Human-powered Sorts and Joins

Summary:

Nowadays, there is growing interest in the crowd sourced database, which enables both human and computer to process records. This paper first discusses how Qurk, a system created by author, can be used to reduce the burden on workflow designers. Then, the paper mainly focuses on how to use human to compare items, particular pictures, for sorting or joining. The paper proposed three different UIs (simple, naïve batching and smart batching) and used feature filtering to do the joining part. In the sorting part, the authors used three kinds of method: rating-based method, comparison-based method and the mix of the two. In the experiment parts, the paper evaluated all the above methods from the perspective of cost, latency and accuracy. By using the proper methods above, the authors found that they can greatly reduce the total cost of queries without sacrificing accuracy.

Pros:

The paper introduces several methods that can decrease human-power sorting and joining cost while maintaining result accuracy. Also, the Qurk, which was created by authors, can be used to reduce the burden on workflow designers. Therefore, the ideas from this paper provide meaningful contributions into the real-worlds human-powered database.

Cons:

First of all, the paper has several syntax errors. For examples, at the bottom of page 3, the author writes “This is allows”, which should be written as “This allows”. Also, sometimes the authors miswrote ‘HITs’ as ‘hits’. Therefore, I highly recommend the author to do more proofreading.

Secondly, in the beginning, the author did not point out the structure of the paper. This makes it difficult for readers to follow the author due to the lack of big picture of the paper. In addition, there are several times when the earlier part refers to the later content in the paper. I do not think it’s a wise choice since it might add complexity for understanding the paper, which in turn will make reader lose interests in the paper.

Finally, in 4.2.3, the paper does not validate the use of metrics, a modified version of Fleiss K and T, to evaluate whether the sort task is feasible and how closely RATE responds to COMPARE, respectively. Without these validations, the paper can hardly convince reader of its conclusions based on these two metrics.

Questions:
In section 5, the paper used both the comparison-based method and rating-based method for sorting. But, why didn't the paper discuss the using of mix of the comparison-based and rating-based method for sorting?
Executing SQL over Encrypted Data in the Database-Service-Provider Model

Summary:
The development of Internet technologies enables the emergence of ‘software as service’ model. The model gives customers the chance to use software over Internet. This article introduces the ‘database as service’ model that enables customer to create, store, modify and retrieve data from a remote server via Internet. Privacy issues can occur when using ‘database as service’ model. Particularly, the article focuses on the privacy concern under situation where the service provider is not trustful. To deal with this privacy issue, the authors create a model that can execute SQL queries over encrypted data. In the model, customers store and create the encrypted data on server that can only be decrypted by owners. There are two parts of executing a SQL query: server query and client query. During the server query, the server selects the corresponding rows over the encrypted data. Then, the result of the server query is sent to the client, which is followed by the client query. After decrypting the result and executing the query, the client gets the correct result of the original query. After explaining the model, the authors do several experiments to validate that the model can minimize the computation at the client sites.

Pros:
Firstly, this article creates a split-query strategy to deal with the privacy concern that occurs in the Data-Base-Service model. This is a good strategy in that it takes both the privacy and query execution time into consideration. While the strategy is secure enough to protect data privacy, it can also reduce the time consumption in the client sites.

Secondly, the article does a good job in explaining its proposed service-provider architecture. It tells readers how the data is encrypted and decrypted, how to translate original query conditions to encrypted data query conditions, how to implement relational operators over encrypted data, and how to split the query into client query and server query. Those details make it easy for readers to understand the whole process of this architecture.

Cons:
First of all, neither of the figures in Section 6 has the metric for query execution time. Without such metric, the article can hardly convince us of all the results it proposes based on these figures.

Secondly, in experiment 1, the article studies how the execution time consumption is affected by different numbers of buckets. However, the author only compared the result of 2 buckets and one of 8 buckets. The comparison between only two conditions is not sufficiently enough to validate the proposed conclusion in the article. Similarly, the author cannot conclude that their architecture “does not introduce overhead” only based on the results of two conditions.
In addition, although authors give the numbers of buckets in each condition, they do not specify how they determine the numbers of buckets. Since each table has several columns and each column can have different number of buckets. The article does not points which column determines the numbers of buckets in the experiments.

Questions:
In section 6, the article mentions to use “equ-width and equ-depth histograms to partition the data for two different classes queries”. So, for which class each of the partition algorithms is used?