Mining Test Oracles of Web Search Engines

Wujie Zheng[1], Hao Ma[2], Michael R. Lyu[1], Tao Xie[3] and Irwin King[1,4]
[1]Computer Science and Engineering, The Chinese University of Hong Kong, China
[2]Internet Services Research Center (ISRC), Microsoft Research, Redmond, USA
[3]Department of Computer Science, North Carolina State University, USA
[4]AT&T Labs Research, San Francisco, USA
{wjzheng, lyu, king}@cse.cuhk.edu.hk, haoma@microsoft.com, xie@csc.ncsu.edu

***Abstract*—Web search engines have major impact in people's everyday life. It is of great importance to test the retrieval effectiveness of search engines. However, it is labor-intensive to judge the relevance of search results for a large number of queries, and these relevance judgments may not be reusable since the Web data change all the time. In this work, we propose to mine test oracles of Web search engines from existing search results. The main idea is to mine implicit relationships between queries and search results, e.g., some queries may have fixed top 1 result while some may not, and some Web domains may appear together in top 10 results. We define a set of items of queries and search results, and mine frequent association rules between these items as test oracles. Experiments on major search engines show that our approach mines many high-confidence rules that help understand search engines and detect suspicious search results.**

I. Introduction

Web search engines are becoming more and more important for people to search for information in the World Wide Web. Given a query, a good search engine should return desired search results that possess various properties such as relevance, authority, and freshness. Providing inadequate search results could mislead or dissatisfy users. As an example, Figure 1 shows the clarification message put in the official PuTTY (a free telnet/ssh client) Website due to the unexpected change of Google's search results.

2010-05-17 Google listing confusion

Several users have pointed out to us recently that the top Google hit for "putty" is now not the official PuTTY site but a mirror that used to be listed on our Mirrors page.

The official PuTTY web page is still where it has always been:

http://www.chiark.greenend.org.uk/~sgtatham/putty/

Fig. 1. Declaration from the official PuTTY Website for Google's search result change

However, it is difficult to test search engines due to the lack of test oracles. In particular, since the Web data and the information need of users keep changing, the desired search results may change along the time, even when the search engines do not change. Existing approaches on search engine testing/evaluation rely on relevance judgments of search results, collected either explicitly [1] or implicitly [2]. It is labor-intensive to manually label a large number of relevance judgments of search results, i.e., test oracles for the queries, and these relevance judgments may not be reusable due to the dynamic nature of the Web. On the other hand, implicit relevance judgments such as clickthrough data (the set of results that the users click on) suffer from various biases such as the position bias and summary bias [3]. In particular, if a desired result is not found by a search engine, there is no clickthrough data of it.

In this work, we propose to reduce the effort of manual labeling by mining pseudo test oracles of Web search engines. Previous work on specification mining [4], [5], [6] suggests that one can mine likely invariants or frequent patterns as specifications (i.e., test oracles) from the execution of existing tests. Violations of these mined test oracles are suspicious and may reveal potential faults of the systems under test. Using such approaches, testers of Web search engines can label only the suspicious search results, which are often in a small number, without missing many bugs.

However, mining specifications of Web search engines is a non-trivial task. Many interesting patterns of search engines may need to be mined from search results of multiple days or multiple search engines. We need to integrate all these search results for mining, regardless of the changes in a search engine's implementation or the differences in different search engines' implementations. Existing specification mining approaches often mine patterns with regard to system implementations, and therefore are not suitable for Web search engines.

To address this problem, we define a set of items for search results of Web search engines, and mine rules between these items as pseudo test oracles. Our approach first defines items of queries, search results, matches between queries and search results, and search engine identities. These items reveal many aspects of the search results, and are not affected by the differences of the search engine implementations. Search results of different search engines in different time are transformed to itemsets of these items. Our approach then applies association rule mining [7] to mine rules between the items. The mined rules are saved as pseudo test oracles. Given new search results, our approach detects the search results that violate the mined rules, and presents them to testers for manual labeling.

978-1-4577-1639-3/11/$26.00 © ASE 2011, Lawrence, KS, USA

We evaluate our approach on search results of Google and Bing for 4232 queries within 4 months, which were collected by ourselves. We choose Google and Bing to test because they are the most popular search engines nowadays. These two search engines, together with many other search engines powered by them (e.g., Yahoo! Search is now powered by Bing and AOL Search is powered by Google), possess more than 90 percent search market share in U.S. [8]. The queries consist of 800 common queries used in the KDD-Cup 2005 competition task [9] and 3432 hot queries of Google and Yahoo[1]. We collected the search results of the queries from December 25, 2010 to April 21, 2011. Our approach mines many high-confidence rules that help to understand search engines. Our approach can also detect suspicious search results that may reveal potential search engine faults.

II. Background of Association Rule Mining

Our approach is based on a data mining technique called *association rule mining* [7]. For ease of discussion, we briefly review some basic terminologies of association rules.

Let $I = \{i_1, i_2, \ldots, i_m\}$ denote a set of *items*. Let D be a database of transactions, where each transaction T is a set of items such that $T \subseteq I$. A collection of zero or more items is called an *itemset*. T contains an *itemset* X if $X \subseteq T$. The *support* of an *itemset* X, denoted as sup(X), is the number of transactions in D that contain X. We call an itemset X a *frequent itemset* if its support is large, i.e., sup(X)> *minsup*, where *minsup* is a threshold of support. For example, consider a datebase $\{\{a, b, c\}, \{a, d, e\}, \{a, b\}\}$. The support of the itemset $\{a\}$ is 3 and the support of the itemset $\{a, b, c\}$ is 1. If $minsup = 2$, the frequent itemsets are $\{a\}$, $\{b\}$, and $\{a, b\}$.

Definition 1 (Association rule) *An* association rule *is an implication expression of the form $X \Rightarrow Y$, where $X \subseteq I$ and $Y \subseteq I$ are two disjoint* itemsets*, i.e., $X \cap Y = \emptyset$.*

We can measure the importance of an association rule $X \Rightarrow Y$ using *support* and *confidence*. The *support* of $X \Rightarrow Y$ is equal to the support of the union set $X \cup Y$. The *confidence* of the rule $X \Rightarrow Y$, denoted as conf($X \Rightarrow Y$), is the percentage of transactions containing X that also contain Y. For example, in the example database described above, the support and confidence of the rule $a \Rightarrow b$ is 2 and 2/3, respectively.

For a database of itemsets, the problem of mining association rules is to find all association rules with $support \geq minsup$ and $confidence \geq minconf$, where *minsup* and *minconf* are the corresponding support and confidence thresholds, respectively.

III. Approach

Figure 2 presents an overview of our approach. The main idea of our approach is to mine rules between items of queries and search results automatically. An item describes a property of the input (query) or the output (search results), e.g., a word in the query, the URL domain of a top 10 search result, and whether the URLs contain the query. Our approach first extracts input/output items from existing queries and search results. Our approach then mines association rules of the items. The mined rules are saved as pseudo test oracles. Given new queries and search results, our approach detects suspicious search results that violate mined rules, and presents them to testers for manual labeling.

[1]Bing used to have a service of hot queries named Bing xRank, which however has been shut down.

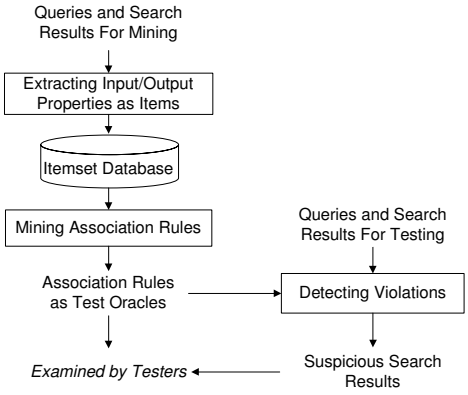

Fig. 2. Approach

TABLE I
Example Items

Category	Item Description	(Example) Items
Query	The query.	Q:ase 2011
Query	A word in the query.	QW:ase
Query	The query type (hot, common).	HotQ
Query	The number of words in the query.	OneWord
Query	The number of words in the query.	TwoWords
Search Result	The domain of the top 1 search result.	top1: continuinged.ku.edu
Search Result	The domain of a top 10 search result.	top10: continuinged.ku.edu
Search Result	The Alexa Ranks of the top 10 results' top private domains are all greater than 1,000.	ALLGE1K
Match	The whole query does not appear in the title of any top 10 result.	NoTitleHasQ
Search Engine	The search engine that returns the search results.	SE:google

A. Extracting Items from Queries and Search Results

We consider items of four broad categories: (i) items based on the query, (ii) items based on the results, (iii) items based on the matching between the query and the results, and (iv) search engine identities. Items of the query category consist of the query words, the query type, the number of words in the query, etc. Items of the search result category consist of the URL domain of the top 1 search result, the URL domains of top 10 search results, etc. Items of the matching category consist of whether the query appears in the title of any top 10 search result, etc. Items of the search engine category consist of the search engine names. Table I provides a list of example items. In general, any properties of the queries and the search

results, such as the link and traffic information of the Webpage that a search result points to, can be used as items.

Given a set of search results (and their queries), our approach extracts the items and builds a database of itemsets for the search results.

B. Mining Association Rules

Our approach employs association rule mining to mine implicit rules between different items. It first mines frequent itemsets from the database. In practice, we may not be interested in rules that contain too many items, because they are too complex to understand and the implications are more likely to be coincidences. Therefore, our approach uses a threshold for the length of rules (i.e., the number of items in the rules), denoted as *maxL*. Given a length threshold *maxL*, our approach adopts the Apriori algorithm [7] to generate frequent itemsets up to length *maxL* iteratively. Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore $(k+1)$-itemsets.

Our approach then generates the rules with high confidences from the frequent itemsets. Given a frequent itemset $X = \{x_1, x_2, ..., x_n\}$, we can generate $2^n - 2$ association rules from it. However, these rules could be highly redundant. Therefore, our approach generates only the rules whose righthand side has just one item from the frequent itemsets. That is, for each x_i in X, our approach generates a rule $\{x_1, ..., x_{i-1}, x_{i+1}, ..., x_n\} \Rightarrow x_i$.

Our approach also designs a set of controlling schemes, including left-hand-side patterns, right-hand-side patterns, and stop words, to guide the rule generation. These patterns specify what items are allowed or disallowed to be parts of association rules. For example, left-hand-side patterns specify the items that can be employed in the left-hand-side of rules. A pattern can represent many possible items, e.g., a "top10:" pattern represents all possible items for the top 10 search results. In this way, we do not need to enumerate all possible items in advance.

Testers may examine the mined rules to learn and examine search engines' behaviors. If there is any rule that seems unreasonable, there may be drawbacks in some modules of the search engine under test.

C. Detecting Violations of Mined Rules

After mining association rules, our approach automatically detects violations of these rules in any given search results. Our approach ranks the mined rules in descending order of confidence and support. Given a set of search results, our approach transforms them to a database of itemsets, and then checks the rules as follows. From top to bottom, our approach picks a rule and checks it against all the itemsets. If the rule is violated by any itemset, which represents a query and its search results, our approach outputs the violation as well as the rule. The testers can then examine the violation, i.e., a suspicious search result, manually.

IV. Evaluation

A. Data Collection

We collect two sets of queries for the evaluation. The first set consists of 800 queries that were used for evaluation in KDD-Cup 2005 Competition [9]. The second set consists of 3432 queries that are collected from Google Trends and Yahoo! Buzz from November 25, 2010 to April 21, 2011. These two indexes provide the hottest queries submitted to the corresponding search engine everyday.

We collect the search results of the prepared queries from December 25, 2010 to April 21, 2011. We apply the Web services of Google and Bing to collect the top 10 search results of each query every day. In total, we collect 390797 ranked lists of search results (each list contains the top 10 search results of a query).

B. Mining Rules as Test Oracles

We apply our approach to mine rules from search results of Google and Bing during December, 25, 2010 to March 31, 2011. We set $minsup = 200$, $minconf = 0.95$, and $maxL = 2$. For the rules that indicate the best top 1 search results of queries, there may be much fewer supporting documents. Therefore, we mine this kind of rules separately by specifying the left-hand-side and right-hand-side patterns and set $minsup = 20$.

```
1.top10:quotes.nasdaq.com, => top10:finance.yahoo.com,
                                  : 314/314=1.0
2.top10:finapps.forbes.com, => top1:finance.yahoo.com,
                                  : 262/262=1.0
3.top10:absoluteastronomy.com, => SE:bing, : 7657/7657=1.0
4.Q:facebook, => top1:facebook.com, : 182/182=1.0
```

Fig. 3. Example association rules of items

Figure 3 shows some examples of the mined rules. These rules can be classified into three categories: implications between Websites, the different opinions of search engines to certain Websites, and the best top 1 results of queries.

Rules 1 and 2 are example rules between Websites. Rule 1 says that there are 314 itemsets (queries and results) where the top 10 search results contain "quotes.nasdaq.com". In all these 314 itemsets, the top 10 search results always contain "finance.yahoo.com". In other words, the rule says that "quotes.nasdaq.com" being ranked top 10 often implies "finance.yahoo.com "being ranked top 10 for the same query. Rule 2 describes a similar rule. It says that when the top 10 search results contain "finapps.forbes.com", the top 1 search result is always "finance.yahoo.com" in the database.

Rule 3 is an example rule that shows different opinions of search engines to certain Websites. It says that if the top 10 results contain "absoluteastronomy.com", the search engine is likely to be Bing (the confidence is 1.0). In other words, Google seldom ranks the Website as one top 10 result for queries while Bing often does. No matter the Website is good or not, such rules are helpful for understanding the differences

of search engines, and may help testers find drawbacks of search engines' spiders or ranking functions.

Rule 4 is an example rule of the best top 1 search results of queries. It says that for the query "facebook", "facebook.com" is always ranked top 1 by the search engines in the collected search results. Violations of such rules, which might be caused by spamming or phishing Websites, can confuse users and cause user dissatisfaction. On the other hand, many queries may not have stable top 1 results since their meanings are ambiguous and the Web data keep changing. Therefore, it is important to identify whether a query has the most suitable top 1 search result automatically.

Note that we do not always need the search results of multiple search engines in multiple days for mining the rules. For example, implications between Websites can be mined from search results of a single search engine in one day. The different opinions of search engines to certain Websites can be mined from search results of multiple search engines in one day. The best top 1 results of queries can be mined from search results of a single search engine in multiple days.

C. Detecting Violations

We also apply our approach to check the mined rules against the search results of Google and Bing between April 1, 2011 to April 22, 2011. Figure 4 shows an example violation of the mined rules.

```
Q:where to login to john carroll university email, =>
top1:mirapoint.jcu.edu, : 172/180=0.96
```

Fig. 4. An example suspicious search result

The high confidence of the rule in Figure 4 suggests that for the query "where to login to john carroll university email", the URL "mirapoint.jcu.edu" is often the best answer. Both Google and Bing agree on this result for most of the time. We check this URL manually, and find that it is the entrance of the webmail system of the John Carroll University. However, on April 1st, 2011, Bing violates this rule. We check the search results of the query of Bing in that day. The top 1 search result of Bing is the URL "http://www.jcu.edu/index.php", which points to the homepage of the John Carroll University. A manual investigation of the URL shows that it is not easy to get the answer of the query, i.e., the entrance of the mail system, from the URL. Therefore, the change of the top 1 search result for this query is inadequate. Collecting such suspicious cases automatically can help testers identify problems in the search engines more quickly.

V. Related Work

Our approach is related to dynamic specification mining, which mines test oracles from the execution of existing tests. Existing approaches mainly mine three kinds of specifications: temporal models [4], algebraic models [6], and operational models [5], [10]. These approaches focus on mining models of a specific system implementation. Instead, our approach designs a set of items for the inputs and outputs of search engines, so as to integrate the search results of different search engines in different time for mining.

Using the mined test oracles, our approach can reduce the efforts of manually labeling search results, essentially a test selection task. Various kinds of code coverage criteria have been proposed for test selection [11]. Dickinson et al. [12] employed clustering analysis to select executions from clusters for result inspection. These approaches select tests based on the information related to the system implementation, while our approach mines frequent patterns in the system level and thus can easily integrate the tests of different systems in different time.

VI. Conclusion

We propose to mine test oracles of Web search engines from existing search results. We define a set of items of queries and search results, and mine frequent association rules between these items as test oracles. We collect a data set that contains the search results of two major search engines, namely Google and Bing, for 4232 queries in a period of 4 months. Evaluation on this data set shows that our approach mines many high-confidence rules whose violations are suspicious search results for manual investigation.

Acknowledgment

This work was supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 415311 and CUHK 413210), NSF grants CCF-0725190, CCF-0845272, CCF-0915400, CNS-0958235, and ARO grant W911NF-08-1-0443.

References

[1] K. S. Jones and C. van Rijsbergen, "Report on the need for and provision of an "ideal" information retrieval test collection," in *British Library Research and Development Report 5266, University of Cambridge*.

[2] T. Joachims, "Evaluating retrieval performance using clickthrough data," in *Text Mining*, 2003, pp. 79–96.

[3] Y. Yue, R. Patel, and H. Roehrig, "Beyond position bias: examining result attractiveness as a source of presentation bias in clickthrough data," in *WWW*, 2010, pp. 1011–1018.

[4] G. Ammons, R. Bodík, and J. R. Larus, "Mining specifications," in *POPL*, 2002, pp. 4–16.

[5] M. D. Ernst, J. Cockrell, W. G. Griswold, and D. Notkin, "Dynamically discovering likely program invariants to support program evolution," *IEEE Trans. Software Eng.*, vol. 27, no. 2, pp. 99–123, 2001.

[6] J. Henkel and A. Diwan, "Discovering algebraic specifications from java classes," in *ECOOP*, 2003, pp. 431–456.

[7] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB*, 1994, pp. 487–499.

[8] "comscore," http://comscore.com/Press_Events/Press_Releases/2010/10/comScore_Releases_September_2010_U.S._Search_Engine_Rankings.

[9] Y. Li, Z. Zheng, and H. K. Dai, "KDD CUP-2005 report: facing a great challenge," *SIGKDD Explorations*, vol. 7, no. 2, pp. 91–99, 2005.

[10] W. Zheng, M. R. Lyu, and T. Xie, "Test selection for result inspection via mining predicate rules," in *ICSE, Companion Volume*, 2009, pp. 219–222.

[11] J. C. Huang, "An approach to program testing," *ACM Comput. Surv.*, vol. 7, no. 3, pp. 113–128, 1975.

[12] W. Dickinson, D. Leon, and A. Podgurski, "Finding failures by cluster analysis of execution profiles," in *ICSE*, 2001, pp. 339–348.