Evaluation and Refinement of Clustered Search Results with the Crowd

AMY X. ZHANG, Massachusetts Institute of Technology, USA
JILIN CHEN, WEI CHAI, JINJUN XU, LICHAN HONG, and ED CHI, Google, USA

When searching on the web or in an app, results are often returned as lists of hundreds to thousands of items, making it difficult for users to understand or navigate the space of results. Research has demonstrated that using clustering to partition search results into coherent, topical clusters can aid in both exploration and discovery. Yet clusters generated by an algorithm for this purpose are often of poor quality and do not satisfy users. To achieve acceptable clustered search results, experts must manually evaluate and refine the clustered results for each search query, a process that does not scale to large numbers of search queries. In this article, we investigate using crowd-based human evaluation to inspect, evaluate, and improve clusters to create high-quality clustered search results at scale. We introduce a workflow that begins by using a collection of well-known clustering algorithms to produce a set of clustered search results for a given query. Then, we use crowd workers to holistically assess the quality of each clustered search result to find the best one. Finally, the workflow has the crowd spot and fix problems in the best result to produce a final output. We evaluate this workflow on 120 top search queries from the Google Play Store, some of whom have clustered search results as a result of evaluations and refinements by experts. Our evaluations demonstrate that the workflow is effective at reproducing the evaluation of expert judges and also improves clusters in a way that agrees with experts and crowds alike.

CCS Concepts: • **Information systems → Presentation of retrieval results**; *Evaluation of retrieval results*; *Relevance assessment*; Clustering and classification; • **Human-centered computing → Empirical studies in collaborative and social computing**;

Additional Key Words and Phrases: Clustered search results, clusters, cluster evaluation, crowdsourcing, human computation, search engines, mobile app stores

ACM Reference format:
Amy X. Zhang, Jilin Chen, Wei Chai, Jinjun Xu, Lichan Hong, and Ed Chi. 2018. Evaluation and Refinement of Clustered Search Results with the Crowd. *ACM Trans. Interact. Intell. Syst.* 8, 2, Article 14 (June 2018), 28 pages.
https://doi.org/10.1145/3158226

Authors' addresses: A. X. Zhang, Massachusetts Institute of Technology, Computer Science Artificial Intelligence Lab, 32 Vassar St., Cambridge, MA 02139; email: axz@mit.edu; J. Chen, Google, 1600 Amphitheatre Pkwy., Mountain View, CA 94043; email: jilinc@google.com; W. Chai, Google, 1600 Amphitheatre Pkwy., Mountain View, CA 94043; email: chaiwei@google.com; J. Xu, Google, 1600 Amphitheatre Pkwy., Mountain View, CA, 94043; email: jjxu@google.com; L. Hong, Google, 1600 Amphitheatre Pkwy., Mountain View, CA 94043; email: lichan@google.com; E. Chi, Google, 1600 Amphitheatre Pkwy., Mountain View, CA 94043; email: edchi@google.com.

ACM 2160-6455/2018/06-ART14 $15.00
https://doi.org/10.1145/3158226

1 INTRODUCTION

When exploring a space using search, learning about the structure of the space helps with orienting and discovery. For instance, when searching for cameras to buy, it may be useful to discover that there are mirrorless cameras, in addition to digital SLR cameras. Research has shown that *clustering* search results into topical categories, such as shown in Figure 1, can provide structure for users and also aid in exploration [8, 23], helping users discover new items and new categories of items. Given the number and diversity of potential results, clustering also provides visual structure to differentiate between results. This makes it easier for users to home in on certain categories they are interested in and ignore others, for instance, focusing on potstickers instead of pierogis in a search for dumpling recipes.

However, it is difficult to generate high quality clusters for the wide variety of search queries that may appear in search applications. The relationships of results to each other and to higher level categories are often highly subjective and dependent on the query. While algorithms have been developed that can generate search result clusters [16, 24, 27], these automatic approaches often result in groupings that users do not find coherent or distinct [5]. For instance, a clustered search result for "wallpaper" on an app store might have a "forest wallpaper" cluster and a separate "waterfall wallpaper" cluster, as determined by a clustering algorithm. However, there may be apps that overlap between these two categories appearing in both clusters, making them seem indistinct from each other. Since both are related to nature, it might make more sense if the two clusters were merged together under a "nature" category. Resolving these kinds of issues is often a manual and time-consuming process involving experts tuning algorithms or tweaking the final clustered results. Therefore, while clustered search results have appeared in many research prototypes, few currently exist in large-scale production systems.

How then can we build a system that scales to hundreds or thousands of search queries and also has appropriate quality controls in place? One way to be able to scale cluster evaluation and refinement would be to use paid crowd workers instead of experts. Prior research in crowdsourcing has involved humans in the cluster *generation* process by using crowds to suggest clusters and add items to clusters to create a final clustering [1, 9]. However, this is a labor-intensive process that does not scale when there may be thousands of items per search result and thousands of potential searches. Instead we consider using clustering algorithms, which are fast and free, to quickly produce fully formed but imperfect clusters at scale. Then we can involve the crowd in the *evaluation* and *refinement* stages by having them assess each clustered search result as a whole and make small fixes as necessary. This allocation of resources would use the strengths and also compensate for the weaknesses of both algorithms and crowds.

In this article, we describe Refinery, a hybrid machine and crowd workflow to generate high quality clustered search results at scale. Refinery begins by using a suite of algorithms to produce initial clusterings and then uses the crowd to evaluate the output of the algorithms, refine the organization of the final clusters, and provide finishing cosmetic fixes. To have production-level quality control today, multiple experts would need to take hours to inspect and refine clusters for even a small number of highly-trafficked search queries. Taking inspiration from an internal expert workflow used for clustering search results on the Google Play Store that we describe in detail in this article, Refinery performs the same function faster and cheaper, and with the ability to parallelize, and thus can scale to thousands of search queries.

From our evaluation with 120 highly trafficked queries from the Google Play Store, we find that a crowd-powered workflow produces results that approaches and is occasionally better than results currently shown on the app store that were refined over many hours by experts. We also evaluate each step of the workflow and find that, overall, the crowd and experts are in agreement

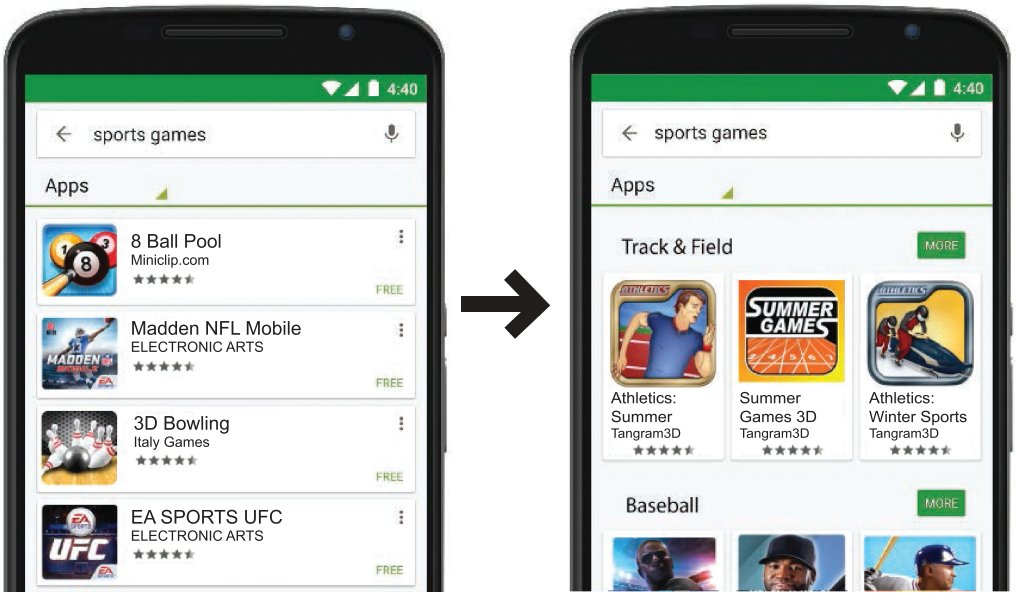

Fig. 1. On the left is a search result in the form of an ordered list while, on the right, the search result is clustered into topical categories.

about the refinements. We conclude by considering how Refinery generalizes to different search applications, as well as how to best combine the efforts of crowds, experts, and machine learning algorithms so that their respective strengths are maximized.

This article demonstrates how cluster evaluation and refinement can be done with the crowd without sacrificing quality or incurring excessive cost and time. Using the Refinery workflow, production search engines powering anything from shopping sites to media sites and beyond can more readily deploy quality clustered search results to benefit exploration and discovery.

2 RELATED WORK

Our system involves a hybrid machine and crowd approach, and thus we draw upon prior work in both automatic and crowdsourced approaches to cluster generation and evaluation.

2.1 Clustered Search Results

Cluster analysis is a well-known area of research, comprising many algorithms and techniques. Early research in clustering search results focused on developing algorithms for clustering text documents on the fly based on text features from a bag-of-words model [12]. Later research expanded this work to items such as web pages [24, 33] or images [4]. Research has demonstrated that a grouped representation of search results as opposed to a list is often more appealing [8] and provides for a more interactive and exploratory experience [18, 23]. However, clustered search results have not yet made it into many large-scale production search engines. This may be because the raw output of search result clustering algorithms is often of low quality, creating clusters that are incoherent with many overlapping clusters. Thus, existing search engines utilizing clustered search results such as Carrot2 [29] or Yippy[1] have yet to become mainstream. Even sites like

[1]http://yippy.com.

Netflix and Amazon that have a detailed tagging system or taxonomy to characterize items and display grouped categories on their home page still only provide dynamic search results in an ordered list.

2.2 Automatic Cluster Evaluation

There are many different clustering algorithms available, with each emphasizing slightly different criteria, and also many parameters to tune within each algorithm [15]. Unlike classification problems, the set of clusters is not known *a priori*, so what makes a good clustering can be difficult to define. Without a gold standard produced by humans, cluster evaluation must be done internally, by examining the data that was used to perform the clustering.

Most widely used internal evaluation metrics, such as the Silhouette coefficient [26] or the Davies-Bouldin index [13], attempt to score clusterings based on two main characteristics: coherence, that is, intra-cluster similarity and distinctness, that is, low inter-cluster similarity. However, these characteristics do not encapsulate all that could be taken into account when examining a cluster, and thus the metrics do not necessarily result in the best clusterings [22]. Internal evaluation metrics also are biased toward algorithms that use the same metrics to perform clustering.

2.3 Crowdsourcing Cluster Evaluation

Due to issues with internal evaluation metrics, many researchers prefer to evaluate clusters using external metrics, such as conducting user studies or interviews in a qualitative way [11]. Others choose to compare their results against a gold standard created by experts [22]. However, both of these approaches are time-consuming and expensive. Also, they do not scale when there are many sets of clusterings to evaluate, such as for a large number of search results. It would not be feasible to construct clusterings by experts for every major search result. It is also not enough to evaluate an algorithm based only on a subset of queries and expand to the rest because a clustering algorithm may perform well for the results of one search query and do a poor job for another. To ensure high quality clusters across many different search queries, a comprehensive evaluation that can scale to large numbers of queries is necessary.

One way to scale evaluation is to replace experts with the crowd. Some researchers have tried to conduct user study evaluations of clustered search results using paid crowdworkers [27]. Researchers evaluated different clustered search results using the crowd by rating whether labels of pairs of clusters within a result were different, which aimed to test for distinctness, and rating whether the top cluster labels within a search result matched the intent of the query, which aimed to test for coverage.

However, little research exists that has rigorously examined particular approaches to gathering crowd cluster evaluations and compared them to the evaluations of experts. Nor have researchers explored design considerations to empirically test how to best gather cluster evaluations from the crowd. However, there have been promising studies in other domains that have shown that crowd annotators can perform just as accurately as expert annotators, even in comparatively specialized areas such as medicine and linguistics [34]. Researchers have also shown how disagreements between crowdworkers may not necessarily be the result of poor quality but can actually signal ambiguity in text [14].

2.4 Crowdsourcing Cluster Generation

Given that creating clusters can be a subjective problem, some recent research has studied generating clusters using crowdsourcing. This would be faster and more scalable than having experts manually curate clusters but is still labor-intensive. Some approaches break the clustering task down into small tasks such as assigning items to clusters, determining item-to-item similarity, and

coming up with categories for items [1, 3, 9]. Such approaches struggle with the constraint that each worker may only see a small and unrepresentative portion of the data and thus workarounds must be adapted to ensure a coherent and seamless final result [31]. Previous approaches circumvent this by showing users more items for context [1] or allowing the workers to sample the data for more items [6]. In contrast, our approach provides workers with a global context to work with throughout the process. This leads to some constraints on the number of clusters and items per cluster we can reasonably show to a worker. However, this tradeoff may be appropriate for clustered search results, where adding more clusters on a page has diminishing returns and only a few items can be shown per cluster.

Additionally, cluster generation with the crowd becomes infeasible when generating clustered search results due to issues of scalability. When there are potentially thousands of items to cluster for a single search query and thousands of search queries to satisfy a proportion of search traffic, crowdsourced cluster generation quickly becomes prohibitively costly and time-consuming. Some approaches combine machine-learning techniques with crowdsourcing, such as using crowd-generated inputs to feed into a clustering algorithm [17, 30]. While hybrid approaches may scale better than fully human ones, requiring many item-to-item comparison tasks per clustering still does not scale well when there may be thousands of clusterings.

Our approach focuses on directing all the efforts of the crowd in the *evaluation and refinement* phase, where crowdworkers can inspect and improve the output of algorithms as a whole. We do this to eliminate any tasks at the item level, which further improves scaling. Additionally, this frees up our system to allow any manner of cluster generation. This is unlike systems that embed constraints, such as that each item can only belong to one cluster [6] or a specific algorithm must be used [9]. Indeed, the workflow we present allows one to attempt a number of algorithms and then picks the best clustering.

To construct Refinery, we consult research that examines crowdsourcing workflows for complex generative tasks. In particular, concepts such as decomposition of complex tasks into smaller ones [2], iterative improvement [21], and combining crowd results with heuristics [9] are helpful for considering how to design the Refinery workflow.

2.5 Crowdsourcing Cluster Refinement

Like our article, other research has explored using people to clean up algorithm outputs and has also developed tools for cluster or topic model refinement. These interfaces support common refinement operations such as merging clusters, splitting clusters, removing items from clusters, and adding items to a cluster [10, 19]. Some work explores giving workers a global context by allowing them to sample and query the entire dataset as opposed to working with a fixed set of items [6]. Other tools have focused on experts, including interactive interfaces to allow algorithm experts to fix problems within clustered search results and refine the features of the clustering algorithm [7].

Researchers have also examined how refinement systems could be better designed to support non-expert refinement of clusters. Some work studies the operations non-experts wish to make to fix clusters, finding that desired refinement operations differ from the low-level operations often supported in human-in-the-loop systems [20]. Researchers point to the need to limit complexity by reducing items, directing users to a subset of possible tasks, and prioritizing simple refinements. Other work has explored visualizations to support labeling tasks, finding a tradeoff between simple presentations that are quicker to comprehend versus complex visualizations that illuminate more nuanced relations [28]. This prior work helps to inform the high-level refinements that we enable as well as the tradeoffs in design we make to keep tasks simple while still providing a global context.

3 PROBLEM BACKGROUND

There are many benefits to clustering search results, including the ability to explore the space of search results to discover new categories of items, drill down to particular categories, and ignore categories that are not interesting. Currently, searches made to the Google Play Store return apps shown in a list, as seen on the left side in Figure 1. This is also a common search result interface for many store, catalogue, and document search engines found on the web. An ordered list is a reasonable interface for navigational queries such as "pokemon go" that are looking for a particular item. However, for certain types of queries, a clustered search result may be more desirable. For instance, a query that is more broad or categorical, such as "sports games" could benefit from a clustered interface that breaks down apps from different sports.

However, determining what is a good clustered search result can be hard to quantify and usually requires a holistic evaluation by experts. We begin by describing in detail the criteria that experts use to evaluate clustered search results as well as the actions they perform on a raw clustering algorithm output to turn it into a production-ready search result. The information about experts' workflow is gathered from an internal expert workflow used to evaluate and refine clusters for the Google Play Store, through discussions with experts and review of their internal documentation. The experts in question are software engineers, some of whom are in charge of devising and tweaking the features of algorithms, as well as some of whom are in charge of the product and its design and development. The experts are generally familiar with the algorithms underlying the clusters and how internal evaluation metrics are calculated. They are also knowledgeable about the app space, including popular apps, major categories of apps, and the general distribution of topic keywords, as well as search queries, including searching behavior and common search terms in the product. The authors of this article make up a subset of the experts involved in this process.

The expert workflow has been responsible for the successful release of clustered search results for under 100 hand-picked high volume search queries to the app store. Aspects of this workflow inform the process and design of Refinery, our crowd-powered workflow that seeks to replace the work of experts.

3.1 Cluster Evaluation

Given a clustered search result for a query, as generated by a clustering algorithm, a group of engineers responsible for the algorithm inspect the result manually to evaluate it. The raw result of the clustering algorithm on a search result is a set of clusters, each with one or a few main topics and a set of apps that align with those topics. There may be several clustering algorithms or a single algorithm with different parameters attempted that each return slightly different search results. After manual inspection and experimentation with the algorithms, the engineers settle on a single clustered search result that has the best quality.

There are many characteristics that experts consider when determining the quality of a clustered search result or deciding that one set of clusters is better than another set of clusters. These characteristics are documented in an internal shared rubric collaboratively constructed by the experts and consulted whenever evaluating a clustered search result. We describe the characteristics in detail as we use some of the same ones later when performing evaluations of the Refinery workflow. Many of them are general qualities of good clusters and echo characteristics that automatic metrics attempt to capture, while some are specific to the application of clustered search results.

—Coherence: Items and main topics within the same cluster should be similar to each other.
—Distinctness: Items and main topics from different clusters should be as distinct as possible.

—Coverage: An obvious or important category given the query should not be missing. Also, most of the items in the search result should fit into one of the clusters. There could be a "more" cluster that has unclustered items, but it should not be too large.
—Relevance: Cluster topics and items should be related to the query. The cluster topics should not be a synonym or super concept of the query. Items should be related to cluster topics.
—Balance: Different cluster should be balanced in size. If only a few clusters contain most items, then clicking on a cluster would return users to a long list of items.

3.2 Cluster Refinement

Once the experts come to an agreement around a single clustered search result, this result goes through a refinement process where they fix any remaining problems with the clusters. When it comes to refinement, there are several ways in which the experts may alter the clusters.

—Merge two clusters together if they are too related given the query (a "piano" cluster topic and "keyboard" cluster topic in a query for "instruments") or if there is a lot of overlap in the items or topics so that two clusters are not distinct.
—Delete a cluster if it is of low quality (no clear relation between the main topics or items) or too granular given the query.
—Delete a topic from a cluster if it does not go with the other main topics of the cluster.

These changes are manual edits to the final output of a clustering algorithm and are not signals toward re-running of the algorithm. We discuss a more human-in-the-loop approach to using edits toward improving algorithms in future work.

3.3 Titling of Clusters

Finally, experts must go through and compose a name for each cluster at the conclusion of the refinement process. Experts often use the top topics for the cluster as the title, sometimes grouping two topics with an "&", or using the topics as inspiration for the title. Following is the expert criteria for titles.

—Relevance: The title should be relevant to all items in the cluster, not just a subset, and not be overly broad given the query or a synonym of the query.
—Coherence: If the title is of form "X & Y," the two topics should be related and not contradict each other.
—Distinctness: If the title is of form "X & Y," the two topics should not be too similar or redundant.

This process can be time-consuming, taking an hour of an expert's time to fine-tune each query. There is also exchanging of different evaluation and refinement proposals among experts due to disagreements. The existence of disagreements between experts demonstrates the subjectivity that comes with clustering items into topics. While this workflow has allowed for the creation of high quality clustered search results for tens of queries in a reasonable amount of time, it would not suffice when expanding to thousands of queries.

4 THE REFINERY WORKFLOW

Our design of the Refinery workflow takes inspiration from and is designed to replace the aforementioned existing expert workflow. We also use the existing expert workflow as a comparison when conducting evaluations of our process. We demonstrate the Refinery workflow using the case of an app store search engine, though the designs and overall workflow could be used in a variety of applications, which we discuss further.

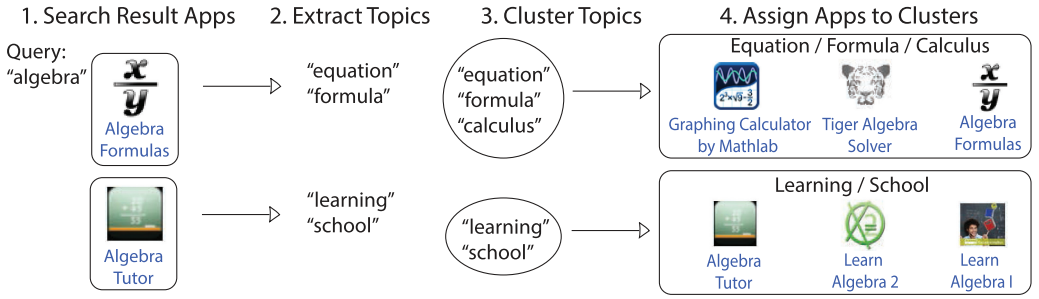

Fig. 2. An overview of the machine portion of the workflow, which focuses on the generation of clusters. First, topics are extracted from apps. Then, topics are clustered using a suite of clustering algorithms. Finally, apps are assigned to clusters in a greedy fashion.

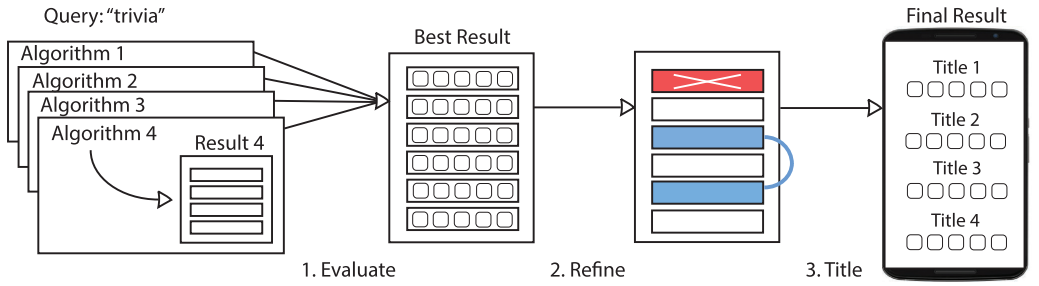

Fig. 3. An overview of the crowd portion of the Refinery workflow, which is focused on evaluation and refinement of clusters. After running a suite of different clustering algorithms, the Evaluate step picks the best-clustered search result from several different algorithm outputs. The Refine step finds clusters to merge or delete. Finally, the Title step adds titles to each cluster to create the final result.

Refinery is a hybrid machine and crowdsourcing workflow that replaces expert evaluation and refinement with the crowd. Using the crowd to generate clustered search results would result in tens to hundreds of tasks to place items into clusters and additional tasks to come up with cluster topics and to give clusters names. That is not to mention the significant overhead for duplication, verification, and resolving of conflicts, of which there are likely to be many. Instead of using the crowd to generate clusters from scratch, as prior research has done, we use well-known clustering algorithms as a free first step that quickly gets us to a clustered search result, and then use the input of the crowd to judge and clean up the result.

We first describe the cluster generation process handled by machine learning algorithms, shown in Figure 2, before describing in detail the three stages of our crowd workflow, as shown in Figure 3, and informed by the production workflow used by experts on the Google Play Store. Following this section, we conduct an evaluation of each step in the crowd portion of the workflow and also compare our final results with clustered search results that went through the entire expert process.

4.1 Cluster Generation

To begin with, each app in the store is assigned several topics, extracted from the information provided by the app developer, such as title, description, and reviews. The topics are extracted using an algorithm for entity recognition. The topics are then mapped to entities in an internal knowledge graph, similar in structure to public ones such as DBpedia or Freebase. For instance, a first-person-shooter app might have topics such as "action game," "battle," and "sniper." Then

clustering is performed over the topics present in a search result. For the purposes of clustering search results, a suite of clustering algorithms were developed by engineers that are each variations on well-known clustering algorithms. We describe these algorithms briefly as they are not the focus of this article.

Each algorithm takes as input the topic keywords associated with the apps that get returned for a query and then clusters the topic keywords into topical groups. The clustering is performed on topic keywords and not the apps themselves as apps update their descriptions and new apps enter the app store many times daily. Clustering by app would quickly lead to stale clusters, while clustering by topic keywords allows broad topical clusters to be maintained while individual apps within the clusters can update seamlessly. Additionally, clustering by topic keyword makes naming the clusters an easier task, as the keywords suggest candidate titles for the clusters.

At a high level, all the algorithms use a hierarchical agglomerative clustering approach with different variations. For instance, some use single-link while others use complete-link clustering. Some compute the similarity metric between topic keywords using a model that maps apps and their topic keywords into an embedding space and calculates the cosine similarity. Others make use of relationships that have been manually encoded between topic keywords, such as synonym (e.g., "puzzle" and "puzzle game") or parent-child relationships (e.g., "game" and "puzzle game") to determine if two apps or topic keywords have ontological similarity. We do not go into more detail as the algorithms are not the focus of this article and are well known.

A suite of algorithms was developed instead of a single one, as engineers found that different approaches produced favorable results for different queries, and no one algorithm consistently outperformed the others, echoing prior work [15]. This is because different algorithms approach a problem in different ways and thus favor different things. For instance, using complete-link as opposed to single-link clustering may produce more coherent clusters generally but may also produce poorer results when there are large outliers [22]. These algorithms were developed over many months of tuning and manual inspection of the output by engineers. Yet they still often create flawed results, containing clusters that are too similar to be distinct or containing topic keywords that do not belong together. This underscores the difficulty in encoding "common-sense" knowledge into computer algorithms.

Given a set of topical clusters for a search query, a search result interface can then be built by ordering the clusters and assigning apps in a greedy way to the first cluster where there is a topic match. The best ordering can then be found by trying all orderings and picking the one that has the most balanced number of apps per cluster. This aspect of the workflow is the same as what experts currently do in the expert workflow.

We now turn to the three stages of the crowd workflow, as shown in Figure 3. They are as follows:

— Evaluate: Pick the best-clustered search result from the raw output of a suite of clustering algorithms.
— Refine: From a single clustered search result, find clusters to merge or delete.
— Title: Choose titles for each cluster based on the most used topic keywords from apps in the cluster to create the final result.

4.2 Evaluate

The first stage of the crowd portion of the workflow takes in as input any number of clustered search results and has the crowd assign ratings to each result to pick the best one. As mentioned in the cluster generation step, we chose to develop a suite of clustering algorithms and have the crowd evaluate the output of each. This stage can be used on its own if only evaluation of different

Fig. 4. A mockup of the first page of the Evaluate step. Screenshots of the real interface are not shown for space reasons. Users see the same query and a different search result produced by a different algorithm on each page (a). For each search result, users must rate each cluster (b) before rating the overall organization (c), and provide qualitative reasoning (d).

clusterings is needed. It can also be used iteratively to rule out or tune an algorithm. Conversely, it can be disregarded if a single algorithm has been decided on previously. Extending Refinery to other types of items may require different algorithms suited for that data and other ways of representing the items. We discuss some potential starting points for different types of data further in the article.

The first page of the evaluation template can be seen in Figure 4. In a single task, users see a series of pages, each showing a different clustered search result for the same query. Each result is generated by a different algorithm in the suite. Workers must rate the individual clusters in the

result, as well as rate the result as a whole and give qualitative feedback. We randomize the order in which workers see the different search results because we saw that people tended to spend more time on and rate more highly the first search result they see. Ratings are collected across workers in parallel so that this step can run quickly, even with many ratings and queries. We take the average across workers as the score for each particular algorithm. We consider the algorithm with highest score as the best, which we then send to the next step in the workflow.

We now address the major design decisions we made, summarized below:

—We show five apps per cluster and up to seven clusters in a search result to the worker.
—We ask for ratings for each individual cluster before asking for a rating for the entire search result.
—The question about the search result is phrased, "*Rate how well organized or grouped the clusters are in this search result.*"
—We filter out tasks results that took less than 150 seconds or don't hover over apps, as well as ones that incorrectly answer a question about the number of clusters.
—We ask workers about their familiarity with the topic of the search query and use this to weigh ratings.

4.2.1 Combating Information Overload. First, there is a great deal of information encoded in a set of clusters that need to be succinctly communicated to the worker. For instance, in the app store presentation of results, there are many clusters per clustered search result, each covering many apps, with each app containing a title, icon, and a description. To mitigate this issue, for a single search result, we only show for each cluster the five apps with the highest download count. This is more apps than would appear in a mobile phone view and equal to the number of apps in a tablet view. The algorithms are also set to create a maximum of seven clusters. While apps may have membership in more than one cluster, we do not allow duplicate apps in the search result view. Lacking a title, we display instead the topic keywords that are most represented in the cluster, picking the top three with highest coverage over the items. All these decisions serve to mimic how the result would look in production. As users may not know the items in question, we display more information about each item upon hovering over the icon. Users can also click on the app to see its page on the app store website. Figure 4 shows an example of a clustered search result using our template.

One way we might have cut down on the information to present would be to break the search result down to individual clusters or apps. However, we chose to have crowdworkers evaluate the clustered search result *as a whole* instead of inspecting individual clusters or item-to-cluster relationships. This is because several of the qualities that experts use to evaluate clustered search results require an understanding of the result as a whole. For instance, when it comes to distinctness, knowledge of what different clusters are there are important to determining what is redundant. Additionally, it is impossible to grade a clustered search result based on coverage of important concepts if not all clusters are visible.

4.2.2 Encouraging Detailed Inspection. Unlike many tagging and rating tasks that need only quick judgments, we found from conducting expert evaluations that understanding how well clusters are organized takes thoughtful inspection of the individual clusters and items. This is due to the abundance of information within each search result as well as the many dimensions in which a cluster may be evaluated. For this reason, we ask for ratings for each individual cluster, as a way to encourage workers to inspect each cluster one at a time, before considering the search result as a whole, as seen in Figure 4(b). We also ask users to spend around a minute inspecting each clustered search result.

Given the inspection needed, it may be difficult to determine which workers are inspecting the clusters closely and providing a reasoned evaluation. Also, since the ratings are all somewhat subjective, it would be difficult to weed out workers based on their ratings alone. To aid our filtering of the worker results, we record time spent on the task as well as time spent hovering over the items to get descriptions. In our evaluation, we filtered out results that took less than 150 seconds or had no time spent hovering over apps. We also include a question on the last page asking the number of clusters in the final search result to filter out workers who are not paying attention. Finally, we ask workers to provide qualitative reasoning for their rating for us to understand what they considered during their inspection, as shown in Figure 4(d).

4.2.3 Determining What Question to Ask. We tried several variations of questions to ask. We asked about the worker's overall opinion, stating, "*Rate your overall opinion of this search result*" as well as a Bayesian Truth Serum variant that asked, "*How do you think other people will rate the quality of this search result?*" Bayesian Truth Serum, which uses raters' meta-knowledge about what other people know, has been shown to produce less subjective results in some cases, such as guessing the rating of a movie [25]. We also tried to ask directly about organization: "*Rate how well organized or grouped the clusters are in this search result.*" Overall, the question about organization performed the best. From the qualitative responses, we saw that people mentioned more personal reasons when asked the question about overall opinion, for example, "*I don't really want to play kiddy games so those can easily be left off.*" The Bayesian Truth Serum format did not improve results, perhaps because a peer's opinion about a clustered search result is less obvious than, say, about a movie. We also found that asking multiple questions about different aspects of the clustered search results, such as coverage or relevance, led to most people simply putting the same rating for all the questions. In the end, we choose to ask the question about organization, as seen in Figure 4(c).

4.2.4 Taking into Account Expertise. We found while conducting expert evaluations that having some prior knowledge of the specific domain of the query and apps made us better evaluators. For instance, the search query "rts" is shorthand for "real-time strategy," a sub-genre of strategy video games. If a user is unfamiliar with this subgenre, it would be difficult to understand how apps should cluster into more specific topics. Thus, on the final page of the template (not shown), we ask workers to rate on a five-point Likert scale their prior knowledge of the search query and familiarity with the apps presented. We use this information to weight workers' ratings when averaging them together for the final score. This question is not shown in the sample template in Figure 4.

4.2.5 Comparison with Expert Evaluation. While experts in the expert workflow judged the search results using a number of explicitly-stated characteristics, we chose to ask crowdworkers more broadly about the overall organization. This was because we were concerned about overwhelming workers with terms they may not be familiar with. Also, some of the evaluation characteristics in the expert workflow were not relevant in the crowd evaluation because of their lack of domain knowledge or lack of information due to our template. For instance, coverage is hard for a crowd worker to determine if they do not know the app space well. Similarly, balance cannot be determined because we do not show the entirety of each cluster to the worker. Since we didn't ask about specific characteristics, these may lead different crowdworkers to emphasize different characteristics they care about, creating more noisy rating data than experts.

4.3 Refine

The second step of the crowd portion of the workflow takes the best-rated clustered search result and asks workers to select manipulations to be made to the search result to improve it. We chose

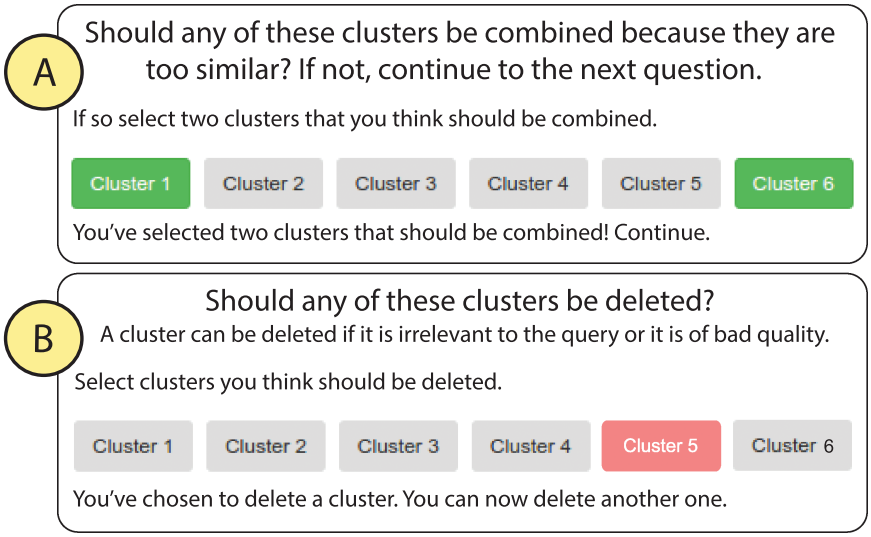

Fig. 5. The question for the Merge task (a) and the question for the Delete task (b). Each question appears in its own separate template similar to the EVALUATE template, except the question replaces the qualitative reasoning portion.

to allow two different types of manipulations, taken from the ways that experts currently refine clusters. They can merge two clusters that are too indistinct or delete a cluster that is incoherent or irrelevant, as seen in Figure 5. In Figure 5(a), we show the question for asking workers to suggest clusters to merge, and in Figure 5(b), we show the question for suggesting clusters to delete. Each type of manipulation is a separate template and thus a separate task that can be run in parallel. The template for each task looks similar to the template in the Evaluate step, except that the qualitative reasoning part in Figure 4(d) is replaced with the respective merge or delete question. As demonstrated in the workflow diagram in Figure 3, merge and delete tasks are performed on the single clustered search result that was rated the best according to the crowd for a query.

In the following, we explain design decisions we made while building these two templates, summarized below:

—We ask workers to rate individual clusters and the clustered search result before considering refinement tasks.
—We separate merging and deleting into their own tasks, and workers can only suggest one merge action.
—We use thresholding to implement a merge or delete an action.

4.3.1 Encouraging Inspection. Again, we chose to ask workers to individually rate each cluster to encourage them to inspect clusters one by one. We also ask the question about overall organization again to have the workers thinking specifically about the organization of the clusters before they consider clusters to merge or to delete. To aid in the decision-making process, hovering over the buttons for a cluster in the question highlights the cluster in the view of the clustered search result. As before, we also ask a question about the number of clusters as a check as well as a question where users rate their expertise.

4.3.2 Keeping the Task Simple. To keep the task simple for workers, we separate the potential actions of merging and deleting into their own respective tasks. For the Merge task, we only allow

workers to select one pair of clusters to merge, as adding the possibility to select more makes both the decisions and the visualization more complicated. However, we do allow users to suggest as many clusters to delete as they wish. This is in line with experts' behavior, who we saw, overall, perform more deletes than merges and rarely more than one merge in a single result. Workers are able to undo their suggestions and are also not required to suggest something to merge or delete. This is because sometimes there may not be anything that should be merged or deleted from the original algorithm output. Requiring workers to put something might result in adding more noise to the results.

4.3.3 Using Thresholding to Pick Final Actions. To combine different workers' suggestions together, we chose to use thresholding instead of adding a verification step using the crowd. From pilot tests, we saw that there were often many suggestions for both merge and delete tasks that had only one person suggesting that change, and those overall had low quality. In contrast, the suggestions where several workers picked the same action reduced the noise from the suggestions. In the end, we chose to set a threshold for each type of task so that a certain number of workers needed to suggest a merge or a delete before implementing the change. Thus, we simply perform the change if a threshold was passed, in the interest of cost and time, as we determined from the pilots that the changes were of high quality. We also did not want to introduce more complexity or steps into the workflow and increase cost per query and overall time.

However, a system with more checks in place could incorporate a crowd verification step, such as reusing the template from the Evaluate step and showing crowds the clustered search result before and after the change for the crowd to evaluate. Indeed, in a further section on evaluation of the Refinery workflow, we performed this step to have a separate crowd evaluation of the merge and delete changes. As with before, all tasks are run in parallel. We stayed away from iterative improvement workflows in general as they are more time-consuming. This means that there may be conflicts between the merge and delete changes. In the case of two different actions suggested on the same cluster, we chose to go with the merge action since it is rarer and thus a stronger signal.

4.3.4 Comparison with Expert Refinement. Unlike the expert refinement step, the crowd refinement step does not allow workers to delete a topic from a cluster if it does not go with the other topics in the cluster. We chose to not allow this action because it may require more domain knowledge about the underlying algorithms and the app and topic space to know if the change would be successful. This may mean that well-defined clusters that have one mis-clustered topic may be deleted in the crowd workflow or left as is, lowering the overall quality of the search result. In the future, interactive tools for crowdworkers to attempt changes and revert them might allow for more complicated actions.

4.4 Title

After the refinement step, the crowd picks titles for each of the clusters in the clustered search result. Up until this point in the workflow, we have shown users the top three topic keywords for each cluster in the clustered search result as a way for users to get a sense of the topics that each cluster comprised. In a way, these clusters also served as a makeshift title for the clusters in the template view that workers saw. However, we want cluster titles that would work in a production setting. The existing topic keywords would not always work as good titles as some topic keywords are synonyms of each other and some reference specific aspects of the cluster but do not describe the entire cluster.

Thus, to select the title, we have crowdworkers create a title by selecting from the top topic keywords of the cluster. We noticed when experts picked titles in the expert workflow that they

Fig. 6. A sample question for the Title task for the query "baby." In the template for this step, there is one question for each cluster with more than one main topic keyword. Like prior steps, the entire search result is shown for each task.

often simply picked from one or two of the existing topic keywords describing the cluster as the title.

Figure 6 shows how a question for a cluster might look. We allowed workers to pick one or two of the topic keywords as a title, given that sometimes a second topic keyword would provide additional contextualization. In the template, workers would see the entire clustered search result much like in Figure 4(a). Then for each cluster that had more than one topic keyword, workers would pick one or two of the topic keywords as the new title. We then picked the topic keyword or keywords that were most suggested by the crowd as the new title of the cluster. We decided to show users the entire search result again as choosing the title for one cluster might affect how one would choose the title for another cluster. For instance, titles may want to emphasize how two clusters are different instead of highlighting how they are similar to enhance distinctness. Thus workers might be answering several questions, like in Figure 6, for a single task.

4.4.1 Comparison to Expert Titling. In the expert workflow, experts could write novel titles using free-text if the top topics sounded awkward as a title or weren't good descriptors. However, this would be difficult to aggregate across many workers' suggestions and would require additional crowd steps to vote on the best suggestions. While our crowd approach provides a cheap and fast alternative, there may be cases where the title is not as good as expert-written ones. Further in the article, we discuss how attributes used to cluster items can provide a shortcut toward titles for other types of items.

5 EVALUATION

We now turn to an evaluation of the Refinery workflow that was run using real search queries and apps taken from the Google Play Store. In total, we put 120 search queries through the workflow. These queries are chosen from a list of high volume queries on the app store. They were targeted for clustering by product managers because they were more broad, categorical queries that might benefit from clustering, as opposed to queries about specific apps.

Fifty-seven of the 120 queries went through an internal expert workflow, which comprises the same cluster generation step using the same suite of clustering algorithms and a cluster evaluation, refinement, and titling step performed by experts. The results of the expert workflow appear on the public-facing app store as clustered search results to millions of users.

For each step of the crowd portion of the workflow, we evaluate the results produced by the crowd using the judgment of two experts. We also conduct evaluations of alternative templates

in each step to arrive at our final workflow templates. The two experts used in the evaluation are authors of this article as well as members of the internal team employed to do expert evaluation and refinement of Google Play Store clusterings.

We also conducted an end-to-end evaluation by comparing the final results of Refinery against the 57 clustered search results that went through the expert workflow. Given that our Refinery workflow could be run quickly in parallel and on demand using non-expert crowdworkers, it would be a useful replacement for the expert workflow if it produced comparable results as the clustered search results on the app store, which have gone through many hours of expert evaluation and refinement.

5.1 Comparison of Crowd Evaluation

We begin with an evaluation of the first step of the Refinery workflow: the EVALUATE step. Taking 120 search queries, we generated four clustered search results for each query using four algorithms in the suite of clustering algorithms developed for this purpose. Each clustered search result appeared on a separate page in the EVALUATE template. A single Mechanical Turk task would then consist of four pages with a clustered search result on each page, all for the same query. We estimated that workers would spend around 6 minutes (1.5 minutes per result), and thus we paid 70 cents per task (17.5 cents per result), basing our pay on a equivalent wage of $8 per hour. We collected ratings from 50 workers for each query so as to gather more data at this point in our study. In the next section, where we present the results of our comparison, we discuss the tradeoff between number of ratings and agreement with experts in more detail. For comparison purposes, we randomly picked 60 queries out of the 120 and had two experts go through and rate each of the algorithm outputs for each query on a five-point Likert scale. Each expert spent around 1 minute studying each cluster search result, or a total of 4 hours per expert for 240 clustered search results.

We summarize our findings below:

- —Experts have fair agreement (Cohen's Kappa of 0.396) in pairwise comparisons with each other. This highlights the relative subjectivity of the task.
- —The crowd agrees with experts (Cohen's Kappa of 0.403) about as often as experts agree with each other.
- —However, we need many crowd ratings (around 40) to achieve comparable performance, though more could be done to pre-screen workers, since we drop around half.
- —Alternative interfaces such as side-by-side comparisons of clusterings, or averaging ratings of individual clusters within a clustering perform poorly.
- —Alternative rating questions such as asking for their overall opinion or a Bayesian Truth Serum variant did not perform as well as asking about organization.

5.1.1 Agreement between Experts. As mentioned, the task of cluster evaluation is difficult because it can be very subjective. There are many characteristics that can be considered toward assessing the quality of clusters, and people may assign different levels of importance to each characteristic. We noticed when comparing discrepancies between expert opinions that one person might care more about a misplaced app while another person might care more about two clusters that are similar. One person might be pickier than another person overall, transposing their ratings down a level. Also, there may be minor differences between two clustered search results for the same query, making it difficult to determine which is better. For this reason, there are several ways to consider inter-rater agreement. On way would be to consider how well experts agree when it comes to picking which clustered search result is better out of a pair. When we looked at pairs of results within a query, experts had a Cohen's Kappa score of 0.396, signifying fair agreement. We also computed how well experts agreed about the relative rank of the four

Table 1. Inter-Rater Reliability Comparing Search Results

Rating Type	Pairwise IRR	IRR (Experts Agree)
Experts	0.396	1.0
Final EVALUATE template	**0.403**	**0.669**
Side-by-side preference template	0.194	0.267
Average of individual cluster ratings	0.258	0.450
Overall opinion question	0.255	0.419
Bayesian Truth Serum opinion question	0.347	0.618

Note: Results for picking the better search result when comparing two search results for the same query. There were 360 pairs in total (four algorithms for 60 queries). We show the inter-rater reliability (Cohen's Kappa) overall as well as when we constrain to cases when experts agree. Results are shown between two experts, and between the experts and the crowd for our final template which has workers rate four results sequentially and asks about organization. Below that, we also show results between experts and the crowd for ratings from a side-by-side template, from averaging scores for individual clusters, and from asking about overall opinion or a Bayesian Truth Serum variant.

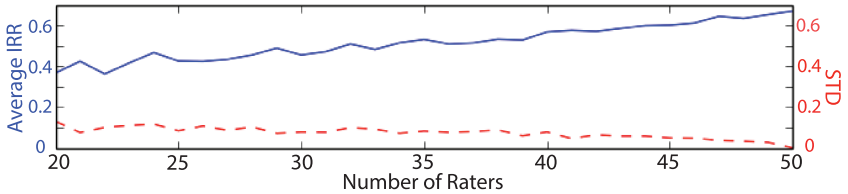

Fig. 7. μ and σ of the IRR with experts when randomly sampling from the 50 worker ratings. Twenty random samples were taken for each number of raters, constrained to the cases when experts agreed.

results for each search query. We compute this as an average of Spearman rank correlations for each query. Between the two experts, this was 0.384, which also shows fair agreement on within-query ranking. The Spearman correlation of the experts' ratings across all queries was 0.547 ($p < .0001$), which indicates that experts' ratings across queries were moderately correlated.

5.1.2 Results of Crowd Compared to Experts. When we compare crowd ratings to experts' ratings, we find overall that *the crowd agrees with experts about as often as experts agree with each other.* When it came to comparing pairs of clustered search results within queries, the inter-rater reliability between the crowd and experts was 0.403 (Cohen's Kappa). Interestingly, when we looked only at the algorithm pairs where both experts agreed on the better one, which happened 29% of the time, the inter-rater reliability scores between the crowd and experts improved to 0.669, as seen in Table 1. As for average within-query ranking, the crowd had an average correlation with experts of 0.406 (Spearman's rank). However, when it came to correlating scores across all the queries, the average correlation was lower than for experts ($\rho = 0.311$, $p < .0001$). This may be because the same crowdworker had to rate all four algorithm results for each query in one task but different workers rated different queries. We also demonstrate how the pairwise agreement between crowds and experts alters when we reduce the number of raters from 50 and randomly sample, as seen in Figure 7. This finding dovetails with prior work demonstrating that many crowd annotations are necessary to achieve the highest quality in relation to experts [14]. While more ratings perform better, we still achieve a Cohen's Kappa of 0.363 for all pairs and 0.567 for when experts agree using 40 ratings. Likewise, average within-query correlation is 0.367. The number of ratings to which we can limit while still achieving similar performance as experts includes the

ratings we discarded due to workers spending under 150 seconds on the task, not hovering over any of the apps, or incorrectly answering the question about number of clusters. Out of the 50 possible ratings per query, this amounted to on average 26.77 ratings discarded. Future work to reduce costs could involve better pre-filtering or pre-training of the workers.

5.1.3 Comparison with Alternatives. We now discuss the results of experimenting with alternative templates and ratings. Table 1 shows results for each alternative's agreement with the experts. One of the earlier templates that we piloted was a side-by-side approach that took two different clustered search results and put them next to each other. It then asked the worker to pick the result they thought was better on a Likert scale. The rationale was that it might be easier for users to visually compare two search results on a single page and make a judgment about which is better than to rate one at a time. However, in our evaluation of the different templates, we found that the side-by-side ratings did not perform well, as see in Table 1. Analysis of the qualitative responses showed that the majority of people tended to simply pick the clustered search result that had more clusters. This would be reasonable if the two results were of similar quality, as the one with more clusters would result in more apps shown and ostensibly better coverage. However, this preference persisted even in cases when the search result with more clusters had more incoherent clusters compared to the search result with less clusters. For instance, one worker explained their preference by stating, "*More games to choose from even though some of the games weren't relevant.*" This preference also continued when we had users inspect and rate one search result at a time, the same as in Figure 4, before finally showing a final side-by-side view where they then rated their preferred side. Indeed, in a pilot with 5,040 ratings of pairs of search results, we saw that 28% of the time, workers would rate one side higher when looking at them individually, but then state a preference for the other side in the side-by-side view.

We also considered deriving a crowd rating from ratings of individual clusters. As shown in the Evaluate template, we ask users for a binary rating of each cluster before we ask about the overall clustered search result to facilitate inspection. We tried to additionally use the individual ratings of each cluster as a rating for the overall clustered search result by averaging the individual cluster ratings together. However, this performed poorly, as seen in Table 1. Thus, we chose not to consider templates that break up clustered search results, for instance to ask workers to rate individual clusters without seeing the full search result. If individual cluster ratings perform poorly even when all clusters were presented together, then tasks where users only see one cluster at a time without the rest of the search result may provide an even poorer indication of overall quality.

Finally, we show results in Table 1 for using the alternative questions we tried when asking for clustered search result ratings. A question about overall opinion and a Bayesian Truth Serum variant both performed worse than the question we eventually used, which asks about cluster organization.

5.2 Evaluation of Crowd Refinement

After choosing the best-clustered search result, we moved to the Refine step. We had 120 Merge tasks and 120 Delete tasks run in parallel on the 120 queries. Each task was completed by 20 workers. We paid 20 cents per task with an estimate of 1.5 to 2 minutes per task. We set a threshold of 6 out of 20 and 5 out of 20 for Merge and Delete, respectively, which we determined experimentally. Any merge or delete suggestions that passed their threshold were performed. After this filtering step, 51 actions passed the threshold and 47 results out of 120 were altered. To evaluate the changes, we had two experts go through the 51 actions and validate them. If the expert felt that completing the merge or delete would improve the clustered search result, then it was marked correct. If the expert was unsure or thought the result would get worse, then it was marked

incorrect. In total, the two experts agreed on 88.2% of the actions, with a inter-rater reliability of 0.55 (Cohen's Kappa). Out of 51 actions, both experts approved of 78% of the actions and one or more experts approved of 90% of the actions.

5.2.1 Crowd Evaluation of Refinement Changes. We also performed a crowd evaluation of the changes. Reusing the EVALUATE template from the first step, we created tasks where workers would see the clustered search result before the change (BEFORE) as well as after the change (AFTER) in randomized order. Overall, the crowd rated the AFTER as better 83% of the time. When looking at the clustered search results that improved from BEFORE to AFTER, the improvement in the crowd rating was an average of 0.19. In contrast, when looking at the clustered search results that were rated worse after the change, the change in rating was an average of −0.05. This difference in rating change between the improved versus worsened results was also significant ($t = 2.08$, $p < 0.05$, $ES = 1.03$). Thus, most clustered search results improved after the refinement step, and even for the search results that did not improve, they were rated around the same as before instead of worsening.

5.3 Evaluation of Crowd Titling

After performing the merge and delete changes that the crowd suggested and that passed the thresholds, the crowd performed the third step in the workflow: suggesting titles for each of the clusters. As in the previous step, we had 20 workers put suggestions for titles for each clustered search result. We paid 15 cents per task, with an estimate of 1 to 1.5 minutes on average. As stated earlier, suggestions were asked only for the clusters that had more than one topic keyword. Out of 120 clustered search results, there were 362 clusters that the crowd was asked to suggest a title for. We had two experts go through each of the titles and validate them. If a title fit the criteria mentioned earlier in the expert workflow for new titles, then the titles were approved. Overall, the experts agreed on 90.3% of the titles, though the Cohen's Kappa measure of agreement was fair (0.396). In total, 86% of the titles were approved by both experts, and 96% of the titles were approved by at least one expert.

5.4 End-to-End Comparison

After all 120 queries went through the Refinery workflow, we compared results from the 57 queries that also went through the expert workflow and are now serving live traffic on the Google Play Store. These clustered search results went through an hour or more of manual refinement by experts to reach their current state. To evaluate the two search results, we had two experts rate them both on a five-point Likert scale. From 114 total ratings, the two experts agreed 57% of the time and had an inter-rater reliability of 0.38 for five categories (Cohen's Kappa).

We summarize our findings below:

- —The quality of the Refinery search results approaches the quality of the expert-refined search results (Likert rating of 2.54 versus 2.82, respectively).
- —Qualitatively, much of the improvements in the expert workflow trace back to better titles, while occasionally, the Refinery workflow finds more clusters that are still distinct.
- —The quality of the Refinery search results improves over the quality of the original output of the algorithms (Likert rating of 2.54 versus 1.25, respectively).
- —The Refinery workflow is less costly than the expert workflow and more scalable.

5.4.1 Comparing Experts and Refinery. The average rating from 0 to 4 for the app store clusters was 2.82 ($\sigma = 0.78$). The average rating for the Refinery clusters was 2.54 ($\sigma = 0.97$). Averaging both experts' ratings together for each search result, in 51% of the queries, the Refinery result was

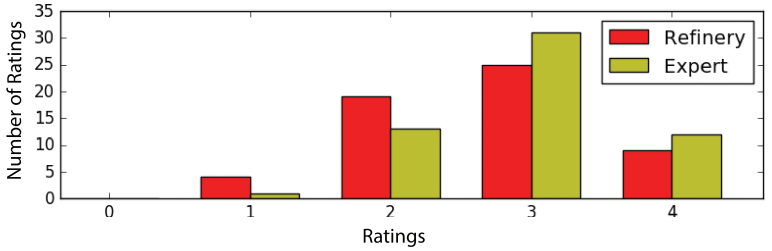

Fig. 8. The histogram of total number of ratings between 0–4 on a Likert scale by two experts comparing the output of the Refinery workflow versus the Expert workflow.

rated equal or higher to the app store result. In 49% of the time, the app store result was rated higher. The histogram in Figure 8 shows all the ratings by both experts for outputs of the Refinery workflow as well as the expert workflow.

As an example, the left of Figure 9 shows the clustered search result for the query "trivia" from the Refinery workflow. The right of Figure 9 shows the equivalent search result from the expert workflow and which appeared on the app store. Comparing these two results, we can see that the Refinery result breaks the apps into more detailed categories while not sacrificing distinctness for the most part. In this case, both experts rated the Refinery result as better than the expert result. The Refinery result was also rated an average of 1.5 better than the raw output of the four algorithms for this query.

Another example with the same number of clusters for each condition is shown in Figure 10, where the left shows the clustered search result for the query "zoo" from the Refinery workflow while the right shows the equivalent search result from the expert workflow. As can be seen, the clusters are similar across the two clusterings, though the names of the second clusters are slightly different ("educational game" versus "family zoo games"). We also notice some overlap on the expert-refined clustering, with apps called "Zoo Robot" appearing in both the first and last clusters. In this case, both experts gave each search result a rating of 3, while the average rating for the raw output of the four algorithms was 1.

When examining the clustered outputs more carefully, we found that Refinery sometimes had titles for clusters that were overly general and did not adequately describe the cluster separately from the other clusters. This was because the source of the titles was topic keywords that are generally written as opposed to written for the query. For instance, a "simulation game" or "casual game" cluster does not make much sense when the query is about a specific type of game, such as "cars." In the fewer cases where Refinery performed better than the expert workflow, we noticed that, much like the "trivia" example, the Refinery workflow was able to isolate distinct categories that the expert workflow missed or grouped together. As another example, the search query "relaxing music" has three clusters in the expert clustering, with titles "Meditation Music," "Nature Sounds," and "Classical Music Apps." In contrast, the Refinery clustering has six clusters; in addition to the three in the expert clustering, it also had clusters titled "Insomnia," "Health," and "Lullaby."

5.4.2 Comparing Algorithms and Refinery. The average expert rating for the four raw algorithm outputs for the 57 queries was 1.25, compared with the average expert rating for Refinery clusters of 2.54. This was a statistically significant difference ($t = 6.038, p < .0001, ES = 1.750$). When looking at the four algorithms individually, the expert ratings of their raw output ranged from a low of 1.06 ($\sigma = 1.05$) to a high of 1.48 ($\sigma = 1.16$), with a significant difference between the overall best

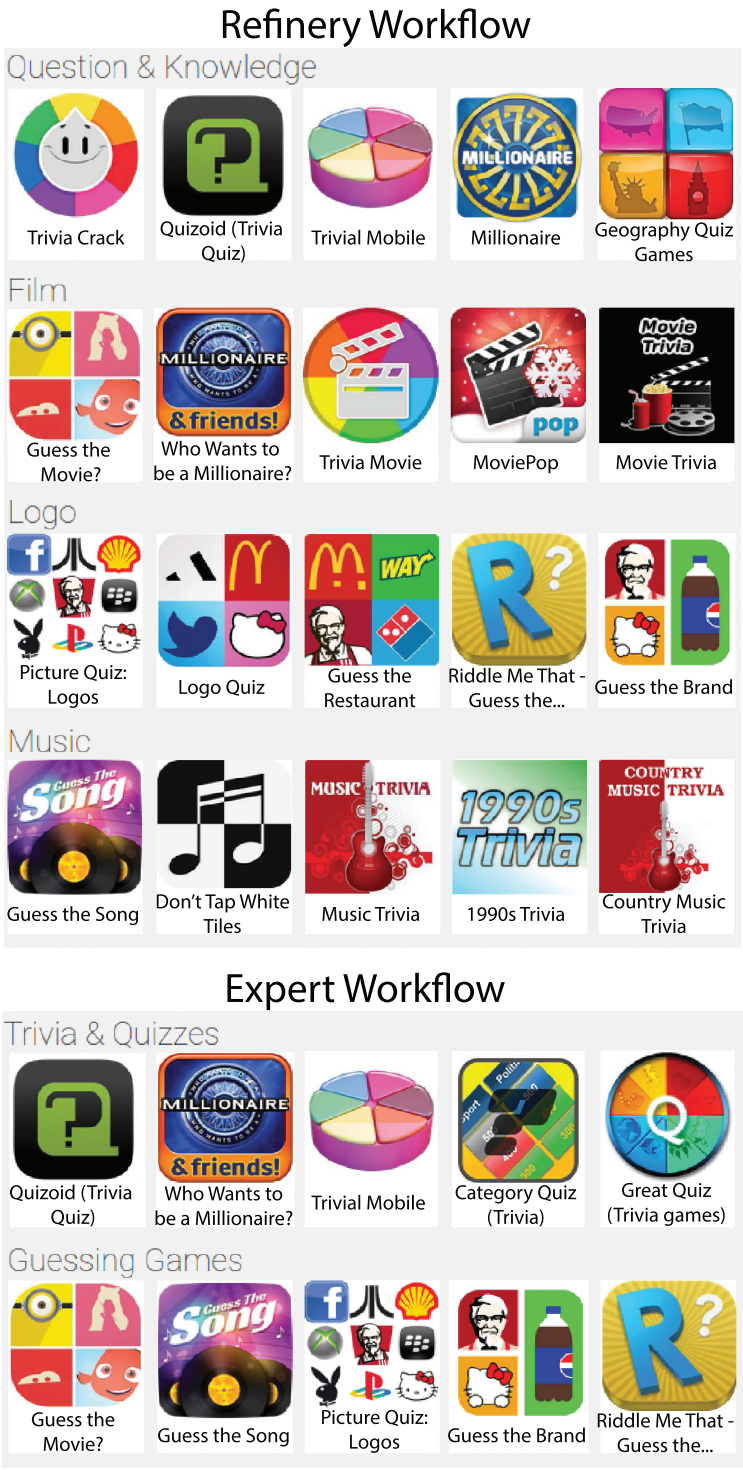

Fig. 9. One the top is the clustered search result for the query "trivia" that went through the Refinery workflow, while below is the clustered search result that went through the expert workflow.

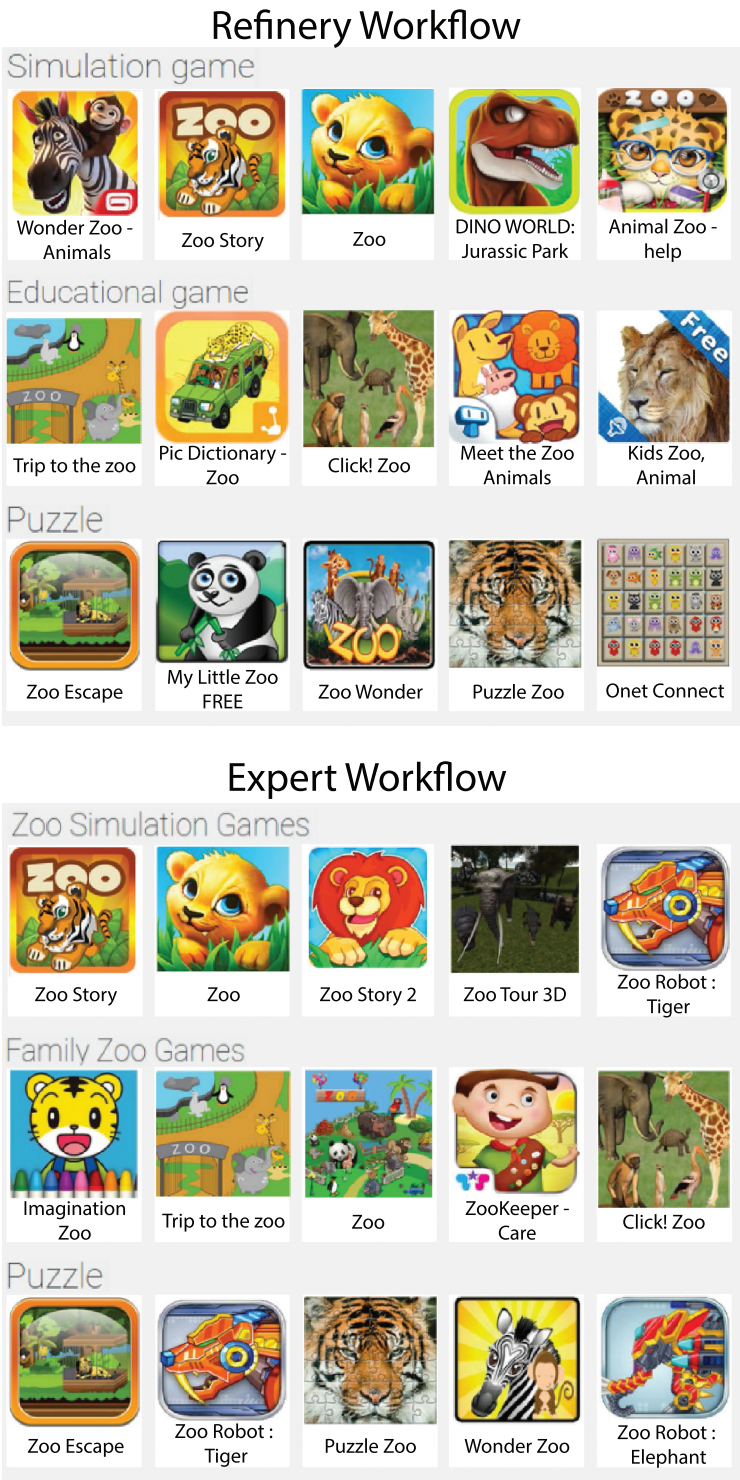

Fig. 10. One the top is the clustered search result for the query "zoo" that went through the Refinery workflow, while below is the clustered search result that went through the expert workflow.

and worst algorithm ($p < .001$). The crowdworker ratings in the Refinery workflow reflected the expert ratings, with 21 out of 57 queries eventually using the algorithm with the highest average expert rating, while only 8 queries used the algorithm with the lowest average expert rating. The crowd ratings of the output selected from the four algorithms after the first stage of the workflow were generally low at 1.31 on average. This suggests that perhaps a future cost-saving measure could whittle down the potential algorithms to a smaller number that go into the Refinery workflow. We also saw no significant difference between which algorithms tended to receive merge or delete refinements.

5.5 Costs of Experts Compared to the Crowd

Finally, we compare the cost of the expert workflow with Refinery. In the expert workflow, a query may take an hour for experts to inspect, evaluate, and refine. Assuming an engineer works 2,000 hours a year and makes a salary of $100,000, an hour would cost around $50 in labor. Using the crowd workflow, the Evaluate step with 40 ratings per query and 70 cents per task would cost $28, the Refine step with 40 suggestions per query would cost $8, and the Title step would cost $3. While this is already more cost effective than using experts, the bulk of the cost is in the Evaluate step, which can be further reduced. First, as mentioned, about half of the ratings were discarded due to not passing several filters we set. More effort could be made to pre-screen or pre-train workers, which would reduce the number of ratings we would need to collect to a number more like 20 or 25. Also, when there are many algorithm outputs to compare, it might be preferable to use a generic automatic metric, such as the Silhouette coefficient, to remove the worst performing ones, and then use the crowd to rate the top two or four. In addition to being cheaper than experts, with crowds we can scale up to many queries easily, making the workflow faster for large batches of queries. We can also employ crowdworkers on an as-needed basis while experts are much more inelastic in supply.

6 DISCUSSION

From our evaluation conducted on real app store data against an existing production workflow used for the Google Play Store, we found that Refinery produced results that approach and occasionally exceed the quality of the results on the app store that took many hours of experts' time. We consider how our findings generalize to other search queries and other use cases and how experts, crowds, and algorithms could be utilized so as to highlight their respective strengths.

6.1 Generalizing to Other Search Queries

In our article, we specifically focused on broad queries describing a category of apps, such as "sports games," which signal exploration, and rejected queries signaling a particular result desired, such as queries containing a specific app name. Our decisions to include or reject specific queries were made by inferring users' intent from the search query and picking queries that appeared to have unambiguous, exploratory intents. Because of this initial filtering step, we also allowed crowdworkers to infer on their own the intent of the query when judging the search results. To expand this work to more ambiguous queries, future work could involve gathering a deeper understanding of users' different intents through log analysis of what they downloaded or through surveying searchers. Then, we could ask crowdworkers to judge the search results for each of the main intents provided as opposed to allowing them to infer intent on their own. Questions for crowdworkers here could be about whether there exists one or more clusters within the clustered search result that satisfy a given intent.

When considering the type and number of queries possible using Refinery, our cost per query still constrains us to a certain number of queries before becoming prohibitively expensive. For

instance, thousands of queries may be reasonable but millions would not be. However, there are diminishing returns to implementing clustered search results for the long tail of search queries. Other strategies such as finding more common rephrases of rarer queries could reduce this tail and re-use existing search results.

Another possibility is re-using individual clusters from search results that have gone through Refinery. Because our clusters are made up of weighted set of topic keywords as opposed to individual apps, they are more likely to be re-usable in a different search result. There are some issues with this approach however. First, the title made for the cluster in the original search result may not transfer well to a new search result because it is emphasizing a particular aspect of the cluster that made more sense in the original context. Additionally, if multiple clusters taken from different search results are re-used in a new search result, there is the risk of these clusters being insufficiently distinct when placed together. A cluster may now also be too broad or too narrow given the relative specificity of the new query in comparison to the original one. Finally, there is the question of what to do with the leftover items that do not fall into one of these clusters. Given these issues, it may be a better strategy to refine and re-use clusters at the search result level, where clusters can be evaluated in context, as opposed to mix and match pre-refined clusters to construct a new clustered search result.

As mentioned earlier, there are also queries that may never be suitable to clustering because they return relatively few results or are seeking a particular item instead of exploring a broader category. This could be determined by examining usage patterns or surveying searchers.

6.2 Generalizing to Other Applications

Though our implementation was evaluated on an app store, the Refinery workflow can generalize to many applications that could be improved with clustering. For instance, any online collection of items, from online shopping to recipe repositories to web design themes, to name just a few, could benefit from clustered search results and could use the Refinery workflow. Items could be tagged with labels by user contributors, community members, employees, machine learning algorithms, or crowdworkers. These labels could feed into a number of suitable off-the-shelf clustering algorithms such as the ones we used. The Evaluate template could also flexibly be used in a pre-processing step to aid engineers toward tuning the parameters of a single algorithm or whittling down the set of algorithms to try using a sample of queries.

However, the quality of the raw output of the algorithms as well as the input to the Title step does depend on the quality of the item labels. This may require some manual generation and cleaning of tags or automatic approaches when tags do not exist and are expensive to create. Much research dedicated to crowd labeling [32] as well as automatic label extraction from text and images explore making this process cheaper and better. However, many existing online collections already have high quality tags and even expert-generated ontologies of tags for the purpose of search or recommendation, such as on Netflix or Etsy. Yet most of these sites still do not show clustered search results beyond some global, expert-defined categories for a few high-level queries or on their home page. For instance, both Netflix and Amazon show different categories of items on the home page but only return search results in a ranked list.

Our template designs can also generalize to other types of items, with some modifications. Different types of items, such as ones that are primarily text, would need to alter how the clusters are presented to the user, but otherwise could use a similar workflow. An ever-present challenge is determining how to present all the necessary information to workers without overwhelming them. In the app store case, we only show a limited number of items and topic keywords per cluster and only allowed a limited number of clusters. For many evaluations of end user systems, this is sufficient since that is all most end users will see. In cases where there are more items

or clusters that need to be part of the evaluation, it may be feasible to show a random portion of the items or the clusters to different crowdworkers and combine the results. However, this portion should constitute a significant part of what is there, otherwise workers may be getting too restricted of a picture to make a reasoned evaluation. Another possibility is to use a "sample and search" approach [6] to let crowdworkers actively explore the entire cluster.

Finally, given the aforementioned downsides of automatic and gold-standard evaluation methods, crowd evaluation of clusters may be a faster and cheaper way for researchers and algorithm developers to get an evaluation of their clustered output that approaches the quality of expert evaluation. Our work is one of the first to evaluate holistic cluster evaluation with the crowd using comparisons with experts.

6.3 Interplay of Crowd and Expert Refinement

Our research findings also suggest the pros and cons of using crowds versus experts for refinement tasks, and the importance of combining them when appropriate. In our use case, we found that a small set of changes made by the crowd to the raw algorithm output led to a significant improvement in quality. This is useful for systems that need refinement capabilities at scale as it involves little overhead, time, and money but makes an outsize impact.

However, these changes do not cover all the actions that could be done to alter a clustering. For instance, one could imagine asking the crowd to move a topic keyword or an item from one cluster to another or delete a topic keyword or item from a cluster. We piloted several templates for these actions and found that the crowd made reasonable suggestions given the clustered search result that they saw. However, difficulties arose when it came to actually implementing actions based on the suggestions that the crowd made. For instance, an item that is incorrectly clustered could signal a poor topic keyword assignment, or that a topic keyword should be moved, or that just the item needs to be moved. Likewise, moving a topic keyword and its items from one cluster to another might also have unintended consequences, where the result is two clusters that are more indistinct. This is because the interface the crowd sees is not the entire picture, and crowds do not have the same amount of insight that experts do into how the algorithms work and how the topic keywords are assigned.

From our early experiments, we believe that one solution for these particular tasks might be to let crowds suggest problem areas and then have experts make the final decisions about what to do. When experts are analyzing the clusters, they bring in their understanding of why the algorithms separated the topic keywords the way they did, which topic keywords cover more items, and how much overlap there is between two topic keywords. The crowd, on the other hand, can point out problem areas that experts overlook and find problems in much shorter time than it would take experts. This saves experts a great deal of time as they no longer need to inspect every result and instead can focus on the clusters that crowds deemed problematic. This also makes experts' jobs easier as they only need to approve or make decisions based off a set of suggestions instead of devising their own solutions. For other kinds of tasks where the implementation of a fix is more clear, such as the ones we included in Refinery, less expert involvement may be needed.

Another solution may be to develop dynamic templates that update when crowdworkers move a topic or remove an app. Prior work in interactive interfaces for expert manipulation of clustered search results could be useful here [7]. More experimentation is needed to understand how crowdworkers could pilot such an interface.

6.4 Pros and Cons of Humans and Algorithms

In much of the research around cluster evaluation and refinement, clusters manually created by experts are seen as the gold standard. However, there may be times when manual clustering may

not perform well. This may happen because there are too many items for a person to keep in their head, or there are nuances in the dataset that the person might not realize. In early experiments with manually crafted clusters using experts, we discovered that, for many queries, algorithms would find groupings that experts would overlook. In addition, using humans to manually create clusters is extremely time-consuming and expensive, while an algorithm is fast and free.

On the other hand, this article makes it clear that algorithms can make mistakes that are obviously wrong to a human. Given the respective strengths and weaknesses of humans versus algorithms, a sensible approach is to use algorithms for the initial, low-level, detailed work, and then have humans work on higher-level evaluation and refinement. This runs counter to the technique used by many crowd-powered clustering methods that break down the clustering task into small, low-level components for workers to complete. There was initial concern that our tasks would be too subjective, too high level, or involve too much introspection for crowdworkers to provide a good signal. To mitigate these issues, we made several design decisions to present only essential information and to guide the worker through the process. In the end, our results show the crowd managed to agree as much as experts, and our method is cheaper and easier to scale than many methods that involve humans at an earlier stage.

7 LIMITATIONS AND FUTURE WORK

While we considered how our findings may be extrapolated to clusters other than search results on an app store, we did not evaluate other use cases. Specialized implementations of our workflow could require unforeseen changes. However, as our specific application affects millions of app store users every day, this project has had to contend with issues of scale and high standards of quality control that are imperative in industry, but may not come in to play in research prototypes.

Our article is also limited in that we only allowed limited forms of refinement. Future work could explore converting our template from a static one into a dynamic one where workers could try out changes, see the result, and be able to undo. Also, envisioning a human-in-the-loop cycle, it would be interesting to consider how the suggestions and evaluations that workers make could feed back into the algorithms to improve them.

Finally, an issue that would be interesting to explore is how to insert some aspect of personalization into the clustered search results while still maintaining high quality. For instance, the clusters or the apps within the clusters could be ordered differently according to different people's interests. More studies would need to be done to see how successful this would be for users and how cluster evaluation and refinement would be impacted.

8 CONCLUSION

In this article, we designed and developed Refinery, a hybrid machine and crowd workflow for generating, evaluating, and refining clustered search results at scale. This workflow was evaluated against an expert-powered workflow currently providing clustered search results for one of the largest app stores in the world, the Google Play Store. Taking into account the respective strengths and weaknesses of algorithms and humans, we chose to involve algorithms in the cluster generation portion and reserve the crowd for evaluation and refinement of the algorithm outputs. We found that the Refinery workflow produces results that approaches and is occasionally better than results on the app store refined over many hours by experts. Our results and findings can serve researchers and practitioners seeking to evaluate or refine clusters using human involvement in a scalable way.

ACKNOWLEDGMENTS

The authors would like to thank the members of the Seamless Interactive Recommenders team at Google for their feedback, as well as all the crowdworkers that participated in our experiments.

REFERENCES

[1] Paul André, Aniket Kittur, and Steven P. Dow. 2014. Crowd synthesis: Extracting categories and clusters from complex data. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW'14)*. ACM, New York, NY, 989–998. DOI: http://dx.doi.org/10.1145/2531602.2531653

[2] Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: A word processor with a crowd inside. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology (UIST'10)*. ACM, New York, NY, 313–322. DOI: http://dx.doi.org/10.1145/1866029.1866078

[3] Jonathan Bragg, Daniel S. Weld, and others. 2013. Crowdsourcing multi-label classification for taxonomy creation. In *1st AAAI Conference on Human Computation and Crowdsourcing (HCOMP'13)*. AAAI, Palm Springs, CA.

[4] Deng Cai, Xiaofei He, Zhiwei Li, Wei-Ying Ma, and Ji-Rong Wen. 2004. Hierarchical clustering of WWW image search results using visual, textual and link information. In *Proceedings of the 12th Annual ACM International Conference on Multimedia (MULTIMEDIA'04)*. ACM, New York, NY, 952–959. DOI: http://dx.doi.org/10.1145/1027527.1027747

[5] Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. 2009. A survey of web clustering engines. *Comput. Surv.* 41, 3, Article 17 (July 2009), 38 pages. DOI: http://dx.doi.org/10.1145/1541880.1541884

[6] Joseph Chee Chang, Aniket Kittur, and Nathan Hahn. 2016. Alloy: Clustering with crowds and computation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16)*. ACM, New York, NY, 3180–3191. DOI: http://dx.doi.org/10.1145/2858036.2858411

[7] Shuo Chang, Peng Dai, Lichan Hong, Cheng Sheng, Tianjiao Zhang, and Ed H. Chi. 2016. AppGrouper: Knowledge-based interactive clustering tool for app search results. In *Proceedings of the 21st International Conference on Intelligent User Interfaces (IUI'16)*. ACM, New York, NY, 348–358. DOI: http://dx.doi.org/10.1145/2856767.2856783

[8] Hao Chen and Susan Dumais. 2000. Bringing order to the web: Automatically categorizing search results. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'00)*. ACM, New York, NY, 145–152. DOI: http://dx.doi.org/10.1145/332040.332418

[9] Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. 2013. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 1999–2008. DOI: http://dx.doi.org/10.1145/2470654.2466265

[10] Jaegul Choo, Changhyun Lee, Chandan K. Reddy, and Haesun Park. 2013. UTOPIAN: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE Trans. Vis. Comput. Graph.* 19, 12 (Dec. 2013), 1992–2001. DOI: http://dx.doi.org/10.1109/TVCG.2013.212

[11] Justin Cranshaw, Raz Schwartz, Jason I Hong, and Norman Sadeh. 2012. The livehoods project: Utilizing social media to understand the dynamics of a city. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM'12)*. AAAI, Dublin, Ireland, 58–68.

[12] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. 1992. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92)*. ACM, New York, NY, 318–329. DOI: http://dx.doi.org/10.1145/133160.133214

[13] David L. Davies and Donald W. Bouldin. 1979. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.* 1, 2 (Feb. 1979), 224–227. DOI: http://dx.doi.org/10.1109/TPAMI.1979.4766909

[14] Anca Dumitrache, Lora Aroyo, and Chris Welty. 2015. Achieving expert-level annotation quality with crowdtruth. In *Proceedings of Workshop on Biomedical Data Mining, Modeling, and Semantic Integration at the 16th International Semantic Web Conference (ISWC'15)*. Springer-Verlag, Bethlehem, PA.

[15] Vladimir Estivill-Castro. 2002. Why so many clustering algorithms: A position paper. *ACM Spec. Interest Group Knowl. Discovery Data Min. Explor. Newsl.* 4, 1 (June 2002), 65–75. DOI: http://dx.doi.org/10.1145/568574.568575

[16] Paolo Ferragina and Antonio Gulli. 2005. A personalized search engine based on web-snippet hierarchical clustering. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web (WWW'05)*. ACM, New York, NY, 801–810. DOI: http://dx.doi.org/10.1145/1062745.1062760

[17] Ryan Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. 2011. Crowdclustering. In *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11)*. Curran Associates Inc., 558–566. http://dl.acm.org/citation.cfm?id=2986459.2986522

[18] Marti A. Hearst and Jan O. Pedersen. 1996. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*. ACM, New York, NY, 76–84. DOI: http://dx.doi.org/10.1145/243199.243216

[19] Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2014. Interactive topic modeling. *Mach. Learn.* 95, 3 (2014), 423–469.

[20] Tak Yeon Lee, Alison Smith, Kevin Seppi, Niklas Elmqvist, Jordan Boyd-Graber, and Leah Findlater. 2017. The human touch: How non-expert users perceive, interpret, and fix topic models. *Int. J. Hum.-Comput. Stud.* 105 (2017), 28–42.

[21] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 2009. TurKit: Tools for iterative tasks on mechanical turk. In *Proceedings of the ACM Special Interest Group on Knowledge Discovery and Data Mining Workshop on Human Computation (HCOMP'09)*. ACM, New York, NY, 29–30. DOI: http://dx.doi.org/10.1145/1600150.1600159

[22] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to Information Retrieval*, Vol. 1. Cambridge University Press, Cambridge.

[23] Gary Marchionini. 2006. Exploratory search: From finding to understanding. *Commun. ACM* 49, 4 (April 2006), 41–46. DOI: http://dx.doi.org/10.1145/1121949.1121979

[24] Stanislaw Osinski and Dawid Weiss. 2005. A concept-driven algorithm for clustering search results. *IEEE Intell. Syst.* 20, 3 (May 2005), 48–54. DOI: http://dx.doi.org/10.1109/MIS.2005.38

[25] Dražen Prelec. 2004. A Bayesian truth serum for subjective data. *Science* 306, 5695 (2004), 462–466.

[26] Peter Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20, 1 (Nov. 1987), 53–65. DOI: http://dx.doi.org/10.1016/0377-0427(87)90125-7

[27] Ugo Scaiella, Paolo Ferragina, Andrea Marino, and Massimiliano Ciaramita. 2012. Topical clustering of search results. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*. New York, NY, 223–232.

[28] Alison Smith, Tak Yeon Lee, Forough Poursabzi-Sangdeh, Jordan Boyd-Graber, Niklas Elmqvist, and Leah Findlater. 2017. Evaluating visual representations for topic understanding and their effects on manually generated topic labels. *Trans. Assoc. Comput. Linguist.* 5 (2017), 1–16.

[29] Jerzy Stefanowski and Dawid Weiss. 2003. Carrot2 and language properties in web search results clustering. In *Proceedings of the 1st International Atlantic Web Intelligence Conference on Advances in Web Intelligence (AWIC'03)*. Springer-Verlag, Berlin, 240–249.

[30] Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. 2011. Adaptively learning the crowd kernel. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11)*. Omnipress, 673–680.

[31] Vasilis Verroios and Michael S. Bernstein. 2014. Context trees: Crowdsourcing global understanding from local views. In *Proceedings of the 2nd AAAI Conference on Human Computation and Crowdsourcing (HCOMP'14)*. AAAI.

[32] Jinfeng Yi, Rong Jin, Anil K. Jain, Shaili Jain, and Tianbao Yang. 2012. Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In *Proceedings of the 25th International Conference on Neural Information Processing Systems—Volume 2 (NIPS'12)*. Curran Associates Inc., 1772–1780.

[33] Oren Zamir and Oren Etzioni. 1999. Grouper: A dynamic clustering interface to web search results. In *Proceedings of the Eighth International Conference on World Wide Web (WWW'99)*. Elsevier North-Holland, Inc., 1361–1374. http://dl.acm.org/citation.cfm?id=313234.313054

[34] Haijun Zhai, Todd Lingren, Louise Deleger, Qi Li, Megan Kaiser, Laura Stoutenborough, and Imre Solti. 2013. Web 2.0-based crowdsourcing for high-quality gold standard development in clinical natural language processing. *Journal of Medical Internet Research* 15, 4 (2013).

Received December 2016; revised August 2017; accepted October 2017