www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 4 Issue 8 Aug 2015, Page No. 13674-13677

Identification of User Search Behavior through Task Trail Clustering

[1]Hima G., [2]Jasila E.K.

[1] M-Tech Student, Computer Science and Engineering, MES college of Engineering
Malappuram, Kerala, India
himagnair@gmail.com

[2] Assistant Professor, Computer Science and Engineering, MES college of Engineering
Malappuram, Kerala, India
jasilaabhilash@gmail.com

Abstract: ***Web log is a pouch of valuable information that records users search queries and related actions on the internet. By mining the recorded information, it is possible to exploit the users underlying goals, interests and search behaviors. In order to mine information from web logs, the web logs should be segmented into sessions or tasks by clustering the queries. In this work, Task Trail is introduced to understand user search behaviors. A Task can be defined as set of semantically relevant queries issued to satisfy an atomic user information need. A task trail represents all user activities within the particular task, such as query reformulations, URL clicks. In most of the previous works, web search logs have been studied mainly at session or query level where users may submit several queries within one task and handle several tasks within one session. Although previous studies have addressed the problem of task identification, little is known about the advantage of using task over session or query for search applications. Instead of analyzing Session Trails or Query Trails, Task Trails can be analysed to determine the user search behaviour much more efficiently. By separating different task trails from a session, it can be used in several search applications such as determining user satisfaction, predicting user search interests, and suggesting related queries***.

Keywords: Query clustering, Search engine, Task- based clustering, User search interest, Web log.

1. Introduction

Web logs[1] are a pouch of valuable information that records search queries and related actions of a user on internet. Web logs can be categorized into two types such as Search logs and Browse logs. Search logs are collected from search engines and record the interaction details between search engines and users. These details include queries submitted to search engines, search results returned to users, and clicks made by users. Browse logs are usually collected from client-side browser plug-ins or proxies of Internet Service providers. They record all URLs visited by users, irrespective of search engines and web servers. Web log query clustering is a technique for discovering similar queries on a search engine. The driving force of the development of query clustering techniques comes from the requirements of modern web searching. The web log query clustering techniques can be mainly of Query-level, Session-level and Task-level. The Query level clustering analyses each query in the web log separately. The session-level query clustering technique clusters a set of queries issued by the user of a web search engine within a particular time period. Task-level query clustering clusters a set of non-contiguous queries issued by a user to carry out a particular task. After clustering the queries into sessions or tasks, the web log can be analysed and required knowledge can be extracted. The need of web log analysis is to determine the user behaviour such as user satisfaction, to predict user search interest and to suggest related queries on internet.

2. Related works

Researches are always been conducted to improve the efficiency of web log segmentation with maximum accuracy. This chapter briefly presents some of such effective approaches to segmentation of web log.

Neha Bagoria, and Nirmala Huidrom proposed a threshold based algorithm[2] for Web log clustering. It is a session-level clustering of web log. Threshold based algorithm is the simplest and very basic method that was used for the identification of user-session. The algorithm depends on the proper selection of the threshold value. Based on the threshold value, the sessions are identified. The procedure starts with the proper selection of threshold value. Based on this value, the following condition is checked:

- If the inter-arrival time between the consecutive queries from a web log is less than the threshold value, then the two queries are considered to be in the same session.
- If the inter-arrival time between the consecutive queries is greater than the threshold value, then the two queries are considered to be in different sessions i.e. the first query is the last element of the current session and the second query is the first element of the next session.

Zhenshan Hou, Mingliang Cui [3] proposed a hierarchical conceptual clustering algorithm, COBWEB. The root node is the highest level concepts of the conceptual frame work; while the root node contains the information of all instances. The input objects of the algorithm are described with categorical attribute-value pairs. The COBWEB algorithm adopts the category utility to instruct the tree construction.

D. Beeferman and A. Berger et.al, proposed an agglomerative clustering algorithm[4] for the segmentation of

web search log. It is a task-level segmentation method. The first step of this method is to construct a query-page bipartite graph with one set of the nodes corresponding to the set of queries submitted by the user, and the other set of nodes corresponding to the sets of clicked pages. When a user clicks on a page, a link is created between the query and the page on the bipartite graph. After the bipartite graph is obtained, an agglomerative clustering algorithm is used to discover similar queries and similar pages. During the clustering process, the algorithm iteratively combines the two most similar queries into one query node, then the two most similar pages into one page node. This process of combination of queries and pages is repeated until a termination condition is satisfied. The main reason for not clustering all the queries first and then all the pages next are that two queries may seem unrelated prior to page clustering because they link to two different pages but they may become similar to each other if the two pages have a high enough similarity to each other and are merged later. After the bipartite graph is constructed, the agglomerative clustering algorithm is applied to obtain clusters of similar queries and similar pages.

Claudio Lucchese et.al proposed Query Clustering using Weighted Connected Component (QC-WCC) method[5]. QC-WCC is a graph based algorithm. Upon the query similarity function, an undirected graph is built for queries within a session. The vertices of the graph are queries and the edges are similarity scores between queries. The weighting function is a similarity function that can be easily instantiated in terms of the distance functions. After removing the suspicious edges with scores below a threshold, any connected component of the remain graph is identified as a task. There are mainly two types of similarity measures considered such as Content-based similarity $(\mu_{Content})$ and Semantic-based similarity $(\mu_{Semantic})$. Two queries are considered as similar in content if they share some common terms. Semantic-based similarity is the similarity in the meanings of two queries.

Claudio Lucchese et.al proposed QC-HTC algorithm[5] for task-level web log segmentation. QC-HTC is a variation of the QC-WCC algorithm, which does not need to compute the full similarity graph. Since queries are submitted one after the other by the user, the QC-HTC algorithms takes advantage of this sequentiality to reduce the number of similarity computations needed by QC-WCC. The first step of the algorithm aims at creating an approximate fine-grained clustering of the given time-gap session. Every single web-mediated task generates a sequence of queries and each web-mediated task is observed as a set of fragments, i.e. smaller sets of consecutive queries, and fragments of different tasks are interleaved in the query log because of multi-tasking. The algorithm exploits the sequentiality of user queries, and tries to detect the above fragments, by partitioning the given time-gap session into sequential clusters. The second step of the algorithm merges together the set of fragments (tasks) when they are related, trying to overcome the interleaving of different tasks. Here, the assumption is that a cluster of queries can be described well by just the chronologically first and last (head and tail) of its queries. It reduces the computational cost of the algorithm.

Zhen Liao and Yang Song et.al proposed Query Task Clustering algorithm (QTC)[6] for task-level clustering of web search log. QTC Algorithm is based on the observation that consecutive query pairs are more likely belonging to same task than non-consecutive ones. QTC prefers to first compute the similarities for consecutive query pairs by time stamps. For example, given a sequence of 4 queries q_1, q_2, q_3, q_4 , QC-WCC needs 6 times of pair-wise relevance computations. For QTC, if q_1 is similar to q_2 and q_2 is similar to q_3, there is no need to compute the relevance between q_1 and q_3 any more. If q_1 is similar to q_2 but q_2 is not similar to q_3 , QTC still has to compute the relevance between q_1 and q_3 to avoid the task interleaving. For sessions having multiple tasks, if some tasks have more than two consecutive queries, the time cost can still be reduced for the same reason. In the worst case that all tasks are short and interleaved with each other, QC-SP has the same time complexity as QC-WCC.

In this paper a new approach for web log segmentation is proposed by following the task trail.

3. Proposed work

In this section a new and efficient web log segmentation method is introduced. The main concept of steps involved in the web log segmentation process is explained here.

3.1 Task Definition

Web log contains users search queries and related actions on the internet. Web logs also contains a set of users, and each user has a sequence of consecutive behaviors $e_1, e_2, \ldots e_n$ where each e_i can be a search behavior or a browse behavior[7]. A search behavior is a single query submitted to a search engine. A browse behavior belongs to one of the following activities: 1) user starts to surf from the homepage of the browser; 2) user types a URL address in the browser; 3) user pastes the URL address from other place into browser; 4) user clicks a bookmark or favourite page in the browser; 5) user clicks the "back" or "forward" button in the browser; 6) user clicks an anchor link or a search result. A Query-Trail q represents a sequence of user behaviors ${e_1}^q, {e_2}^q, \ldots {e_m}^q$ of one user u, starting from a query, followed by a sequence of browsing behaviors triggered by this query. A Session Trail s is a sequence of user behaviors ${e_1}^s$, ${e_2}^s, \ldots {e_m}^s$ of one user u, where user behaviors are consecutive in search logs and any two consecutive behaviors e_i, e_{i+1} occurred within time threshold ɵwhere Task Trail t is a sequence of user behaviors ${e_1}^t$, ${e_2}^t, \ldots {e_m}^t$ of one user u occurred within one session, where all user behaviors collectively define an atomic user information need.

3.2 Task Extraction

A task can be defined as a set of semantically relevant query trails within a session. Two queries can be grouped into same task if they satisfy any of the following: (1) they are identical; (2) one is a part of the other (e.g., "sea" and "sea food"); (3) two partially agree to each other (e.g., gate result" and "gate score"); (4) one is a typo of the other (e.g., "machnie learning" and "machine learning"). These rules can be used in the annotation process and propose an efficient clustering framework to group queries into tasks. The basic ideas of our clustering framework are described as follows. First, since tasks are extracted out from each session, we follow the time threshold method to segment logs into sessions by choosing a time threshold u. we quantitatively compute the similarity between any two queries. Last, queries similar to each other are clustered into the same task. A SVM classifier can be used to learn the weights of various features of query similarity function.

3.3 Query Similarity

A linear SVM can be used to compute the similarity between two queries. A labelled data set should be constructed to learn a good query similarity function for task classification. The labels include same task and different task. 11 features are used to measure the similarity between queries. These features can be

categorized into two groups such as time based (temporal) and query word based. The details of these features are given in the following table. Where frequently searched but meaningless words are selected as stop words. The column weight in the table lists the weight of each feature for similarity function. The whole labeled data set can be split into five folds for cross validation. Each time three folds can be used for training, one fold used for tuning parameter, and the rest one fold can be used for testing. Studies on these features showed that using temporal features can only achieve about 70 percent accuracy, using word features can achieve 91 percent accuracy, and combining them can achieve 93 percent accuracy.

3.4 Semantic Similarity Computation Method

WordNet[8] is a semantic database used to establish the connections between four types of Parts of Speech (POS) - noun, verb, adjective, and adverb. The smallest unit in a Wordnet is synset, which represents a specific meaning of a word. It includes the word, its explanation, and its synonyms. So using wordnet is an effective method to find the semantic similarity between two words. Sentences are made up of words, so it is reasonable to represent a sentence using the words in the sentence. This method dynamically forms the semantic vectors solely based on the compared sentences. Recent research achievements in semantic analysis are also adapted to derive an efficient semantic vector for a sentence. Given two sentences, T_1 and T_2, a joint word set is formed:

$$T = T_1 \cup T_2$$
$$= (w_1\ q_2 \ldots\ w_m)$$

The joint word set T contains all the distinct words from T_1 and T_2. Since inflectional morphology may cause a word to appear in a sentence with different forms that convey a specific meaning for a specific context, the word form is used as it appears in the sentence. The joint word set, T, can be viewed as the semantic information for the compared sentences. Each sentence is readily represented by the use of the joint word set as follows: The vector derived from the joint word set is called the lexical semantic vector, denoted by $\tilde{s}$. Each entry of the semantic vector corresponds to a word in the joint word set, so the dimension equals the number of words in the joint word set. The value of an entry of the lexical semantic vector, $\tilde{s_i}$(i=1,2,…m) is determined by the semantic similarity of the corresponding word to a word in the sentence. The information content of a word is derived from its probability in a corpus. Each cell is weighted by the associated information I(w_i) and I($\tilde{w_i}$). Finally, the value of an entry of the semantic vector is:

$$s_i = \tilde{s}\ .\ \mathrm{I}(w_i)\ .\ \mathrm{I}(\tilde{w_i})$$

where w_i is a word in the joint word set, $\tilde{w_i}$ is its associated word in the sentence. The use of I(w_i) and I($\tilde{w_i}$) allows the concerned two words to contribute to the similarity based on their individual information contents. The semantic similarity between two sentences is defined as the cosine coefficient between the two vectors:

$$S_{sim} = \frac{s_1 . s_2}{\|s_1\| \|s_2\|}$$

3.5 Clustering Queries into Tasks

From the intuition that consecutive queries more likely belong to the same task than non-consecutive ones, a better approximation is to compute the pair-wise similarity for all consecutive query pairs. In this work, a clustering algorithm Query Clustering using Modified Bounded Spread method (QC-MBSP) is proposed for task extraction, as shown in Algorithm 1.

Algorithm 1

Input : Query set Q, Cut-off threshold b, Bounded length bl
Output : A set of tasks Θ
Initialization: Θ = ϕ; Query to task table L=ϕ, M=ϕ
Steps:

```
1. // Initialize same queries into one task.
2. cid = 0;
3. for i=1:|Q|-len do
4.    if M[Q_i] exists then
5.       add Q_i into Θ(M[Q_i])
6.    else
7.       M[Q_i]=cid++
8. if |Θ|=1 return Θ
9. for len=1:bl do
10.   for i=1:|Q|-len do
11.      // if two queries are not in the same task,
12.      if L[Q_i] ≠ L[Q_i] then
13.         // Compute lexical similarity
14.         S_1 ← lsim(L[Q_i],L[Q_{i+len}])
15.         // Compute semantic similarity
16.         S_2 ← ssim(L[Q_i],L[Q_{i+len}])
17.         S = S_1 + S_2
18.         if S ≥ b then
19.            merge Θ(Q_i) and Θ(Q_{i+len})
20.            modify L;
21.   // Break if there is only one task
22.   if |Θ| =1 break;
23. return Θ;
```

3.6 User Satisfaction Determination

After the search process, to understand whether a user was satisfied or not in search process, several indirect feedback signals can be used as measures.

● Clicks: The total number of clicks to perform a particular task can be taken as a signal of user satisfaction on that task. Clicking on search results often indicates the relevance between queries and clicked pages.

● Dwell time: Dwell time can also be considered as a signal of user satisfaction. It is because users are more likely to stay on useful pages.

● Markov Model Success Score: The Markov model can be used to model the user's search activities as a sequential process. The Markov model takes queries, clicks, dwelltime(> 30seconds) as states Q, SR, SR_{Long}, respectively. Two Markov models can be built to compute the likelihood of user satisfaction and dissatisfaction. When a new users search activities are given, the score of Markov Models can be computed to determine the label of user satisfaction.

3.7 User Interest Prediction

User search interests can be represented by their queries. Summarizing queries into topics can help understanding user search interests at a higher level. Given two queries submitted by one user, they may come from: (1) different sessions (inter-sessions); (2)same session (intra-session); (3) different tasks in different sessions (inter-tasks among sessions); (4) different tasks in same session (inter-tasks within sessions); (5) same task in same session (intra-task). All these five sources can provide query pairs. Besides, capturing user search interests at topic level is useful to understand user

behaviors. For example, average topic similarity between query pairs from different sessions can help tracing the user search interests during a relative long period. Topic similarity between query pairs from same session can reflect user search interests in a relative short time.

4. Results and Analysis

For the experimental study, one week web log data sets of a particular user is extracted. The data set consists of user browsing logs and search logs from a widely used browser plug-in. It contains URL visits of the user and also contains machine ID, User clicked/visited URLs, as well as queries related to user clicks, a referrer URL where current URL comes from and Time stamps of user events. The web log is segmented at task level, so that the user search behavior can be easily identified. Implementations were done in Java. From the experimental results it can be seen that, the BSP clustering algorithm does not consider the semantic similarity between the query words. But in the case of modified BSP algorithm considers the semantic similarity between the query words also.

Table 1 represents the task identification for BSP clustering algorithm corresponds to different query words. From the table 6.1, it can be observed that the existing BSP algorithm does not consider the semantic similarity between query words before clustering. It only consider the lexical similarity between the query words. That is, the query words with similar spelling but different meaning and different spelling with same meaning are clustered together regardless of
their meaning.

From Table 2, it can be observed that the modified BSP algorithm also considers the semantic similarity between the query words. Since it consider both the lexical similarity and semantic similarity, the query words with same meaning regardless of its spelling are clustered together.

Label	Query A	Query B
Same Task	gmail.com facebook.com face cream for men ticket reservation for cinema	login gmail facebook.com/home ice creams flavours train ticket reservation
Different Task	Healthy food habits Pets Images Heart attack symptoms Red alcoholic drink beautiful kid wallpapers	Pets for sale Variety frokes Cardiac arrest details A bottle of wine pretty baby images

Table 1: Labelled Query Pairs before using wordnet

Label	Query A	Query B
Same Task	gmail.com facebook.com Heart attack symptoms Red alcoholic drink beautiful kid wallpapers	login gmail facebook.com/home Cardiac arrest details A bottle of wine pretty baby images
Different Task	face cream for men ticket reservation for cinema Healthy food habits Pets Images	ice creams flavours train ticket reservation Pets for sale Variety frokes

Table 2: Labelled Query Pairs after using wordnet

Attributes	Precision (%)	Recall (%)
Temporal+Lexical	78.84	73.63
Temporal+Lexical+Semantic	82.61	76.71

Table 3: Accuracy Comparison

From Table 3, it can be observed that, when all the temporal, lexical and semantic attributes are used for query similarity computation, it gives better precision and recall value than using the temporal attributes and lexical attributes only.

5. Conclusion

Web log segmentation can be done in query level, session level, or task level. Task trail is a sequence of user behaviors occurred within one session, where they collectively define an atomic user information need. Following task trail is an effective method to segment the web log and also to determine the user search behaviour. Web logs are segmented into sessions by choosing a time threshold. Queries similar to each other are clustered into same task after computing the query similarity. From the extracted tasks, user search behaviour can be determined.

References

[1] Ryen W. White, Jeff Huang, "Assessing the Scenic Route: Measuring the Value of Search Trails in Web Logs", ACM 978-1-60558-896-4/10/07, Geneva, July 2010.
[2] Nirmala Huidrom, Neha Bagoria, “Clustering Techniques for the identification of Web User Session” , International Journal of Scientific and Research Publications, Volume 3, Issue 1, January 2013.
[3] Zhenshan Hou, Mingliang Cui, Ping Li, "Session Segmentation Method Based on COBWEB", IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, 2012.
[4] D. Beeferman, A. Berger, “Agglomerative Clustering of a Search Engine Query Log”, Proc. ACM SIGKDD,2000.
[5] Lucchese, Claudio, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, Gabriele Tolomei., “Identifying Task-based Sessions in Search Engine Query Logs”, Proceedings of the Fourth ACM International Conference on Web Search and Data Mining – WSDM, 2011.
[6] Claudio Lucchesez, Salvatore Orlando, “Identifying Task based Sessions in Search Engine Query Logs” , Proc. ACM 978-1-4503-0493-1/11/02, 2011.
[7] Liao, Y. Song, Y. Huang, L. He, Q. He,, " Task Trail: An Effective Segmentation of User Search Behavior", IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 12, pp. 3090-3102, 2014.
[8] Li, D. McLean, Z. Bandar, J. O'Shea and K. Crockett, " Sentence similarity based on semantic nets and corpus statistics", IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 8, pp. 1138-1150, 2006.