Evaluating Answer Quality/Efficiency Tradeoffs

Ugur Cetintemel
University of Maryland
ugur@cs.umd.edu

Björn T. Jónsson
University of Maryland
bthj@cs.umd.edu

Michael J. Franklin
University of Maryland
franklin@cs.umd.edu

C. Lee Giles
NEC Research Ins. & UMIACS
giles@research.nj.nec.com

Divesh Srivastava
AT&T Labs–Research
divesh@research.att.com

Abstract

For many emerging applications and environments, information systems designers and implementers must consider the tradeoffs between efficiency and the quality of query answers. This flexibility, while allowing numerous opportunities for optimization, complicates the development of various system components and their performance evaluation. In this short paper, we first outline the issues that arise in evaluating answer quality/efficiency tradeoffs. We then describe how we address these issues in two ongoing projects that adopt notions of answer quality from the field of Information Retrieval.

1 Introduction

Traditionally, database systems have been designed based on the idea that for a given query on a given database, there is a single, correct answer that must be returned. It is clear, however, that for many emerging applications and environments it is neither practical nor desirable to enforce such a stringent requirement. In particular, when scaling query processing to wide-area environments such as the World Wide Web, variable factors such as heterogeneity, availability, usage cost, query imprecision, and data quality impose new constraints on database systems. In such environments, users are not likely to get the perfect answers to their queries within a reasonable time, if ever. For these reasons, many recent database-related projects have been exploring techniques for providing approximate or incomplete answers, e.g., [HHW97, FJS97, ABF$^+$97, BDF$^+$97, GM98].

Performance studies and benchmarks for database systems have traditionally focused on throughput, response time, or other metrics related to the efficiency of the query execution. When the guarantee of a single, correct answer is removed, however, these metrics are clearly insufficient — the quality of answers must also be taken into consideration. From a systems implementation and performance perspective, allowing a flexible approach to answer quality raises a number of interesting opportunities and challenges. Opportunities arise from the ability to *trade off the quality of answers vs. system performance.* For example, it becomes possible to develop query optimization algorithms that change the results returned by a query — something that is clearly not allowed in traditional database query optimization. The additional degree of freedom that is introduced by a flexible approach to answer quality complicates the development of various system components and their evaluation.

Performing solid systems work in a world where users no longer expect a single correct answer, however, requires some basic issues to be addressed. For example: How do we quantify the quality of answers? How do we choose reasonable solutions from the vast array of possible approaches? How do we evaluate the effectiveness of our proposed methods? This short paper examines some possible answers to these questions by discussing two ongoing projects in which we have

Proceedings of the 5th KRDB Workshop
Seattle, WA, 31-May-1998

(A. Borgida, V. Chaudhri, M. Staudt, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-10/

developed systems that allow some flexibility in the quality of the answers they provide. In both of these projects we have adopted ideas that were originally developed in the Information Retrieval (IR) community. IR systems typically deal with natural language and unstructured data, both of which add a degree of ambiguity into the data retrieval process. IR systems are developed with such ambiguity as a fundamental assumption. Before describing this work, we first present a quick overview of some basic IR techniques for coping with imprecision in data and queries.

2 IR Approaches to Answer Quality

2.1 Ranked Retrieval vs. Exact Match

Early IR systems were based on a boolean model of querying, in which user needs were specified as boolean combinations of search terms. Likewise, existing database systems provide similar exact match query semantics (e.g., SQL). Current opinion in the IR community is that formulating effective boolean retrieval queries requires significant expertise [Tur94]. As in the IR case, the limitations of the exact match approach are quickly becoming apparent in emerging data retrieval applications and environments. In such environments queries are posed by hordes of novice users, who may have only a vague notion of what they are looking for. Such users are likely to run into an inherent problem with exact match queries, namely that effective query formulation requires some knowledge of the contents of the database. Without such knowledge, users are likely to ask queries that return no answer or return an excessively large answer.

An answer to these problems in the IR context is *ranked retrieval*, where documents are scored based on relevance to the query and the most relevant documents are returned in order of decreasing score. This model intuitively assists the user in two ways: 1) The best matches are seen first, and so the user can quickly find interesting documents, and 2) If there are no good matches, the returned documents may assist the user in figuring out what query to pose next. Extensive studies have shown that the ranked retrieval model gives better results than the boolean model [Tur94].

We believe that ranked retrieval is also a better query model for many emerging database applications, for exactly the same reasons as in the IR case. There has been some work related to this, for example the query interface of [Bro97], where the system uses summary screens to lead the user through a query refinement dialogue, until the posed queries have an answer set that is easy to display in ranked order. Of course, much work is needed to quantify the tradeoffs between quality of answers and performance of query processing in these two models, but for the rest of this paper we assume the ranked retrieval model.[1]

2.2 Metrics

In IR, two basic metrics are used to capture the satisfaction of users with returned answers. These are *precision* and *recall*. Precision is defined as the fraction of the returned answers that are relevant to the user's query. Recall is defined as the ratio of relevant documents that are returned to the user to the total number of relevant documents. (Note that the traditional database requirement of returning a single, correct answer can be rephrased as: precision = recall = 100%. This assumes that the user correctly and exactly knows what query to submit.)

In the ranked retrieval model, precision and recall can be cleverly combined into a single metric called *non-interpolated average precision* [Har96]. This metric is particularly appropriate when the user requests for the top k relevant documents. In many IR scenarios, users can improve their precision-recall (and, hence, the non-interpolated average precision) by interacting with the system. Two typical types of interaction are *query refinement*, in which users refine their queries by adding or removing terms until they are satisfied with the returned results, and *relevance feedback*, in which users indicate the documents they deem relevant and/or irrelevant, and the system automatically reformulates the original query based on this feedback.

In order to compute these metrics, we need to know if an individual document is relevant to a query, and what all the relevant documents are. In the TREC series [Har96], which serves as the major benchmarking activity for IR systems, relevance of documents to particular queries are determined by a board of human experts. When processing queries in environments such as the World Wide Web, such expert judgements are not available, and one has to rely on feedback from users to determine relevant documents, and (approximations to) the non-interpolated average precision

3 Unsafe Query Optimizations

The concept of *unsafe optimizations*, which has been used extensively in IR, refers to techniques that are used to speed up query evaluation by removing some guarantees about the answers. Due to the inherent fuzziness of IR queries, the query evaluation engine has the added flexibility of choosing *which data to read*, and many unsafe optimizations involve skipping portions of inverted lists that are deemed to have little

[1] Note that all the methods we describe can also be used with exact-match queries. They simply make more sense with ranked retrieval.

impact on the quality of the results. Some such optimizations are quite successful in that they maintain the quality of results while reducing the query response time significantly [TF95].

In [JFS98] we proposed a new class of unsafe optimizations that interact with the buffer manager. We proposed a *buffer-aware* extension to IR query evaluation algorithms, which dynamically changes the query evaluation strategy based on buffer contents. This approach trades off the quality of the results in return for savings in disk reads. To evaluate the effects of our extension we applied it to an efficient IR algorithm, and compared the quality of answers with and without the extension.

We ran query refinement sequences based on benchmark queries against a benchmark document collection. For this setup, there exists a list of relevant documents for each benchmark query. These lists allowed us to calculate a metric called *non-interpolated average precision*, which cleverly combines precision and recall into a single number [Har96]. Using this metric, we were able to gauge the potential degradation of the results in an objective manner. We observed that while quality of answers was virtually unaffected, using the buffer aware extension always improved the disk performance significantly; sometimes by as much as 70% for the whole refinement sequence, and as much as 97% for the last refinement. Details of the results, as well as the proposed buffer-aware extension, can be found in [JFS98].

We expect ranked retrieval for structured databases to be amenable to unsafe optimizations of the kind described above. The new wide-area environment, however, opens up yet another class of unsafe optimizations that apply at the network/server level. These include not querying a certain relevant database, not waiting for data from a loaded server before returning the answer to the user, and accessing a cheaper/faster server with less accurate data. These optimizations could be triggered by network delays or could be based on cost estimates for accessing remote databases.

4 User Profiles for Data Push

Another area in which the quality of returned results can be traded off against efficiency is the management of user profiles used for routing information in publish/subscribe systems. A user profile encodes an indication of the data needs and interests of a user. From the user's viewpoint, a profile provides a means of *passively* retrieving relevant information. A user can submit a profile to a push-based system once, and then continuously receive data that are (supposedly) relevant to him or her in a timely fashion without the need for submitting the same query over and over again. This automatic flow of relevant information helps the user keep pace with the ever-increasing rate of information generation. From the system point of view, profiles fulfill a role similar to that of queries in database or information retrieval systems. In fact, profiles are a form of continuously executing query. In a large publish/subscribe system, the storage and access of user profiles can be resource-intensive. Additionally, given the fact that the user interests are changing over time, the profiles must be updated accordingly to reflect up to date information needs. The need for scalable, efficient profile management was clearly demonstrated by experience with the SIFT system [YGM96].

In our work on user profiles [CFG98], we have developed MM, an incremental algorithm for constructing and maintaining user profiles for filtering text-based data items. MM can be tuned to trade off effectiveness (i.e., accuracy of the filtered data items) and efficiency of profile management. The algorithm receives relevance feedback information from the users about the documents that they have seen (i.e., a binary indication of whether or not the document was considered useful), and uses this information to improve the current profile. One important aspect of MM is that it represents a user profile as multiple keyword vectors whose size and elements change dynamically based on user feedback.

In fact, it is this *multi-modal* representation of profiles which allows MM to trade off effectiveness and efficiency. More specifically, the algorithm can be tuned using a threshold parameter to produce profiles with different sizes. Let us consider the two boundary values of this threshold parameter to illustrate this tradeoff: When the threshold is set to 0, a user profile is represented by a single keyword vector, achieving extremely low overhead for profile management, but seriously limiting the effectiveness of the profile. At the other extreme, if the threshold is set to 1, we achieve an extremely fine granularity user model. The profile size, however, equals the number of relevant documents observed by the user, making it impractical to store and maintain the profile. Therefore, it is more desirable to consider intermediate threshold values which will provide an optimal effectiveness/efficiency tradeoff for a given application.

We evaluated the utility of MM by experimentally investigating its ability to categorize pages from the World Wide Web. More specifically, we used pages from Yahoo! [Yah98]. Yahoo! is a popular web guide that organizes web pages into a topic hierarchy formed by human editors. We used non-interpolated average precision as our primary effectiveness metric and focused on the profile size for quantifying the efficiency of our approach. We demonstrated that, compared to minimum-sized profiles, we can achieve significantly higher precision values with a modest increase in profile size. Additionally, we were able to achieve precision values with small profiles that were comparable to, or in some cases even better than those obtained with maximum-sized profiles. The details of the algorithm, experimental settings, and the results are discussed in [CFG98].

5 Conclusions

In summary, emerging environments and applications require that information systems implementers consider the tradeoffs between efficiency and answer quality. Making intelligent choices for such tradeoffs requires a rethinking of metrics and evaluation procedures for databases and other information systems. This additional flexibility opens up various novel implementation opportunities. We described two ongoing projects that provide examples, based on notions of answer quality from IR, of the kinds of tradeoffs that are possible and the kinds of evaluations that can be done on such approaches.

Acknowledgments

Franklin, Cetintemel and Jónsson's work has been partially supported by the NSF under grants IRI-9501353 and IRI-9409575, by ONR under contract number N66001-97-C8539 under DARPA order number F475, by Rome Labs agreement number F30602-97-2-0241 under DARPA order number F078, by NEC, by Bellcore, and by an IBM Shared University Research Award.

References

[ABF$^+$97] Laurent Amsaleg, Philippe Bonnet, Michael Franklin, Anthony Tomasic, and Tolga Urhan. Improving responsiveness for wide-area data access. *IEEE Data Engineering Bulletin*, 1997.

[BDF$^+$97] Daniel Barbará, William DuMouchel, Christos Faloutsos, Peter J. Haas, Joseph Hellerstein, Yannis E. Ioannidis, H. V. Jagadish, Theodore Johnson, Raymond T. Ng, Viswanath Poosala, Kenneth A. Ross, and Kenneth C. Sevcik. The New Jersey data reduction report. *IEEE Data Engineering Bulletin*, 20(4):3–45, 1997.

[Bro97] Kurt P. Brown. DBGuide industrial presentation. *ACM SIGMOD Conference*, Tucson, AZ, USA, 1997.

[CFG98] Ugur Cetintemel, Michael Franklin, and C. Lee Giles. Constructing user profiles incrementally: A multi-modal approach. Submitted for Publication, 1998.

[FJS97] Christos Faloutsos, H. V. Jagadish, and Nikolaos Sidiropoulos. Recovering information from summary data. In *Proc. of the VLDB Conf.*, Athens, Greece, August 1997.

[GM98] Phillip B. Gibbons and Yossi Matias. New sampling-based summary statistics for improving approximate query answers. In *Proc. ACM SIGMOD Conf.*, Seattle, WA, USA, June 1998.

[Har96] Donna Harman. Overview of the fourth Text REtrieval Conference (TREC-4). In *Proc. of the Fourth Text REtrieval Conference (TREC-4)*, National Institute of Standards and Technology, Gaithersburg, MD, 1996.

[HHW97] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online aggregation. In *Proc. ACM SIGMOD Conf.*, Tucson, AZ, USA, May 1997.

[JFS98] Björn T. Jónsson, Michael Franklin, and Divesh Srivastava. Interaction of query evaluation and buffer management for information retrieval. In *Proc. ACM SIGMOD Conf.*, Seattle, WA, USA, June 1998.

[TF95] Howard Turtle and James Flood. Query evaluation: Strategies and optimization. *Information Processing & Management*, November 1995.

[Tur94] Howard Turtle. Natural language vs. boolean query evaluation: A comparison of retrieval performance. In *Proc. ACM SIGIR Conf.*, Dublin, Ireland, 1994.

[Yah98] Yahoo!. `http://www.yahoo.com/`, 1998.

[YGM96] Tak W. Yan and Hector Garcia-Molina. Efficient dissemination of information on the internet. *IEEE Data Engineering Bulletin*, 19(3), September 1996.