Human-in-the-loop Outlier Detection

Human-in-the-loop Outlier Detection

Chengliang Chai
Tsinghua University
chaicl15@mails.tsinghua.edu.cn

Lei Cao
CSAIL, MIT
lcao@csail.mit.edu

Guoliang Li
Tsinghua University
liguoliang@tsinghua.edu.cn

Jian Li
Tsinghua University
lijian83@@tsinghua.edu.cn

Yuyu Luo
Tsinghua University
luoyy18@mails.tsinghua.edu.cn

Samuel Madden
CSAIL, MIT
madden@csail.mit.edu

ABSTRACT

Outlier detection is critical to a large number of applications from finance fraud detection to health care. Although numerous approaches have been proposed to automatically detect outliers, such outliers detected based on statistical rarity do not necessarily correspond to the true outliers to the interest of applications. In this work, we propose a human-in-the-loop outlier detection approach HOD that effectively leverages human intelligence to discover the true outliers. There are two main challenges in HOD. The first is to design human-friendly questions such that humans can easily understand the questions even if humans know nothing about the outlier detection techniques. The second is to minimize the number of questions. To address the first challenge, we design a clustering-based method to effectively discover a small number of objects that are unlikely to be outliers (aka, inliers) and yet effectively represent the typical characteristics of the given dataset. HOD then leverages this set of inliers (called context inliers) to help humans understand the context in which the outliers occur. This ensures humans are able to easily identify the true outliers from the outlier candidates produced by the machine-based outlier detection techniques. To address the second challenge, we propose a bipartite graph-based question selection strategy that is theoretically proven to be able to minimize the number of questions needed to cover all outlier candidates. Our experimental results on real data sets show that HOD significantly outperforms the state-of-the-art methods on both human efforts and the quality of the discovered outliers.

SIGMOD'20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3389772

CCS CONCEPTS

• **Information systems** → *Crowdsourcing*; • **Computing methodologies** → **Anomaly detection**.

KEYWORDS

outlier detection; human-in-the-loop; question selection

ACM Reference Format:
Chengliang Chai, Lei Cao, Guoliang Li, Jian Li, Yuyu Luo, and Samuel Madden. 2020. Human-in-the-loop Outlier Detection. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3318464.3389772

1 INTRODUCTION

Given a dataset, outliers are the objects that are significantly different from the others (aka, inliers) in the dataset. Outlier detection is of paramount importance in a wide variety of applications such as fraud detection for credit cards, insurance, health care, fault detection in safety critical systems, military surveillance for hostile activities [15], and outliers detection and repairing in visualization tasks [33].

Due to the importance of outlier detection, numerous outlier detection methods [2, 3, 5, 23, 41] have been proposed. In general, these methods can be divided into two categories: supervised and unsupervised methods. The supervised methods require labeled outliers to train a binary classification model. However, since outliers typically correspond to rare events, there is no enough number of available labeled outliers that are sufficient to train an accurate model in many real applications. Although active learning techniques can be leveraged to guide humans to label the objects that are more likely to be outliers, they cannot solve the problem of *lacking outliers*. Crowdsourcing-based methods, like CrowdER [44], leverage humans to judiciously select questions in order to reduce the cost. However, they incur huge costs when used to address the outlier detection problem, because they ask humans to label many pairs of objects. As confirmed in our experiments, existing methods [14, 42] do not perform well.

On the other hand, unsupervised techniques do not rely on any labeled data [22], and they identify the objects that

*Guoliang Li is the corresponding author.

	Name (a_1)	Customers (a_2)
o_1	Apple iPhone 7th 16GB white	cu_1, cu_2, cu_3
o_2	Apple iPhone 7th 32GB black	cu_2, cu_3, cu_4, cu_5
o_3	Apple iPhone 8th 32GB	cu_5, cu_6, cu_7
o_4	iPhone 5s	cu_1
o_5	Huawei Mate 10	cu_2, cu_5, cu_8, cu_9
o_6	Huawei Mate 20	$cu_3, cu_8, cu_9, cu_{10}$
o_7	Huawei P30 pro Black 64GB	cu_4, cu_8, cu_{10}
o_8	P20 Pro	cu_8, cu_9
o_9	Sumsung S7 White	$cu_4, cu_{11}, cu_{12}, cu_{13}$
o_{10}	Sumsung Galaxy S7	cu_1, cu_{11}, cu_{13}
o_{11}	Sumsung Note 2	cu_{12}, cu_{14}
o_{12}	iPhone 6th case	cu_{17}, cu_{18}
o_{13}	Huawei Mate Earphone	cu_{10}, cu_{11}
o_{14}	Google Pixel 3	$cu_{10}, cu_{16}, cu_{17}, cu_{18}$
o_{15}	Google Pixel 2	$cu_{11}, cu_{16}, cu_{17}$
o_{16}	Google Pixel 3 XL	$cu_{15}, cu_{17}, cu_{18}$

Table 1: Records in a Product Dataset

have few neighbors as outliers, where an object and its neighbors should be very similar to each other. However, simply relying on these data driven (unsupervised) methods tends to erroneously flag normal objects as outliers (false positive) or miss true outliers to the interest of applications (false negative). Next, we use a toy example to demonstrate this.

Given a dataset with two attributes (product name & customers who viewed the product) as shown in Table 1, most of the objects are cellphones in different brands (o_1–o_{11},o_{14}–o_{16}). Obviously, o_{12} and o_{13} are different from the majority, because they are cellphone case and earphone respectively. Therefore, in this dataset o_{12} and o_{13} are considered as the *true outliers*, which probably were erroneously inserted into this cellphone database by the product manager. However, applying the above unsupervised techniques on this dataset, we obtain a set of outliers $\{o_4, o_7, o_8, o_{11}, o_{12}\}$ as shown in Fig. 1(b), which do not agree with the true outliers. o_4, o_7, o_8, o_{11} tend to be misclassified as outliers, because they are not close enough to any other object in this dataset. For example, o_4 only has one common word "iPhone" with o_1, o_2, o_3 and has a significantly different customer list. Therefore, although it is also an Apple cellphone same to o_1, o_2, o_3, it was detected as an outlier incorrectly. On the other hand, the true outlier o_{13} is missed, because it is close to objects o_5 and o_6, which also contain the word "Huawei Mate".

In this work, we focus on designing an approach that effectively leverages human intelligence in outlier detection to solve the above problems. Unlike the supervised outlier detection methods, we do not target on training a classification model to detect outliers. Instead, we propose to first generate some outlier candidates using unsupervised outlier algorithms, and then use humans to identify the true outliers based on the machine generated results. However, to effectively leverage the human efforts, several challenging problems have to be solved. First, although outliers typically are rare, a large number of outlier candidates will still be produced when detecting outliers from large data sets. It will be very expensive if we rely on the human to evaluate each of these candidates one by one. Therefore, when interacting with the human, questions have to be carefully designed such that each question can cover as many outlier candidates as possible to minimize the human efforts. Second, human often do not have a deep understanding on the applications, nor the unsupervised outlier detection techniques. Therefore, we have to design some effective outlier explanation mechanisms that intuitively show why each outlier candidate has been flagged to be a potential outlier. Otherwise, it will be extremely hard for the human to validate the outlier candidates and thus lead to low quality results.

To address these challenges, we propose a human-in-the-loop outlier detection framework HOD, that accurately identifies the true outliers with minimal human efforts. Essentially, it unifies the merits of unsupervised outlier detection algorithms and human intelligence. Our solution is based on the *inlier observation*. That is, given a set of inliers, the inliers surrounding an outlier candidate constitute the context in which this candidate occurs, so called *context inliers*. These context inliers represent the typical characteristics that a normal object in this area is expected to show. Therefore, humans can verify the status of an outlier candidate based on its similarity to its context inliers. More specifically, an outlier candidate typically corresponds to an inlier if it is similar to one context inlier, while it tends to be an outlier if it is considered to be different from all its context inliers.

HOD fully leverages this insight. First, we design a clustering-based method to discover a set of context inliers that is compact yet sufficient to represent the typical characteristics of the whole data set. HOD first conducts clustering on the objects that the unsupervised outlier detection methods believe are unlikely to be outliers. Then leveraging a small set of inlier examples acquired through human feedback, HOD learns a probability for each pair of objects in a cluster that together indicates how possible this cluster corresponds to an *inlier cluster*. Only one object from each inlier cluster is selected as a context inlier. On the other hand, the objects in the less certain clusters are considered as outlier candidates in addition to those produced by the outlier detection methods. In this way, HOD avoids producing misleading context inliers and also captures the outliers missed by the outlier detection algorithms using small human efforts. Moreover, we propose a context-aware question selection mechanism that minimizes the human efforts in verifying the outlier candidates. Instead of directly asking the human if a given object is an outlier or not, we use a multi-object questioning mechanism. This not only allows the human to easily verify

the outlier candidates by contrasting them against their corresponding context inliers, but also reveals the opportunities to verify the status of multiple objects using one single question. Based on the questioning mechanism, we encode the outlier candidates, the context inliers, and their relationships into a bipartite graph. We then model our question selection problem as an edge covering problem, and efficiently solve this problem with a small approximation ratio to the optimal solution.

In conclusion, the key contributions of this work include:

(1) We propose a human-in-the-loop outlier detection approach (HOD) that is able to accurately discover outliers using as few human efforts as possible (see Section 2).

(2) We propose a question selection method that minimizes the number of questions with the theoretical guarantee, thus effectively reducing the human efforts (see Sections 3, 4).

(3) We design a clustering-based method that discovers a set of representative inliers, effectively serving as the context for the easy evaluation of outlier candidates (see Section 5).

(4) Our experiments conducted on real-world datasets confirm that HOD improves the precision and recall of outlier detection by more than 40% and 20% compared with the unsupervised outlier detection methods, while only using half of the human efforts in comparison to the baseline approaches. Besides, given the same budget, HOD improves the F1-score by more than 30% compared with the active learning-based methods. HOD also outperforms crowdsourcing-based methods on both quality and cost (see Section 6).

2 OUR OVERALL APPROACH

In this section, we first formally define the human-in-the-loop outlier detection problem (Sec. 2.1), and then overview the high level design of our HOD approach (Sec. 2.2).

2.1 Problem Definition

Definition 2.1. Human-in-the-loop Outlier Detection: Given a dataset $O = \{o_1, o_2, ..., o_N\}$, each object has M attributes $A = \{a_1, a_2, ..., a_M\}$. Assume there is a set of outliers $\hat{O} \subset O$ that is unknown apriori. Human-in-the-loop outlier detection aims to discover all outliers in $\hat{O}$ from O with the minimum human efforts.

For example, consider the toy product dataset in Table 1, which has 16 records with two attributes, namely the product name and the IDs of the customers who have viewed the product. In this dataset, most of the objects correspond to different brands of cellphones, such as Apple, Huawei, etc. However at the same time, there are also a few products that are not cellphones hidden in the data. As shown in Table 1, o_{12}, o_{13} are cellphone cases and earphones. These two objects are considered as outliers by the application, because this dataset is supposed to only contain cellphones.

2.2 Overall Approach

Context Inlier Observation. The HOD approach is built on our context inlier observation. Suppose we have a set of inliers known beforehand. If the human believes that an outlier candidate is similar to a known inlier, we can deduce that it is also an inlier with high possibility. Otherwise, if the candidate is different from any inlier, it tends to be an outlier. For example, in our product dataset, assume we already know that o_6 ("Huawei Mate 20") is an inlier. Given two outlier candidates o_7 and o_8, they fall into the same category with o_6, as they are also cellphones made by Huawei. On the other hand, o_{13} will be considered as an outlier, as it is an earphone, different from the known inliers which are all cellphones.

If we can acquire a set of objects that are guaranteed to be inliers, these inliers can be used as references to verify the status of the outlier candidates, so called *context inliers*.

Problems to Solve. To effectively leverage this inlier observation, several problems have to be solved. First, an outlier candidate set has to be produced that covers most of the true outliers. Second, in real applications the inliers typically are not given beforehand. Instead, they have to be discovered from the data. Therefore, an inlier discovery method has to be designed that: (1) is reliable and not producing any outlier erroneously; (2) produces inliers that represent the typical characteristics of the given dataset. Moreover, to minimize the human efforts, the questions for the human to answer have to be carefully designed and selected. Intuitively, to verify the status of a given candidate, we can ask the human to compare it against each context inlier. However, a large number of questions will be produced.

In this work, we successfully solve the above problems by designing three techniques, namely an outlier candidate generation method, a clustering-based context inlier discovery algorithm, and a bipartite graph-based question selection strategy, which are applied step by step as sketched below.

2.2.1 Outlier Candidate Generation. We use an ensemble of different outlier detection algorithms to produce the outlier candidates. Although there are many outlier detection algorithms, none of them fits all datasets with diverse data characteristics. Thus, we invoke several outlier detection algorithms $\mathcal{AG} = \{\mathcal{A}_1, \mathcal{A}_2, ...\mathcal{A}_{|N|}\}$ that generate $|N|$ different sets of outliers. We then compute the union of these sets as our outlier candidate set $\mathcal{U}$ to avoid the missing of true outliers. Naturally, other objects left in O constitute the inlier candidates $\mathcal{I}$. As shown in Fig. 1(b), $\mathcal{I} = \{o_1, o_2, o_3, o_5, o_6, o_9, o_{10}, o_{13}, o_{14}, o_{15}, o_{16}\}$ and $\mathcal{U} = \{o_4, o_7, o_8, o_{11}, o_{12}\}$.

As confirmed in our experiments, HOD is not sensitive to specific detection algorithms and their input parameters used in generating outlier candidates. In other words, using HOD, users do not have to carefully select the outlier detection algorithms and tune their input parameters.

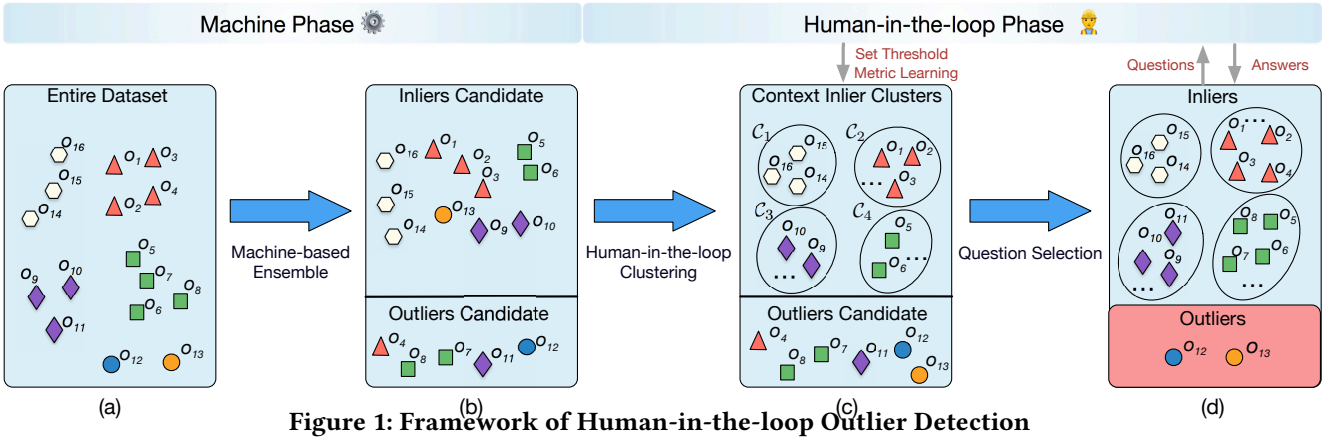

Figure 1: Framework of Human-in-the-loop Outlier Detection

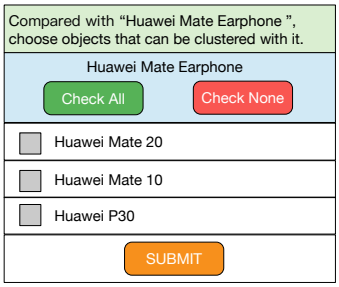

Figure 2: Example Question for Humans

2.2.2 Clustering-based Context Inlier Discovery. The context inliers are discovered from $\mathcal{I}$. To this end, we aim to cluster objects in $\mathcal{I}$ and generate a collection $\mathcal{IC} = \{C_1, C_2, ..., C_{|\mathcal{IC}|}\}$, where each $C_i (C_i \subset \mathcal{I})$ denotes a set of inliers that can be clustered together. Only one inlier from each cluster is selected as a context inlier. In this way, we produce a small set of context inliers to effectively represent the typical characteristics of the whole dataset.

Moreover, to ensure the *reliability* of the discovered context inliers, we leverage the human feedback to learn a function that for each pair of objects, computes a probability to indicate how possible this object pair belongs to the same cluster. The reason is that even if two objects are clustered together by a classical clustering algorithm which typically uses some standard distance functions to determine their similarity, they do not necessarily fall into the same category by the semantics of the application. This is also one of the key reasons that the data driven outlier detection techniques do not work well in capturing true outliers. These probabilities are then used in HOD to cluster objects.

In addition, the clustering algorithm often produces some small clusters which do not necessarily correspond to context inliers. Therefore, HOD moves the objects in these small clusters to $\mathcal{U}$ as outlier candidates. This not only ensures the reliability of the context inliers, but also avoids the missing of the true outliers. For example, now $\mathcal{U} = \{o_4, o_7, o_8, o_{11}, o_{12}, o_{13}\}$, because o_{13} is moved into the outlier candidate set. Thus the clustering algorithm aims at generating high-quality clusters of inliers while not involving outliers.

2.2.3 Question Generation. HOD aims to verify if each object $o \in \mathcal{U}$ is an outlier or not. Intuitively, to achieve this, we can ask the human to compare each outlier candidate against every context inlier. However, since the number of outlier candidates and the context inliers potentially is large, this naive method will cost huge amount of human efforts.

Multi-Object Questions. To solve this problem, we first design a multi-object question interface used to interact with the human, as shown in Fig. 2. It effectively leverages the context inliers observation.

Definition 2.2. Multi-object Question in HOD: Each question q consists of two parts. One corresponds to a single target object and the other contains at most k optional objects or options. The human is asked to mark the options that share the same category/similar property with the target through the checkboxes.

Note that either a context inlier or an outlier candidate can be used as the target. If a question aims to verify if a given candidate is an outlier, this question should use the candidate as the target and the h nearest context inliers of this candidate as the options. On the other hand, in the case that one context inlier corresponds to the nearest context inlier of multiple candidates, a question can be constructed that uses this context inlier as target and the outlier candidates as options. Thus, a multi-object question can verify the status of multiple candidates at once.

Question Selection. Given a set of outlier candidates and the context inliers, a question selection strategy decides on how to select the targets and the options to construct the multi-object questions. An optimal question selection strategy should minimize the costs used in the verification of the outlier candidates, as defined below.

Definition 2.3. Question Selection for Outlier Detection: Given a dataset O and the size k of a question (i.e., a target object and k options), the problem is to produce a minimal number of questions that are sufficient to verify the status of all candidates $\in \mathcal{U}$.

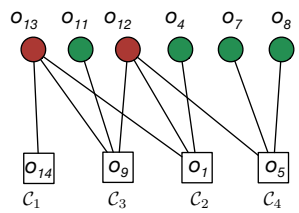

Figure 3: Graph Model

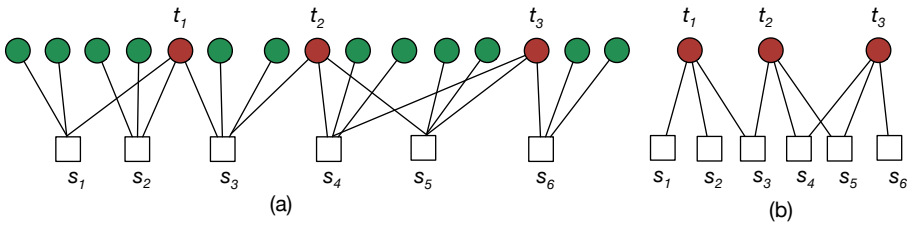

Figure 4: A Case for the Graph Model

To solve this problem, we first encode the outlier candidates and the context inliers into a bipartite graph, where one part of vertices correspond to the outlier candidates in $\mathcal{U}$ and the second part of vertices correspond to the context inliers in $\mathcal{IC}$. An edge indicates that a context inlier is among the h context inliers nearest to the corresponding outlier candidate.

For example, suppose $k = 3$ and $h = 3$. Given the dataset in Table 1, as shown in Fig 1(c), we first split them into outlier candidates ($\mathcal{U} = \{o_4, o_7, o_8, o_{11}, o_{12}, o_{13}\}$) and context inliers collection ($\mathcal{IC} = \{\{o_1, o_2, o_3\}, \{o_5, o_6\}, \{o_9, o_{10}\}, \{o_{14}, o_{15}, o_{16}\}\}$). Then the optimal solution generates 4 questions to identify the status of the objects in $\mathcal{U}$, i.e., $\{\underline{o_9}, o_{13}, o_{11}, o_{12}\}$, $\{\underline{o_1}, o_{13}, o_{12}, o_4\}$, $\{\underline{o_5}, o_{12}, o_7, o_8\}$, $\{\underline{o_{14}}, o_{13}\}$, where o_9, o_1, o_5, o_{14} are the targets.

Intuitively, a question with k options can cover at most k edges. The goal is to cover all edges using the minimal number of questions. In the next section (Sec. 3), we will formally show that this problem is NP-hard. In Sec. 4 we will give an approximate solution that solves this problem with a small approximation ratio. The clustering-based context inliers discovery method will be presented in Sec. 5.

Discussion: the Scope of HOD. Although the focus of HOD is to directly discover all outliers from a given dataset using minimal human efforts, it can also be used as a tool to obtain a set of labels to train a classification model which is used later to classify outliers from the new data produced by the application. It tends to be more effective than the existing active learning methods [14, 42], since using the same number of questions, HOD can find more outliers with higher precision, as confirmed in our experiments (Sec. 6).

3 QUESTION SELECTION PROBLEM

In this section, we analyze the complexity of the question selection problem using a bipartite graph-based model.

Suppose we already know the true outliers in $\mathcal{U}$ and have a perfect context inlier set $\mathcal{IC}$ in advance. The bipartite graph is constructed as follows.

Definition 3.1. Question Selection Bipartite Graph: Given the context inliers and the outlier candidates $\mathcal{U}$, a bipartite graph $\mathcal{G} = (\mathcal{S}, \mathcal{T}, \mathcal{E})$ is constructed, where vertices in $\mathcal{S}$ denote the context inliers and vertices in $\mathcal{T}$ denote the candidate object in $\mathcal{U}$. For a vertex $t \in \mathcal{T}$ whose ground truth is an inlier, there is an edge between t and the vertex $s \in \mathcal{S}$ that represents the nearest context inlier to t. For a vertex $t' \in \mathcal{T}$ whose ground truth is an outlier, there are h edges connecting t' and the vertices in $\mathcal{S}$ corresponding to t''s h nearest context inliers.

For example, as shown in Fig. 3 ($h = 3$), $o_4, o_7, o_8, o_{11}, o_{12}, o_{13}$ correspond to the outlier candidate $\mathcal{U}$. o_{14}, o_1, o_9, o_5 correspond to the context inliers $\mathcal{IC}$, representing inlier clusters C_1, C_2, C_3, C_4 respectively. For ease of understanding, given the ground truth, we color the true inlier (outlier) as Green(Red).

Among the outlier candidates, o_{13} is an outlier (colored in Red) and o_{14}, o_1, o_9 are o_{13}'s top-3 nearest context inliers, there are 3 edges (o_{13}, o_{14}), (o_{13}, o_1) and (o_{13}, o_9). In addition, since o_7 (colored in Green since its ground truth is in fact an inlier) is the nearest to context inlier o_5, there is an edge (o_7, o_5). Next, we define the bipartite graph-based version of our question selection problem (Definition 3.2).

Definition 3.2. Question Selection on Graph: Each question q contains x vertices in the bipartite graph $\mathcal{G}$, where $x \in [1,k]$. An edge $e \in \mathcal{E}$ is covered by question q if two nodes connected by e are included in q. A vertex in $\mathcal{T}$ is considered to be resolved if all edges associated to it are covered. The problem is to construct the minimal number of questions that cover all edges (or resolve all vertices in $\mathcal{T}$) in $\mathcal{E}$.

For example, given the graph in Fig. 3 (constructed based on Table 1), we can generate five questions $\{\underline{o_{13}}, o_{14}, o_9, o_1\}$, $\{\underline{o_{12}}, o_1, o_5, o_9\}$, $\{\underline{o_5}, o_7, o_8\}$, $\{\underline{o_1}, o_4\}$ and $\{\underline{o_9}, o_{11}\}$. However, a better solution is to generate four questions $\{\underline{o_{14}}, o_{13}\}$, $\{\underline{o_9}, o_{13}, o_{11}, o_{12}\}$, $\{\underline{o_1}, o_{13}, o_{12}, o_4\}$ and $\{\underline{o_5}, o_{12}, o_7, o_8\}$, where o_{14}, o_1, o_9, o_5 are the representatives of inlier clusters C_1, C_2, C_3, C_4 respectively. Therefore, all edges are covered. o_4, o_7, o_8, o_{11}, o_{12}, o_{13} are resolved. Based on the above definition, covering all edges in $\mathcal{E}$ equals to resolving all vertices in $\mathcal{T}$. So we use those two terminologies interchangeably. For the ease of presentation, sometimes we also use the notation of the cluster to denote the context inlier sampled from it. For example, $\{\underline{o_{14}}, o_{13}\}$ can be written as $\{\underline{C_1}, o_{13}\}$.

Theorem 3.3. *The question selection problem is NP-hard.*

Proof. We omit the proof because of the space limit and put it into the technical report [7]. □

4 GRAPH-BASED QUESTION SELECTION

As shown above, the question selection problem is NP-hard even if we assume that the ground truth outliers are already

	o_4	o_7	o_8	o_{11}	o_{12}	o_{13}
C_1	0.5	0.4	0.2	0.5	0.4	0.45
C_2	0.78	0.3	0.3	0.45	0.48	0.48
C_3	0.45	0.2	0.2	0.78	0.45	0.48
C_4	0.4	0.75	0.8	0.3	0.45	0.35

Table 2: Examples of Candidate-Cluster Distances

known beforehand, we propose a bipartite graph-based question selection strategy that effectively solves this problem with a small approximate ratio, without relying on such assumption. We first show how to construct the bipartite graph $\mathcal{G}$. Then we introduce our question selection algorithm and show its theoretical property.

4.1 Graph Construction

A bipartite graph $\mathcal{G}$ is constructed based on Def. 3.1 (Sec. 3). The objects in $\mathcal{IC}$ and $\mathcal{U}$ correspond to the nodes in $\mathcal{S}$ and $\mathcal{T}$ of $\mathcal{G}$ respectively. To produce the edges in $\mathcal{E}$, we need to find the h closest context inliers in $\mathcal{IC}$ for each candidate in $\mathcal{U}$. For ease of understanding, we use the cluster which a context inlier represents to denote the corresponding context inlier. For example, C_2 denotes context inlier o_1. Note the context inliers $\mathcal{IC}$ are discovered using our inlier clustering method which will be presented later in Sec. 5.

We define p_{ij} as the probability that o_i and o_j can be categorized into the same cluster. This probability is also used in our inlier clustering to produce $\mathcal{IC}$. So we will discuss how to compute this probability in Sec. 5 together with our clustering method. The distance between an outlier candidate and a cluster is then measured using candidate-cluster likelihood defined below.

Definition 4.1. Candidate-Cluster Likelihood: Given $u \in \mathcal{U}$ and $C \in \mathcal{IC}$, $q(u, C)$ denotes the likelihood that u can be clustered into C. $q(u, C) = q(C, u) = p_{uu^*}$, where $u^* = \arg\max_{u' \in C} p_{uu'}$.

We show how this distance is computed using o_4 and C_2 in Fig. 1(c) as example. Suppose $p_{14} = 0.55$, $p_{24} = 0.6$ and $p_{34} = 0.78$, then $q(o_4, C_2) = 0.78$, because p_{34} corresponds to the largest probability.

For each $u \in \mathcal{U}$, we use $closest(u)$ to denote its closest inlier cluster. $hclosest(u)$ denotes the h closest inlier clusters of u. Table 2 gives the distances between the outlier candidates and inlier clusters in Fig. 1(c). For example, $closest(o_4) = C_2$. $hclosest(o_4) = \{C_1, C_2, C_3\}$ when $h = 3$.

By Definition 3.1, an edge exists between u and its $closest(u)$ if u is an inlier. On the other hand, h edges exist between v and nodes in $hclosest(v)$ if v is an outlier. However, apparently in practice we do not know the ground truth in advance. We solve this problem by presuming the outlier candidates as outliers or inliers based on their distances to the corresponding closest clusters. Although the presumed

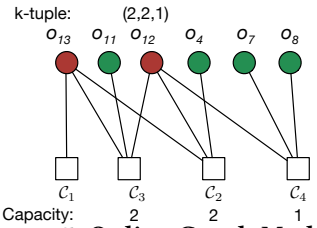

Figure 5: Online Graph Model

ground truth is not precise, it can be used as a good start point to solve the question selection problem.

More specifically, when building the graph $\mathcal{G} = (\mathcal{S}, \mathcal{T}, \mathcal{E})$, given one $u \in \mathcal{U}$, if $q(u, closest(u)) > 0.5$, we color u as `Green` and insert an edge connecting u and $closest(u)$ to $\mathcal{E}$. Otherwise, it is colored as `Red`. h edges are added into $\mathcal{E}$ that connect u and the nodes in $hclosest(u)$.

For example, given Table 2, we build a graph as Fig. 5. Since $q(o_{12}, closest(o_{12})) = 0.48 < 0.5$, we color it to `Red` and connect it with C_2, C_3 and C_4. Since $q(o_4, closest(o_4)) = 0.78 > 0.5$, it is colored as `Green` and connected with C_2.

4.2 Approximate Solution

In this section, we present our approximate solution to the graph-based question selection problem, or in short *AQS*. We then show that AQS has a small approximate ratio. AQS consists of 3 steps. The goal is to resolve all vertices in $\mathcal{T}$.

Step 1. Evaluate `Green` nodes: AQS first tackles the `Green` nodes that are likely to be inliers. These green nodes are evaluated based on the $\mathcal{S}$ connected with them, where these nodes in $\mathcal{S}$ correspond to the context inliers.

Given one node s_i in $\mathcal{S}$, AQS first finds out all `Green` nodes connected with s_i, denoted as $g(s_i)$. For each s_i, based on the size of $g(s_i)$, AQS generates $\lceil \frac{|g(s_i)|}{k} \rceil$ questions. Each question has s_i as the target and the `Green` nodes in $g(s_i)$ as the options. Among these questions, the options of $\lfloor \frac{|g(s_i)|}{k} \rfloor$ questions are fully occupied by k `Green` nodes.

However, if $k\%|g(s_i)| \neq 0$, one question still has $k - k\%|g(s_i)|$ options that remain unused. These leftovers are marked as available and will be used later to evaluate red nodes. For example, C_4 is connected with 2 `Green` nodes. Since $k = 3$, AQS generates $\lceil \frac{2}{3} \rceil = 1$ question. The question has one option remaining available. So its capacity is 1.

Step 2. Evaluate some `Red` nodes for free: We use $\mathcal{S}'$ to denote a subset of $\mathcal{S}$, where each $s \in \mathcal{S}'$ is connected with at least one `Green` node. For example, $\mathcal{S}' = \{C_2, C_3, C_4\}$ in Fig. 5. $\mathcal{T}'$ denotes a subset of `Red` nodes in $\mathcal{T}$. For each $t \in \mathcal{T}'$, there exist at least one edges that connect t to the nodes in $\mathcal{S}'$. Thus potentially the red nodes in $\mathcal{T}$ can be verified using the available capacities of the questions created in Step 1, without having to create new questions.

For each $t \in \mathcal{T}'$, we use a h-tuple to denote the available capacities of the nodes in $\mathcal{S}'$ t connects to. For example, as shown in Fig. 5, $\mathcal{T}' = \{o_{12}\}$. o_{12} is connected to C_2, C_3 and

C_4. Since the available capacities of C_2, C_3 and C_4 are 2, 2, and 1, the h-tuple of o_{12} is (2,2,1).

Note o_{13} is not in $\mathcal{T}'$. The reason is that in graph $\mathcal{G}$ it is connected to C_1, while C_1 is not used to evaluate the green nodes in Step 1, and hence is not listed in $\mathcal{S}'$.

AQS then iterates each $t_i \in \mathcal{T}'$. Given a t_i, if all green nodes in its h-tuple have remaining capacities, t_i is assigned as options to the h questions created in Step 1, which use the green nodes connected to t_i as targets. Then the capacities recorded in h-tuple decrease by 1.

For example, the h-tuple of o_{12} is (2,2,1), indicating that the status of o_{12} can be resolved using the existing questions. More specifically, o_{12} is assigned to the existing questions that use C_2, C_3 and C_4 as target. Meanwhile, the capacities of C_2, C_3 and C_4 decrease to (1,1,0). Therefore, no Red node can be assigned to the question that uses C_4 as target.

Those questions are then submitted to the human. For each question, if a Green node in the options is believed to match with its target, it will be indeed an inlier. Therefore, its status is resolved. Otherwise, AQS changes this green node to Red. Meanwhile, by Def. 3.1, in the bipartite graph each Red node u should connect to h nodes in $\mathcal{S}$, corresponding the h nearest context inliers (inlier clusters) of u. Since in the current graph, u only has one edge connecting to its nearest context inlier, $h-1$ new edges have to be created. The unresolved Red nodes are evaluated in the next step.

Step 3. Evaluate the rest Red nodes: We first give some notations used in this step. $\mathcal{T}_u$ denotes the set of unresolved Red nodes (potential outliers). $\mathcal{S}_u$ denotes a subset of $\mathcal{S}$, where each $s \in \mathcal{S}_u$ connects to at least one red node in $\mathcal{T}_u$. $r(s_i)$ represents the set of unresolved Red nodes that are connected to one $s_i \in \mathcal{T}_u$. In our example, $\mathcal{T}_u = \{o_{13}\}$, $\mathcal{S}_u = \{C_1, C_2, C_3\}$ and $r(C_1) = r(C_2) = r(C_3) = 1$. New questions have to be created to evaluate the Red nodes in $\mathcal{T}_u$.

To achieve this, AQS provides two alternative algorithms suitable for different situations.

Red Node (Outlier Candidate) Driven Solution (OC). When forming a question a red node is used as the target, and its corresponding h nodes are used as options. $\lceil \frac{h}{k} \rceil$ questions are produced for each unresolved red node. Thus, in total $\lceil \frac{h}{k} \rceil |\mathcal{T}_u|$ questions are produced. In our example, $\lceil \frac{h}{k} \rceil \cdot |\mathcal{T}_u| = 1$.

Inlier Cluster (Context Inlier) Driven Solution (IC). For each $s_i \in \mathcal{S}_u$, we generate $\lceil \frac{|r(s_i)|}{k} \rceil$ questions. Each question uses s_i as the target and those red nodes as the options. In total $\sum_{s_i \in \mathcal{S}_u} \lceil \frac{|r(s_i)|}{k} \rceil$ questions will be generated. In our example, it generates 3 questions.

Cost-based Algorithm Selection. The cost model of OC solution corresponds to cost(OC) = $\lceil \frac{h}{k} \rceil |\mathcal{T}_u|$, while the cost model of IC corresponds to cost(IC) = $\sum_{s_i \in \mathcal{S}_u} \lceil \frac{|r(s_i)|}{k} \rceil$.

Therefore, if $cost(OC) < cost(IC)$, we choose solution OC. Otherwise, we choose solution OC. In our example, solution OC is chosen, which produces one new question $\{\underline{o_{13}}, C_1, C_2, C_3\}$.

These questions are then submitted to the human. If Red node does not match any node in $closest(u)$, it is declared as an outlier. Otherwise, it is an inlier.

4.3 Approximate Ratio

Next, we establish the approximate ratio of AQS.

Theorem 4.2. *Given a bipartite graph $\mathcal{G}$ which contains n^+ Green nodes and n^- Red nodes with h edges, if the number of options in each question is set as k, the AQS algorithm has an approximate ratio* $1 + \frac{n^+ + \lceil \frac{h}{k} \rceil k n^-}{n^+ + h n^-} + \frac{1}{W^*}$, *where W^* denotes lower bound of the question selection algorithm.*

Proof. We omit the proof because of the space limit and put it into the technical report [7]. □

Note as typically used in our experiments, k usually is set to the same value with h. In this case the approximate ratio is equal to $2 + \frac{1}{W^*}$. Since W^* is guaranteed to be much larger than 1, this is a small approximate ratio.

5 CONTEXT INLIER DISCOVERY

As discussed in Section 2.2, HOD includes context inliers in the multi-object question. These context inliers assist the human to verify the outlier candidates. However, if an object used as a context inlier is in fact an outlier, it would lead to misleading questions and in turn the mis-classification of the outlier candidates.

In this work, we design a clustering-based method to reliably discover the context inliers. The key idea is to apply a clustering method on $\mathcal{I}$ and only consider the objects in the big clusters as inliers $\mathcal{IC}$. This way, the small *outlier clusters* that are not discovered by the outlier detection algorithms will be safely excluded from context inliers. One object will be selected from each of these big clusters as a context inlier. This not only reduces the number of context inliers, but also ensures the diversity of the context inliers, effectively representing the typical characteristics of the given dataset.

Intuitively, we can use any existing clustering method to cluster the inlier candidates. However, since clustering is an unsupervised technique, due to the lack of human supervision, the formed clusters tend to contain objects that are not considered to be similar in the application domain [26].

To solve this problem, we introduce a clustering method that effectively leverages human input to ensure the quality of the produced clusters. More specifically, using a small number of examples provided by the human, we first learn a function that for each pair of objects, computes a probability indicating how possible this object pair belongs to the same cluster (Section 5.1). We then again leverage human feedback

to establish a probability cutoff that is used to produce the clusters, similar to the distance threshold used in classical clustering methods such as DBSCAN [16] (Section 5.2). The clustering algorithm is also presented in this section. In the end, we show how to use the clustering results to refine the outlier candidates produced by the unsupervised outlier detection methods (Section 5.3).

5.1 The Learned Clustering Probability

Here we use d_{ij} to denote the distance between a pair of objects o_i and o_j computed based on Euclidean distance or Jaccard distance. Using a small set of training examples, HOD learns a function F that maps d_{ij} to $p_{ij} \in [0, 1]$, representing the probability that o_i and o_j belong to the same cluster, so called *clustering probability*.

To acquire training examples, we first sample some object pairs from O and then ask the human if each object pair belongs to one cluster or not. If the answer to a given pair is positive, $p = 1$. Otherwise $p = 0$. For example, we can ask the human if "Huawei Mate 10" and "Huawei Mate 20" in Table 1 belong to the same cluster (category). Using as input the labeled object pairs and the distances between the objects in these pairs, HOD employs the logistic regression model to learn the function $F^{\dagger}$. As shown in Equation (1), o_i^m denotes the value on m-th attribute of o_i. $d(o_i^m, o_j^m)$ denotes the similarity between o_i and o_j on attribute m.

$$p = \frac{1}{1 + e^{-\theta^T x}}, x = (x_1, ..., x_M), x_m = d(o_i^m, o_j^m),\ m \in [1, M]. \tag{1}$$

When computing this similarity score, different distance functions can be used on different attributes. For example, we can use Euclidean distance for numerical attributes and Jaccard or Edit distance for text attributes.

Therefore, our method naturally supports data with mixed data types. θ corresponds to a vector of parameters to be learned, which capture the significance of each attribute. Each element in feature vector x represents the distance value computed on one attribute.

5.2 Threshold Setting

HOD uses a cutoff threshold τ to determine if two objects should be clustered together. A pair of objects will be assigned to the same cluster if the clustering probability produced by F is larger than τ. Therefore, τ is critical for the quality of the produced clusters. In this work, we propose a binary search-based method to establish an appropriate τ.

Binary Search-based Method. HOD first uniformly divides the probability range [0,1] into $|R|$ subranges $R = \{r_1, r_2, ..., r_{|R|}\}$. r_i is considered to be smaller than r_j if the upper bound of

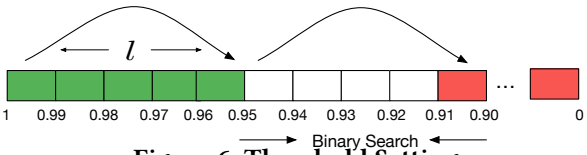

Figure 6: Threshold Setting

r_i is smaller than the lower bound of r_j. These subranges are sorted in descending order. For example, if $|R| = 100$, $r_1 = (0.99, 1]$, $r_2 = (0.98, 0.99]$, and $r_{100} = [0, 0.01]$. All object pairs are distributed into the corresponding range based on the probabilities computed by function F. One object pair is sampled from each subrange as the representative. Human is then asked to evaluate if these object pairs belong to the same cluster or not.

Intuitively, if the human says Yes to an object pair in subrange r_i, it is highly likely that the answers to the object pairs in subranges $\{r_j | j < i, r_j \in R\}$ are also Yes. This is because these object pairs have larger probabilities to be clustered together than those in r_i. If the answer is instead No, then the object pairs in subranges $\{r_j | j > i, r_j \in R\}$ also cannot be clustered together. An appropriate threshold τ should be able to separate the Yes object pairs and the No object pairs. In other words, τ should correspond to the largest Yes subrange.

Based on this observation, we design a binary search-based method to locate the τ threshold. It effectively minimizes the number of object pairs to be evaluated by the human. That is, it starts by asking the human to check the object pair sampled from subrange $r_{\frac{|R|}{2}}$. If the answer is Yes, the human will proceed to check the subrange $r_{\frac{3}{4}|R|}$. Suppose the user eventually stops at the subrange $r_k = [low, high]$. If the answer is Yes, $\tau = low$. Otherwise $\tau = high$.

A Conservative Strategy. However, although the above binary search-based method prunes a large number of subranges, it may generate a relatively low threshold and thus form invalid clusters. It happens in the cases that the object pair in a skipped subrange is actually a non-matching pair, but is incorrectly marked as Yes, because an object pair was identified as a matching pair before, and it lies in a subrange with a index smaller than the skipped subrange.

To avoid such problems, we propose a conservative strategy that starts from the subrange with the high probability (small index in R), where the object pairs are likely to be clustered together, as shown in Fig. 6. More specifically, in the t-th human identification iteration, we ask the human to check the $(l \times t)$-th subrange, where l denotes the number of subranges skipped in each iteration, for example $l = 5$ in Fig. 6. If the pairs in the $(l \times t)$-th subrange are marked as Yes, all pairs in the subranges with indices smaller than $l \times t$ are marked as Yes (color those subranges in green in Fig. 6). The algorithm jumps to the $(l \times (t+1))$-th subrange to conduct the next iteration of human evaluation. Otherwise, the algorithm marks all pairs in the subranges with an index

[†]Other learning models such as Random Forest can be applied here if they can produce a confidence score to represent the clustering probability.

	$\|O\|$	# Attr	# Outliers
Product	1070	5	74
Paper	10578	8	817
Glass	214	9	9

Table 3: Datasets

larger than $l \times t$ as No (color those subranges in red). A binary search is conducted between subranges $l \times (t-1)$ and $l \times t$.

Clustering Algorithm. Using the learned clustering probabilities and the threshold τ, we design a clustering algorithm to find the collection of context inliers $\mathcal{IC}$. It first picks a pivot o randomly from $\mathcal{I}$, computes $p_{oo'}, o' \in \mathcal{I}$, and add those objects into a cluster if $p_{oo'} > \tau$. Then it removes the objects in this cluster from $\mathcal{I}$, picks another pivot, and repeats until all objects in $\mathcal{I}$ are assigned to some clusters.

5.3 Outlier Candidate Refinement

Next, we show how to use the clustering results to identify the potential outliers missed by the unsupervised outlier methods and prune the outlier candidates that do not have to go through the human evaluation phase. As confirmed in our experiments, these intuitive methods are effective in reducing the human efforts yet improving the accuracy of outlier detection.

Identifying Missed Outliers. By [23], big clusters with a large number of objects tend to be inlier clusters. So HOD divides the produced clusters as big clusters and small clusters using the method introduced in [23]. We then move the objects in the small clusters to the outlier candidate set $\mathcal{U}$. More details are described in our experiments (Sec. 6).

Pruning Outlier Candidates. To achieve this, we leverage the candidate-cluster likelihood $q(u, closest(u))$ between an outlier candidate u and its closest inlier cluster (Def. 4.1). For each $u \in \mathcal{U}$, if $q(u, closest(u)) > \tau$ which corresponds to the threshold learned through human feedback as shown in Section 5.2, u will be directly declared as an inlier. This is because very possibly u also belongs to $closest(u)$. On the other hand, if $q(u, closest(u)) < 1-\tau$, u will be declared as an outlier without having to go through the human evaluation. These objects are then removed from $\mathcal{U}$.

6 EXPERIMENTAL EVALUATION

We evaluate our HOD approach on both the human efforts and the quality of the produced outliers. HOD is implemented in Python. The experiments were run on a Ubuntu server with Intel 2.4GHz Processor and 32GB memory.

6.1 Experiment Setting

Datasets. We evaluate HOD using three real datasets, namely Product dataset, Paper dataset and Glass dataset. The statistics of the datasets are summarized in Table 3.

Product. We crawl 1070 different brands of cellphones from Amazon. Each product has 5 attributes: Asin, Title, Brand, Also viewed, Bought together. Outliers correspond to accessories of cellphones such as case, charger, earphone etc.

	Clustering	Question Selection	Total time
Product	10min	4min30s	14min30s
Paper	10min	18min48s	28min48s
Glass	10min	4min18s	14min18s

Table 4: Time of outlier detection

Paper. We crawl 130 Google Scholar pages. Each page contains publications of a senior researcher in Computer Science. In total we obtain 10,578 papers (objects). Each object consists of eight attributes: Title, Authors, Date, Venue, Volume, Issue, Pages and Publisher. Outliers correspond to the papers that are written by other researchers, but erroneously assigned to this researcher by Google Scholar. Since the co-authors, research topics, or publication venues of these outlier papers are different from the papers authored by the 130 researchers, human can easily capture these outliers.

Glass. Glass [1] is an outlier benchmark dataset with 214 objects and 9 attributes. Each attribute represents one property of glass. In total, it contains 7 types of glasses. The outliers correspond to the tableware glass, because only a small number of objects belong to the tableware glass.

Evaluation Metrics. We evaluate both the human efforts and the precision/recall of outlier detection. The human effort is measured as the total number of questions needed to complete the outlier detection task including the costs in the training and clustering step, while precision and recall measure the quality of the identified outliers. Using L_T to denote the true outliers in the produced outlier set L, the precision $p = \frac{|L_T|}{|L|}$. The recall $r = \frac{|L_T|}{|\hat{O}|}$.

User Study. 10 students (7 males, 3 females aged from 18 – 24) from our group participate in the evaluation. None of them is expert in outlier detection. Each question includes five objects ($k = 5$). Before starting the evaluation, we demonstrate the participants using two example questions from each dataset. Two batches of questions are generated sequentially. The questions within each batch are independent with each other. Thus, users can answer questions in parallel. This accelerates the evaluation process and we show the time usage of our proposed method.

We report how long it takes to detect outliers in Table 4. First, we give each user 10 min to label, browse the dataset, and understand the task requirement. Next, the users start to answer questions. Since they work in parallel, most datasets can be evaluated in 20 min and all datasets can be handled within half an hour, as shown in Table 4.

To handle human errors, we assign each question to three different users and aggregate their answers using the majority voting (MV), following the typical error-tolerance practice in crowdsourcing [18, 36, 37]. More specifically, given a question in Fig.2, it is assigned to three users. Correspondingly, for each question, three answers are collected and aggregated using MV. A similar approach is also applied to the baselines for fair comparison.

Training Step. As shown in Section 5, HOD trains a logistic regression model to learn the probability that two objects belong to one cluster. Therefore, we need to acquire some training examples through human feedback. We first sample some object pairs and then ask the human if these pairs of objects belong to one cluster. To acquire a similar number of Yes pairs and No pairs, the number of close object pairs is similar to the number of distant object pairs.

In the evaluation, we acquired 50 labeled object pairs. Using our multi-object question interface defined in Def. 2.2, this only costed 10 questions, much less than the number of questions generated for evaluating outlier candidates.

Setting Up Clustering Threshold. Setting up the threshold τ used in clustering also costs a few questions. As illustrated in Section 5.2, we set the threshold by checking object pairs in the 100 probability subranges. Eventually we obtain the $\tau = 0.8$ for dataset Product, $\tau = 0.8$ for Paper and $\tau = 0.7$ for dataset Glass, after asking 5 or 6 questions.

Comparison with the State-of-the-art. We compare HOD with the well known unsupervised outlier detection methods including KNN [40], LOF [5], Cluster method [23] and Ensemble method [3] as well as active learning based methods AI^2 [42] and AAD [14].

(1) Unsupervised Methods:

1) KNN computes an outlier score for each object. The score is defined as the distance from the object to its K-th nearest neighbor in a given data set.

2) LOF computes a LOF value for each object as the outlier score. For a given object, the LOF score is equal to the ratio of the average local density of the K nearest neighbors of the object and the local density of this object itself.

3) Cluster method adopts the idea that inlier objects belong to large and dense clusters, while outliers belong to either small or sparse clusters. Therefore, it first clusters on the entire dataset, ranks them based on the size of the clusters in descending order, and divides these clusters into large or small ones. More specifically, suppose $\mathcal{IC} = \{C_1, C_2, ..., C_{|\mathcal{IC}|}\}$ is the set of clusters, where $|C_1| \geq |C_2| \geq ... \geq |C_{|\mathcal{IC}|}|$. Given a β, we define b as the boundary to separate large and small clusters if $\frac{|C_b|}{|C_{b+1}|} \geq \beta$.

4) Ensemble first uses the above three outlier detection algorithms to detect outliers separately and then takes a majority voting on the results. If an object is regarded as an outlier by most algorithms, it will be considered as an outlier.

We compare HOD with all above methods. To ensure a fair comparison, we make sure that different approaches use the same parameters (K in KNN, K in LOF and β in Cluster) as HOD. On each dataset we evaluate 4 groups of parameters, that is, $para_1 = \{20, 20, 3\}$, $para_2 = \{30, 30, 4\}$, $para_3 = \{40, 40, 5\}$, $para_3 = \{50, 50, 6\}$ respectively, where $para_1 = \{20, 20, 3\}$ indicates that we set K in KNN and LOF as 20 and β in Cluster as 3. In the Ensemble approach and the ensemble-based outlier candidate generation phase of HOD, the same set of parameters are used. Given these parameters, we compute the outlier scores using the above algorithms and rank the objects in descending order by the scores. The top-n objects with largest outlier scores are selected as outliers. The parameter n influences the quality of detected outliers. For these unsupervised outlier detection algorithms, we select a n that leads to the highest F1-score ($f_1 = \frac{2pr}{p+r}$). But for HOD, when we ensemble machine-based algorithms to acquire outlier candidates, we fix n as $0.1 \times |O|$ – a clear bias to these unsupervised methods.

(2) Active Learning based Methods: We compare with two methods: AI^2 [42] and AAD [14]. The details are discussed in Related work (Section 7).

(3) Entity Matching (EM) based Methods: We compare with two methods: CrowdER [44] and Magellan [24].

1) CrowdER uses crowdsourcing techniques to identify matching pairs. Specifically, it generates all pair of objects, prunes some of them that are likely to be non-matching based on the likelihood and some pruning techniques [45], and finally asks the human to label the rest pairs through cluster-based questions [44]. We first leverage CrowdER to cluster on $\mathcal{I}$. Given the clustering results, we use clusters of size larger than 1 as context inliers and apply our question selection strategy on top of them.

2) Magellan asks the human experts to label some matching or non-matching objects and then train a random forest to process the dataset. For fair comparison, in Magellan, we ask the experts to label the same number of object pairs as HOD does. Next, we use Magellan to train a model to predict whether two objects can be clustered and use the model to conduct clustering on the inlier candidates and generate context inliers similar to HOD. Then we use our question selection method to identify outliers.

Evaluation on Alternative Strategies. We also compare HOD with three baselines. 1) ClusterAll. We design a baseline algorithm that first clusters on the entire dataset, split the cluster into inlier candidate subset and outlier candidate subset, and evaluate the objects in the latter subset using our proposed algorithm AQS in Section 4.2. This experiment shows the significance of our outlier ensemble and clustering strategy. 2) CheckAll. Given $\mathcal{IC}$ and the to-be-evaluated outlier candidates $\mathcal{U}$, CheckAll identifies outliers by evaluating every candidate in $\mathcal{U}$ without using the outlier candidate pruning strategy. 3) CandAll. CandAll is an alternative of AQS . The difference from AQS is that when forming questions, CandAll uses each outlier candidate $u \in \mathcal{U}$ as the target and the objects in $hclosest(u)$ as options.

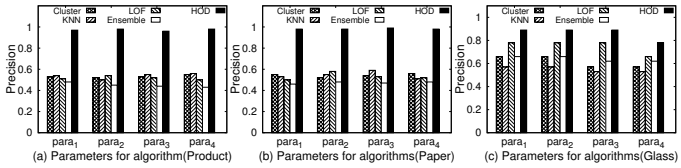

Figure 7: Evaluate State-of-the-art Works: Precision

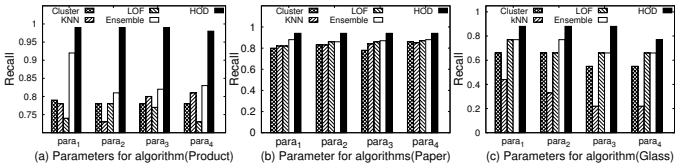

Figure 8: Evaluate State-of-the-art Works: Recall

6.2 Comparison with the State-of-the-art

We compare with three categories of state-of-the-art outlier detection methods, namely unsupervised methods (Section 6.2.1), active learning based-methods (Section 6.2.2), and crowdsourcing entity matching (EM) methods (Section 6.2.3). By default the parameter h is set as 5.

6.2.1 Comparison with the Unsupervised Method. In this set of experiments, we compare HOD against state-of-the-art unsupervised outlier detection approaches KNN, LOF, Cluster method and Ensemble method by measuring the precision (Fig. 7) and recall (Fig. 8) under various parameter settings.

For the **precision** of Product dataset, as shown in Fig. 7(a), in all cases Ensemble achieves the lowest precision. For example, as using $para_3$, Ensemble has a precision of 44%, while the precision of other machine-based approaches is 53% (Cluster), 55% (KNN), and 52% (LOF) respectively. This is because Ensemble computes the union of different approaches and involves many inliers into outlier candidates.

However, all these methods are far worse than our method HOD. For example, when we use $para_4$, HOD has a precision of 98% – more than 40% higher than that of Cluster (55%), KNN (56%), LOF (50%) and Ensemble (43%) on average. This is because HOD leverages the human input to locate inliers in the outlier candidates, while these inliers are mis-classified as outliers by these machine-based algorithms. In addition, as the parameters vary, the precision of HOD does not vary much. This is because we ensemble different algorithms to compute the candidates, covering as many outliers as we can. Therefore, our algorithm is not sensitive to the parameters.

For dataset Paper in Fig. 7(b), HOD also outperforms these machine-based algorithms. For example, when using $para_1$, HOD has a high precision of 96%, while the precision of other methods is 55% (Cluster), 53% (KNN), 50% (LOF) and 46% (Ensemble) respectively.

For Glass, when using $para_3$, HOD has a precision of 89%, while the precision of other methods is 57% (Cluster), 53% (KNN), 78% (LOF) and 62% (Ensemble), as shown in Fig. 7(c).

For **recall** of Product dataset, as shown in Fig. 8(a), Ensemble performs better than other machined-based approaches. For example, when using $para_2$, Ensemble has a recall of 81%, which is higher than that of Cluster (78%), KNN (73%) and LOF (78%). This is because the ensemble method obtains more true outliers than any single method. Our HOD still performs the best among all approaches. For instance, when using $para_3$, HOD has a recall of 98%, while the recall of other methods are 80% (Cluster), 82% (KNN), 88% (LOF) and 90% (Ensemble) respectively. This is because HOD leverages both the outlier ensemble and clustering approaches to avoid the missing of true outliers. Similarly, for Paper and Glass dataset, HOD outperforms the others and is not sensitive to the parameters as shown in Fig. 8(b) and (c).

6.2.2 Comparison with the Active Learning Method. For a fair comparison, we design the same number of questions for AI^2, AAD and HOD. We apply $para_3$ to HOD when comparing with the other two methods. Under this situation, HOD costs 78, 352, and 52 questions on the three datasets respectively. For AI^2, we conduct 10 human evaluation iterations. Using Product dataset as example, we ask 78 questions in total. Therefore, we ask $X = \lceil \frac{78}{10} \rceil$ questions in each iteration. When we test AAD, 78 iterations are conducted, because AAD only selects one question to ask in each iteration.

As shown in Fig. 9(a), HOD performs better than the other two methods on *precision*. On Product dataset, HOD has a precision of 96%, while the precision of AI^2 is only 19%. This is because in AI^2 the training data is extremely unbalanced – containing much more labeled inliers than outliers. AAD has a high precision 92%, because it relies on the human to label the objects and does not use the trained classifier to predict. However, AAD still performs worse than HOD, because AAD requires the users to directly label each outlier. This is error-prone. Our multi-object question in HOD instead provides users the context inliers as the references during the evaluation process. Therefore, it is much easier to answer.

As for *recall*, as depicted in Fig. 9(b), HOD also performs better than the other two methods. For example, on Paper dataset, HOD has a recall of 96%, while the recall of AAD is only 85%. This is because HOD discovers almost all outliers in the machine-based outlier detection and clustering phases and then uses human input to effectively filter out the false outliers in the following step. However, AAD utilizes the machine learning model to recommend the potential outliers for the user to label, while the machine learning model tends to miss true outliers because of the imbalanced training data (insufficient number of labeled outliers). AI^2 has a recall of 92%, which is higher than AAD. This is because in addition to the human evaluated outliers, AI^2 also automatically produces outliers from the unlabeled data using the classification model, although the classification model is not accurate enough due to the imbalanced training data.

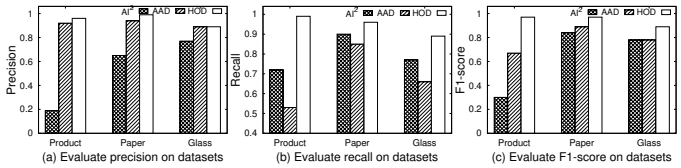

Figure 9: Evaluate Active Learning Works

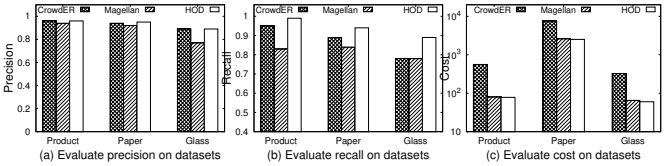

Figure 10: Evaluate EM-based Approaches

Finally, we report the F1-scores of these methods on the three datasets in Fig. 9(c). HOD has the highest F1-score on all these three datasets. For example, on Paper dataset, HOD has an F1-score 95%, while the F-1 scores of AI^2 and AAD are 84% and 86% respectively. On Glass dataset, HOD has an F1-score 83%, while AI^2 and AAD have the same F-1 score at 78%.

6.2.3 Comparison with the Entity Matching Methods. In this section, we compare HOD with EM methods CrowdER [44] and Magellan [24]. For CrowdER [44], we ask the users to process the dataset using its proposed method and compare the precision, recall and cost with HOD. CrowdER uses cluster-based questions to interact with users. We set the cluster size as k for fair comparison. For a fair comparison with Magellan [24], we set the same budget of cost, and compare its precision and recall against our HOD.

For precision, in Fig. 10(a) HOD has a similar precision with CrowdER on three datasets. CrowdER achieves a good performance because it leverages humans to address all pairs. HOD achieves a similar performance, because it also incorporates human knowledge, i.e., using some labels to train a model and learn a threshold for clustering. Besides, we cluster on a pure candidate set whose objects are likely to be inliers. This leads to a good clustering result. We can also observe that HOD has higher precision than Magellan. For example, on dataset Product, HOD has a precision of 96%, while the precision of Magellan is only 83%. This is because HOD can judiciously select the threshold for clustering after training, while Magellan relies on the model for clustering.

For recall, HOD outperforms CrowdER by about 5%. For example, on the Product dataset, HOD has a recall of 99%, while the recall of CrowdER is 95%. The reason is that CrowdER uses clusters of size larger than 1 as context inliers. This does not perform well, because some of small clusters are also likely to be outliers. Besides, HOD also outperforms Magellan (84%) for the similar reason to that of precision.

For cost, since we use the same number of labels for HOD and Magellan for fair comparison, they have the same cost. In Fig. 10(c) CrowdER takes much more cost than HOD on all datasets. For example, on the Product dataset, HOD only asks 78 questions, while CrowdER asks 560. This is because CrowdER needs to generate multi-object questions to cover all pairs of objects in $\mathcal{I}$ and there are a large number of pairs ($|\mathcal{I}|^2$). Although CrowdER prunes some of pairs using a threshold [44] and leveraging the transitivity [45], it still costs too much. Therefore, our method not only saves the cost but also achieves high quality.

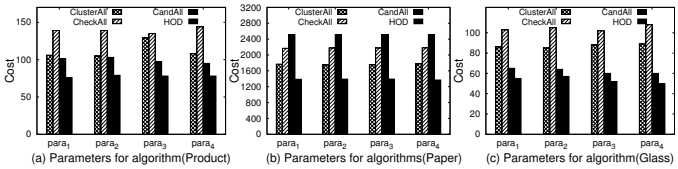

Figure 11: Evaluate Baselines: Human Efforts

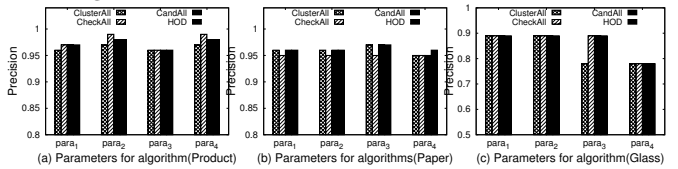

Figure 12: Evaluate Baselines: Precision

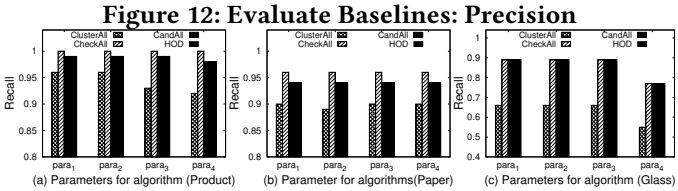

Figure 13: Evaluate Baselines: Recall

6.3 Evaluation on Our Techniques

We compare HOD against the three alternative algorithms (Section 6.1) by measuring human efforts (number of questions asked) (Fig. 11), precision (Fig. 12) and recall (Fig. 13).

For the *human efforts*, HOD significantly outperforms all baseline algorithms, because it uses the AQS algorithm to select questions which is shown to have a good theoretical guarantee. For example, as shown in Fig. 11(a), when we use $para_1$, HOD only uses 76 questions and costs about 30% less than ClusterAll, because ClusterAll clusters on the entire dataset and generates many small clusters that have to be evaluated by human. CheckAll costs about 2 times more than HOD, because HOD prunes some objects that are very likely to be inliers or outliers and hence do not have to be evaluated by human. HOD costs about 25% less than CandAll, because HOD fully uses the capacities of each question. As another example, when using $para_2$, ClusterAll, CheckAll and CandAll use 105, 139 and 103 questions respectively, while HOD only uses 79 questions.

Similarly, for dataset Paper, HOD outperforms the three baseline algorithms as shown in Fig. 11(b). For example, when using $para_3$, HOD costs 1394 questions, while ClusterAll, CheckAll and CandAll cost 1752, 2177 and 2512 questions respectively. On this dataset, CandAll costs nearly two times

Cost, Precision, Recall		Question Selection		
		HOD	CheckAll	CandAll
Clustering	HOD	78, 97%, 99%	144, 97% , 100%	98, 97%, 99%
	CrowdER	560, 96% , 95%	632, 96%, 96%	580, 96% , 95%
	Magellan	80, 94%, 83%	142, 96%, 85%	102, 94%, 83%

Table 5: Impact Comparison of Cost and Quality on the Product dataset.

more than HOD. This is because many inliers are hidden in the outlier candidate set. Unfortunately, unlike HOD, CandAll does not take any advantage of inliers which only need one comparison against the nearest context inlier to verify.

For dataset Glass, HOD still outperforms the three baseline algorithms as shown in Fig. 11(c). For example, when using $para_3$, HOD costs 52 questions, while ClusterAll, CheckAll and CandAll cost 88, 60 and 102 questions respectively.

For the **precision**, HOD and CandAll are expected to achieve the same precision because they use the same bipartite graph to produce questions. We can see from Fig. 12 (a) and (b) that all approaches achieve similar high precision, more than 96%. On Product dataset, the precision of HOD is slightly lower than CheckAll, because a few candidates that are regarded as outliers directly by our candidate pruning strategy are in fact inliers. However, the pruning is shown to be very effective in recognizing the obvious inliers which are very close to some context inliers. Therefore, HOD and ClusterAll even achieve a slightly better precision than CheckAll on Paper dataset. CheckAll has to check all candidates in $\mathcal{U}$ even if most of them are in fact inliers. Sometimes inliers are mis-classified as outliers in human evaluation, because the questions created for inliers only contain one context inlier as option to save costs. However, in the rare cases this introduces errors due to the lack of information for human to make correct decision.

For the **recall**, similar to the precision, HOD and CandAll performs the same. On all datasets (Fig. 13) HOD and CheckAll achieve a similar high recall, because they have the same outlier candidates and use human to identify the outliers. However, HOD outperforms ClusterAll. For example, on Product dataset, when using $para_1$, HOD has a recall of 99%, while the recall of ClusterAll is 96%. This is because in addition to the outlier candidates acquired by the ensemble of outlier detection techniques, HOD also incorporates more true outliers into $\mathcal{U}$ by treating the objects in small cluster as outlier candidates.

On Paper (Glass) dataset, HOD has a recall of 94% (88%), while the recall of ClusterAll is 89% (66%) using $para_2$.

6.4 Evaluation on Question Selection

Next, we show in Table 5 how different techniques for clustering and question selection perform on the Product dataset in both cost and quality. Each column denotes the methods that use different clustering methods but the same question-selection methods. Each row denotes the methods that use

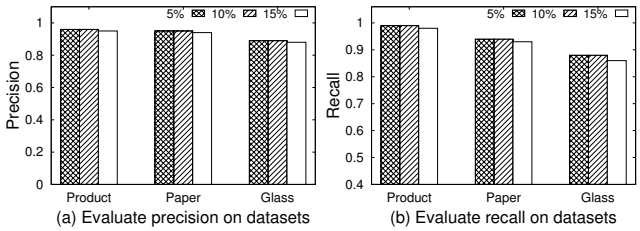

Figure 14: Evaluate experts errors

the same clustering method but different question-selection methods. We have the following observations.

First, clustering strategies have large impact on the cost and quality (i.e., comparing the rows in the same column). For cost, CrowdER takes much larger cost than HOD and Magellan. The reason is that CrowdER requires to ask humans to label many pairs of objects, while HOD and Magellan ask the humans to label a small number of objects for clustering. For quality, HOD has similar quality with CrowdER but achieves much higher quality than Magellan. This is because even if Magellan also asks the humans to label the similar number of questions with HOD, Magellan does not leverage the humans to judiciously determine an appropriate threshold used in clustering and thus it may involve potential outliers during the clustering, leading to missing true outliers. In summary, HOD is able to get high clustering quality, while achieving lower cost.

Second, different question selection methods perform differently on cost (i.e., comparing the columns in the same row). For cost, HOD saves half of the cost compared to CheckAll and saves 25% cost compared to CandAll, because HOD prunes those unnecessary pairs, and CheckAll and CandAll still need to ask some unnecessary pairs. For quality, question selection strategies achieve the similar precision and recall, because they both leverage the human intelligence to check the outlier candidates. In summary, HOD is able to prune some unnecessary objects, without sacrificing the quality.

6.5 Evaluation of Human Errors

We simulate the error rate of experts as 5%, 10% and 15% respectively and test the precision, recall and cost. The experimental results are shown in Fig. 14. In Fig. 14(a) for different error rates of experts, the precision is nearly the same. For example, on the Product dataset, the precision of users with 5% and 10% error rate is 96% and the precision of 15% is 95%, because the majority voting is good at handling the expert errors. However, the precision is still not perfect due to other types of errors induced by pruning methods or a small h. For recall, we observe the similar phenomenon with the precision. On the dataset Product, the recalls are 99%, 99% and 98% respectively for different error rates.

7 RELATED WORK

Outlier Detection Techniques. A large number of outlier detection techniques have been proposed in the literature.

Outlier detection techniques can be broadly classified into supervised methods and unsupervised methods.

Supervised Outlier Detection. Techniques in this class assume that abundant labeled outliers and inliers are available beforehand. Using these labeled data, a classification model is then trained, which classifies a testing object as an outlier or inlier. However, supervised outlier detection has two severe problems. First, outliers in the training set typically are far fewer than inliers. This leads to the imbalanced class distribution and thus impacts the classification accuracy [38, 43, 46]. Second, it is difficult to obtain a large number of high quality labels, especially outlier labels because of the rarity nature of outliers [2, 41]. Due to these problems, supervised outlier detection methods are not prevalent in real applications.

Unsupervised Outlier Detection. Techniques [3, 5, 6, 23, 40, 52] in unsupervised class are widely used in real world, because they do not require any training data. In general, these techniques detect outliers based on the observation that inliers in the data set usually is much more frequent than outliers. However, the accuracy of unsupervised techniques tends to be low as we described in Section 1. We thus propose to use human input to improve the accuracy of the unsupervised techniques. However, although human is involved, unlike the supervised techniques, we do not require the human to explicitly mark the objects as inliers or outliers, which is typically hard and extremely time-consuming. Instead, by carefully selecting the questions to be answered by the human as shown in Fig. 2, our HOD is able to accurately discover outliers with the least human efforts.

Active Learning-based Methods Given a human cost budget, the active learning methods choose objects from the dataset, ask the human to label these objects and train a classifier iteratively until the budget is used up. In [14, 42], active learning has been used to detect outliers.

(1)AI^2 [42]: Each time AI^2 selects X objects for the users to label, in which $\frac{X}{2}$ objects come from an unsupervised method and the other $\frac{X}{2}$ objects are provided by the supervised method (random forest). After the X objects are labeled, it retrains the supervised model and selects another X objects until the budget is used up. Finally, it uses the model to predict the rest unlabeled data.

(2)AAD [14]: First, AAD extracts features for each object, and uses a supervised method to train a classifier. Next, based on the modeling results, AAD selects one object that is most likely to be an outlier for the human to label. Then it iteratively retrains the model to adjust the weight of each feature and selects a new outlier candidate.

Although the active learning methods effectively reduce the labeling efforts of the human, they do not solve the unique challenge in outlier detection, that is, the lacking of ground truth outliers in the real applications due to the rarity nature of outliers. Therefore, their outlier detection accuracy is still much lower than our HOD approach as confirmed in our experiments.

Crowdsourcing. Crowdsourcing leverages the human cognitive ability to solve many significant data management and analytics tasks [8–13, 19, 21, 27, 28, 30, 32, 34, 35, 39, 47, 48]. Crowdsourcing entity matching (EM) [20, 24, 44] focuses on identifying records that refer to the same entity, while our proposed method HOD focuses on identifying outliers that are significantly different from others. In essence, these two problems are contradictory. Further, EM targets on text data (strings), while HOD generally supports various types of data including strings and numerical values. Even if EM-based techniques can be leveraged in the clustering step of HOD, they always suffer from either high cost (asking many pairs of objects) or low quality (leading to poor clustering results due to the influence of potential outliers during clustering). Some other existing works focus on data cleaning by experts [4, 17, 25, 29, 31, 49–51]. SampleClean [25] is a framework that only requires the human to clean a sample of data, and utilizes the cleaned sample to obtain unbiased query results with confidence intervals. However, it mainly focuses on processing aggregate queries rather than outlier detection. QOCO [4] is a query-oriented system that asks the crowd workers to clean inconsistent records or missing values. Our HOD instead is for outlier detection. To the best of our knowledge, this is the first work on crowdsourcing outlier detection. Traditional crowdsourcing EM approaches do not work well on this problem as confirmed in our experiments, because they either incur much cost or low quality.

8 CONCLUSION

In this paper, we propose a human-in-the-loop approach HOD that leverages the human input to accurately discover outliers from a dataset. HOD features a clustering algorithm to reliably generates context inliers that are used as context to contrast against the outlier candidates in the human evaluation phase. This facilitates the evaluation of outlier candidates. We then map the question selection problem to a bipartite graph-based edge cover problem, which is proven to be NP-hard. An approximate algorithm is then designed that solves this problem with theoretical guarantee. This effectively minimizes the human efforts used in the evaluation. Our experimental results on multiple real datasets show that HOD significantly outperforms the alternative approaches on both human efforts and outlier quality.

Acknowledgement. This paper is supported by NSF of China (61925205, 61632016), Huawei, and TAL education.

REFERENCES

[1] https://archive.ics.uci.edu/ml/datasets/.

[2] N. Abe, B. Zadrozny, and J. Langford. Outlier detection by active learning. In *KDD*, pages 504–509, 2006.

[3] C. C. Aggarwal. Outlier ensembles: position paper. *SIGKDD Explorations*, 14(2):49–58, 2012.

[4] M. Bergman, T. Milo, S. Novgorodov, and W. C. Tan. Query-oriented data cleaning with oracles. In *SIGMOD*, pages 1199–1214, 2015.

[5] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *SIGMOD*, pages 93–104, 2000.

[6] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Min. Knowl. Discov.*, 30(4):891–927, 2016.

[7] C. Chai, L. Cao, G. Li, J. Li, Y. Luo, and S. Madden. *Human-in-the-loop Outlier Detection.*

[8] C. Chai, J. Fan, and G. Li. Incentive-based entity collection using crowdsourcing. In *ICDE*, pages 341–352, 2018.

[9] C. Chai, J. Fan, G. Li, J. Wang, and Y. Zheng. Crowdsourcing database systems: Overview and challenges. In *ICDE*, pages 2052–2055, 2019.

[10] C. Chai, G. Li, J. Fan, and Y. Luo. Crowdchart: Crowdsourced data extraction from visualization chart. *IEEE Trans. Knowl. Data Eng.*, 2020.

[11] C. Chai, G. Li, J. Fan, and Y. Luo. Crowdsourcing data extraction from visualization chart. In *ICDE*, 2020.

[12] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. Cost-effective crowdsourced entity resolution: A partial-order approach. In F. Özcan, G. Koutrika, and S. Madden, editors, *SIGMOD*, pages 969–984. ACM, 2016.

[13] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. A partial-order-based framework for cost-effective crowdsourced entity resolution. *VLDB J.*, 27(6):745–770, 2018.

[14] S. Das, W. Wong, and et al. Incorporating expert feedback into active anomaly discovery. In *ICDM*, 2016.

[15] C. C. A. S. Edition. *Outlier Analysis.* Springer, 2017.

[16] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996.

[17] J. Fan, G. Li, B. C. Ooi, K. Tan, and J. Feng. icrowd: An adaptive crowdsourcing framework. In *SIGMOD*, pages 1015–1030, 2015.

[18] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: Answering queries with crowdsourcing. In *SIGMOD*, 2011.

[19] Z. Gharibshah, X. Zhu, A. Hainline, and M. Conway. Deep learning for user interest and response prediction in online display advertising. *Data Science and Engineering*, 5(1):12–26, 2020.

[20] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. W. Shavlik, and X. Zhu. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*, pages 601–612, 2014.

[21] S. Hao, C. Chai, G. Li, N. Tang, N. Wang, and X. Yu. Outdated fact detection in knowledge bases. In *ICDE*, 2020.

[22] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2009.

[23] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[24] P. Konda, S. Das, P. S. G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra. Magellan: Toward building entity matching management systems. *PVLDB*, 9(12):1197–1208, 2016.

[25] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, T. Kraska, T. Milo, and E. Wu. Sampleclean: Fast and reliable analytics on dirty data. *IEEE Data Eng. Bull.*, 38(3):59–75, 2015.

[26] B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013.

[27] G. Li, C. Chai, J. Fan, X. Weng, J. Li, Y. Zheng, Y. Li, X. Yu, X. Zhang, and H. Yuan. CDB: optimizing queries with crowd-based selections and joins. In *SIGMOD*, pages 1463–1478, 2017.

[28] G. Li, C. Chai, J. Fan, X. Weng, J. Li, Y. Zheng, Y. Li, X. Yu, X. Zhang, and H. Yuan. CDB: A crowd-powered database system. *PVLDB*, 11(12):1926–1929, 2018.

[29] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. Crowdsourced data management: A survey. *IEEE Trans. Knowl. Data Eng.*, 28(9), 2016.

[30] K. Li and G. Li. Approximate query processing: What is new and where to go? *Data Science and Engineering*, 3(4):379–397, 2018.

[31] K. Li, X. Zhang, and G. Li. A rating-ranking method for crowdsourced top-k computation. In *SIGMOD*, pages 975–990, 2018.

[32] M. Li, H. Wang, and J. Li. Mining conditional functional dependency rules on big data. *Big Data Mining and Analytics*, 03(01):68, 2020.

[33] Y. Luo, C. Chai, X. Qin, N. Tang, and G. Li. Interactive cleaning for progressive visualization through composite questions. In *ICDE*, 2020.

[34] Y. Luo, X. Qin, C. Chai, N. Tang, G. Li, and W. Li. Steerable self-driving data visualization. *IEEE Trans. Knowl. Data Eng.*, 2020.

[35] Y. Luo, X. Qin, N. Tang, and G. Li. Deepeye: Towards automatic data visualization. In *ICDE*, pages 101–112, 2018.

[36] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, pages 211–214, 2011.

[37] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *CIKM*, 2012.

[38] C. Phua, D. Alahakoon, and V. C. S. Lee. Minority report in fraud detection: classification of skewed data. *SIGKDD Explorations*, 2004.

[39] X. Qin, Y. Luo, N. Tang, and G. Li. Deepeye: An automatic big data visualization framework. *Big Data Mining and Analytics*, 1(1):75, 2018.

[40] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD*, pages 427–438, 2000.

[41] I. Steinwart, D. R. Hush, and C. Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 2005.

[42] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li. Aiˆ 2: training a big data machine to defend. In *BigDataSecurity*, pages 49–54. IEEE, 2016.

[43] R. Vilalta and S. Ma. Predicting rare events in temporal domains. In *ICDM*, pages 474–481, 2002.

[44] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

[45] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD*, 2013.

[46] G. M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In *KDD*, pages 359–363, 1998.

[47] T. Zhao, C. Chai, Y. Luo, J. Feng, Y. F. Huang, S. Yang, H. Yuan, H. Li, K. Li, F. Q. Zhu, and K. Pan. Towards automatic mathematical exercise solving. *Data Science and Engineering*, 4:179 – 192, 2019.

[48] T. Zhao, Y. Huang, S. Yang, Y. Luo, J. Feng, Y. Wang, H. Yuan, K. Pan, K. Li, H. Li, and F. Zhu. Mathgraph: A knowledge graph for automatically solving mathematical exercises. In G. Li, J. Yang, J. Gama, J. Natwichai, and Y. Tong, editors, *DASFAA*, 2019.

[49] Y. Zheng, G. Li, and R. Cheng. DOCS: domain-aware crowdsourcing system. *PVLDB*, 10(4):361–372, 2016.

[50] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng. Truth inference in crowdsourcing: Is the problem solved? *PVLDB*, 10(5):541–552, 2017.

[51] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*, pages 1031–1046, 2015.

[52] A. Zimek, R. J. G. B. Campello, and J. Sander. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *SIGKDD Explorations*, 15(1):11–22, 2013.