Knowl Inf Syst
DOI 10.1007/s10115-013-0620-3

REGULAR PAPER

Intent mining in search query logs for automatic search script generation

Chieh-Jen Wang · Hsin-Hsi Chen

Received: 2 January 2012 / Revised: 23 July 2012 / Accepted: 20 January 2013

Abstract Capturing users' information needs is essential in decreasing the barriers in information access. This paper mines sequences of actions called *search scripts* from search query logs which keep large-scale users' search experiences. Search scripts can be applied to guide users to satisfy their information needs, improve the search effectiveness of retrieval systems, recommend advertisements at suitable places, and so on. Information quality, query ambiguity, topic diversity, and document relevancy are four major challenging issues in search script mining. In this paper, we determine the relevance of URLs for a query, adopt the Open Directory Project (ODP) categories to disambiguate queries and URLs, explore various features and clustering algorithms for intent clustering, identify critical actions from each intent cluster to form a search script, generate a nature language description for each action, and summarize a topic for each search script. Experiments show that the complete link hierarchical clustering algorithm with the features of query terms, relevant URLs, and disambiguated ODP categories performs the best. Applying the intent clusters created by the best model to intent boundary identification achieves an F score of 0.6666. The intent clusters then are applied to generate search scripts.

Keywords Intent mining · Query log analysis · Search script generation · Web search enhancement

1 Introduction

Search intents are engendered if users have information needs, and similar search intents are generated by similar information needs. A user's search intent can be as simple as

C.-J. Wang · H.-H. Chen (✉)
Department of Computer Science and Information Engineering,
National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei 10617, Taiwan
e-mail: hhchen@ntu.edu.tw

C.-J. Wang
e-mail: cjwang@nlg.csie.ntu.edu.tw

getting the price of a commodity or as complicated as finding information for planning a trip, which may contain some sub-needs, such as searching for location information, tourist attractions, accommodations, transportations, and so on. Nowadays, search engine companies are proposing many search enhancement techniques to reduce the barriers between information needs and query formulations. For example, Google Instant and Yahoo Direct return search results quickly and automatically as you type and they update the search result pages instantly. These systems predict users' information needs and report the most relevant search results before users finish typing. In this way, users can formulate queries easier than before because they can scan instant feedbacks and do not have to type full query terms. Currently, these enhancement techniques help users get more relevant information quickly and easily if the search intent is simple and clear. Additionally, query expansion is another popular search enhancement technique for industrial search engines. Query expansion focuses on the measures of query similarity and recommends some similar queries to users. Query expansion may encounter problems if information needs refer to more than one subject by several query submissions. This is because some subjects may not be explicitly associated with one another, and there are even no direct clues to expand the related queries. Consider an information need of planning a trip. The transportation and the local climate are important subjects of the information need. When a user submits transportation-related queries like "buying an airline ticket," some queries, such as "cheap airline ticket," "airline ticket sales," etc., may be recommended to provide additional information to the user. Nevertheless, the subjects are dealt with independently by these kinds of search enhancement techniques, and the similarity of any two subject-related queries may be very low for expansion. Thus, the traditional approaches are not able to tailor the subjects of the same information need for recommendations like local climate-related queries.

For a complex information need, relevant information may span many websites of different subject matters, making a single information access insufficient. Instead, accomplishing such a complex information need may require submitting more than one query and browsing through several relevant websites. It is not trivial for users with different backgrounds and search abilities to know what subjects are indispensable and express a clear and exact search query for every subject matter. Using state-of-the-art search enhancement techniques may improve search effectiveness on a subject matter, but it will lose efficiency if the information need refers to several subject matters. In such a case, recommendation systems with sufficient knowledge of various human activities recorded in search query logs for supporting and handling the complex information need become more and more important in the intelligent web search.

Search query logs can be viewed as a large collection of search sessions that keep searching and browsing trails of users to provide an opportunity to deal with the problem. In a session, users represent their search intents by queries and URL clicks. Similar representations demonstrate similar search intents. In this paper, we collect the ways many users complete their search intents and summarize the best solutions for similar information needs. The valuable search experiences embedded in the wisdom of crowds (e.g., search query logs) are mined and provide a solution to guide users in accomplishing their information needs. The solution, called a *search script*, which illustrates collective intelligence of web users mined from search query logs, consists of a sequence of actions. Users can follow the recommended actions step by step to satisfy their information needs. The usefulness of a search script goes beyond the current search enhancement techniques because it ties various subjects of a search intent together to accomplish an information need. The mined search scripts are useful for search need prediction, retrieval effectiveness improvement of retrieval systems, recommending advertisements at suitable places, and so on.

Information quality, topic diversity, query ambiguity, and document relevancy are four challenging issues in search script mining from search query logs. Search query logs suffer from noise because they hold information from naïve users and expert ones. Besides, the same search intent may be expressed in terms of different queries in different term orders. Queries may be ambiguous, and clicked URLs may not be always relevant. Moreover, the intent boundaries are not clear in search query logs. That is, a user's session may contain more than one search intent. In this paper, we propose some strategies to sample users' sessions, determine the relevance of clicked URLs for a query, disambiguate the uses of queries and clicked URLs, explore different clustering algorithms on different search intent representations for intent clustering, extract critical actions in intent clusters and create search scripts, generate natural language descriptions for actions in each search script, and summarize a topic for each search script. In this way, the recommendation to users will be a sequence of ordered actions in natural language descriptions rather than queries only. A set of search scripts are generated to form a database after the search script mining. When a user inputs his/her information need, a search script recommendation system selects the most related search script from the database and recommends it to the user.

In this paper, we present a recommendation system that provides search scripts to guide users through the actions required to satisfy their information needs. The search scripts are automatically generated and are based on the collective intelligence mined from large-scale search query logs. The experimental results show that the search scripts are of high quality and order accuracy, along with covering a wide range of topics of daily life, including shopping, recreation, computers, education, health, travel, etc.

The remainder of this paper is organized as follows. In Sect. 2, we compare our research with others and summarize our contributions. Section 3 gives an overview of the proposed system. Section 4 describes the primary resources used in this study. Section 5 presents the session sampling strategies, relevance estimation of clicked URLs, and disambiguation of queries and clicked URLs for intent clustering. Section 6 describes how to put together sessions of similar search intent in a cluster. In addition, we evaluate the quality of intent clusters by applying them to an intent boundary identification problem. Section 7 shows how to extract critical actions from an intent cluster and summarize a topic for the search script. Section 8 analyzes two action identification algorithms and discusses the findings, along with their implications. Section 9 concludes the remarks.

2 Related work

A session, which is composed of queries and clicked URLs, is considered to be a fundamental unit for intent clustering. Ideally, we require a session to contain a single user's attempt to fill a single information need. In real cases, identifying users and understanding their intents are challenging issues. Practically, a session is often segmented physically by users' actions, such as opening/closing a browser or login/logout from a search engine, or by some heuristic methods, like time cutoffs [21,29] or mean session lengths [16,29]. To distinguish the differences, a single-attempt session and a physically segmented session hereafter are called a *logical session* and a *physical session*, respectively. A physical session may contain more than one logical session. Detecting an intent shift or identifying an intent boundary in a physical session is an important task [8].

Queries are usually short and even ambiguous. To realize the meanings of queries, researchers define taxonomies and classify queries into predefined categories. Broder [7] divided queries into navigational, informational, and transactional types. Nguyen and Kan

[23] characterized queries along four general facets of ambiguity, authority, temporal sensitivity, and spatial sensitivity. Manshadi and Li [20] constructed a hybrid, generative grammar model based on probabilistic context-free rules for classifying queries into finer categories. Gu et al. [13] proposed a cross-domain random walk algorithm that learns query patterns across a large number of searchable domains to discover the query intent. Li et al. [19] used the click graph to determine query intent via semi-supervised learning. Similarly, clicked URLs can be classified into predefined categories. Shen et al. [27] adopted the Open Directory Project (ODP) taxonomy to represent clicked URLs and modeled the topic transition.

Clicked URLs may not be relevant to queries, due to search engine performance and/or users' comprehension. Modeling users' click behaviors has been studied intensively in recent years. Joachims et al. [17] adopted an eye-tracking equipment to observe the users' behaviors when browsing web pages. They found that web pages of higher ranks have higher click probability. The works on click models, for example, cascade model [11], multiple-click model [15], click chain model [14], and dynamic Bayesian network click model [9], predicted the click probability of a web page under different postulations.

Query recommendation is one of the most popular applications of search query log analysis. Suitable queries are recommended to users when they initiate searches. This helps users satisfy their information needs quickly. Methods [3,37] utilizing clickthrough data and search query logs have been proposed. The major issue is how to measure the similarity of queries and recommend similar queries. The similarity may be estimated through using the surface forms of queries, the overlap of clicked document of queries [4,8,35], or the category/content of clicked documents [35].

Understanding the user's search intent underlying a query has been recognized as an important issue of effective user intent mining. The user intent mining generally relies on employing and analyzing search query logs [31], interests shared on social networks [22], and user behavior patterns [25,26,28] to predict users' underlying search intent for improving search performance. Many applications have benefited from taking the users' search intent into account. Ashkan and Clarke [2] developed a method for mining characteristics of the query intent from the content of search engine result pages and the information obtained from query strings for ad clickthrough prediction. Moreover, several studies [12,32] have applied the users' search intent and preference to improve the effectiveness of scientific literature searching.

These previous works consider information from queries and content of clicked URLs for recommending more suitable queries to satisfy users' information needs, but they cannot deal with the cases where information needs refer to more than one subject or information access. Different from their works, we propose a sequence of related and ordered actions to satisfy users' information needs rather than a single subject only. Each action in a script is described by natural language descriptions along with some example queries. To the best of our knowledge, our work learning how users utilize search engines to satisfy their information needs and generating search scripts that can assist users with similar intent has not been investigated previously.

3 System architecture

A search script generation system is outlined in Fig. 1. Due to different backgrounds of users, not all sessions are suitable for intent mining. Besides, a session may contain insufficient information or noise. In this paper, we take a conservative policy to attain a reliable data set for study. Physical sessions in search query logs are selected in the session purification. URL relevance determination estimates the relevance degree of a clicked URL for a query.

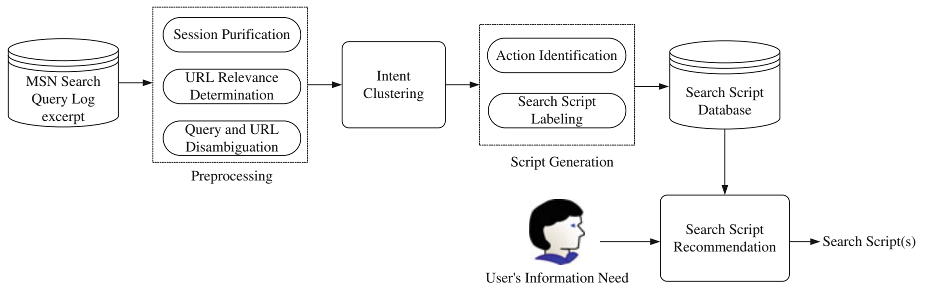

Fig. 1 A system overview of search script generation

Queries and clicked URLs are disambiguated based on the ODP,[1] which is the largest, most comprehensive, and most widely distributed human-compiled taxonomy of websites.

The preprocessed physical sessions are partitioned into intent clusters based on queries, clicked URLs, their corresponding ODP categories, and the category descriptions. In other words, physical sessions of similar search intent are put together as an intent cluster. To evaluate the performance of clustering algorithms on various intent representations, we manually prepared a ground truth. A total of 1,000 physical sessions were sampled, and the intent boundaries of each physical session were annotated. We employed sets of intent clusters created by various models to divide each physical session into logical session(s).

Search scripts are created from the best set of intent clusters in the following way. We identify critical actions from each intent cluster to generate a search script and summarize a topic for the search script. The collection of search scripts created from the set of intent clusters forms a search script database. When a user inputs his/her information need, a search script recommendation system selects the most related search script in the search script database and recommends it to the user. The search script will guide the user in how to accomplish his/her information need.

Figure 2 illustrates an example search script for planning a trip. As shown in Fig. 2, a user inputs a query (e.g., plan a trip), then the recommendation system returns the most related search script to the query, and finally the user can achieve the goal by referring to the action sequence provided by the returned search script. In other words, the goal represents an information need that the user wants to satisfy, and the sequence of actions in the search script contains the plausible steps to reach the goal. A search script consists of a topic and actions in some order. The topic of the search script, for example, Recreation/Travel, is automatically summarized from an intent cluster. Each action consists of a natural language description of the action and some example queries. The search script will be recommended to users who have a search intent of planning a trip.

4 Resource

The MSN Search Query Log excerpt (RFP 2006 data set) [10] is the data source of this work. This data set consists of 14.9 million queries and 12.2 million clicks during a one-month period in May 2006. For each query, its query terms, query time, and the associated session are recorded in the logs. For each click, the clicked URL, the click time, and the associated query are recorded in the logs. In total, there are 7.4 million sessions, where each session records the activity of a user from the time of the first query submitted until a timeout of the search engine or the web browser is closed.

[1] http://www.dmoz.org/about.html.

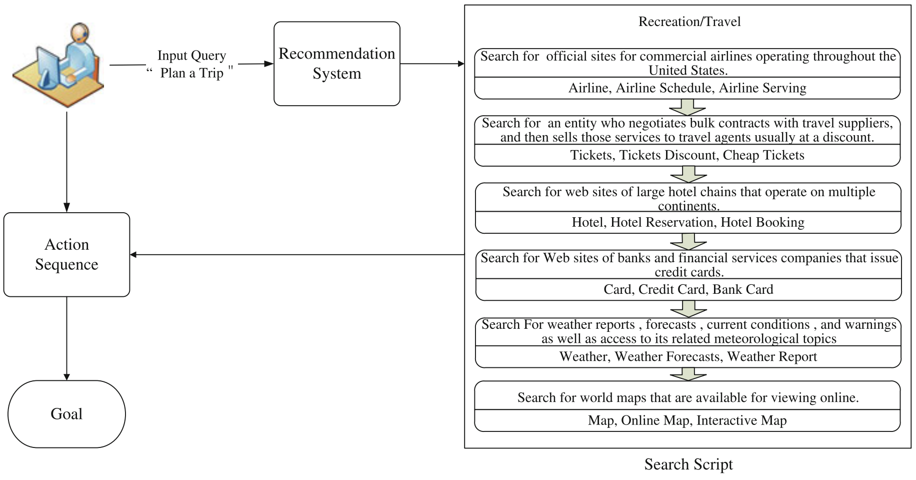
Input Query
" Plan a Trip "
Recommendation System
Recreation/Travel
Search for official sites for commercial airlines operating throughout the United States.
Airline, Airline Schedule, Airline Serving
Search for an entity who negotiates bulk contracts with travel suppliers, and then sells those services to travel agents usually at a discount.
Tickets, Tickets Discount, Cheap Tickets
Search for web sites of large hotel chains that operate on multiple continents.
Hotel, Hotel Reservation, Hotel Booking
Search for Web sites of banks and financial services companies that issue credit cards.
Card, Credit Card, Bank Card
Search For weather reports , forecasts , current conditions , and warnings as well as access to its related meteorological topics
Weather, Weather Forecasts, Weather Report
Search for world maps that are available for viewing online.
Map, Online Map, Interactive Map
Action Sequence
Goal
Search Script

Fig. 2 A search script for planning a trip

The ODP contains more than 4.5 million websites, which are organized into more than 600 thousand paths. A *path* [24] is defined as an ordered hierarchical structure of hyperlink labels from the root category of a directory to a leaf in the ODP. For example, the Google website (http://www.google.com) is assigned a path (Top/Computers/Internet/Searching/Search Engines/Google). The root category is "Top," and "Computers" is a sub-category of "Top." The ODP contains not only website annotations edited by volunteers collaboratively, but also the category description and the path description for websites.

5 Preprocessing

Before intent clustering, we first create a reliable data set by session purification, URL relevance estimation, and category disambiguation. Session purification filters out potential noise in search query logs. URL relevance estimation assigns each URL a relevance degree to a query. The disambiguation algorithm selects more precise ODP categories for a clicked URL.

5.1 Session purification

In this work, we aim to capture complete information needs as possible as we can. A smaller session may not cover the whole set of information; thus, it has a completeness issue. In contrast, a longer session tends to introduce noise if more than one search intent is involved. Therefore, how to trade off the completeness and the noiselessness is important. At the session purification stage, the following filtering strategies are proposed to balance these two issues.

Filtering Strategy #1: A physical session is made up of a sequence of user queries and the clicked URLs. It records the activity of a user from the time of his/her first query submission to the time of a timeout between his/her web browser and the search engine. A user may change his/her search intent in a session, so identifying the intent boundary is important in avoiding including different search intents.

As we do not have a collection of logical sessions for training, we adopt a common time constraint, that is, timeout threshold [21,29], to deal with this problem at this stage. The basic idea is a long duration time between two different queries tends to introduce different search intents. To avoid confusion, we use the term *raw session* to denote the original records with the same session ID in the MSN Search Query Log excerpt.

If the duration between two continuous queries, q_i and q_{i+1}, in a raw session is more than a threshold (e.g., 30 min), we place a cut between these two queries. In other words, the queries along with their clicked URLs before q_{i+1} form a session, and q_{i+1} initiates the beginning of a new session. The same method repeats on the sequence from q_{i+1} until the last query of the raw session. Finally, a raw session is decomposed into one or more smaller sessions.

Filtering Strategy #2: A user may complete his/her goal with few queries in a brief session. For example, she/he may submit a navigation query (e.g., Microsoft), click the official website, and stop the search. A brief session does not provide enough information about the user search intent because we do not know what his/her exact search intent is. His/her search intent may be finding a job in Microsoft or searching for Microsoft's products. The total number of queries in a session is considered as a filtering strategy. Sessions with less than n queries are regarded as brief sessions and will be filtered out. As mentioned in a related work [29], a variety of length cutoffs have been proposed by different research projects. In our experiments, n is set to 3.

Filtering Strategy #3: During searching and browsing, users may click other search engines. The subsequent actions are out of the control of the current search engine, so information is lost after those clicks. The above cases result in an incomplete session. As we have no information about how users interact with other search engines, we adopt a strict strategy—say, sessions containing clicks to other search engines will be removed.

Filtering Strategy #4: Since our system utilizes ODP categories of URLs to disambiguate the search intent of a session, we keep only the sessions in which all clicked URLs appear in the ODP at the training stage. Although many sessions are filtered out with this strategy, sessions with a clear search intent are kept. This is indispensable for the effectiveness of intent clustering. The restriction is relaxed during the testing stage. That is, we will propose an algorithm to deal with the situation where not all URLs can be found in the ODP at the testing stage.

After the filtering process, a total of 17,277 sessions remain. Although not many sessions remain after our strict strategies, this is not a problem because a huge collection of logs is available in the real world. More sessions will be generated if more logs are available. Moreover, the ODP is updated as time goes on; thus, more sessions will remain after the fourth filtering strategy. A *clean* data set is important in deriving *correct* results. The basic idea is avoiding the garbage in and garbage out idea at this stage.

5.2 Relevance determination

Users' clicks depend on various issues, such as performance of search engines, user comprehension, positions of the results, and so on. Clicked URLs express users' interests after reading the corresponding snippets. Thus, a click can be regarded as some sense of relevance voting. Nevertheless, a URL may be irrelevant to the submitted query after a document is browsed. Here, we adopt a function to compute the relevance of a URL while considering position bias [36] at the result page. Given a query q, we collect all of the clicked URLs of q in search query logs at each position p and compute the clickthrough rate (CTR) at each position. Equation (1) defines the relevance degree *relD*(q, u) of URL u for query q.

$$relD(q,u) = \frac{click(q,u)}{\sum_{p=1}^{s} c_p \times \text{CTR}_p} \tag{1}$$

The numerator is the total number of clicks of URL u under query q in search query logs. The denominator can be represented as expected clicks in such a way that c_p denotes the number of URLs u clicked at position p, CTR_p is the CTR at position p in the MSN Search Query Log excerpt, and s denotes the maximum position of URL u clicked in the search results.

5.3 Category disambiguation

We postulate that the sequence of clicked URLs in a session satisfies a user's search intent. Thus, the clicked URLs are coherent and co-related. The clicked URLs surrounding a specific clicked URL form its context. The contextual information is employed to disambiguate the categories of URLs and thus the categories of queries. Of course, the context may be noisy because a clicked URL may not be always relevant to the query or related to the surrounding URLs. The URL relevance degree stated in Sect. 5.2 is introduced into the disambiguation process.

For a URL, we consult the ODP to extract its possible paths. A path in the ODP is represented by $l_1/l_2/\ldots/l_n$, which is an ordered sequence of hyperlink labels $l_1, l_2, \ldots, l_n$ from the root (l_1) of the ODP to the leaf (l_n). The hyperlink labels themselves are categories. The labels on the left in a path dominate the labels on the right. In other words, l_{i+1} is a subcategory of l_i. Since a clicked URL may be classified as more than one ODP category, we have to find its correct meaning. For example, Marriott International's website can be mapped to two possible paths in the ODP. One specifies that Marriott International is an international hotel chain "Top/Recreation/Travel/Lodging/Hotels_and_Motels/Chains" and the other one regards it as a business unit "Top/Regional/North_America/United_States/Maryland/Localities/B/ Bethesda/ Business_and_Economy." Without disambiguation, it is unclear whether a user clicking on Marriott International's URL wants to access Marriott International's hotel functions or view Marriott International's business status.

Categories from different levels denote different specificity. The one in the first level is the root category, and the last is the leaf category. As the categories in the first two levels are too general, such as "Regional," and the categories on the intermediate levels may contain uninformative terms, such as a localized prefix term "B" in the business unit path of Marriott International's website, we shorten the path to include the categories of the third level and the last two levels only. For example, the path "Top/News/News-papers/Regional/United_States/ Indiana" will be shortened into "Newspapers/United_States/Indiana." In this way, a path is condensed into a *concise path*, consisting of three categories.

Assume there are n clicked URLs in a session, that is, $U_1, U_2, \ldots, U_n$ in their clicked chronological order. After looking up the ODP, let a clicked URL U_i corresponds to m paths. The m paths are shortened into m concise paths $CP_{i1}, CP_{i2}, \ldots, CP_{im}$. We arrange all of the concise paths of all of the clicked URLs in a session to form a trellis. Each slice of the trellis contains all of the possible concise paths of a clicked URL, and the slices are sorted by their clicked chronological order. Each node in a slice corresponds to a concise path of a URL. There is a directed edge from every node in a slice j to every node in a slice $j+1$. The goal is to find the most semantically coherent trail through the trellis, where the trail contains a single node from each slice in the trellis.

Figure 3 shows an example of category disambiguation. A total of three URLs have been clicked in a session. Assume URL_1, URL_2, and URL_3 have 1, 2, and 2 concise paths, respectively, after consulting the ODP. There are four possible combinations of trails after traversing all slices in the trellis. Algorithm 1 computes the score of a trail. The score of a node is the sum of similarity between this node and the other nodes in the trail. The similarity of two nodes is the number of common categories between these two nodes. The categories

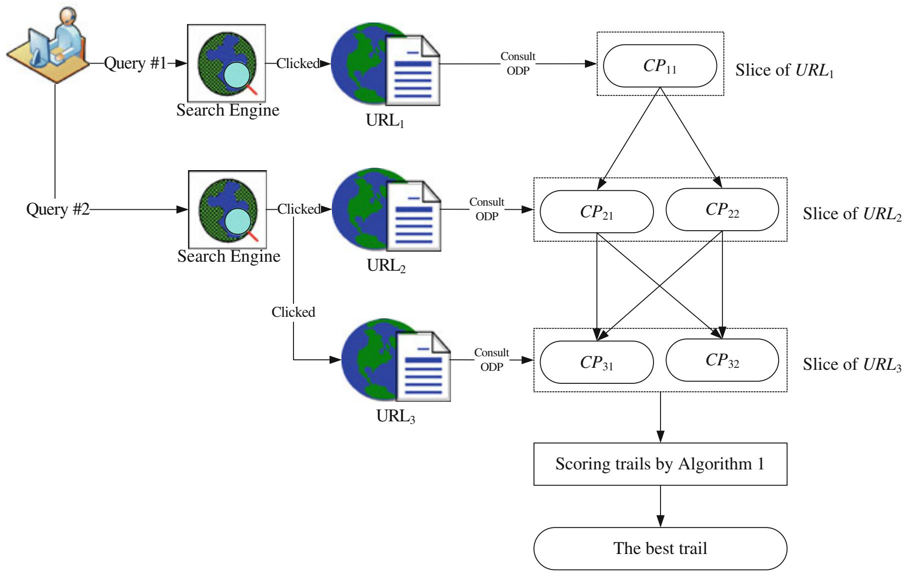

Fig. 3 A category disambiguation example

l_1 and l_3 are the most generic and the most specific categories in a concise path, respectively. The category l_2, which is one level up from l_3, is less specific than l_3. Matching successfully at the more specific level brings clearer meaning than matching at the less specific level; thus, the categories at more specific levels get larger weights. In the experiments, the categories from l_1, l_2, and l_3 are assigned scores 1, 2, and 3, respectively.

Algorithm 1. Computing a score of a trail

```
Input: A trail $p$ in a session
Output: A score of the trail $p$
 1: $s \leftarrow 0$
 2: for each node $n$ in $p$
 3: assume $n$ is the concise path $l_1/l_2/l_3$ of URL $u$ for query $q$
 4:     if ($\exists$ node $m \neq n$ in $p$ and the concise path in $m$ is $l_1/x/y$)
 5:         then $s \leftarrow s + 1 \times relD(q,u)$
 6:     end if
 7:     if ($\exists$ node $m \neq n$ in $p$ and the concise path in $m$ is $x/l_2/y$)
 8:         then $s \leftarrow s + 2 \times relD(q,u)$
 9:     end if
10:     if ($\exists$ node $m \neq n$ in $p$ and the concise path in $m$ is $x/y/l_3$)
11:         then $s \leftarrow s + 3 \times relD(q,u)$
12:     end if
13: end for
14: return $s$ as the score of $p$
```

Table 1 Description of each feature set

Feature name	Description
Q	Query terms as features
U	URLs as features
*rel*U	Relevant URLs as features
C	ODP categories of URLs as features
DC	Disambiguated ODP categories of URLs as features
Q + U	Query terms and URLs as features
Q + C	Query terms and ODP categories as features
Q + *rel*U	Query terms and relevant URLs as features
Q + DC	Query terms and disambiguated ODP categories as features
Q + U + C	Query terms, URLs, and ODP categories as features
Q + *rel*U + DC	Query terms, relevant URLs, and disambiguated ODP categories as features

The score of a trail is the sum of the scores of all of the nodes in the trail. The number of common categories among nodes reflects the degree of intent coherency. Therefore, the trail with the highest score will be selected, and the nodes in the trail contain the disambiguated ODP categories for the clicked URLs.

6 Intent clustering

Two sessions are similar if the search behaviors in them are similar. Sessions of the same search intent will be clustered together to reveal the common actions related to the search intent. Five sets of features are extracted from each session, and the feature weight is determined by binary and *tf-idf* schemes. Two clustering algorithms are employed to cluster sessions of similar search intents. A clustering model is a combination of features and a clustering algorithm. Clustering results will be evaluated by applying them to the application of intent boundary identification, and the best model will be selected for search script generation.

6.1 Feature extraction

Five sets of features, including queries Q, clicked URLs without/with considering relevance (abbreviated as U and *rel*U, respectively), and ODP categories of clicked URLs without/with considering disambiguation (abbreviated as C and DC, respectively) are used to cluster sessions. Table 1 shows the details of each feature and feature combination. In a query, terms are transferred to lower case and stop words are removed. Query terms are the first type of features. Here, a bag-of-words strategy is employed. That is, two queries consisting of the same terms in different orders are regarded as the same query. Queries may suffer from typographical errors, but this issue is neglected in this work. A complete URL is the second type of feature. The third type of feature is similar to the second type except that the relevance degree is considered. The relevance degree determines the feature weight of *rel*U. A concise ODP path is considered in the fourth and fifth types of features. We do not disambiguate the uses of the ODP categories of each URL in the fourth type. In contrast, only the disambiguated ODP categories are selected as features in the fifth type. Different combinations of the above five types of features are shown from the 6th to 11th rows.

The weight of a feature (a query term, a URL, or a category) is determined by two possible schemes: binary or *tf-idf*. In the binary setting, the weight of a feature is set to 1 if the feature appears in the session and 0 otherwise. In the *tf-idf* setting, the weight of a feature is defined in Eq. (2).

$$w_{i,s} = \left(0.5 + \frac{0.5 freq_{i,s}}{\max_s freq}\right) \times \log \frac{N}{n_i} \tag{2}$$

where $freq_{i,s}$ is the frequency of feature i in session s, $\max_s freq$ is the maximum feature frequency in session s, N is total number of sessions, and n_i is the number of sessions in which feature i appears. When the relevance of URLs is considered in a feature set, for example, *rel*U and DC, the weight of a feature is multiplied by *relD*(q, u), a relevance degree of URL u to query q.

6.2 Clustering models

After preprocessing, there are 17,277 sessions for clustering. The complete link and the average link hierarchical clustering algorithms [34] with different sets of features are explored. Euclidean distance determines the similarity between two sessions. In the experimental setup, there are two clustering algorithms, two feature weight representations, and eleven feature combinations. In total, we have 44 clustering models.

The two hierarchical clustering algorithms require a cutoff threshold to separate the agglomerative hierarchical cluster tree into clusters. Different intents may be put into the same cluster when a loose cutoff threshold is adopted. In contrast, a cluster may be divided into more than one smaller cluster when a tight cutoff threshold is used. For all clustering models, setting the threshold to 1 produces more than one cluster and setting the threshold to 2 produces only one cluster. We use Algorithm 2 to search for the largest threshold between 1 and 2 where increasing the value of the threshold that is accurate to five decimal points does not produce a different number of clusters. Each clustering model generates an intent cluster set by a cutoff threshold.

Algorithm 2. Finding a cutoff threshold

```
Input: An agglomerative hierarchical cluster tree
Output: A cutoff threshold $t$
1: $t \leftarrow 1$ and $d \leftarrow 0$
2: while ($d<5$)
3:     $d \leftarrow d + 1$
4:     $n \leftarrow$ the number of clusters using threshold $t$
5:         for ($i \leftarrow 1; i \le 9; i \leftarrow i + 1$)
6:             $s = t + i \times 10^{-d}$
7:             if ($n \neq$ the number of clusters using threshold $s$) then $t \leftarrow s$
8:             end if
9:         end for
10: end while
11: return $t$ as the cutoff threshold
```

6.3 Intent boundary detection

Intent clustering aims to group sessions of similar search intents into an intent cluster. There are two possible approaches to verify the intent clusters. In a direct approach, human assessors examine the sessions in each intent cluster and identify which are correctly placed into the intent cluster and which are not. The benefit of this approach is its fineness, and the drawback is the assessor cost, in particular, checking the results of all clustering models is time consuming. In an indirect approach, the correctness of the intent clusters is verified by applying them to an application. The performance of the application shows which clustering models are better indirectly. The upside of this approach is that verifying large numbers of clustering models is feasible, and the downside is that it cannot tell which session should not be included in an intent cluster.

Here, we adopt the indirect approach to verify the intent clusters. The application in the indirect evaluation is the intent boundary identification. Recall that the sessions in the MSN Search Query Log excerpt are raw sessions that may contain more than one intent. In the experimental setup, we employ each intent cluster set to identify the intent boundary of raw sessions sampled from the MSN Search Query Log excerpt. The performance of identification is dependent on the intent coherency of clusters in the set. The intent cluster set that gains the highest boundary identification accuracy contains the most intent-coherent intent clusters.

Algorithm 3 shows an intent cluster–based boundary detection algorithm. Given a testing session of n queries $q_1, q_2, \ldots, q_n$, there may be a set of intent boundaries, $b_1, b_2, \ldots, b_m$, that is, a set of intent segments $(q_1, q_2, \ldots, q_{b_1})$, $(q_{b_1+1}, q_{b_1+2}, \ldots, q_{b_2}), \ldots, (q_{b_m+1}, q_{b_m+2}, \ldots, q_n)$. At first, we use an approximation strategy to set a potential boundary q_t. Four approximation strategies, including (1) Avg#Queries, a span of n queries forming a potential segment; (2) AvgTime, a span of m minutes forming a potential segment; (3) Q2QTime, the duration of more than 30 min between two continuous queries forming a potential segment; and (4) DynamicCTime, the duration of more than average user comprehension time between a URL click and a query submission forming a potential segment [33], are proposed. In this way, a rough segment, $q_1, q_2, \ldots, q_t$, is derived.

Then, the following procedure selects the most similar intent cluster to adjust the intent boundary. The segment $q_1, q_2, \ldots, q_t$ is represented by queries, clicked URLs, and disambiguated ODP categories. Category descriptions and anchor text of the clicked URLs are also used if they exist. We compute the similarity of the segment with all intent clusters and select the intent cluster with the highest similarity with the segment. We move the boundary to the right to include one more query q_{t+1} and compute the similarity between the enlarged segment and the selected intent cluster. If the similarity becomes lower, that is, q_{t+1} does not belong to the same intent, then we try to move the boundary to the left of q_t, that is, query q_{t-1}, and compute the similarity of the shortened segment with the selected intent cluster. Otherwise, we try to move the boundary to the right again. The right/left movement procedure is repeated until the similarity becomes lower. The intent boundary q_{b_1} is added to the intent boundary set. The above procedure is repeated on $q_{b_1+1}, q_{b_1+2}, \ldots, q_n$ until the final query is considered.

At the training stage, we require that all clicked URLs in a session should be found in the ODP. This is because our system uses ODP categories of URLs in the category disambiguation algorithm. At the testing stage, we have to face the cases where some URLs may not be covered in the ODP. We propose an approximation approach to deal with the ODP coverage problem. This approach condenses an uncovered URL one level at a time and looks up the ODP to check if the condensed URL exists in the ODP. If it does exist, its ODP category is adopted. Otherwise, we condense one more level until either a match is found or a miss is reported. If it still misses, we use the query associated with the uncovered URL to find the approximate ODP categories. The query is submitted to the ODP, and the ten most frequent paths are assigned to the URL.

Algorithm 3. Intent cluster based boundary detection

```
Input: a testing session of $n$ queries $q_1, q_2, \ldots, q_n$, an intent cluster set $H$
Output: a set of intent boundaries, $b_1, b_2, \ldots, b_m$
1: $m \leftarrow 1, i \leftarrow 1$
2: while ($b_m \neq n$)
3:     $j \leftarrow 1$
4:     $q_t \leftarrow$ Approximate $(q_i, q_{i+1}, \ldots, q_n)$
5:     $c^* \leftarrow \mathrm{argmax}_{c \in H}$ Sim$(c, (q_i, q_{i+1}, \ldots, q_t))$
6:     $MaxSim \leftarrow$ Sim$(c^*,(q_i, q_{i+1}, \ldots, q_t))$
7:     if (Sim$(c^*, (q_i, q_{i+1}, \ldots, q_{t+j})) > MaxSim$) then
8:         while (true)
9:             $MaxSim \leftarrow$ Sim$(c^*, (q_i, q_{i+1}, \ldots, q_{t+j}))$
10:            $j \leftarrow j+1$
11:            if (Sim$( c^*, (q_i, q_{i+1}, \ldots, q_{t+j})) < MaxSim$) then
12:                $b_m \leftarrow t+j-1$ and $i \leftarrow t+j$
13:                break while
14:            end if
15:        end while
16:    else if (Sim$( c^*,( q_i, q_{i+1}, \ldots, q_{t-j})) > MaxSim$) then
17:        while(true)
18:            $MaxSim \leftarrow$ Sim$(c^*, (q_i, q_{i+1}, \ldots, q_{t-j}))$
19:            $j \leftarrow j+1$
20:            if (Sim$(c^*, ( q_i, q_{i+1}, \ldots, q_{t-j})) < MaxSim$) then
21:                $b_m \leftarrow t-j+1$ and $i \leftarrow t-j+2$
22:                break while
23:            end if
24:        end while
25:    else
26:        $b_m \leftarrow t$ and $i \leftarrow t+1$
27:    end if
28:    $m \leftarrow m+1$
29: end while
30: return a set of intent boundaries, $b_1, b_2, \ldots, b_m$
```

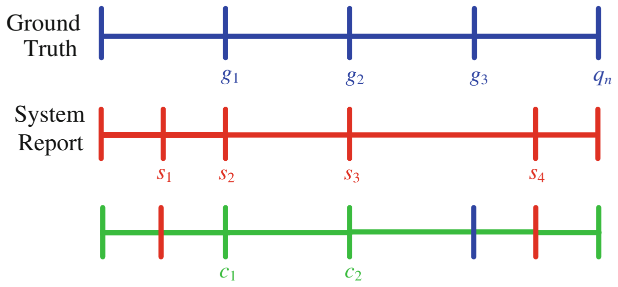

Fig. 4 An evaluation example

Table 2 Using time/query constraints

	Avg#Queries	AvgTime	Q2QTime	DynamicCTime
Recall	0.6203	0.6368	0.6345	0.5229
Precision	0.6195	0.6348	0.6343	0.5235
F-score	0.6196	0.6355	0.6344	0.5217

6.4 Evaluating quality of intent clusters

We randomly selected 1,000 raw sessions from the MSN Search Query Log excerpt, and graduate students majoring in computer science manually annotated the intent boundaries in each session for testing. In total, there were 3,800 queries and 4,651 clicked URLs. After manual annotation, 1,456 boundaries were identified. As previously mentioned, a total of 44 intent cluster sets are generated by different clustering models. Each of the 44 cluster sets was input into Algorithm 3 to detect intent boundaries in the testing sessions.

We evaluated the testing sessions according to the following evaluation metrics. Assume ground truth G and system report S are composed of m and n intent boundaries, respectively. Let $\{c_1, c_2, \ldots, c_k\}$ be k common boundaries between G and S. Precision (P), recall (R), and F score (F) are defined as follows, where $b_i = 1$ if no boundary c exists in G and S such that c is located between c_{i-1} and c_i, otherwise $b_i = 0$.

$$P = \frac{\sum_{i=1}^{k} b_i}{n},\ R = \frac{\sum_{i=1}^{k} b_i}{m},\ F = \frac{2 \times P \times R}{P + R} \tag{3}$$

Figure 4 shows an example. Given a testing session consisting of n queries $q_1, q_2, \ldots, q_n$, there are three intent boundaries in G and four results in S. The common boundaries between them are c_1 and c_2. Only the fragment between c_1 and c_2 forms a correct logical session, so P is 1/5 and R is 1/4.

Table 2 shows the performance of the four baselines that utilize approximation strategies without using intent clusters. The performance of considering comprehension time, that is, DynamicCTime, is behind the other three approximation strategies.

Tables 3 and 4 show the F scores of introducing intent clusters created by the complete link and the average link clustering algorithms into the four baseline models, respectively. Intent clusters generated by different clustering models are compared. Feature weight with binary is better than *tf-idf* when the average link clustering algorithm is adopted. Using U features is better than using Q. This may be because clicked URLs express more clear users' intents and users' queries may be ambiguous. Using C features performs better than using U features. This meets our expectation because an ODP category is a conceptual representation of a URL. Using Q together with U/C is better than using the U/C only. Using Q together with U+C is

Table 3 Introducing the intent clusters created by the complete link clustering algorithm

Feature	Binary + complete link				tf-idf + complete link			
	Avg#Queries	AvgTime	Q2QTime	DynamicCTime	Avg#Queries	AvgTime	Q2QTime	DynamicCTime
Q	$0.6109^{\square}$	$0.6314^{\square}$	$0.6331^{\square}$	$0.6509^{+\blacktriangle\square}$	$0.6193^{\square}$	$0.6386^{+\square}$	$0.6421^{+\square}$	$0.6540^{+*\blacktriangle\square}$
U	$0.6114^{\square}$	$0.6324^{\square}$	$0.6333^{\square}$	$0.6523^{+\blacktriangle\square}$	$0.6193^{\square}$	$0.6392^{+\square}$	$0.6431^{+\square}$	$0.6542^{+*\blacktriangle\square}$
C	$0.6168^{\square}$	$0.6361^{\square}$	$0.6364^{\square}$	$0.6524^{+\blacktriangle\square}$	$0.6205^{\square}$	$0.6406^{+\square}$	$0.6440^{+\square}$	$0.6550^{+*\blacktriangle\square}$
Q + U	$0.6203^{\square}$	$0.6397^{+\square}$	$0.6405^{+\square}$	$0.6553^{+*\blacktriangle\square}$	$0.6227^{\square}$	$0.6431^{+\square}$	$0.6460^{+\square}$	$0.6557^{+*\blacktriangle\square}$
Q + C	$0.6205^{\square}$	$0.6401^{+\square}$	$0.6419^{+\square}$	$0.6564^{+*\blacktriangle\square}$	$0.6247^{\square}$	$0.6445^{+\square}$	$0.6506^{+\square}$	$0.6559^{+*\blacktriangle\square}$
Q + U + C	$0.6236^{\square}$	$0.6412^{+\square}$	$0.6442^{+\square}$	$0.6567^{+*\blacktriangle\square}$	$0.6271^{\square}$	$0.6452^{+\square}$	$0.6507^{+\square}$	$0.6561^{+*\blacktriangle\square}$
*rel*U	$0.6236^{\square}$	$0.6413^{+\square}$	$0.6472^{+\square}$	$0.6571^{+*\blacktriangle\square}$	$0.6272^{\square}$	$0.6452^{+\square}$	$0.6514^{+\square}$	$0.6573^{+*\blacktriangle\square}$
DC	$0.6250^{\square}$	$0.6438^{+\square}$	$0.6492^{+\square}$	$0.6577^{+*\blacktriangle\square}$	$0.6282^{\square}$	$0.6480^{+\square}$	$0.6525^{+\square}$	$0.6583^{+*\blacktriangle\square}$
Q + *rel*U	$0.6319^{\square}$	$0.6509^{+\square}$	$0.6558^{+*\blacktriangle\square}$	$0.6597^{+*\blacktriangle\square}$	$0.6301^{\square}$	$0.6482^{+\square}$	$0.6554^{+*\blacktriangle\square}$	$0.6629^{+*\blacktriangle\square}$
Q + DC	$0.6342^{\square}$	$0.6515^{+\square}$	$0.6595^{+*\blacktriangle\square}$	$0.6625^{+*\blacktriangle\square}$	$0.6310^{\square}$	$0.6495^{+\square}$	$0.6556^{+*\blacktriangle\square}$	$0.6633^{+*\blacktriangle\square}$
Q + *rel*U + DC	$0.6343^{\square}$	$0.6545^{+*\blacktriangle\square}$	$0.6644^{+*\blacktriangle\square}$	$0.6666^{+*\blacktriangle\square}$	$0.6343^{\square}$	$0.6540^{+*\blacktriangle\square}$	$0.6596^{+*\blacktriangle\square}$	$0.6643^{+*\blacktriangle\square}$

Table 4 Introducing the intent clusters created by the average link clustering algorithm

Feature	Binary + complete link				$tf\text{-}idf$ + complete link			
	Avg#Queries	AvgTime	Q2QTime	DynamicCTime	Avg#Queries	AvgTime	Q2QTime	DynamicCTime
Q	$0.6100^{\square}$	$0.6289^{\square}$	$0.6322^{\square}$	$0.6480^{+\square}$	$0.6072^{\square}$	$0.6269^{\square}$	$0.6283^{\square}$	$0.6500^{+\square}$
U	$0.6107^{\square}$	$0.6326^{\square}$	$0.6325^{\square}$	$0.6482^{+\square}$	$0.6081^{\square}$	$0.6287^{\square}$	$0.6285^{\square}$	$0.6502^{+\square}$
C	$0.6146^{\square}$	$0.6349^{\square}$	$0.6355^{\square}$	$0.6518^{+\square}$	$0.6098^{\square}$	$0.6307^{\square}$	$0.6310^{\square}$	$0.6523^{+\square}$
Q + U	$0.6158^{\square}$	$0.6361^{\square}$	$0.6384^{+\square}$	$0.6525^{+\square}$	$0.6113^{\square}$	$0.6307^{\square}$	$0.6310^{\square}$	$0.6530^{+\blacktriangle\square}$
Q + C	$0.6172^{\square}$	$0.6373^{\square}$	$0.6388^{+\square}$	$0.6541^{+*\blacktriangle\square}$	$0.6115^{\square}$	$0.6319^{\square}$	$0.6318^{\square}$	$0.6543^{+*\blacktriangle\square}$
Q + U + C	$0.6172^{\square}$	$0.6373^{\square}$	$0.6394^{+\square}$	$0.6553^{+*\blacktriangle\square}$	$0.6122^{\square}$	$0.6325^{\square}$	$0.6324^{\square}$	$0.6543^{+*\blacktriangle\square}$
*rel*U	$0.6186^{\square}$	$0.6386^{+\square}$	$0.6399^{+\square}$	$0.6556^{+*\blacktriangle\square}$	$0.6127^{\square}$	$0.6333^{\square}$	$0.6336^{\square}$	$0.6546^{+*\blacktriangle\square}$
DC	$0.6187^{\square}$	$0.6403^{+\square}$	$0.6423^{+\square}$	$0.6558^{+*\blacktriangle\square}$	$0.6140^{\square}$	$0.6343^{\square}$	$0.6342^{\square}$	$0.6548^{+*\blacktriangle\square}$
Q + *rel*U	$0.6219^{\square}$	$0.6410^{+\square}$	$0.6430^{+\square}$	$0.6569^{+*\blacktriangle\square}$	$0.6142^{\square}$	$0.6349^{\square}$	$0.6345^{\square}$	$0.6559^{+*\blacktriangle\square}$
Q + DC	$0.6224^{\square}$	$0.6427^{+\square}$	$0.6461^{+\square}$	$0.6585^{+*\blacktriangle\square}$	$0.6165^{\square}$	$0.6377^{\square}$	$0.6377^{\square}$	$0.6564^{+*\blacktriangle\square}$
Q + *rel*U + DC	$0.6271^{\square}$	$0.6431^{+\square}$	$0.6473^{+\square}$	$0.6611^{+*\blacktriangle\square}$	$0.6195^{\square}$	$0.6379^{\square}$	$0.6413^{+\square}$	$0.6610^{+*\blacktriangle\square}$

Table 5 Significance tests of approximation strategies and features/feature combinations using intent clusters created by the complete link clustering algorithm and binary features

	Avg#Queries (■)	AvgTime (•)	Q2QTime (◆)	DynamicCTime
Q (□)		■	■	■, •, ◆
U (○)		■	■	■, •, ◆
C (⊚)		■	■	■, •, ◆
Q+U (☆)		■	■	■, •
Q+C (△)		■	■	■, •
Q+U+C (▽)		■	■	■, •
*rel*U (◿)		■	■	■, •
DC (⊕)		■	■, □, ○	■, •
Q+*rel*U	□, ○, ⊚	■, □, ○	■, □, ○, ⊚, ☆	■
Q+DC	□, ○, ⊚	■, □, ○, ⊚	■, □, ○, ⊚, ☆, △,▽	■
Q+*rel*U+DC	□, ○, ⊚	■, □, ○, ⊚	■, □, ○, ⊚, ☆, △, ▽, ◿, ⊕	■, □

better than using Q together with U/C. Using *rel*U features is better than using all features without considering URLs relevance degree. This indicates that determining the relevance degree of URLs is very important. Using DC feature is better than using *rel*U. This shows that the ODP category after disambiguation can represent the search intent of the clicked URL more clearly. Using Q together with *rel*U/DC is better than using *rel*U/DC only as well. The intent clusters created by integrating the features of Q, *rel*U and DC perform the best.

The chi-squared tests compare the performance of the intent cluster–based approach with the four baselines shown in Table 2. The symbols +, *, ▲, and □ in Tables 3 and 4 show the improvement over the four baseline approximation strategies Avg#Queries, AvgTime, Q2QTime, and DynamicCTime, respectively, and are statistically significant (p value <0.05). Note that the four baselines utilize approximation strategies only without using intent clusters to adjust the intent boundary. As shown in Table 3, the significance tests show that the performance of the intent cluster–based DynamicCTime approach is significantly better than the four baselines, except the AvgTime baseline, when using intent clusters generated by Q, U, and C only with binary weight being adopted. Features from Q together with *rel*U/DC are very important because the performance of the intent cluster–based Q2QTime approach and the intent cluster–based DynamicCTime approach is significantly improved over all the baselines. All intent cluster–based approaches using intent clusters created by integrating the features of Q, *rel*U, and DC significantly outperform the four baselines, except the intent cluster–based Avg#Queries approach.

We further analyze the performance of intent cluster–based approaches using intent clusters generated by the complete link clustering algorithm with binary weight being adopted. Please refer to the left part of Table 3. In Table 5, the impacts of different approximation strategies and feature effectiveness are discussed. The solid symbols ■, •, and ◆ denote that the improvement of the approximate strategies over Avg#Queries, AvgTime, and Q2QTime, respectively, is statistically significant (p value <0.05). The hollow symbols □, ○, ⊚ , ☆, △, ▽, ◿, and ⊕ show that the effectiveness of features over Q, U, C, Q+U, Q+C, Q+U+C, *rel*U, and DC, respectively, is statistically significant (p value <0.05). The former tests are column wise, and the latter ones are row wise. For analyzing the performance of approximation strategies, the experimental results show that DynamicCTime performs better than the other three approximation strategies using intent clusters generated by every feature and feature combination, as shown in Table 3. The performance of DynamicCTime is significantly better than that of the Avg#Queries using intent clusters generated by every feature and feature combination. The performance of DynamicCTime approximation strategy is significantly better than that of AvgTime with almost all features, except the three feature combinations Q + *rel*U, Q + DC, and Q + *rel*U + DC. The performance of DynamicCTime approximation strategy using Q, U, and C only performs significantly better than the corresponding Q2QTime strategy. For the feature effectiveness, intent clusters generated by integrating the features of Q, *rel*U, and DC perform better than the other features for the four approximation strategies, as shown in Table 3. The performance of using intent clusters created by the feature combination Q + *rel*U + DC is significantly better than that of using Q, U, and C only in Avg#Queries and AvgTime approximation strategies. Q2QTime approximation strategy using intent clusters generated by the feature combination Q + *rel*U + DC is significantly outperform that of using the other features, except the two feature combinations Q + *rel*U and Q + DC. DynamicCTime approximation strategy using intent clusters created by the feature combination Q + *rel*U + DC performs significantly better than using that of Q only.

In summary, the intent clusters created by integrating the features of Q, *rel*U, and DC in the complete link clustering algorithm with DynamicCTime perform the best. The model achieving an F score 0.6666 is significantly better than the four baselines in Table 2 (p value <0.001). The results reflect that considering the user comprehension time and intent clusters are quite useful for the intent boundary identification. The best intent cluster set will be studied further in Sect. 7.

7 Search script generation

A search script consists of a sequence of actions to complete an information need. The tasks required to generate a search script from an intent cluster include identifying the critical actions, generating a natural language description for each action, and summarizing the topic. In this section, we will present two action identification algorithms, an action description generation algorithm, and a topic summarization algorithm to generate search scripts in an intent cluster.

7.1 Action identification

To identify the critical actions in an intent cluster, we first transform each session in an intent cluster into a transition graph, where a node denotes a disambiguated ODP category of a clicked URL, and a directed edge from node u to node v means the clicked URL of v immediately follows the clicked URL of u in chronological order. Initially, all the edges radiating outward from a node have the same weight. Then, transition graphs of all of the sessions in the intent cluster are combined into one graph by merging nodes with the same ODP category. The ODP categories denote the purpose of actions in a search script. The weights for an edge are summed during the merge. A search script is extracted from the graph, which contains actions to accomplish an information need.

The basic idea of action identification is to select more significant nodes from a transition graph. We adopt PageRank [1], a link analysis algorithm widely used in search engines for assessing the importance of web pages, to grade the importance of nodes in a graph. The PageRank value measures the degree of significance of a node. A node with more in-links is more significant. In other words, the nodes with low PageRank values, which are less likely to be related to the main search intent of the intent cluster, will be filtered out. In this paper, the PageRank parameters of damping factor, epsilon, and iteration are set to 0.85, 0.000001, and 100, respectively. We compute a PageRank value for each node in a transition graph via the PageRank algorithm. Two action identification algorithms, *TopNodes* and *Connected*, shown in Algorithm 4 and Algorithm 5, respectively, are proposed to select the critical actions, based on the PageRank value of nodes in transition graphs.

The *TopNodes* algorithm sorts the nodes in the descending order of their PageRank values and selects those nodes of PageRank values larger than a threshold. The threshold is determined by the largest PageRank value multiplied by a weight R. This means the selected nodes must be authoritative enough to have at least a fixed percentage (R) of the largest PageRank value. Recall that nodes denote actions. Critical actions are proposed to form a search script by their selection order.

Algorithm 4. *TopNodes*: an action identification algorithm

```
Input: A graph $G$ consisting of a set of nodes $N$ and a set of directed edges $E$, a
weight $R$
Output: Sub-graph(s) of $G$
 1: $H \leftarrow \varnothing$ and $E'' \leftarrow \varnothing$
 2: $n^* \leftarrow \mathrm{argmax}_{n \in N}$ PageRankVal($N, E, n$)
 3: threshold $\leftarrow$ PageRankVal($N, E, n^*$) $\times R$
 4: $H \leftarrow \forall n \in N$, PageRankVal($N, E, n$) > threshold
 5: Sort $H$ and result in a sequence $(n_1, \ldots, n_m)$ where PageRankVal($N,E, n_j$) >=
    PageRankVal($N, E, n_{j+1}$)
 6: $i \leftarrow 1$
 7: while (i < $m$)
 8:       add $(n_i, n_{i+1})$ to $E''$
 9:       $i \leftarrow i+1$
10: end while
11: return $H$ and $E''$
```

Algorithm 5. *Connected*: an action identification algorithm

```
Input: A graph $G$ consisting of a set of nodes $N$ and a set of directed edges $E$, a
weight $R$
Output: A connected sub-graph of $G$
 1: $n^* \leftarrow \mathrm{argmax}_{n \in N}$ PageRankVal($N, E, n$)
 2: $H \leftarrow \{n^*\}, N'' \leftarrow \{n^*\}, \quad E'' \leftarrow \varnothing$
 3: while ($H \neq \varnothing$)
 4:       remove a node $k$ from $H$
 5:       for each node $m \in N$ such that $(k, m) \in E$ or $(m, k) \in E$
 6:             if (PageRankVal($N, E, m$) > PageRankVal($N, E, k$) $\times R$)
 7:                   then add $m$ to $H$ and $N''$ and add $(k, m)$ to $E''$
 8:             end if
 9:       end for
10: end while
11: return $N''$ and $E''$
```

The *Connected* algorithm regards the node of the largest PageRank value as a root and selects a connected subgraph starting from the root. We enforce a criterion on the selection: A node is selected if its PageRank value is at least *R* percentage of the PageRank value of its parent node.

Comparing these two algorithms, the *TopNodes* algorithm ensures that all of the nodes have PageRank values larger than a threshold, but they may not form a connected subgraph. In contrast, the *Connected* algorithm selects a connected subgraph, but the selected nodes may not be the top nodes of higher PageRank values.

7.2 Action description and topic generation

An action corresponds to an ODP category. In the ODP, humans write down a natural language description for each ODP category. For understanding, each action is expressed by a natural language sentence rather than a category. Figure 2 shows a search script example. Each action is marked as "*Search for…*" For example, the third action is "*Search for web sites of large hotel chains that operate on multiple continents.*"

We summarize ODP category descriptions to represent actions. First, we use the Stanford part-of-speech tagger to tag the first sentence of a category description. Next, the first noun following the first verb (e.g., "*is*" and "*contains*") is identified. The entire phrase, starting from the noun to the end of the sentence, is extracted. For readability, we add "*Search for*" to the front of the extracted phrase to form an action description. For example, the first sentence of the category description of the third action is "*This* category*is for central web sites of large hotel chains that operate on multiple continents.*" The first noun of the sentence following the first verb (e.g., "*is*") is "web." Thus, "*Search for*" is added to the front of the phase (e.g., "*web sites of large hotel chains that operate on multiple continents*") to form a natural language description of the action.

A topic summarizes the main search intent in an intent cluster. Recall that a clicked URL is mapped into some proper ODP categories after category disambiguation. An ODP category is represented by a path from the root category to the leaf one. The categories of the 2nd and the 3rd levels of the path are considered as a potential topic because the root category of URLs is "Top," which is meaningless. We exclude potential topics containing "Regional" or "World" category on the 2nd level because they are uninformative for generating topics. In this way, after category disambiguation, each clicked URL corresponds to some potential topics. For each search script, we select the topic mentioned the greatest number of times in an intent cluster.

7.3 Evaluating quality of search scripts

The experiments of the intent boundary identification show that the clustering model using the complete link clustering algorithm with Q (query terms),*rel*U (URLs with relevance degree), and DC (disambiguated categories) features performs the best. A total of 4,020 intent clusters were generated by the clustering model. The intent clusters generated automatically may contain noise. In other words, an intent cluster may contain several search intents. The noise may come from the sessions themselves and/or clustering errors. Search scripts are generated from intent clusters so that it may contain noise as well. The intent purity of a search script is evaluated as follows: (1) single intent, where all actions are classified as the same search intent; (2) major intent, where more than half of the actions correspond to the same search intent; and (3) multiple intents, where more than one search intent is contained in a search script and neither of them dominates the search script.

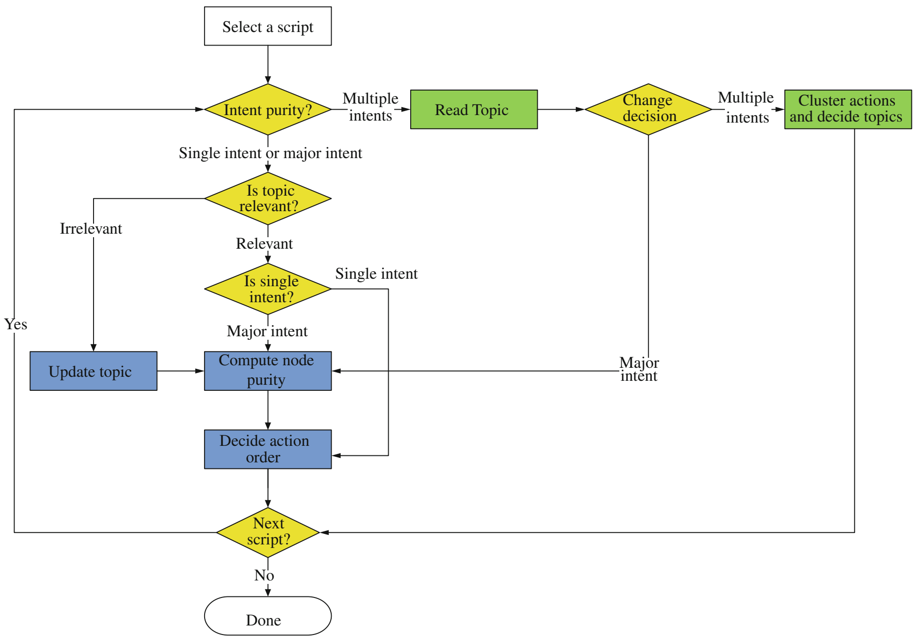

Fig. 5 A search script evaluation procedure

An action identification algorithm generates a search script for each intent cluster. This paper proposes two action identification algorithms, so two search scripts are generated for the same intent cluster. To balance the evaluation loads, a total of ten assessors majoring in computer science and familiar with web searching were recruited for the user study. The search scripts to be examined were divided into ten partitions. As the evaluation is subjective, each search script is examined by two different assessors. Each assessor will check the search scripts generated by the two action identification algorithms in the same intent cluster. To avoid anticipative bias, search scripts were presented to assessors randomly, so that they did not know which search scripts were generated by which algorithms.

Figure 5 shows the evaluation procedure. An assessor first selects a search script and verifies its intent purity. If the search script contains a single intent or a major intent, then the assessor further checks whether the topic generated by the system can capture the search intent of the search script or not. If the system cannot summarize a topic correctly from an intent cluster, the assessor will be asked to provide the correct topic. Moreover, for major intent, there may be some noise actions in the search scripts. Assessors have to point out the noise actions. In this way, we can compute the purity rate of such search scripts. The purity rate is defined as follows.

$$\text{purity rate} = 1 - \frac{\text{number of noise actions}}{\text{total actions}} \tag{4}$$

Besides the correctness of actions, we are also concerned about their order in a search script. Assessors will mark which actions are disordered and list their correct order. As the structure of a search script is a tree, we employ the tree edit distance [6] to measure the order accuracy between the search script generated automatically by our system and the ground truth annotated by assessors. The edit operations between any two trees include

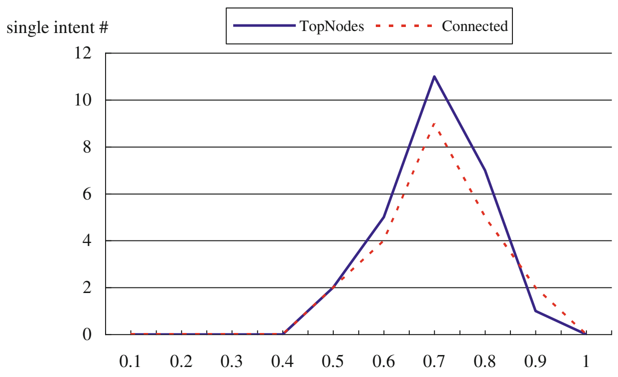

Fig. 6 The number of search scripts containing a single intent with different settings of *R*

insertion, deletion, and substitution. Each edit operation has an associated cost. A sequence of edit operations forms an edit procedure, which changes one tree to another. The cost of an edit procedure is the sum of the costs of its edit operations. Tree edit distance is defined to be the minimum cost of edit procedures between two trees. The order accuracy is defined as follows.

$$\text{order accuracy} = 1 - \frac{\text{tree edit distance}}{\text{total actions}} \tag{5}$$

For those search scripts containing multiple intents, the evaluation platform will present the system-generated topics to assessors. After reading, they may change their decisions from multiple intents to a major intent. If assessors keep their original decisions, they have to point out which actions correspond to which search intent and write down its topic.

A parameter R in the *TopNodes* and *Connected* algorithms affects how many actions will be selected from an intent cluster. In extreme cases, all actions will be selected if R is 0 and only the action with the highest PageRank value will be selected if R is 1. To determine suitable R, we randomly sample twenty intent clusters from a set of intent clusters containing more than two sessions, and we evaluate the search scripts created by the *TopNodes* and *Connected* algorithms with different settings. Figure 6 demonstrates the number of search scripts containing a single intent in relation to R. Both algorithms create the largest number of search scripts containing a single intent when R is set to 0.7. In the experiments, the two algorithms create search scripts from the set of intent clusters proposed by the best clustering model in Sect. 7.3 with this setting.

Search scripts are created from intent clusters. The cost is too high, if all of the search scripts are evaluated by human assessors. We propose two strategies to sample intent clusters for evaluation. An intent cluster containing only one or two sessions cannot provide enough common user behavior to demonstrate the effects of our action identification algorithms. In addition, a search script consisting of one or two actions is more probable to contain a single intent. Thus, we select 1,017 intent clusters containing more than two sessions and the search scripts generated from them consisting of at least three actions.

Since search scripts created from the 1,017 intent clusters are examined by two different assessors, there are a total of 2,034 checks for each action identification algorithm. Table 6 shows the intent purity of search scripts. Over half of the search scripts generated by the *TopNodes* algorithm contain a single intent. This shows the *TopNodes* algorithm is better than the *Connected* algorithm on the intent purity metric. A total of 321 search

Table 6 Intent purity of search scripts

Intent purity	TopNodes	Connected
Single	1,037	1,009
Major	676	704
Multiple	321	321
Total	2,034	2,034

Table 7 Topic relevance/irrelevance under different intent purity

Intent purity	TopNodes		Connected	
	Relevant	Irrelevant	Relevant	Irrelevant
Single	863	174	793	216
Major	458	218	464	240
Total	1,321	392	1,257	456

Table 8 Agreements in purity assessments

	Assessor #2					
	TopNodes			Connected		
	Single	Major	Multiple	Single	Major	Multiple
Assessor #1						
Single	473	43	0	452	59	0
Major	48	271	26	46	281	24
Multiple	0	17	139	0	13	142

scripts contain multiple intents in both algorithms. This occupies 15.78 % of overall search scripts.

Each search script is automatically labeled by our system with a topic denoting its search intent. In the search script evaluation procedure, assessors determine the relevance of the system-generated topic. They do not give judgments to search scripts containing multiple intents. Table 7 shows the results of topic relevance distribution under different intent purity. The topic accuracy is defined in Eq. (6). The search scripts created by the *TopNodes* algorithm have higher topic accuracy (i.e., 0.7711) than those created by the *Connected* algorithm (i.e., 0.7338).

$$\text{topic accuracy} = \frac{\text{\# of topic relevance}}{\text{\# of topic relevance} + \text{\# of topic irrelevance}} \tag{6}$$

Tables 6 and 7 may suffer from personal bias, so we checked inter-assessor agreement further. Table 8 shows agreements in purity assignments by different assessors. For the search scripts created by the *TopNodes* algorithm, both assessors agree that 473, 271, and 139 search scripts contain a single intent, a major intent, and multiple intents, respectively. The search scripts created by the *Connected* algorithm are interpreted in a similar way. Let $I(x)$ denote the total number of search scripts annotated by two assessors with the same intent purity x, where $x \in$ {single, major, multiple}. Equations (7) and (8) define a strict and a lenient accuracy to evaluate the intent purity of search scripts, respectively. The strict accuracy of the *TopNodes* and *Connected* algorithms is 0.4650 and 0.4444, respectively. If we relax

Table 9 Agreements in relevance assessments

	Assessor #2			
	TopNodes		Connected	
	Relevant	Irrelevant	Relevant	Irrelevant
Assessor #1				
Relevant	616	51	591	36
Irrelevant	38	312	39	351

the measurement to include search scripts containing either the single or major intent, the lenient accuracy of the *TopNodes* and *Connected* algorithms increases to 0.7315 and 0.7207, respectively.

$$\text{strict accuracy}_{\text{intent}} = \frac{I\,(\text{single})}{\text{Total scripts}} \tag{7}$$

$$\text{lenient accuracy}_{\text{intent}} = \frac{I\,(\text{single}) + I\,(\text{major})}{\text{Total scripts}} \tag{8}$$

Cohen's Kappa value examines the agreement between two assessors on the assignment of categories, for example, intent purity (single/major/multiple) or topic (relevance/irrelevance). The rule of thumb values of Kappa from 0.40 to 0.59 are considered to be moderate, 0.60–0.79 to be substantial, and 0.80 to be outstanding [18]. Table 8 shows that the Kappa values of the *TopNodes* and *Connected* algorithms on the intent purity aspect are 0.782 with $p < 0.001$ and 0.771 with $p < 0.001$, respectively. Both Kappa values are classified as substantial level.

We also consider the agreement on the relevance assessment of the topic. Let $T\,(y)$ denote the total number of topics annotated by two assessors with the same relevance level y, where $y \in \{\text{relevant, irrelevant}\}$. Table 9 shows the agreements in relevance assessments. The Kappa values of the *TopNodes* and *Connected* algorithms on the relevance aspect are 0.808 with $p < 0.001$ and 0.844 with $p < 0.001$, respectively. Both Kappa values belong to the outstanding level. Strict accuracy and lenient accuracy on the relevance aspect are defined in Eqs. (9) and (10). The strict accuracy of the *TopNodes* and *Connected* algorithms is 0.6057 and 0.5811, respectively. If we relax the restrictions, the lenient accuracy of the *TopNodes* and *Connected* algorithms is 0.7015 and 0.6754, respectively.

$$\text{strict accuracy}_{\text{topic}} = \frac{T\,(\text{relevent})}{\text{Total scripts}} \tag{9}$$

$$\text{lenient accuracy}_{\text{topic}} = \frac{T\,(\text{relevent})}{\text{Total scripts} - I\,(\text{multiple})} \tag{10}$$

Those search scripts containing a single intent and a relevant topic are good ones. Table 10 shows that 405 good search scripts were created by the *TopNodes* algorithm, while the *Connected* algorithm only created 370 good search scripts. The Kappa values of the *TopNodes* and the *Connected* algorithms on the two aspects are 0.879 with $p < 0.001$ and 0.888 with $p < 0.001$, respectively. Both Kappa values are classified as outstanding level. Table 11 lists the relevance assessments for search scripts containing a major intent. The Kappa value of the *TopNodes* and *Connected* algorithms is 0.815 with $p < 0.001$ and 0.775 with $p < 0.001$,

Table 10 Agreements in relevance assessments for search scripts containing a single intent

Single	Assessor #2			
	TopNodes		Connected	
	Relevant	Irrelevant	Relevant	Irrelevant
Assessor #1				
Relevant	405	6	370	5
Irrelevant	7	55	9	68

Table 11 Agreements in relevance assessments for search scripts containing a major intent

Major	Assessor #2			
	TopNodes		Connected	
	Relevant	Irrelevant	Relevant	Irrelevant
Assessor #1				
Relevant	163	10	157	18
Irrelevant	13	85	12	94

respectively. The Kappa values of the *TopNodes* and *Connected* algorithms on the aspect are classified as outstanding and substantial levels, respectively.

Strict accuracy and lenient accuracy are defined in Eqs. (11) and (12). We only consider good search scripts in strict measurement. We relax the metric to include search scripts containing major intent and relevant topic in lenient measurement. The strict accuracy of the *TopNodes* algorithm and the *Connected* algorithm is 0.5443 and 0.5047, respectively. The lenient accuracy of the *TopNodes* algorithm and the *Connected* algorithm is 0.7634 and 0.7189, respectively.

$$\text{strict accuracy}_{\text{intent} \cap \text{topic}} = \frac{I(\text{single}) \cap T(\text{relevant})}{I(\text{single}) + I(\text{major})} \tag{11}$$

$$\text{lenient accuracy}_{\text{intent} \cap \text{topic}} = \frac{I(\text{single}) \cap T(\text{relevant}) + I(\text{major}) \cap T(\text{relevant})}{I(\text{single}) + I(\text{major})} \tag{12}$$

As mentioned before, we define Eqs. (4) and (5) to evaluate the purity rate and the order accuracy of search scripts, respectively. Table 12 shows the average purity rate and the order accuracy. Take the search scripts containing a major intent generated by the *TopNodes* algorithm as an example. A total of 271 search scripts are annotated as the major intent by both assessors. The average number of actions and sessions of each search script is 4.5461 and 7.3152, respectively. Noise actions are actions that are not related to a search script. The average number of noise actions in the 271 search scripts is 345, and the average purity rate is 0.7200. Tree edit distance (TED) means how many operations are needed to revise a search script into a search script in correct order. In this experiment, every operation cost is equal to 1. The average TED of the 271 search scripts is 415, and the average order accuracy is 0.6631. When we also consider search scripts containing a single intent, the average performance of the purity rate and the order accuracy increases to 0.8898 and 0.8272, respectively. The *TopNodes* algorithm is better than the *Connected* algorithm in both average purity rate and average order accuracy.

Table 12 Results of the purity rate and the order accuracy

	TopNodes		Connected	
	Major	Single + major	Major	Single + major
Total search scripts	271	744	281	733
Total actions	1,232	3,131	1,114	2,714
Avg. # of actions	4.5461	4.0148	3.9644	3.7026
Avg. # of sessions	7.3152	9.6745	7.1126	9.3765
Avg. # noise actions	345	345	348	348
Avg. purity rate	0.7200	0.8898	0.6876	0.8718
Avg. TED	415	541	450	571
Avg. order accuracy	0.6631	0.8272	0.5961	0.7896

8 Discussion

In the experiments, the *TopNodes* algorithm performs better than the *Connected* algorithm in nearly all aspects. This is because the *TopNodes* algorithm determines a threshold based on the highest PageRank value in an intent cluster and uses it to select the actions. In this way, it avoids selecting minor or irrelevant actions. In contrast, the *Connected* algorithm determines a threshold based on the PageRank value of the connected parent, so it has a greater probability of selecting minor or irrelevant actions. In our observations, minor actions tend to be annotated as noise by assessors.

For example, some sessions in an intent cluster of the topic*Recreation/Travel* contain actions about baseball games. This means some users may include watching baseball games in their trips. Nevertheless, watching baseball games is not the main activity of their trips, so they do not browse or click many URLs about baseball games. Thus, the sessions are put together to form a *Recreation/Travel* intent cluster rather than *Sports/Baseball* intent cluster. As the frequency of the baseball-related actions is relatively low in the travel intent cluster, their PageRank values are smaller. The *Connected* algorithm, which computes a threshold based on the connected relationship, has a greater probability of accepting minor actions, such as baseball-related actions, in *Recreation/Travel* search scripts. Take another scenario as an example. When users click the URLs related to some media (i.e., newspaper) and see the published advertisements in the media, they may be attracted by the promotional merchandize and have a new search intent of shopping at the moment. Such actions tend to be annotated as noise in a search script as well, because they are not mandatory, that is, the meaning of a search script is not impacted by these actions.

The ordering of actions in a search script is subjective. In some cases, the precedence relationships among actions are clear. Take Fig. 2 as an example. Users survey the possible airlines in the 1st action and try to find a cheap ticket in the 2nd action. Also, the action of buying a ticket from an airline (i.e., the 2nd action) is preferred to precede the action of finding accommodations (i.e., the 3rd action) because users usually make hotel reservations after confirming their flights. The financial service (i.e., the 4th action) is done for the reservation. Nevertheless, the precedence relationships may be fuzzy in some other cases. For example, the 5th and the 6th actions, which refer to finding weather reports and traveling maps, are independent. Either order of one preceding the other is acceptable. To sum up, the ordering of actions in a search script provides a reference procedure for users to satisfy their

Table 13 Agreement between different annotators in order aspect

	TopNodes		Connected	
	Major	Single + major	Major	Single + major
TED	185	221	193	235
Order accuracy	0.8498	0.9294	0.8268	0.9134

information needs, but it may depend on annotators' subjective assessments. The fuzzy cases may influence the order accuracy.

The ordering of actions in a search script is annotated by two different assessors, so the annotated action order may be different. In other words, the two assessors may propose different action orders in fuzzy cases. To evaluate the agreement of assessors in the order aspect, we adopted TED to compare the search scripts corrected by two different assessors. Table 13 shows the TED and the order accuracy. The high-order accuracy illustrates that the action orders annotated by two assessors are similar; thus, the annotation agrees in some sense.

Table 14 shows the topic distribution of the search scripts created by the *TopNodes* algorithm. The top ten topics are listed. The first and the second columns consider the topics of search scripts containing a single intent only and containing a single or a major intent, respectively. Spink et al. [30] and Beitzel et al. [5] analyzed the queries submitted to Excite and AOL search engines, respectively. The topic distributions are listed in the third and the fourth columns for reference. Although the adopted topic categories are not the same in the experiments, the tendency is similar. The top three common topics in the previous studies are (1) Entertainment, (2) Shopping, and (3) Porn in AOL and are (1) Entertainment or recreation, (2) Sex and pornography, and (3) Commerce, travel, employment, or economy in Excite. Our topic distributions are consistent with the prior work, although our topics are generated from the intent clusters rather than queries. The lack of an adult topic in this work is due to the fact that the adult data of the MSN Search Query Log excerpt are placed in another data set.

9 Conclusion and future work

In this paper, we learn users' searching and browsing experiences from the MSN Search Query Log excerpt and generate search scripts of actions to guide users with similar search intents. Several models are proposed for intent clustering. Experiments show that the model integrating the features of query, clicked URLs considering relevance, and disambiguated categories in the complete link clustering algorithm is very useful to group sessions of similar search intents. Two action identification algorithms are proposed to find the critical actions in an intent cluster. We evaluate the search scripts created by action identification algorithms from the aspects of search intent consistency, topic relativity, purity rate, and order accuracy. The user study shows that the *TopNodes* algorithm is better than the *Connected* algorithm in almost all evaluation metrics. The generated search scripts cover the most interesting topics in the real world compared with the topic distribution in other query log data sets, like EXCITE and AOL.

Predicting users' search intents as quickly as possible with the search script database, recommending suitable search scripts to shorten information-seeking time, and selecting proper websites in the search scripts for advertisements are plausible future works. In addition, users from different areas/countries may use different languages to express their search intents. The search scripts may vary due to the culture difference. How to localize the search scripts to meet different lifestyles, cultures, and so on has to be dealt with in the future.

Table 14 Distribution of topic categories for 3 different query logs

Rank	MSN (single)	MSN (single + major)	EXCITE	AOL
1.	Shopping/recreation (11.11 %)	Shopping/recreation (9.75 %)	Entertainment or recreation (19.9 %)	Entertainment (13 %)
2.	Shopping/general merchandize (10.28 %)	Shopping/general merchandize (9.52 %)	Sex and pornography (16.8 %)	Shopping (13 %)
3.	Computers/internet (8.44 %)	Computers/internet (9.52 %)	Commerce, travel, employment, or economy (13.3 %)	Porn (10 %)
4.	Recreation/travel (6.58 %)	Recreation/travel (5.64 %)	Computers or internet (12.5 %)	Research and learn (9 %)
5.	Reference/education (5.35 %)	Reference/education (5.52 %)	Health or sciences (9.5 %)	Computing (9 %)
6.	Arts/television (5.35 %)	Business/financial services (5.41 %)	People, places, or things (6.7 %)	Health (5 %)
7.	Business/financial services (3.50 %)	Arts/television (4.47 %)	Society, culture, ethnicity, or religion (5.7 %)	Home (5 %)
8.	News/newspapers (2.47 %)	News/newspapers (2.35 %)	Education or humanities (5.6 %)	Travel (5 %)
9.	Shopping/health (2.26 %)	Society/relationships (2.23 %)	Performing or fine arts (5.4 %)	Games (5 %)
10.	Games/online games (2.06 %)	Shopping/health (2.23 %)	Non-English or unknown (4.1 %)	Personal and finance (3 %)

Acknowledgments Research of this paper was partially supported by National Science Council, under the contract 99-2221-E-002-167-MY3. We are also grateful to Microsoft Research Asia for the support of MSN Search Query Log excerpt.

References

1. Altman A, Tennenholtz M (2005) Ranking systems: the PageRank axioms. In: Proceedings of the 6th ACM conference on electronic commerce, pp 1–8
2. Ashkan A, Clarke C (2012) Impact of query intent and search context on clickthrough behavior in sponsored search. Knowl Inf Syst 34(2):425–452
3. Baeza-Yates R, Hurtado C, Mendoza M (2005) Query recommendation using query logs in search engines. In: Current trends in database technology—EDBT 2004 workshops, pp 588–596
4. Beeferman D, Berger A (2000) Agglomerative clustering of a search engine query log. In: Proceedings of the 6th ACM SIGKDD international conference on knowledge discovery and data mining, pp 407–416
5. Beitzel SM, Jensen EC, Chowdhury A, Frieder O, Grossman D (2007) Temporal analysis of a very large topically categorized Web query log. J Am Soc Inf Sci Technol 58(2):166–178
6. Bille P (2005) A survey on tree edit distance and related problems. Theor Comput Sci 337(1–3):217–239
7. Broder AZ (2002) A taxonomy of web search. SIGIR Forum 36(2):3–10
8. Cao H, Jiang D, Pei J, He Q, Liao Z, Chen E, Li H (2008) Context-aware query suggestion by mining click-through and session data. In: Proceeding of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 875–883
9. Chapelle O, Metlzer D, Zhang Y, Grinspan P (2009) Expected reciprocal rank for graded relevance. In: Proceeding of the 18th ACM conference on information and knowledge management, pp 621–630
10. Craswell N, Jones R, Dupret G, Viegas E (2009) Proceedings of the 2009 workshop on Web search click data, p 95
11. Craswell N, Zoeter O, Taylor M, Ramsey B (2008) An experimental comparison of click position-bias models. In: Proceedings of the international conference on Web search and Web data mining, pp 87–94
12. El-Arini K, Guestrin C (2011) Beyond keyword search: discovering relevant scientific literature. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, pp 439–447
13. Gu S, Yan J, Ji L, Yan S, Huang J, Liu N, Chen Y, Chen Z (2011) Cross domain random walk for query intent pattern mining from search engine log. In: Proceedings of the 2011 IEEE 11th international conference on data mining, pp 221–230
14. Guo F, Liu C, Kannan A, Minka T, Taylor M, Wang YM, Faloutsos C (2009) Click chain model in web search. In: Proceedings of the 18th international conference on World Wide Web, pp 11–20
15. Guo F, Liu C, Wang YM (2009) Efficient multiple-click models in web search. In: Proceedings of the 2nd ACM international conference on Web search and data mining, pp 124–131
16. Jansen BJ, Spink A, Blakely C, Koshman S (2007) Defining a session on Web search engines: research articles. J Am Soc Inf Sci Technol 58(6):862–871
17. Joachims T, Granka L, Pan B, Hembrooke H, Radlinski F, Gay G (2007) Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search. ACM Trans Inf Syst 25(2):1–27
18. Landis R, Koch G (1977) The measurement of observer agreement for categorical data. Biometrics 33(1):159–174
19. Li X, Wang Y-Y, Acero A (2008) Learning query intent from regularized click graphs. In: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval, pp 339–346
20. Manshadi M, Li X (2009) Semantic tagging of web search queries. In: Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP, vol 2, pp 861–869
21. Montgomery AL, Faloutsos C (2001) Identifying Web browsing trends and patterns. Computer 34(7): 94–95
22. Muhlestein D, Lim S (2011) Online learning with social computing based interest sharing. Knowl Inf Syst 26(1):31–58
23. Nguyen V, Kan M-Y (2007) Functional faceted Web query analysis. In: Query log analysis: social and technological challenges. A workshop at the 16th international World Wide Web conference
24. Perugini S (2008) Symbolic links in the open directory project. Int J Inf Process Manag 44(2):910–930
25. Saleh B, Masseglia F (2011) Discovering frequent behaviors: time is an essential element of the context. Knowl Inf Syst 28(2):311–331
26. Senkul P, Salin S (2012) Improving pattern quality in web usage mining by using semantic information. Knowl Inf Syst 30(3):527–541

27. Shen X, Dumais S, Horvitz E (2005) Analysis of topic dynamics in web search. In: Special interest tracks and posters of the 14th international conference on World Wide Web, pp 1102–1103
28. Shie BE, Hsiao HF, Tseng V (2012) Efficient algorithms for discovering high utility user behavior patterns in mobile commerce environments. Knowl Inf Syst. doi:10.1007/s10115-012-0483-z
29. Silverstein C, Henzinger M, Marais H, Moricz M (1998) Analysis of a very large AltaVista query log. Digital Equipment Corporation, Technical Note
30. Spink A, Jansen BJ, Wolfram D, Saracevic T (2002) From E-sex to E-commerce: Web search changes. Computer 35(3):107–109
31. Wan M, Jönsson A, Wang C, Li L, Yang Y (2011) Web user clustering and Web prefetching using random indexing with weight functions. Knowl Inf Syst 33(1):89–115
32. Wang C, Blei DM (2011) Collaborative topic modeling for recommending scientific articles. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, pp 448–456
33. Wang CJ, Lin KHY, Chen HH (2010) Intent boundary detection in search query logs. In: Proceeding of the 33rd international ACM SIGIR conference on research and development in information retrieval, pp 749–750
34. Ward JH (1963) Hierarchical grouping to optimize an objective function. J Am Stat Assoc 58(301):236–244
35. Wen JR, Nie J-Y, Zhang H-J (2001) Clustering user queries of a search engine. In: Proceedings of the 10th international conference on World Wide Web, pp 162–168
36. Zhang W, Jones R (2007) Comparing click logs and editorial labels for training query rewriting. In: Query log analysis: social and technological challenges. A workshop at the 16th international World Wide Web conference
37. Zhang Z, Nasraoui O (2006) Mining search engine query logs for query recommendation. In: Proceedings of the 15th international conference on World Wide Web, pp 1039–1040

Author Biographies

Chieh-Jen Wang is a Ph.D. Candidate in Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan. His research interests are web mining, information retrieval, and nature language processing.

Hsin-Hsi Chen is a professor in Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan. He served as the chair of the department from 2002 to 2005. From 2006 to 2009, he was the chief director of Computer and Information Networking Center. His research interests are computational linguistics, Chinese language processing, information retrieval and extraction, and web mining. He was the program co-chair of ACM SIGIR 2010 and is the general chair of IJCNLP 2013. He also served as a senior PC member of ACM SIGIR (2006–2009), area/track chairs of ACL 2012, ACL–IJCNLP 2009 and ACM CIKM 2008, and PC members of many conferences (IJCAI, SIGIR, AIRS, ACL, COLING, EMNLP, NAACL, EACL, IJCNLP, WWW, and so on). He has won Google research awards in 2007 and 2012.