner kilometres per liter
2000 4runner fuel economy
km per litre on 2000 toyota 4runner
fuel consumption 2000 4runner
...
2002 suzuki areo mileage
2002 suzuki aero fuel mileage
1991 toyoyota camery le miles per gallon
1991 toyota camry le gas mileage
...

Figure 3: Mining sessions with the same struggling component from query log.

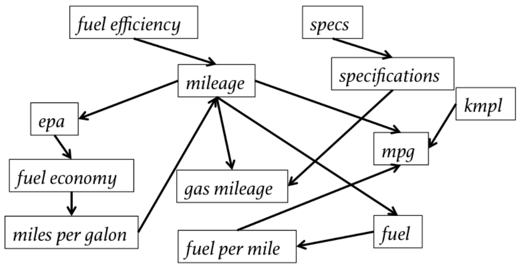

Figure 4: Example of struggling flow graph.

ponent, we then build a *struggling flow graph* to find the effective representation of the struggling component. The purpose of the struggling flow graph is to aggregate multiple users' reformulation behaviors for the terms of a struggling component and mine the effective representations for the given struggling component. A struggling flow graph extendsthe *query flow graph* [6]. In the struggling flow graph, each node represents a struggling phrase of the struggling component. In addition, an edge in the struggling flow graph means that two struggling phrases are consecutive in at least one struggling session.

Here, s_0 denote an on-going struggling session for which we want to provide query suggestions, $S = \{s_1, \ldots, s_n\}$ denote the set of sessions mined by the method described in Section 5. 1 , and $V = \{v_1, \ldots, v_m\}$ denote the unique set of struggling phrases obtained from S by using the classifier described in Section 4.

Formally, the struggling flow graph for sessions S is represented as weighted directed graph $G_S = (V, E, \omega)$.

- V is the set of nodes in the graph. Each node corresponds to a struggling phrase.
- $E \subseteqq V \times V$ is the set of directed edges. If there is at least one session where two struggling phrases v_i and v_j appear consecutively in S, there is a directed edge from v_i to v_j. In addition, for every $v_i \in V$ there is a self-transitive edge from v_i to v_i.
- $\omega : E \rightarrow (0..1]$ is a weighting function that assigns a weight $\omega(v_i, v_j)$ to each pair of struggling phrases $(v_i, v_j) \in E$.

5.3 Finding effective representation using struggling flow graph

Here, we describe how we find the effective representation of the struggling component using the Struggling flow graph. As explained previously, the struggling flow graph is an extension of the query flow graph. In the query flow graph, each node represents a query q and the weight of an edge between two queries q_i and q_j is determined by the probability that the user who issued query q_i and reformulate it as query q_j in a topically coherent session. To generate query suggestions for a given query(ies) q, a random walk is applied to the query flow graph with the random walker starting with query(ies) q. We can extract query suggestions according to the value of the stationary distribution of each query.

We follow the idea of the query flow graph, except for the following differences.

- The struggling flow graph represents a struggling phrase as a node.
- The struggling flow graph incorporates information about whether a query appears to be successful or a failure into the graph.

For the former point, differing from the query flow graph, which aggregates all sessions in the query log into a single graph, the struggling flow graph aggregates only the reformulation behaviors of *terms* rather than *queries*. This allows us to understand the term-level reformulation behaviors of users, which may address the problem of long-tailed queries.

For the latter point, the failure rate in struggling search is high, which increases the probability of failed query suggestions by the random walk in the graph. To address this problem, we incorporate information about whether a query appears successful or a failure into the graph. Suppose we can predict the effectiveness of each struggling phrase in the graph. Our idea is to add high self-transition probability to the effective struggling phrases. Since a node with high self-transition probability is likely to obtain a high random walk score in the stationary distribution, we expect that adding a high self-transition probability to effective struggling phrases will allow us to find effective struggling phrases for the struggling component by applying a random walk to the graph.

5.4 Preparing transition probabilities

In this subsection, we describe how we compute the transition probabilities among the nodes in the struggling flow graph. The assumption behind mining effective struggling phrases is that if one struggling phrase is effective, most users who input it will stop searching as it is effective in helping the users find relevant pages with the information they require. Similarly, if one struggling phrase is ineffective, most users who input it will continue searching as it fails to help the users find the relevant pages. The actual transition probabilities are computed based on the following method. Here, $e(v_i)$ represents the number of sessions that end with struggling phrase v_i with effective clicks and $N(v_i, v_j)$ represents the number of sessions where v_i and v_j occur consecutively. The flow information of struggling phrases that occur only a few times are not trustworthy because they are biased due to sparsity. We define the *degree of trustworthiness* of a struggling phrase to evaluate how trustworthy the flow information of the struggling phrase is, which is represented as $td(v_i)$. As a struggling phrase occurs more times, the degree of trustworthiness of it increases.

$$\mathrm{td}(v_i) = \begin{cases} \frac{e(v_i)}{\sum_{v_j \in V} N(v_j, s_i)}, & \text{if } e(s_i) < \text{threshold} \\ 1.0, & \text{if } e(v_i) \geqq \text{threshold} \end{cases} .$$

The self-transition probability of a node is set as follows.

$$\omega(v_i, v_i) = td(v_i) \frac{e(v_i)}{\sum_{v_k \in V} N(v_j, v_i)} .$$

For the rejected self-transition probability due to low trustable degree, we averagely distribute them to all struggling phrases in the struggling flow graph as random jump probability $\mathrm{rj}(v_i)$.

$$\mathrm{rj}(v_i) = (1 - \mathrm{td}(v_i)) \frac{e(v_i)}{\sum_{v_j \in V} N(v_j, v_i)} .$$

The weight between different struggling phrases v_i and v_j $(i \neq j)$ is as follows:

$$\omega(v_i, v_j) = (1 - \omega(v_i, v_i) - \mathrm{rj}(v_i)) \frac{r(v_i, v_j)}{\sum_{v_j \in V} r(v_i, v_k)} + \frac{\mathrm{rj}(v_i)}{|V|} .$$

Note that the weights calculated based on above weighting functions $w(\cdot)$ are already normalized such that their values represent the transition probability among nodes. In addition, note that, the struggling flow graph uses uniform random jump rather than biased random jump, as is the case of the query flow graph, because our struggling flow graph consists of only related sessions, *i.e.*, sessions with the same struggling component.

5.5 Generating query suggestions

Here, we describe the overall method to generate query suggestions for a given on-going session. Given an on-going session $s_0 = \{q_1, \ldots, q_n\}$, where n represents the number of queries in the session, we first identify the struggling phrases of the session using the method described in Section 4. We then mine the set of sessions S using the method described in Section 5.1 and build the struggling flow graph G_S. Then, we apply a random walk to the graph and extract the top k nodes according to their random walk values in stationary distribution. Finally, we generate the query suggestions by replacing the struggling phrases of the last query in the on-going sessions with the top k struggling phrases.

6. Experiment

In this section, we introduce an experiment conducted to evaluate the effectiveness of the query suggestions generated by the proposed method.

6.1 Baseline

For the baseline, we used the query flow graph proposed by Boldi *et al.* [7]. The query flow graph is a directed graph that aggregates a query log and has been proven very useful for query suggestions and user behavior analysis. Here, we refer to the baseline method as query flow graph method and the proposed method with the struggling flow graph method.

6.2 Evaluated sessions

To evaluate the effectiveness of the proposed method with diverse struggling sessions, we first manually prepared eight types of struggling components by scanning the struggling sessions sampled from the AOL query log. For each struggling component, we manually extracted as many struggling sessions with the struggling component as possible from the AOL query log. Finally, for each struggling component, we randomly sampled 10 struggling sessions from the extracted sessions and used them in the experiment(80 sessions in total). Table 6 shows examples of struggling components and their corresponding session.

Table 6: Examples sessions.

Strugglig component	Example session
gas mileage of car	q_1: 2001 mustang fuel per mile q_2: 2001 mustang fuel efficiency q_3: 2001 mustang gasoline miles
origin of something	q_1: where does the name bonnaroo come from q_2: bonnaroo hisotry q_3: bonnaroo history
best of something	q_1: top 25 children hospital q_2: top 25 children's hospital q_3: number one children's hospital

6.3 Ground truth

We defined a query suggestion that enables a user to locate relevant pages as an *effective* suggestion. To create ground truth query data, for each session described in the previous subsection, we first simulated the session user and attempted to understand the user's search intent as best as possible. For each of the top 10 query suggestions generated by a method, we issued the suggestion to a commercial search engine and examined whether there were relevant pages for the user's information need in the first page of the search results. If there was at least one relevant document in the search results page, the suggestion was considered *effective*; otherwise the suggestion was considered *ineffective*.

6.4 Evaluation metrics

The following metrics were employed to measure the performance of query suggestions.

- **Support rate**: The ratio of sessions for which at least one query suggestion could be offered.
- **Effective support rate@10**: The ratio of sessions in which one or more effective query suggestions were offered in the top 10 query suggestions.
- **MRR@10**: Mean reciprocal rank (MRR) of the top 10 query suggestions.
- **nDCG@10**: Normalized discounted cumulative gain (nDCG) of the top 10 query suggestions. Here, we used binary relevance for the evaluation. An effective query suggestion had relevance of 1, while an ineffective query suggestion had relevance of 0.

6.5 Setting

We used all the prepared labeled struggling sessions prepared (Section 4.2) to train the struggling phrase classifier, which was used to detect struggling phrases of a session. In addition, as with the experiment explained in Section 4.3, we experimented with several cases by changing the number of available queries in a session.

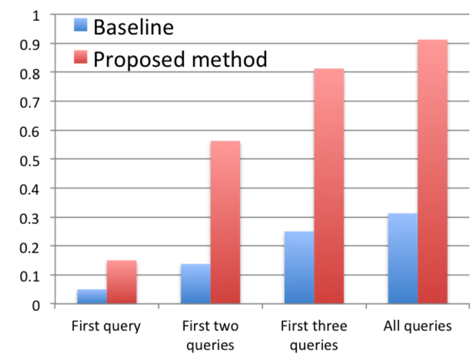

Figure 5: Mean support rate

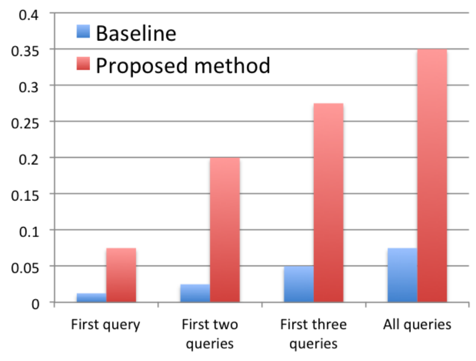

Figure 6: Mean effective support rate@10

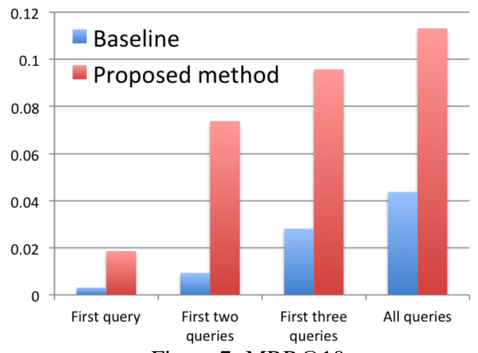

Figure 7: MRR@10

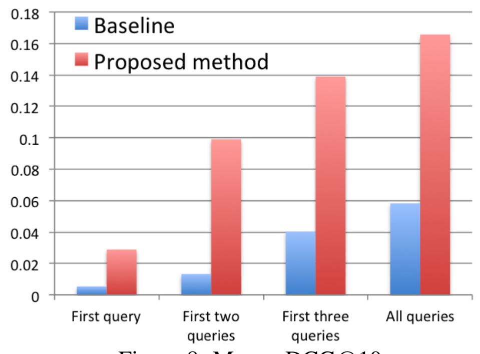

Figure 8: Mean nDCG@10

6.6 Results

Figure 5 shows the mean support rate obtained by the baseline method and the proposed method. As can be seen, the performance to generate query suggestions for struggling sessions improved as the number of available queries increased for both the query flow graph and the struggling flow graph. However, the query flow graph could not generate any query suggestion for 69% of the struggling sessions even when it could use all queries in the session. This was primarily because information needs for many struggling sessions are long-tailed; thus, the queries in struggling search are also long-tailed queries, which may never occur in the query log. On the other hand, the struggling flow graph could generate query suggestions for significantly more sessions than the query flow graph. This shows that considering struggling phrases in a session rather than the queries themselves allows us to address the problem of long-tailed queries. Figures 6, 7, and 8 summarize the mean effective support rate@10, MRR@10 and mean nDCG@10, respectively. From Figure 6, we can observe that the effective support rate@10 is much lower than the support rate for both the query flow graph and the struggling flow graph, which means that only a portion of the sessions were provided effective query suggestions among the sessions for which the given method could provide query suggestions. Similar to the support rate, the struggling flow graph outperformed the query flow graph significantly relative to generating effective query suggestions. The struggling flow graph generated effective query suggestions for 35% of the sessions, and the query flow graph generated effective query suggestions for only 7.5% of the sessions, even when it used all queries of the sessions. From Figures 7 and 8, we can see that the ranking performance increased with an increasing number of available queries for both the query flow graph and the struggling flow graph. The struggling flow graph also outperformed the query flow graph in terms of the ranking performance of effective query suggestions. When using all queries in the session, the proposed method achieved MRR@10 of 0.113 and nDCG@10 of 0.166, and the baseline method achieved MRR@10 of 0.043 and nDCG@10 of 0.058.

Table 7: Example struggling sessions

Query
q_1: average mileage for ford mustang
q_2: 99 mustang mileage

(a)

Query
q_1: history of baseball
q_2: country and origin of baseball

(b)

Table 8: Query suggestions for struggling session in Table 7(a).

Suggestions
Not available.

(a) Suggestion by baseline.

Suggestions
1st: 99 mustang gas mileage
2nd: 99 mustang mileage rate
3rd: 99 mustang fuel mileage
4th: 99 mustang distance
5th: 99 mustang mileage deduction

(b) Suggestion by proposed method.

Two detailed examples of struggling sessions used in the experiment are shown in Table 7(a) and Table 7(b). For the struggling session shown in Table 7(a), the user struggled to find *the gas mileage of ford 99 mastang*. The query suggestions generated by query flow graph are shown in Table 8(a), and those generated by the struggling flow graph are shown in Table 8(b). As can be seen, no query suggestion was generated by the query flow graph because the queries

Table 9: Query suggestions for struggling session in Table 7(b).

Suggestions
1st: history of futball
2nd: history of lacrosse
3rd: baseball history
4th: history of American futball
5th: history of baseball and steroids

(a) Suggestion by baseline.

Suggestions
1st: baseball history
2nd: baseball origin
3rd: baseball origin game
4th: baseball mean
5th: baseball begin

(b) Suggestion by proposed method.

in the session shown in Table 7(a) never occurred in the query log because the information need of the session was a long-tailed information need. For the struggling flow graph method, we first distinguish the struggling component of the session and then mine the effective struggling phrases to generate query suggestions, that maybe effective for the user's information need.

Another example struggling session is shown in Table 7(b), where the user struggled to find information about *the origin of baseball*. The query suggestions generated by the query flow graph are shown in Table 9(a), and the query suggestions generated by the struggling flow graph are shown in Table 9(b). Several query suggestions were generated by the query flow graph; however, none were effective because they were either failed queries for the same information need or a query for another information need. As noted previously, most queries in a struggling session are failed queries and the query flow graph aggregates all information from a query log and does not guarantee consistency of the information need. The struggling flow graph method only retrieved sessions that with potentially the same struggling component, which guarantees consistency to some degree and helps to generate several query suggestions that may satisfy the user's information need.

7. Conclusion

In this paper, we proposed a query suggestion method to help struggling search based on identifying the struggling component of a struggling session and mine effective representations using a struggling flow graph. We conducted an experiment to investigate the effectiveness of the proposed method compared to a baseline. The experimental results show that the proposed method outperformed the baseline method in terms of support rate, effective support rate@10, MRR@10, and nDCG@10. In the future, we plan to improve the model for identifying the struggling phrase and examine methods to improve the prediction of the user's information need.

Acknowledgement This work was supported in part by KAKENHI (#15H01718).

References

[1] Ahmed Hassan, Ryen W White, Susan T Dumais, and Yi-Min Wang. Struggling or exploring?: disambiguating long search sessions. In *WSDM'14*, pages 53–62, 2014.

[2] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *KDD'00*, pages 407–416, 2000.

[3] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *Current Trends in Database Technology-EDBT 2004 Workshops*, pages 588–596, 2005.

[4] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. Query suggestion using hitting time. In *CIKM'08*, pages 469–478, 2008.

[5] 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 *KDD'08*, pages 875–883, 2008.

[6] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna. Query suggestions using query-flow graphs. In *Workshop on Web Search Click Data*, pages 56–63, 2009.

[7] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. The query-flow graph: model and applications. In *CIKM'08*, pages 609–618, 2008.

[8] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *WWW'06*, pages 387–396, 2006.

[9] Gary Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4):41–46, 2006.

[10] David Carmel, Elad Yom-Tov, Adam Darlow, and Dan Pelleg. What makes a query difficult? In *SIGIR'06*, pages 390–397, 2006.

[11] Anne Aula, Rehan M Khan, and Zhiwei Guan. How does search behavior change as search becomes more difficult? In *CHI'10*, pages 35–44, 2010.

[12] Chang Liu, Jingjing Liu, Michael Cole, Nicholas J Belkin, and Xiangmin Zhang. Task difficulty and domain knowledge effects on information search behaviors. In *ASIST'12*, 49(1):1–10, 2012.

[13] Yang Liu, Ruihua Song, Yu Chen, Jian-Yun Nie, and Ji-Rong Wen. Adaptive query suggestion for difficult queries. In *SIGIR'12*, pages 15–24, 2012.

[14] Daan Odijk, Ryen W. White, Ahmed Hassan Awadallah, and Susan T. Dumais. Struggling and success in web search. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1551–1560, New York, NY, USA, 2015. ACM.

[15] Ting Yao, Min Zhang, Yiqun Liu, Shaoping Ma, and Liyun Ru. Empirical study on rare query characteristics. In *WI'11*, pages 7–14, 2011.

[16] Francesco Bonchi, Raffaele Perego, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. Recommendations for the long tail by term-query graph. In *WWW'11*, pages 15–16, 2011.

[17] Francesco Bonchi, Raffaele Perego, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. Efficient query recommendations in the long tail via center-piece subgraphs. In *SIGIR'12*, pages 345–354, 2012.

[18] Idan Szpektor, Aristides Gionis, and Yoelle Maarek. Improving recommendation for long-tail queries via templates. In *WWW'11*, pages 47–56, 2011.

[19] Ryen W White and Susan T Dumais. Characterizing and predicting search engine switching behavior. In *CIKM'09*, pages 87–96, 2009.

[20] Rosie Jones and Kristina Lisa Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM'08*, pages 699–708, 2008.