Human Computation and Crowdsourcing: Works in Progress and Demonstration Abstracts
AAAI Technical Report CR-13-01

A Framework for Adaptive Crowd Query Processing

Beth Trushkowsky
AMPLab, UC Berkeley
trush@cs.berkeley.edu

Tim Kraska
Brown University
tim_kraska@brown.edu

Michael J. Franklin
AMPLab, UC Berkeley
franklin@cs.berkeley.edu

Abstract

Search engines can yield poor results for information retrieval tasks when they cannot interpret query predicates. Such predicates are better left for humans to evaluate. We propose an adaptive processing framework for deciding (a) which parts of a query should be processed by machines and (b) the order the crowd should process the remaining parts, optimizing for result quality and processing cost. We describe an algorithm and experimental results for the first framework component.

Introduction

Existing search engines are a good starting place for information retrieval tasks, but can yield poor results when they have difficulty interpreting query predicates. For example, in an image search query for "people against a light background", a search engine may incorrectly filter some images by misapplying the predicate regarding background color, yielding fewer correct results than a search for just "people". Similarly, searching a quotations search engine for "funny quotes about computers" may be challenging if it cannot discern if a quote is humorous. For queries consisting of a conjunction of boolean predicates, some predicates may be well suited for automated processing, while others are better left for *crowd processing* (Parameswaran and others 2013).

Thus given a query, we need to choose how to divide the query processing work between the automated search tool and the crowd to optimize for query quality while keeping crowd costs low. The first challenge is determining the subset of predicates to use as search terms in the automated search., i.e., which predicates we should *"push down"* into the search engine. An additional challenge is choosing in which order the crowd should evaluate the remaining predicates, similar to predicate ordering in databases (Hellerstein and others 1993). Unlike with a search engine, we can process the predicates one after another with the crowd; which order the predicates are evaluated may influence cost and quality. To minimize crowd costs, predicates should be ordered to reduce the number of worker evaluations needed to filter items in the query result.

As we will not know a priori which predicates are best suited for the crowd, nor the predicates' cost or selectivity, we propose an adaptive query processing approach that adjusts its strategy based on observed performance. In this work-in-progress paper, we describe an adaptive processing framework for attacking predicate pushdown and ordering. We then describe an algorithm and experimental results for predicate pushdown as applied to image search.

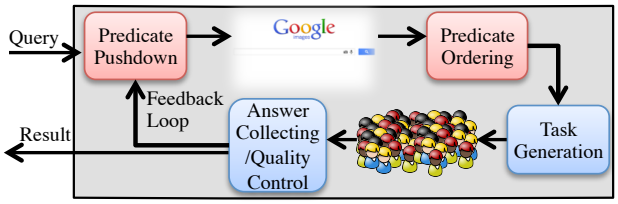

Figure 1: Adaptive query processing framework

Architecture

The adaptive query processing framework (Figure 1) takes a query with n predicates describing the constraints that all results must satisfy. The query is processed with an initial search engine query using a subset of the predicates and then human workers evaluate the remaining predicates.

Predicate pushdown The *predicate pushdown* component decides which subset of the predicates to evaluate using automated search. It determines which predicates yield lower quality results from the search engine. We describe an adaptive algorithm and experimental results for this component applied to the image search problem in the next section.

Predicate ordering The *predicate ordering* component is responsible for deciding in which order the crowd should process the predicates on the items returned from the search engine. Some predicates may cost more than others, where cost may entail worker think time, number of votes to reach consensus, etc., in addition to monetary cost. Choosing an evaluation order that "fails fast" will minimize query processing cost. This component is future work.

Adaptive Predicate Pushdown

For the predicate pushdown component, we propose a greedy algorithm that searches the predicate combination space by executing search engine queries using fewer and fewer predicates until it finds the best combination for the search engine. We first define the notion of paths that the

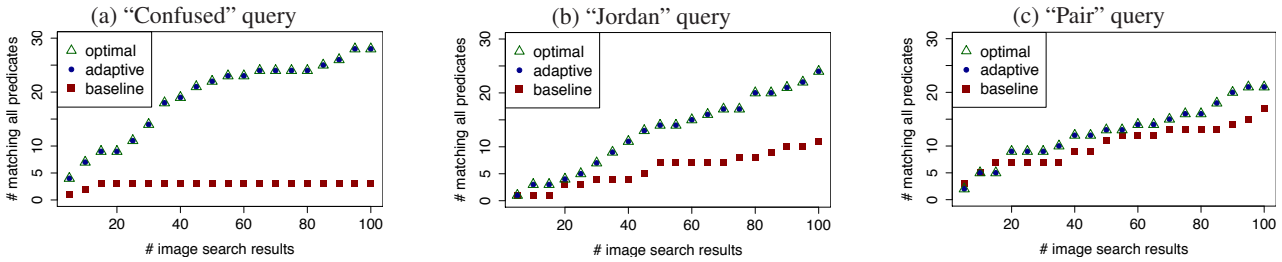

Figure 2: Number of query results matching all predicates after processing the first 100 images from Google search.

algorithm chooses between. A *path* is a particular combination of predicates that are executed by the automated search engine. A path's set of *children* includes the combinations of predicates that result when one predicate is removed.

Algorithm core The algorithm begins exploring the performance of the path with all predicates used in the search engine, as well as that path's children. It proceeds in *stages* as follows. In each stage, the crowd evaluates all predicates for the first batch of m search results from each of the current paths being compared. A performance score, e.g., "relative" recall (the set of total relevant results is the union of correct results from all paths) or precision, is computed for each path, and the winner for the stage is decided via statistical analysis, described below. The winning path and its children paths are explored in the next stage. The algorithm terminates when (a) the stage winner is the same as in the previous stage, or (b) a stage's winning path has no children.

Choosing stage winner To determine the winning path for a stage, we want to show that the path with the current highest score is significantly better than the others. Recall that we compute each path's performance score as they process more and more results. After processing m results, we generate a set of measurements for each path, one for each step in $1, ..., m-1, m$. As the query processing algorithm is typically comparing three or more paths, we employ the ANOVA statistical test to determine if there are significant differences amongst the paths' scores. Once we observe that the paths are significantly different, we must also show that the path with the highest score is significantly better than each of the other paths under consideration. For pairwise comparisons between paths, we perform post-hoc analysis via the TukeyHSD test. If either test fails to show significance, we conclude there is not yet a winning path and more search results ($m+1, m+2...$) need to be evaluated.

Query name	Free-form text	Predicates
Confused	confused person using a computer alone	(1) confused, (2) using a computer, (3) alone
Jordan	Michael Jordan with basketball full body black background	(1) with basketball, (2) full body, (3) black background
Pair	two people using computers white background	(1) two, (2) using computers, (3) white background

Table 1: Image search queries used in experimental results.

Experimental Results: Image Search

Setup We explore the image search problem for three queries, described in Table 1. For each combination of predicates (paths) we use the Google search API[1] and extract the image search results. We use Amazon's Mechanical Turk for crowd processing. Workers indicate "yes"/"no"/"unsure" for each predicate applied to each image; we use majority vote across workers. The algorithms compare paths after $m = 20$ images have been crowd-processed. We use relative recall as a path's performance score, and a p-values of 0.05.

Results We compare the query result performance of the statistical adaptive algorithm to a baseline algorithm that pushes all predicates into the search engine. Figure 2 depicts the number of correct images found by each algorithm as results from the search engine are processed. "Optimal" is the best path, determined by evaluating all combinations.

In all cases, the baseline algorithm's performance shows that pushing all predicates into the search engine can lead to poor result quality. The adaptive algorithm chooses the correct predicate combination to push down for each query, aligning with the optimal algorithm. For some queries, the difference in performance between paths can be quite stark, e.g., for the "Confused" query pushing down the two predicates "confused" and "using computer" yields vastly better results. The adaptive algorithm is also able to capture optimal paths that may seem unintuitive. In the "Jordan" query, pushing down "black background" has the highest recall. We hypothesize this is because images of Michael Jordan with a black background are more likely to depict him in a stylized manner dunking a basketball.

Conclusion

The crowd can help search engines process query predicates that are more amenable to human evaluation. We show that our adaptive framework can get better results than search engines alone by deciding which predicates should be evaluated by the crowd. Future work includes addressing predicate ordering to optimize crowd processing cost.

References

Hellerstein, J. M., et al. 1993. Predicate migration: optimizing queries with expensive predicates. SIGMOD 1993.

Parameswaran, A., et al. 2013. Datasift: An expressive and accurate crowd-powered search toolkit. Technical report, Stanford University.

[1] http://developers.google.com/custom-search