Caching Search Engine Results over Incremental Indices

Roi Blanco
Yahoo! Research
Barcelona, Spain
roi@yahoo-inc.com

Edward Bortnikov
Yahoo! Labs
Haifa, Israel
ebortnik@yahoo-inc.com

Flavio Junqueira
Yahoo! Research
Barcelona, Spain
fpj@yahoo-inc.com

Ronny Lempel
Yahoo! Labs
Haifa, Israel
rlempel@yahoo-inc.com

Luca Telloli
Barcelona Supercomputer Center
Barcelona, Spain
telloli.luca@bsc.es

Hugo Zaragoza
Yahoo! Research
Barcelona, Spain
hugoz@yahoo-inc.com

ABSTRACT

A Web search engine must update its index periodically to incorporate changes to the Web, and we argue in this work that index updates fundamentally impact the design of search engine result caches. Index updates lead to the problem of *cache invalidation*: invalidating cached entries of queries whose results have changed. To enable efficient invalidation of cached results, we propose a framework for developing *invalidation predictors* and some concrete predictors. Evaluation using Wikipedia documents and a query log from Yahoo! shows that selective invalidation of cached search results can lower the number of query re-evaluations by as much as 30% compared to a baseline time-to-live scheme, while returning results of similar freshness.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Performance

Keywords

Search engine, Search results, Cache, Real-time indexing

1. INTRODUCTION

Search engines are often described in the literature as building indices in batch mode [1]. That is, the phases of crawling, indexing and serving queries occur in generations, with generation $n+1$ being prepared in a staging area while generation n is live. The length of each crawl cycle is measured in weeks, implying that the index may represent data that is several weeks stale. In reality, modern search engines try to keep at least some portions of their index relatively up to date. This is realized by modifying the live index rather than replacing it with the next generation. Such search engine indices are said to have incremental indices.

Caching of search results has long been recognized as an important optimization step in search engines [2]. An underlying assumption, however, has been that the same request, when repeated, will result in the same response previously computed. Hence, returning a cached entry does not degrade the application. This does not hold in incremental indexing scenarios, where the corpus is continuously updated and thus the results of any query can potentially change at any time. Namely, the engine must decide whether to re-evaluate repeated queries, thereby reducing the effectiveness of caching, or to save computational resources at the risk of returning stale results. Existing solutions are as simple as foregoing caching altogether and applying time-to-live policies on cached entries to ensure worst-case staleness bounds.

Our goal is to selectively invalidate only those queries whose results are affected by the updates to the underlying index. We formulate this as a *prediction* problem, in which a component that is aware of both the new content being indexed and the contents of the cache, invalidates cached entries it estimates that have become stale. To this end, we propose an architecture for incorporating predictions into search engines, and measure the performance of several prediction policies.

2. CACHE INVALIDATION PREDICTORS

Cache invalidation predictors (or CIP's) bridge the indexing and runtime processes of a search engine, which typically do not interact in search engines operating in batch mode, or limit their interaction to synchronization and locking. Invalidation prediction means that the cache needs to become aware of documents coming into the indexing pipeline. We envision building a CIP in two major pieces (Figure 1):

The synopsis generator: resides in the ingestion pipeline, *e.g.*, right after the *tokenizer*, and is responsible for preparing synopses of the new documents coming in.

The invalidator: receives synopses of documents prepared by the synopsis generator, and through interaction with the runtime system, decides which entries to invalidate.

Our architecture allows composing different synopsis generators with different invalidators, yielding a large variety of behaviors. Below we show how the traditional age-based time-to-live policy (TTL) fits within the framework, and proceed to describe several policies of synopsis generators and invalidators, which we later compose in our experiments.

TTL: age-based invalidation. Age-based policies consider each cached entry to be valid for a certain amount of time after evaluation. Each entry is expired once its age reaches τ. At the two extremes, $\tau = 0$ implies no caching

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

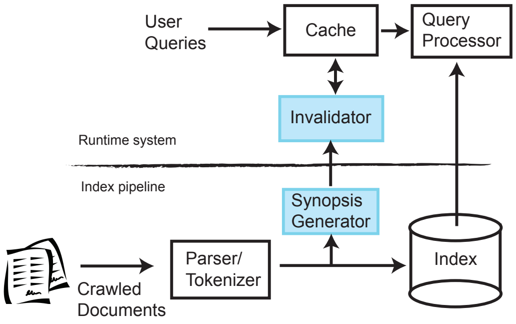

Figure 1: CIP Architecture.

as results must be recomputed for each and every query, whereas with $\tau = \infty$ no invalidation ever happens.

Synopsis generation and selective invalidation. To improve over TTL, we exploit the fact that the cached results for a given query are its top-k scoring documents. By approximating the score of an incoming document to a query we can try to predict whether it affects its top-k results.

The synopsis generator attempts to send compact representations of a document's score attributes, albeit to unknown queries. Its main output is a vector of the document's top-scoring TF-IDF terms [1] for which the document might have a high score. To control the length of the synopsis, the generator sends a fraction of top terms in the vector. Selective (short) synopses will lower the communication complexity of the CIP but will increase its error rate.

Once a synopsis is generated, the CIP invalidators make a simplifying yet mostly accurate assumption that a document (and hence, a synopsis) only affects the results of queries that it *matches*: a synopsis matches query q if it contains all of q's terms in conjunctive query models, or any term in disjunctive models. Then, the invalidator may invalidate all queries matched by a synopsis (note that match computation can be implemented with an inverted index over the cached query set). Alternatively, it can use the same ranking function as the underlying search engine to compute the score of the synopsis with respect to cached query q, and invalidate q iff the computed score exceeds that of q's last result. This projection is feasible for many ranking functions, e.g., TF-IDF, probabilistic ranking, etc. [1]. However, it is imperfect for an incremental index. Cached scores for an incremental index might degrade over time as term statistics of the index drift away.

Finally, similarly to TTL, CIP invalidates all queries whose age exceeds a certain time-to-live threshold. This bounds the maximum staleness of the cached results.

3. EXPERIMENTS

We evaluate multiple CIP instances in a realistic setting. We use the history log of Wikipedia, and assess the performance of predictors on 10,000 cached queries, sampled from the Yahoo! query log. We process the document revisions in single-day batches (*epochs*). In parallel with applying the CIP, we compute the "ground truth" oracle by indexing the epoch and running all queries on epoch boundary, retrieving the top-10 documents per query. A CIP is evaluated by comparing its decisions to the ground truth. We measure the ratio of *false positives* (queries which have been evaluated unnecessarily) versus the ratio of *stale traffic* (queries for which stale results are returned).

We contrast instances of CIP against the TTL policy for a variety of parameters in Figure 2. Our results show that for every point of TTL, there is at least one point of CIP that obtains a significantly lower stale traffic for the same value of false positives. For example, tolerating 6% of stale traffic requires below 20% of false positives, in contrast with TTL's 44.6%. When highest freshness is required, CIP performs particularly well – the number of query evaluations is 30% below the baseline.

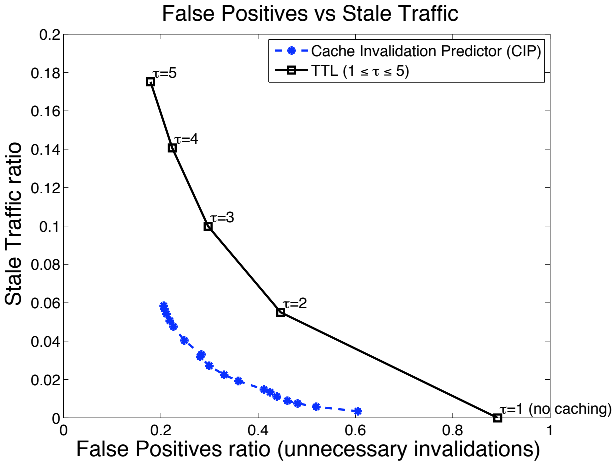

Figure 2: Stale traffic versus False Positives for the best cases of CIP versus the TTL baseline. The τ parameter is measured in epochs.

4. FINAL REMARKS

The implication of our results to the design of caching systems is the following. False positives impact negatively the cache hit rate as they lead to unnecessary misses in our setting. Consequently, selecting a policy that enables a low ratio of false positives is important for performance. With our CIP policies, it is possible to set a desired ratio of false positives as low as 0.2. Lowering the ratio of false positives, however, causes the ratio of stale traffic to increase, which is undesirable when the degree of freshness expected for results is high. A caching system designer must confront such a trade-off and choose parameters according to the specific requirements of precision and performance. Our CIP policies enable such choices and improve over the TTL solution.

5. REFERENCES

[1] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press / Addison Wesley, New York, NY, 1999.

[2] Evangelos P. Markatos. On Caching Search Engine Query Results. *Computer Communications*, 24(2):137–143, 2001.