# Query Driven Data Labeling with Experts: Why Pay Twice?

Eyal Dushkin[1] Shay Gershtein[1] Tova Milo[1] Slava Novgorodov[2]
[1] *Tel Aviv University* [2] *eBay Research*
[1]{eyaldush, shaygl, milo}@post.tau.ac.il [2]snovgorodov@ebay.com

## ABSTRACT

Data has become a major priority for customer facing businesses of all sizes. Companies put a lot of effort and money into storing, cleaning, organizing, enriching and processing data to better meet user needs. Usually in large scale systems such as big e-commerce sites these tasks involve machine learning methods, relying on training data annotated by domain experts. Since domain experts are an expensive resource in terms of monetary costs and latency, it is desired to design algorithms that minimize the interaction with them.

In this paper we address the problem of minimizing the number of annotation tasks with respect to a set of queries. We present a dedicated algorithm based on efficient labeling, that dictates the strategy for constructing a minimal set of classifiers sufficing to answer all queries. Our approach not only reduces monetary costs and latency, but also avoids data redundancy and saves storage space. We first consider a typical scenario of two expressions per query, and further discuss the challenges of extending our approach to multiple expressions. We examine two common models: *batch* and *stream* configurations, and devise *offline* and *online* algorithms, respectively. We analyze the number of annotations, and demonstrate the efficiency and effectiveness of our algorithm on a real-world dataset.

## 1 INTRODUCTION

Data has become a major priority for customer facing businesses of all sizes. Companies put a lot of effort and money into storing, cleaning, organizing, enriching and processing data to better meet user needs. For example, news articles published on news websites, are often annotated, e.g., tagged with "World Cup 2018" or "Elections in the United States", enabling readers to easily consume relevant content, hereby improving personalization and accessibility. E-commerce websites invest in generating a reliable catalog of products combining human and machine intelligence [4, 10, 17]. This allows potential customers to find the best matching product either by navigating through faceted categories, or by executing search queries. Since catalogs are often huge and cannot be maintained solely by human experts, automatic solutions based on machine learning (ML) are also employed. Combining both experts and ML is a widely-used approach also in fraud detection applications [9], text categorization [15] and other classification tasks [19, 20]. One of the most common usages of ML with *human in the loop* is harnessing domain experts to generate adequate labeled data as a baseline for supervised learning. Thus, generating sufficient annotations, while minimizing domain experts' effort, is highly desired. This trade-off between high quality and low cost is the holy grail of training data preparation. Much research has been devoted in the literature to minimizing the interaction with regular crowds [8, 12, 20]. However, these techniques are best suited for mundane classification tasks, whereas for questions that require particular expertise, as shown in [6], gathering multiple answers from various crowd workers does not always produce a desired accuracy level. In light of this, we present a novel query driven approach with *human experts* that avoids redundant labeling tasks and finds the minimal set of necessary tasks. We note that our approach may be combined in a regular crowdsourced schema with probabilistic settings. To illustrate our approach, consider the following example.

*Example 1.1.* A database *Products* contains a relation *Shirts* with attributes *product_id*, *product_title*, *product_description*, *product_image*, *product_price*, *color* and *material*. This relation contains a list of shirts sold on an e-commerce website. While the product title and description are provided by the seller, its color and material are usually missing and should be automatically extracted from the title, description or image using ML classifiers. Assume a customer performs a keyword-based search for orange cotton shirts, which translates into the following SQL query via NLP-based methods[1]:

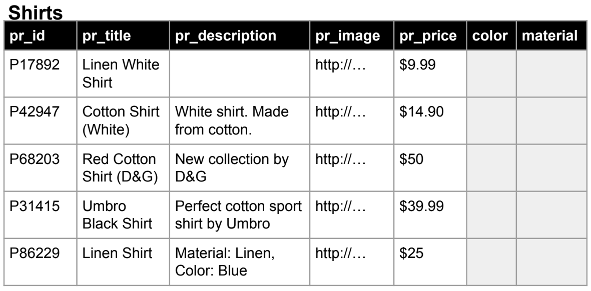
```
SELECT * FROM Shirts
WHERE `color` = 'Orange'
AND `material` = 'Cotton';
```

Providing an answer to such queries, should rely on correctly enriched color and material attributes values for every product. Toward this end, one may train various classifiers with respect to suitable labeled data. Classifiers shall be incorporated to answer "Is this product's color is X and material is Y?". Given that any classifier needs $N$ labeled examples for training, in order to answer the query above one can either train *two* classifiers: one for detecting orange shirts and one for detecting cotton shirts, and separately apply them for each product. This requires $2N$ labeled examples. Alternatively, one could train a more specific classifier that detects orange cotton shirts, using only $N$ labeled examples. Clearly, for this specific case, it is better to choose the latter. However, given a general list of queries with different values of colors or materials, minimizing the number of classifiers may be difficult.

In this paper we propose an algorithm that minimizes the number of classifiers (labeling tasks) that are sufficient to answer all queries. We focus on the case of having at most two expressions per query, which is the most common case in e-commerce search [3]. The extension of our approach to multi-criteria queries is discussed in the future work section. We present a dedicated algorithm based on efficient labeling, that dictates the strategy for constructing a minimal set of classifiers sufficing to answer all queries. Our approach not only reduces monetary costs and latency, but also avoids data redundancy and saves storage space entailed in the enriched attribute values. The contributions of this paper can be summarized as follows:

- We formulate the problem of experts labeling effort minimization with respect to a database and a list of queries.

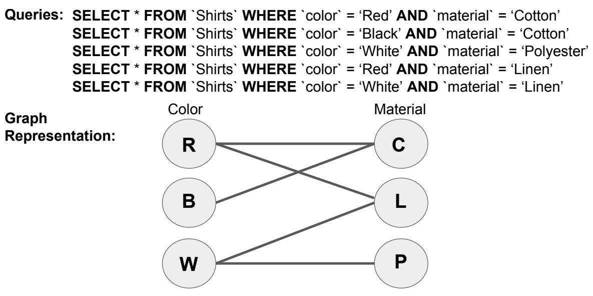
[1]This methods involve NER-based solutions and assumed to be given, hence it is out of the scope of this paper.



Shirts

| pr_id | pr_title | pr_description | pr_image | pr_price | color | material |
|---|---|---|---|---|---|---|
| P17892 | Linen White Shirt | | http://... | $9.99 | | |
| P42947 | Cotton Shirt (White) | White shirt. Made from cotton. | http://... | $14.90 | | |
| P68203 | Red Cotton Shirt (D&G) | New collection by D&G | http://... | $50 | | |
| P31415 | Umbro Black Shirt | Perfect cotton sport shirt by Umbro | http://... | $39.99 | | |
| P86229 | Linen Shirt | Material: Linen, Color: Blue | http://... | $25 | | |

**Figure 1: 'Shirts' relation example.**



Queries: **SELECT** * **FROM** \`Shirts\` **WHERE** \`color\` = 'Red' **AND** \`material\` = 'Cotton'
**SELECT** * **FROM** \`Shirts\` **WHERE** \`color\` = 'Black' **AND** \`material\` = 'Cotton'
**SELECT** * **FROM** \`Shirts\` **WHERE** \`color\` = 'White' **AND** \`material\` = 'Polyester'
**SELECT** * **FROM** \`Shirts\` **WHERE** \`color\` = 'Red' **AND** \`material\` = 'Linen'
**SELECT** * **FROM** \`Shirts\` **WHERE** \`color\` = 'White' **AND** \`material\` = 'Linen'

**Figure 2: Queries and their graph representation.**

- We propose an algorithm that solves the special (yet highly common) case of having at most two expressions per query and provide theoretical analysis of this algorithm.
- We extend the solution to a streaming scenario, where queries are processed piece-by-piece in a serial fashion, and provide approximation guarantees for our approach.
- We conduct an experimental evaluation with a real-world setting, demonstrating the efficiency and effectiveness of our approach, with respect to the number of annotation tasks and the additional storage required.

The paper is organized as follows. The next section defines the model and the problem statement and describes the technical details of the solution (Section 2). We then present our experimental evaluation (Section 3). Finally, we discuss related and future work (Sections 4-5).

## 2 OUR APPROACH

We start by explaining the data model composed of queries and attributes, followed by the description of the algorithm and its online fashion variant for the streaming scenario.

### 2.1 Preliminaries and Model

**Data:** Our model consists of a database $\mathcal{D}$ with relation $\mathcal{R}$, a set of attributes $F$ occupied with values for all $t \in R$ and a set $MF$ of attributes with missing values for some (or all) $t \in R$. Each of the attributes has its domain, the set of all possible values per attribute, i.e., $dom(A_i) = \{v_i^1, v_i^2, ..., \}$.

**Binary Classifier:** Missing attribute values can be discovered with full certainty by constructing a proper *binary* classifier based on labeled data generated by domain experts[2]. Formally, a binary classifier maps every tuple $t \in R$ with its predefined attributes values to $\{0, 1\}$, and can be used for filling holes in missing attribute values. For example, followed our running example, the predefined attributes $F$ are all the *product_∗* attributes (title, image, price, etc.) while two missing attributes are *color* and *material*. In order to reveal missing values for colors, one can learn binary classifiers for various colors that indicate whether a tuple $t$ has the objective color, e.g., $C_{red}(t) = 1$ if $t$ is red. Figure 1 depicts a small sample of such relation that contains products from "Shirts" category. Note that since data is provided by various sellers, the information is concealed within different patterns, usually semi-structured or free-text fields. In some cases the relevant missing values exist only in the title, in other in the description, or in a combination of both. Hence, extracting the desired attributes is a difficult task, which needs well-trained classifiers. We assume that the construction of every classifier is the same. We discuss how to relax this assumption in Section 5, which is a part of our ongoing work.

[2]A common method of *multi-label classification* amounts to independently training one binary classifier for each label [16]. There are other solutions, e.g., reduction to the *multi-class classification* problem or adaption methods, which are out of the scope of this work.

**Queries:** In addition to the information stored in $\mathcal{D}$, there is a set of queries $Q$ that is used to extract the relevant tuples from $\mathcal{D}$. The queries assumed to be only over missing attribute values[3]. Formally, the query is of the following form:

```
SELECT * FROM R WHERE `A_1` = 'v_1_j1' AND
`A_2` = 'v_2_j2' AND ... AND `A_n` = 'v_n_jn';
```

We denote the queries that specify all the attributes from $MA$ as *full queries*, while the queries that specify only subset of $MA$ as *partial queries*.

**Query Expressiveness:** We assume that queries may specify up to two equality expressions over a set of missing attributes values $MF$. While this sounds limiting, having one or two tokens is the most common case both in e-commerce [3] and general purpose search engines [7]. Extending our solution to multi-criteria queries is a part of the open problems and ongoing effort discussed in Section 5.

**Graph Model:** We represent the set of queries in an undirected graph $G = (V, E)$, where the vertices $V$ are the values that appear in the queries (unique per attribute) and the edges $E$ correspond to the queries such that if a query clause consists of two equality expressions with values $v_1^j$ and $v_2^l$ of attributes $A_1$ and $A_2$, resp., we construct an edge between their corresponding nodes in the graph. To support partial queries with one equality expression, we use self edges on the corresponding node. Figure 2 illustrates the set of unique queries and their corresponding graph representation. The graph consists of 6 nodes, 3 unique value per each of the attributes and 5 edges (the number of queries). Note that the graph is not necessarily connected. For example, removing in the mentioned example the "white linen shirt" query, splits the graph into two separate connected components. In addition, the provided example has no self-loops since all queries have two expressions. Adding a query such as:

```
SELECT * FROM `Shirts` WHERE `color` = 'Black';
```

generates a self-loop in the graph ("B" node).

### 2.2 The Algorithm

We start by presenting the *offline* algorithm assuming the set of queries is given as a whole. We extend our solution to support streaming fashion in the next subsection. Considering the general case, the algorithm should determine the set of suitable binary classifiers: either a *query-oriented* classifier corresponding to specific query (denoted by "label the edges") or a *predicate-oriented* classifier corresponding to single attribute value (denoted by "label the nodes"). The goal is to minimize the number of classifiers that are sufficient to answer all queries. Note that in order to answer all queries, our algorithm must determine which edges and nodes to assign a classifier.

Assuming a relation $\mathcal{R}$ and a set of queries $Q$. The algorithm first constructs a graph representation of the queries, as described

[3]Clauses that involve $F$ attributes can be filtered without any classification.

in previous section. For every connected component $C$ in the graph, the algorithm determines whether labeling edges or nodes (in accordance with query-oriented and predicate-oriented classifiers, resp.) by calculating $|E_C|$ and $|V_C|$. If $|E_C| \geq |V_C|$, the algorithm labels nodes[4] and edges otherwise. Since we focus on connected components, the number of edges $|E_C|$ is at least $|V_C| - 1$. Therefore, the algorithm labels edges if and only if $|E_C| = |V_C| - 1$, i.e., if the connected component is a tree.

Lemma 2.1. *Given a relation $\mathcal{R}$ and a set of queries $Q$ with a corresponding graph $G = (V, E)$, the algorithm minimizes the number of classifiers that are sufficient to answer all queries.*

Proof. Since connected components are independent, it suffices to prove correctness with respect to a single connected component $C$. Assume that there is a better solution with a set of classifiers $T$, $|T| < min(|E_C|, |V_C|)$. If $C$ is a tree with $|E_C| = |V_C| - 1 < |V_C|$ then there is an edge $(v_{q_i}, v_{q_j})$ with no query-oriented classifier from $T$ and at least one of $v_{q_i}$ or $v_{q_j}$ does not have a predicate-oriented classifier in $T$. Thus, the query corresponds to this edge is insoluble. On the other hand, let $|V_C| \leq |E_C|$, then some $v_{q_i}$ does not have a predicate-oriented classifier in $T$, thus all $(v_{q_i}, u) \in E$ must have query-oriented classifiers, $d = |(v_{q_i}, u) \in E|$. Since a query-oriented classifier matches a single edge, the number of edges is reduced by $d$ while the number of nodes is reduced by at most $d$, since otherwise $v_{q_i}$ and its neighbours form a tree. Thus, applying these query-oriented classifiers induces a sub-graph $C' \subset C$ with at least $|V| - d$ nodes and $|E| - d$ edges. Since $min(|V| - d, |E| - d) > |T| - d$ it follows, by induction, that the set of classifiers is insufficient to answer all queries. □

Corollary 2.2. *Let $q$ be a partial query, $v_q$ its corresponding vertex and $C_{v_q}$ the connected component containing $v_q$, then the set of classifiers derived by labeling all nodes in $C_{v_q}$ is a minimal set that suffices to answer all queries projected to $C_{v_q}$.*

Proof. Since partial query with one attribute corresponds to a self-loop in a graph, a cycle is created. Therefore, the connected component is not a tree and its nodes should be labeled. □

### 2.3 Online Algorithm

We now assume that the queries arrive in a streaming manner, i.e., piece-by-piece in a serial fashion, and the algorithm makes a decision based on limited information.

The practical motivation for this setting is the common scenario where the already existing classifiers are not sufficient to provide accurate answers to users' query, thus a deficient result is retrieved based on a simple full-text query against a text index. to improve for future times where such a query will be issued, suitable labeling tasks are generated.

Here again we assume a relation $\mathcal{R}$ and a set of queries $Q$ arriving in a streaming manner, one at a time, and our algorithm makes labeling decisions that may later turn out to be sub-optimal. Our algorithm initializes an empty graph and for every query $q$ being processed, it updates (or creates) the connect component $C_q$ with the corresponding edge and nodes. Similar to the offline version, it labels the new edge if $C_q$ is a tree, and the nodes o/w.

Lemma 2.3. *Given a query $q$ being processed from a stream of queries, $v_q$ its corresponding vertex and $C_{v_q}$ the connected component containing $v_q$. If the algorithm decides to label the node $v_q$, the optimal strategy henceforth w.r.t to $C_{v_q}$ is labeling nodes.*

[4]This decision has no effect in the offline case, but helps in the streaming scenario.

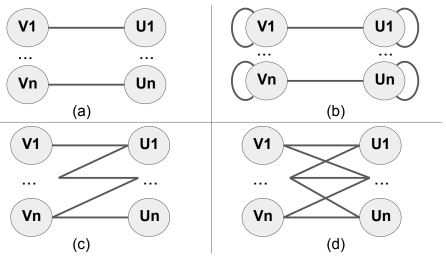


**Figure 3: Worst case analysis of "label the edges" strategy.**

Proof. Given a query $q$ with vertex $v_q$ within connected component $C$. Assume that $|E_C| \geq |V_C|$ for the first time immediately after updating the graph, and the algorithm decides to label $v_q$. Now, since every processed query $q'$ with $v'_q$ in $C$ contributes at most one vertex and at least one edge to $C$, it follows that $|E_C| - |V_C|$ can only increase and $|E_C| \geq |V_C|$ holds. □

For every connected component $C$, we refer to the point when $C$ abandoned its tree structure, as the *swapping point*, since our strategy swapped from label edges to label nodes.

**Relative Bounds Analysis:** First, as proven in Lemma 2.3, once we have swapped strategy from labeling edges to labeling nodes, the optimal strategy is labeling nodes henceforth. Thus, our error derives in between those states, if exist. Till the swapping point $|V_C| - 1$ edge classifiers were generated and in the worst case, each one of them turns out to be redundant. On the other hand, since every solution must contain at least $|V_C| - 1$ classifiers, our approximation ratio is at most 2.

We now illustrate two unfortunate cases. In Figure 3a there are $|E|$ connected components, each consists of two nodes and one edge. Figure 3b makes the original strategy sub-optimal, where each connected component has one redundant edge classifier. In this case, the ratio is $(n + 2n)/(2n) = 3/2$. Figures 3c and 3d present the worst case scenario where the graph is a tree till the swapping point. Therefore, the approximation ratio in the case is $(|V_C| - 1 + |V_C|)/|V_C| = 2 - 1/|V_C|$, demonstrating we can make the relative error as close as we like to 2.

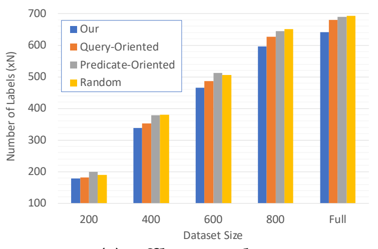
## 3 EXPERIMENTAL EVALUATION

### 3.1 Experimental setup

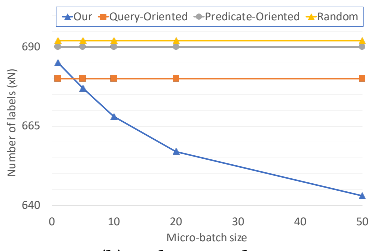
**Competitors:** To evaluate our approach we implemented our algorithm and its three natural competitors. Given a query graph:

- **Predicate-Oriented** - Regardless of any relationships among the queries, the algorithm always labels nodes.
- **Query-Oriented** - Regardless of any relationships among the queries, the algorithm always labels edges.
- **Random** - The algorithm randomly decides whether to label nodes or edges, till it has sufficient information to answer all queries.

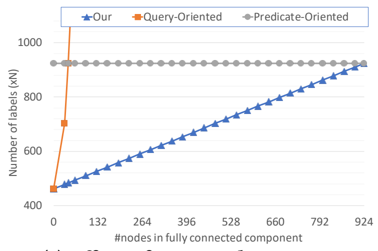
**Dataset:** The dataset contains a real-world public dataset taken from BestBuy with around 1000 search queries over the electronics domain [1]. Each query is written in a structured format, e.g., the query "LG TV" is represented as {"*Brand*" : "*LG*", "*Category*" : "*TV*"}), which allows a straightforward execution of our experiment. The dataset contains 7 different attributes (Category, Model, Brand, ScreenSize, Price, Storage, RAM) with 97% of the queries specify up to 3 attributes: Category, Model, Brand. Out of this subset, around 95% of the queries are of length one or two (66.5% and 29.2% respectively), which corresponds to our assumption on the number of attributes in the e-commerce search queries being small. Interestingly, the longest query in the dataset contains only four attributes (Category, Brand, Model, ScreenSize).



(a) Offline results.



(b) Online results.



(c) Effect of queries dispersion.

**Figure 4: Experimental results.**

## 3.2 Evaluation results

We evaluate our algorithm and its competitors with respect to two configurations: offline and online. Considering the offline configuration, we first execute the algorithms against two types of sets: (i) the full set of relevant queries (that contain one or two expressions), and (ii) randomly selected subsets of varying sizes (200, 400, 600, 800). Considering the online configuration, we simulate random arrivals of successive queries from a stream of queries, and measure the number of classifiers generated by every competitor. Furthermore, we evaluate a *micro-batching* approach, where each iteration is executed with respect to a window of queries with varying sizes (5, 10, 20, 50).

**Offline Evaluation:** Results are depicted in Figure 4a. As expected, our algorithm outperforms the competitors and reduces the number of classifiers as the dataset size increases. Observe that the query-oriented algorithm is constantly superior to the predicate-oriented algorithm. In general, the predicate-oriented algorithm performs better when there is a small set of attributes which covers multiple queries. On the other hand, for queries that do not share many attributes the query-oriented algorithm performs better. Since our algorithm decides the best strategy per connected component, it outperforms both competitors. While reducing "only" 5%, it may save hundreds of thousands of dollars annually in workers cost [2] and terabytes of storage (entailed in the labeled data and the additional attributes in the relation).

**Streaming Scenario Evaluation:** Results are depicted in Figure 4b. We can see that even in small micro-batches of size 5, our algorithm outperforms all competitors, and significantly surpasses the competitors for larger windows. Observe that only in the extreme case of completely online scenario it is sub-optimal.

**Queries Dispersion Evaluation:** In this set of experiment, we examine the effect of attributes overlapping between queries on the number of labeling tasks. To examine it, we fix the size of the graph with $n$ nodes and vary its density by increasing the size of connected components, from 2 to $n$, which corresponds to the queries dispersion. Figure 4c depicts the results for our algorithm and its two deterministic competitors: predicate-oriented and query-oriented algorithms. First, we can see that our algorithm is consistently superior with comparable competitors in the two extreme cases. In between those cases, our algorithm beats the competitors by large margins and when $|E| = |V|$, it conducts approximately two times less labeling tasks.

## 4 RELATED WORK

In recent years, crowd workers and human experts are widely employed with various tasks to amend the performance of supervised ML models, e.g., contributing to feature selection [13], learning of semantic attributes [18] and others. Several systems propose hybrid mechanisms [5, 14, 17] that interweave humans and machines. One family of algorithms focus on reducing the error of crowd annotators [4, 8], e.g., combining crowd and machines for multi-predicate classification tasks [11]. In contrast to the probabilistic models employed in these algorithms, they are less suitable for the experts-based setting. To the best of our knowledge, this is the first attempt that aim at minimizing the effort of experts in annotation tasks and the required storage entailed by obtaining classifiers sufficing to answer a set of queries.

## 5 FUTURE WORK

The most intriguing direction is extending our approach to support longer queries, which is our main ongoing process. Simply extending our graph representation to queries with multiple equality expressions results in a hypergraph representation, thus other approaches might be more suitable. In addition, supporting general weights for various classification tasks, is interesting. Finally, extending our evaluation to additional datasets with different types of queries is a part of our future work.

**Acknowledgements** This work has been partially funded by the Israel Innovation Authority, the Israel Science Foundation, Len Blavatnik and the Blavatnik Family foundation.

## REFERENCES

[1] 2017. BestBuy dataset. https://dataturks.com/projects/Mohan/Best%20Buy%20E-commerce%20NER%20dataset. (2017).
[2] 2017. How to Organize Data Labeling. https://altexsoft.com/blog/datascience/how-to-organize-data-labeling-for-machine-learning-approaches-and-tools/. (2017).
[3] 2017. Retail Site Search Queries Are Getting Shorter. https://www.marketingcharts.com/industries/retail-and-e-commerce-76709. (2017).
[4] Ron Bekkerman and Matan Gavish. 2011. High-precision phrase-based document classification on a modern scale. In *KDD 2011*.
[5] Justin Cheng and Michael S Bernstein. 2015. Flock: Hybrid crowd-machine learning classifiers. In *CSCW*. ACM, 600–611.
[6] Benoît Groz, Ezra Levin, Isaac Meilijson, and Tova Milo. 2016. Filtering with the Crowd: CrowdScreen revisited. In *ICDT 2016*.
[7] Ido Guy. 2016. Searching by Talking: Analysis of Voice Queries on Mobile Web Search. In *SIGIR 2016*. 35–44.
[8] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. 2013. Adaptive task assignment for crowdsourced classification. In *ICML*. 534–542.
[9] Ariel Jarovsky, Tova Milo, Slava Novgorodov, and Wang-Chiew Tan. 2018. GOLDRUSH: Rule Sharing System for Fraud Detection. *PVLDB* 11, 12 (2018).
[10] Ece Kamar, Severin Hacker, and Eric Horvitz. 2012. Combining human and machine intelligence in large-scale crowdsourcing. In *AAMS 2012*. 467–474.
[11] Evgeny Krivosheev, Fabio Casati, Marcos Baez, and Boualem Benatallah. 2018. Combining Crowd and Machines for Multi-predicate Item Screening. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 97.
[12] Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J Franklin. 2016. Crowdsourced data management: A survey. *TKDE* 28, 9 (2016), 2296–2319.
[13] Besmira Nushi, Adish Singla, Andreas Krause, and Donald Kossmann. 2016. Learning and feature selection under budget constraints in crowdsourcing. In *HCOMP 2016*.
[14] Akash Das Sarma, Ayush Jain, Arnab Nandi, Aditya Parameswaran, and Jennifer Widom. 2015. Surpassing humans and computers with JELLYBEAN: Crowd-vision-hybrid counting algorithms. In *HCOMP 2015*.
[15] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)* 34, 1 (2002), 1–47.
[16] Mohammad S Sorower. 2010. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis* 18 (2010).
[17] Chong Sun, Narasimhan Rampalli, Frank Yang, and AnHai Doan. 2014. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *Proceedings of the VLDB Endowment* 7, 13 (2014), 1529–1540.
[18] Tian Tian, Ning Chen, and Jun Zhu. 2017. Learning Attributes from the Crowdsourced Relative Labels.. In *AAAI*, Vol. 1. 2.
[19] Xiaohang Zhang, Guoliang Li, and Jianhua Feng. 2016. Crowdsourced top-k algorithms: An experimental evaluation. *PVLDB* 9, 8 (2016), 612–623.
[20] Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. 2015. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD 2015*. 1031–1046.