Summary

This paper concentrates on how to utilize human to compare items for executing two basic computational operations - joins and sorts in a declarative database management system. For join operation, it proposed three batching strategies, simple, naive batching, and smart batching, to reduce the number of HITs in crowdsourcing. For sort operation, it presented three approaches: comparison-based, rating-based, and a hybrid of the two. This paper also proposed a number of optimizations, including techniques like batching and replacing pairwise comparisons by numerical ratings, to improve the performance. As shown on several real-world datasets, the system proposed in this paper is able to greatly reduce the total money cost without sacrificing accuracy.

Three (or less) Pros

- This paper considered sorting and joining in real crowdsourcing system, and many practical issues are addressed (rather than a toy).
- This paper proposed optimization techniques, like batching and replacing pairwise comparisons by numerical ratings, to improve the performance.

Three (or less) Cons

- Scalability problem. Even by using the best batching approach proposed in this paper (placing multiple pairs of records in a single HIT but workers are asked to find all matches among all of the records.), with a modest database size of 10,000 records, assuming a reasonable HIT size $k = 20$, it would require $250,000$ HITs ($2,500$ with $0.01$ per HIT).[1]
- This paper failed to address more general questions regarding different scenarios, such as different types of workers and compensation schemes, different algorithms and strategies, in a hard-coded manner.
- This paper considered different aspects of implementation instead of giving rigorous theoretical results.

How to improve the paper further if you were the authors.

In the future, I will conduct experiment in realistic scenario and large scale, trying to optimize the performance. At the meantime, I will deliver rigorous theoretical results on which the system is based.

Reference

(1) Summary
This paper considers finding max in crowdsourcing environments. It provides efficient heuristics and optimization strategies, and evaluate them empirically which can be tuned using parameters like execution time, monetary cost, and quality of the result (selecting appropriate parameters by making tradeoff between these three quantities). The results explored multiple models and led to various aspects of realistic crowd-sourced max algorithms.

(2) Three (or less) Pros
- This paper developed a general framework for reasoning about max algorithms in crowd-sourced settings.
- This paper provided effective and efficient strategies in selecting appropriate max algorithm parameters.
- This paper exploited a number of models for worker error and user cost, as well as various error and cost models.

(3) Three (or less) Cons
- This paper assumed that the probability that each human provides the right answer is available, but in our case this information is not known.
- This paper dedicated to task splitting algorithms for specific operations, which may not be applicable to other types of query.
- This paper assumed the order between two elements is independent of preceding elements, which may not be the case in some scenarios.

(4) How to improve the paper further if you were the authors.
In the future, I will make this work more flexible and robust by considering more practical issues and removing some assumptions. As a result, it will be able to handle different scenarios, such as different types of human workers and compensation schemes, different algorithms and strategies. In addition, designing a crowd-powered optimizer is also a direction.