Static Query Result Caching Revisited

Rifat Ozcan, Ismail Sengor Altingovde, Özgür Ulusoy
Department of Computer Engineering
Bilkent University, 06800, Ankara, Turkey
{rozcan, ismaila, oulusoy}@cs.bilkent.edu.tr

ABSTRACT
Query result caching is an important mechanism for search engine efficiency. In this study, we first review several query features that are used to determine the contents of a static result cache. Next, we introduce a new feature that more accurately represents the popularity of a query by measuring the stability of query frequency over a set of time intervals. Experimental results show that this new feature achieves hit ratios better than those of the previously proposed features.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *search process;* H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *Performance evaluation (efficiency and effectiveness).*

General Terms
Performance, Experimentation.

Keywords
Search engine, static caching, query result caching.

1. INTRODUCTION
Caching query results and inverted lists for individual query terms are two basic methods for improving the efficiency of large scale search engines. It is shown that the latter approach may lead to higher cache hit ratios, simply because the same query terms may appear in several different queries (e.g., see [1]). On the other hand, caching query results would provide more gains in terms of efficiency, especially when the network communication dominates the query processing costs. Furthermore, with the recent advances in the computer hardware, it is now possible to keep the entire inverted index in the main memory, which may reduce the advantages of list caching. A recent study discusses several search engine architectures and concludes that for the optimal performance, a cache should be divided into two parts to include query results and inverted lists [1].

Thus, we envision that caching query results deserves a more detailed evaluation considering its premises in terms of efficiency. This paper concentrates on static query result caching and in particular discusses several query features that can be used to choose the queries to be cached. In addition to widely used query frequency feature, we review and evaluate two other features for the static caching purposes. A new feature based on the stability of query frequency over a set of time intervals (e.g., days, weeks, etc.) is also introduced and its performance is evaluated.

WWW 2008, April 21-25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

2. STATIC QUERY RESULT CACHING
It is known that query frequency distribution of a web search engine follows a Zipf distribution [6]. The static caching mechanism in search engines exploits this fact and attempts to fill the cache with the most popular queries (and their results) by analyzing query logs. It is static in the sense that the cache is populated by the selected queries in batch mode and it is not modified until the next batch update.

Markatos [4] analyses the EXCITE query log and shows that static query result caching is a good choice for small cache sizes, but dynamic caching is better for large cache sizes. In [3], a static-dynamic caching policy is proposed: cache is divided into two parts and one part is reserved for static caching and the other is used for dynamic caching. Both of these works use the query frequency feature to determine the static cache content and fill the cache with the most frequent queries. In [2], an admission policy is proposed to detect infrequent queries that will not be submitted in the future in a dynamic caching environment. Their admission policy uses two stateless features, namely length of the query and number of non-alphabetical characters in the query, and one stateful feature, which is based on the frequency of the queries. In this work, we describe and experiment with several other features in addition to the query frequency.

3. QUERY FEATURES
In what follows, we describe a set of features that can be extracted from a query log and used for selecting the queries to be kept in a static cache.

- **Query frequency (QF)**: The query frequency is a simple yet effective way of determining the queries to be cached [3, 4].
- **Query shareness (QS)**: The shareness of a query is defined in [6] as the number of distinct users who issued the query. This feature may be an indicator of the future popularity of a query, i.e., if only a few users have submitted the same query several times, it may be more probable that the query frequency may experience a significant drop in the future.
- **Query result clicks (QRC)**: Query logs contain the URLs which are clicked by the users after issuing the query. The number of clicks may give an idea about how useful a query is and whether it worth to cache it. For instance, a query with no clicks would probably mean that the users are not satisfied with what is returned, so it might be less possible that the users would pose the same query in the future.
- **Query frequency stability (QFS)**: In this paper, we introduce a new feature to more accurately represent the past popularity of a query and to obtain a more realistic estimate of its future popularity. This feature represents the change in a query's popularity during a time interval. The underlying intuition for this feature is that to be cached, the queries

should be frequent *and* remain frequent over a time period rather than being very frequent only for a short while but disappear afterwards. For instance, in our query log (see Section 4) spanning 12 weeks, the query "whitney houston" appears 491 times in a particular week but no more than 10 times in all other weeks. This query may be cached based on its query frequency, but it is less likely to occur in the future requests. On the other hand, a query having a rather moderate frequency that is uniformly distributed among several weeks is more likely to be encountered in the future, but may not be cached by the query frequency feature. In order to represent this notion of stability in the frequency of queries during a set of time intervals and remedy the shortcomings of the QF feature, we introduce a new feature, QFS, as follows.

Assume that query q_i has the total frequency of f_i in a query log which spans a time period T. Consider that this time period is divided into n equal time intervals and query q_i has the following values of frequency $F_i = \{f_{i1}, f_{i2}, \ldots, f_{in}\}$, one for each T/n time units. Then, the stability of query q_i is defined by the following formula:

$$QFS_i = \sum_{j=1}^{n} \frac{\left| f_{ij} - f_{\mu i} \right|}{f_{\mu i}}$$

where $f_{\mu i} = f_i / n$ is the mean frequency of q_i during T.

4. EXPERIMENTAL RESULTS

We use a subset of the AOL Query Log (http://imdc.datcat.org/collection/1-003M-5) which contains around 20 million queries of about 650K people for a period of 3-months. Our subset contains around 5.5 million queries. Queries submitted in the first 6 weeks of three months constitute the train set (to fill the static cache) with 2.8 million queries. The remaining 2.7 million queries are reserved as the test set. In this paper, the requests for the next page of results for a query are considered as a separate query, as in [5].

The first experiment aims to compare the cache hit ratio of the optimal cache content and the content obtained by the query frequency feature, which is widely used in the literature. The optimal cache content is determined as follows. Queries in the test set are sorted in a decreasing order based on their frequency. Then, queries which do not exist in the train set are dropped from this list. The remaining ranked list can be used to fill the cache of a given size, which would yield the best possible hit ratio for this train and test sets. Figure 1 shows the result of this experiment.

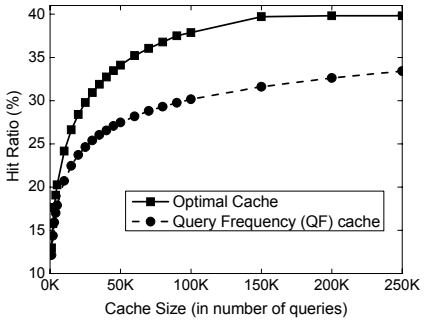

Figure 1. Comparison of the hit ratios for the optimal cache and the cache filled by query frequency feature.

In Figure 1, cache size is given as the number of queries that are stored. It can be seen that there is a significant hit ratio difference between the optimal cache content and the cache content obtained by the QF feature, especially when the cache size is larger than 25K queries. This shows that the state-of-the-art method of choosing the most frequent queries to fill the static cache is far from the optimal and there is still some room for further research.

Table 1 shows the cache hit ratios achieved by using each feature mentioned in Section 3. The QFS feature is computed by using the daily query frequencies, which constitutes a vector of 42 entries (as the train set includes 6 weeks). The findings in Table 1 reveal that the proposed QFS feature performs better than the state-of-the-art QF feature and the other two features. In particular, QFS achieves a (relative) improvement of up to 6% over QF in terms of the cache hit ratio.

Table 1. Hit ratios (%) of query features vs. cache sizes

Cache Size	QF	QS	QRC	QFS
25K	24.6	24.8	23.9	**25.9**
50K	27.4	28.0	26.6	**29.0**
75K	29.1	29.5	28.1	**30.7**
100K	30.2	31.1	29.2	**31.7**
150K	31.6	31.5	30.8	**33.6**
200K	32.6	31.8	31.9	**33.8**
250K	33.4	32.0	32.8	**34.0**

5. CONCLUSION

In this study, we show that solely using the query frequency may not be adequate to reach the optimal hit ratio for the static query result caching. We introduce a more accurate feature based on the stability of frequency and show that it achieves better hit ratios. The future work involves investigating the performance of the QFS feature by using query logs covering a longer time period and in a dynamic caching environment.

6. ACKNOWLEDGMENTS

This work is partially supported by The Scientific and Technical Research Council of Turkey (TÜBİTAK) by the grant # 105E065.

7. REFERENCES

[1] Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Plachouras, V., and Silvestri, F. The impact of caching on search engines. In *Proc. of ACM SIGIR 2007*, 183-190.

[2] Baeza-Yates, R., Junqueira, F., Plachouras, V., and Witschel, H. F. Admission policies for caches of search engine results. In *Proc. of SPIRE 2007*, 74-85.

[3] Fagni, T., Perego, R., Silvestri, F., and Orlando, S. Boosting the performance of Web search engines: Caching and prefetching query results by exploiting historical usage data. *ACM TOIS 24*, 1 (Jan. 2006), 51-78.

[4] Markatos, E. P. On caching search engine query results. *Computing Communications 24* (2001), 137-143.

[5] Sanderson, M., and Dumais, S. Examining repetition in user search behavior. In *Proc. of ECIR 2007*, 597-604.

[6] Xie, Y., and O'Hallaron, D. Locality in search engine queries and its implications for caching. In *Proc. of IEEE INFOCOM 2002*, 1238-1247.