Using Discard-based Search for Indexed Search

Mahadev Satyanarayanan, Christine Henderson[†],
Brian Adams[†], Rahul Sukthankar[‡]

March 2007
CMU-CS-07-114

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[†]University of Pittsburgh Medical Center (UPMC), [‡]Intel Research Pittsburgh

Abstract

Automated indexing of complex data such as images remains a challenging problem today in spite of extensive research. An alternative approach, called *discard-based search,* uses code fragments called *searchlets* to perform content-based computation in response to a specific query. In this paper, we describe a new, two-phased usage model for discard-based search. In the first phase, human experts use discard-based search to create searchlets that reflect their classification expertise. In the second phase, these searchlets are used to preprocess complex data for indexing. This hybrid approach preserves the positive characteristics of indexed search, while offering flexibility in the creation of searchlets and in tuning their precision-recall characteristics. Most importantly, it offers ample opportunity for human expertise and new knowledge to be efficiently incorporated into the process of indexing complex data.

This research was supported by the National Science Foundation (NSF) under grant number CNS-0614679, and by grant number 1UL1 RR024153-01 from the National Center for Research Resources (NCRR), a component of the National Institutes of Health (NIH) and the NIH Roadmap for Medical Research. The findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, NCRR, NIH, Carnegie Mellon University, UPMC, or Intel.

Keywords: Indexed search, interactive search, Diamond, OpenDiamond, medical images, Stentor, Google, Yahoo!, searchlets, filters, meta-data, tags, annotations, non-indexed data, human expertise

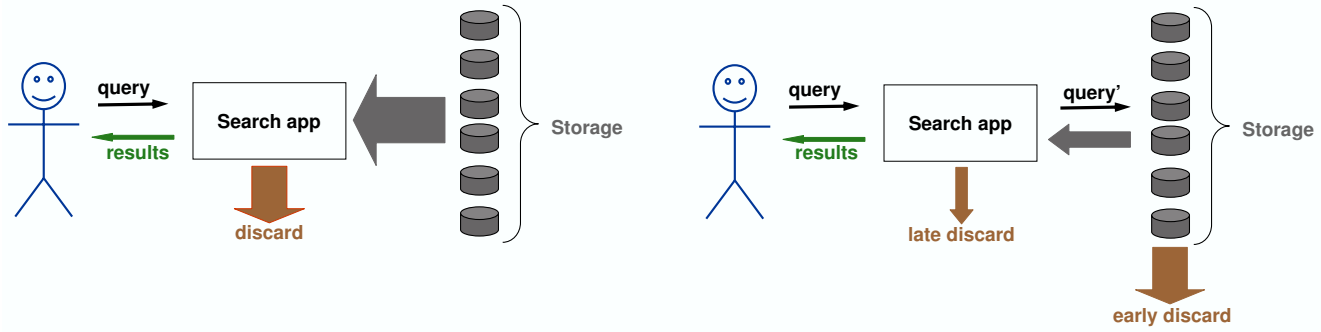
1 Indexed Search

Indexed search has been phenomenally successful in many contexts. Examples include: Web search engines such Google, Yahoo! and Lycos; relational databases such as Oracle, IBM DB2 and Microsoft SQL Server; and Internet-scale image and video services such as Flickr, YouTube, and Informedia. Indexed search rests on three assumptions: first, that the entire space of queries can be anticipated in advance; second, that it is possible to preprocess data to create indexes; third, that the preprocessing can label the data using query-relevant index terms (also called "tags" or "meta-data" or "annotations"). In the recent past, approaches such as the ESP Game [8] have been created to motivate users to provide these labels.

Historically, text has been the data type most thoroughly studied from an indexing viewpoint. Indexing more complex, multi-dimensional data such as images has been studied in systems such as QBIC [3]. However, in spite of extensive research, automated indexing of complex data remains a challenging problem today for several reasons. First, automated methods for extracting semantic content from many data types are still rather primitive. This is referred to as the *semantic gap* [6] in information retrieval. Second, the richness of the data often requires a high-dimensional representation that is not amenable to efficient indexing. This is a consequence of the *curse of dimensionality* [1, 2, 11]. Third, realistic user queries can be very sophisticated, requiring a great deal of domain knowledge that is often not available to the system for optimization. Fourth, expressing a user's vaguely-specified query in a machine-interpretable form can be difficult. These problems constrain the success of indexed search.

2 Discard-based Search

A radically different approach to searching complex data, called *discard-based search,* is being investigated in the Diamond project [5]. The Diamond approach does not attempt to preprocess all data in advance of future queries. Rather, it performs *content-based computation on demand in response to a specific query.* The purpose of the computation is to eliminate objects that are clearly not results for the query. An optimized version of this concept called *early discard* is implemented by the OpenDiamond™ platform for interactive search. The optimization rejects most of the irrelevant data as early as possible in the pipeline from storage to user. This improves scalability since it eliminates a large fraction of the data from most of the pipeline. Figure 1 illustrates the concepts of discard-based search and the early-discard optimization.



(a) Basic Concept (b) Early Discard Optimization

Figure 1: Discard-based Search

Since the knowledge needed to recognize irrelevant data is domain-specific, discard-based search requires domain-specific algorithms to be executed on data objects during a search. Early discard requires all or part of these algorithms to be executed close to storage. Ideally, discard-based search would reject all

irrelevant data without eliminating any desired data. This is impossible in practice because of a fundamental trade-off between false-positives (irrelevant data that is not rejected) and false-negatives (relevant data that is incorrectly discarded) [2]. The best one can do in practice is to tune a discard algorithm to favor one at the expense of the other. Different search applications may make different trade-offs in this space.

3 Relative Merits

The strengths and weaknesses of indexed search and discard-based search complement each other:

Search Speed: Because all data is preprocessed, there are no compute-intensive or storage-intensive algorithms to be run during an indexed search. It is therefore much faster than discard-based search, possibly by many orders of magnitude.

Server security: The early-discard optimization, which is essential for scaling discard-based search to large data volumes, requires searchlet code to be run close to servers. The closer to storage that searchlets can be executed, the more effective the optimization. While a broad range of sandboxing techniques [9], language-based techniques [10], and verification techniques [7] can be applied to reduce risk, the essential point remains that potentially untrusted code may need to run on trusted infrastructure during a discard-based search. This risk does not exist with indexed search, where the preprocessing is done offline by trusted code on trusted infrastructure.

Precision and Recall: The preprocessing for indexed search represents a specific point on a precision-recall curve, and hence a specific design choice in the tradeoff space between false positives and false negatives. In contrast, discard-based search can change this tradeoff dynamically as a search progresses through many iterations. An expert user with extensive domain-specific knowledge may tune searchlets toward false positives or false negatives depending on factors such as the purpose of the search, the completeness of the search relative to total data volume, and the user's expert judgement of results from earlier iterations in the search process. It is also possible to return a clearly-labelled sampling of false negatives during a discard-based search to alert the user to what he might be missing in the results.

New knowledge: The preprocessing for indexing can only be as good as the state of knowledge at the time of indexing. New knowledge may render some of this preprocessing stale. In contrast, discard-based search is based on the state of knowledge of the user at the moment of searchlet creation or parameterization. This state of knowledge may change (hopfully for the better!) even during the course of a search. For example, the index terms used in labelling a corpus of medical data may later be discovered to be incomplete or inaccurate. Some cases of a condition that used to be called "A" may now be understood to actually be a new condition "B." Short of re-indexing the entire corpus, this new knowledge cannot be incorporated into indexed search. Note that this observation is true regardless of how the index terms were obtained, including game-based tagging approaches such as ESP.

User expertise: Discard-based search takes better advantage of the expertise and judgement of the user conducting the search. There are many degrees of freedom, including searchlet creation and parameterization, through which this expertise and judgement can be expressed. In contrast, indexed search limits even the most expert user to the intrinsic quality of the preprocessing that produced the index.

4 Using Discard-based Search for Indexed Search

The strengths and weaknesses presented in the previous section suggests the value of combining discard-based search and indexed search. The resulting hybrid is a new usage model for discard-based search, and is the key innovation described in this document. Although this usage model was first identified in the context of the OpenDiamond platform, it is relevant to any software or hardware tool that can support discard-based search.

The hybrid approach has two phases. In the first phase, a human expert conducts discard-based searches for queries that characterize a set of index terms. The search may be conducted on the entire corpus of data, or on a representative sample of it. As typical with discard-based searches, the process may involve iterative refinement, where each iteration involves modifications to searchlet code or its parameters by the expert. The process converges for a specific index term when the expert is satisfied with the balance of false positives and false negatives produced by the searchlet. This can be repeated for multiple index terms, yielding a separate searchlet for each term.

The second phase of the hybrid approach involves use of the searchlets from the first phase in "batch" mode, that is, without user interaction. Each searchlet produced by an expert or group of experts can be used as preprocessing code for indexed search. Although this was not the original purpose of searchlets, it is a new use for them that is entirely compatible their original design. Since searchlets are designed for execution close to storage rather than close to the user, they are already structured to function effectively without user interaction. A simple implementation of the second phase is to just re-run the original discard-based search application on new data, without a user. Data that is not discarded by a searchlet is labelled with the index term corresponding to that searchlet. Since there is no user to further prune this data, the incidence of false positives is likely to be higher than with an interactive search based on the same searchlet.

This usage model is a generalization of a system design that was developed for applying Diamond to the Stentor image database at UPMC. This design context is described in Section 5. Some optimizations and refinements of the basic hybrid search approach are then described in Section 6.

5 Case Study: the Stentor System at UPMC

Stentor is a image processing and communication system that was originally developed at UPMC, and is now a commercial product of Philips. This system uses a coding scheme and transmission protocol called Dynamic Transfer Syntax (DTS) to enable high-resolution images to be efficiently viewed over networks of relatively low bandwidth [4]. DTS preserves image fidelity since it avoids lossy data compression.

A substantial fraction of medical images generated in UPMC hospitals is stored on Stentor servers. The ability to search these images with Diamond would be a valuable capability. Unfortunately, the I/O and CPU demands of searchlet execution would place substantial load on mission-critical Stentor servers. This is true even if the searchlets were executed on Diamond servers that are close to but distinct from Stentor servers; in that case, the CPU demand may be acceptable but there would still be substantial I/O and network load from searchlet execution. What is desired is a solution that has the flexibility and cognitive power of discard-based searches, without its runtime computational overhead. A sketch of our solution follows.

A Diamond system, separate from the Stentor system, receives copies of images generated from patient care at UPMC. These images are also stored on Stentor servers, without any changes to the current workflow. Searchlets are developed on the Diamond system by UPMC's medical experts. On a periodic basis (once a day, for example), these searchlets are run on recently-collected images to generate index terms. Primary keys (that is, unique image identifiers) connect these index terms to image content on Stentor servers.

The index terms and primary keys are stored in a separate relational database server that is independent of both Stentor servers and Diamond servers. This index server may also store additional annotations received over time from medical experts. These annotations may contradict or support the classifications provided by searchlets. It is a policy decision whether to obtain them lazily (that is, sporadically by experts when they feel motivated) or eagerly (as part of the workflow surrounding generation of new images). Thus Diamond searchlets can be viewed as providing the initial annotation for images, with later human input providing additional annotation. The identity of the entity providing an annotation (specific Diamond searchlet or specific human expert) can be part of the database schema. This ensures that the full provenance of classification can be captured and accurately preserved.

6 Refinements and Optimizations

In this section we describe some extensions to the hybrid search approach described in Section 4. Some of these extensions make it more efficient, while others enrich its expressiveness.

6.1 Supporting Diverse Opinions

Even for the same index term, it is possible for two different experts to produce slightly different searchlets. This reflects the ambiguity inherent in the most challenging classification tasks faced by an expert, and is reflected in common discourse by the phrase "expert opinion." Qualifying an index term with an individual or collective identity is a simple way of representing this ambiguity: for example, "melanoma-by-Doctor-X" and "melanoma-by-Doctor-Y" or "incipient-glaucoma-Mayo-Clinic" and "incipient-glaucoma-Harvard". The corresponding searchlets can be viewed as executable forms of expert opinion.

6.2 Exploiting Searchlet Structure

In the OpenDiamond platform, a typical searchlet is composed of stages called *filters,* each of which can independently discard objects. The interconnection and ordering of filters that make up a searchlet are specified in a *searchlet configuration file.* Each storage node iterates through data objects and presents them to individual filters for evaluation. Only objects that pass through all filters that are encountered in a searchlet are forwarded to the front-end.

It is common for many searchlets to share the same underlying set of filters, sometimes even with the same parameter values. In those cases, considerable computational effort may be saved in second phase of the hybrid search approach by factoring out filter executions that are common. For example, if a face detection filter is part of many searchlets, it needs only to be executed once for that group of searchlets rather than once per searchlet. To exploit this optimization, a global ordering for filter execution can be derived from a collection of searchlet configuration files at the start of the second phase of hybrid search.

6.3 Exporting Cached Filter Results

The OpenDiamond platform maintains persistent (i.e., on-disk) caches for the results of filter executions. If the same filter, with the same parameters, is to be executed on a previously-examined data object, the filter re-execution can be avoided by simply re-using the cached result. This situation occurs frequently in discard-based search, as a user refines searchlets over many iterations of a search. In the hybrid search approach, this situation could occur in the second phase if a group of data objects that were examined long ago are re-processed in the light of new knowledge (as discussed in Section 3). The new searchlets that are constructed to reflect the new knowledge may still embody many of the old filters with the same parameter values. Exploiting cached filter results may significantly improve the efficiency of the second phase.

The ability to export OpenDiamond cache state into an external data representation format (such as the Microsoft Excel `csv` format, or tabular format for input to a SQL database) may amplify the value of caching filter results. A cryptographic hash of a filter and specific parameter values can be viewed as an index term. Exporting cache state from the OpenDiamond platform can be viewed as a way of obtaining low-level index terms for a collection of data objects. These low-level index terms can then be further processed and merged with external information to yield index terms for higher-level concepts. A searchlet configuration file implicitly defines this processing, but exporting cache state enables broader forms of processing to be used in the second phase of hybrid search.

6.4 Tracking Updates to the Data Repository

An important benefit of discard-based search is that it is well-suited to dynamic collections since no re-indexing is required when data is added to (or removed from) the repository. A naive implementation of the proposed hybrid approach would not offer these benefits since the index terms may not reflect the current state of the data repository.

Fortunately, since the majority of repository updates in real-world applications involve the addition of new data, one can maintain a current index simply by running the set of searchlets from the first phase on the new data as it is added to the repository. Alternately, one could employ the mechanisms of the OpenDiamond platform's caching model to efficiencly track the filters that have been executed for each object in the repository, and update the index using speculative filter execution at periodic intervals.

7 Conclusion

We have described a new usage model for discard-based search that preserves many of its strengths while alleviating some of its weaknesses. The proposed hybrid search approach involves two phases. In the first phase, human experts use discard-based search to create searchlets that reflect their classification expertise. In the second phase, these searchlets are used to preprocess complex data for indexing. This hybrid approach preserves the speed and server security characteristic of indexed search, while offering flexibility in the creation of searchlets and in tuning their precision-recall characteristics. Most importantly, it offers ample opportunity for human expertise and new knowledge to be efficiently incorporated into the process of indexing complex data.

Acknowledgements

OpenDiamond is a trademark of Carnegie Mellon University. All unidentified trademarks mentioned in this paper are properties of their respective owners.

References

[1] BERCHTOLD, S., BOEHM, C., KEIM, D., KRIEGEL, H. A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space. In *Proceedings of the Symposium on Principles of Database Systems* (Tucson, AZ, May 1997).

[2] DUDA, R., HART, P., STORK, D. *Pattern Classification*. Wiley, 2001.

[3] FLICKNER M, SAWHNEY H, NIBLACK W, ASHLEY J, HUANG Q, DOM B, GORKANI M, HAFNER J, LEE D, PETKOVIC D, STEELE D, YANKER P. Query by Image and Video Content: The QBIC System. *IEEE Computer 28*, 9 (September 1995).

[4] HUFFMAN, J., CHANG, P., BETANCOURT, C., AND LIU, J. C. iSyntax: A Flexible Representation for Image Data. Tech. Rep. MKT0.0-WP-02 Rev 9/15/05, Phillips Medical Systems, 2005.

[5] HUSTON, L., SUKTHANKAR, R., WICKREMESINGHE, R., SATYANARAYANAN, M., GANGER, G.R., RIEDEL, E., AILAMAKI, A. Diamond: A Storage Architecture for Early Discard in Interactive Search. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies* (San Francisco, CA, April 2004).

[6] MINKA, T., PICARD, R. Interactive Learning Using a Society of Models. *Pattern Recognition 30* (1997).

[7] NECULA, G. C., AND LEE, P. Safe Kernel Extensions Without Run-Time Checking. In *Proceedings of the 2nd Symposium on Operating Systems Design and Implementation* (Seattle, WA, October 1996).

[8] VON AHN, L., AND DABBISH, L. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (April 2004).

[9] WAHBE, R., LUCCO, S., ANDERSON, T. E., AND GRAHAM, S. L. Efficient Software-based Fault Isolation. In *Proceedings of the 14th ACM Symposium on Operating Systems Principles* (Asheville, NC, December 1993).

[10] WALLACH, D. S., BALFANZ, D., DEAN, D., AND FELTEN, E. W. Extensible Security Architectures for Java. In *Proceedings of the 16th ACM Symposium on Operating Systems and Principles* (Saint-Malo, France, October 1997).

[11] YAO, A., YAO, F. A General Approach to D-Dimensional Geometric Queries. In *Proceedings of the Annual ACM Symposium on Theory of Computing* (May 1985).