Human-Powered Blocking in Entity Resolution: A Feasibility Study

Weiling Li Jongwuk Lee Dongwon Lee
The Pennsylvania State University
University Park, PA 16802, USA
{wul135, jxl90, dongwon}@psu.edu

ABSTRACT

Entity Resolution (ER) is the problem of *matching* the records that refer to the same entity within or across two or more data sources. In recent years, *human-powered* ER solutions have been proposed so that challenging ER tasks, that machines cannot do well, can be helped by human workers. While successful in achieving high matching accuracy, existing human-powered ER methods did not incorporate a core technique, *i.e.*, *blocking*, for improving the scalability of the ER process. To address this issue, this paper carries out the feasibility study to validate whether the blocking technique can be integrated into the human-powered ER. Specifically, we first propose two variations of human-powered blocking methods. We then validate their effectiveness in improving the scalability of the ER process through simulated crowdsourcing and AMT-based experiments in synthetic and real-life datasets, respectively.

Categories and Subject Descriptors

H.2.0 [**Database Management**]: General

General Terms

Design, Management

Keywords

Crowdsourcing, Entity Resolution, Blocking

1. INTRODUCTION

Entity Resolution (*ER*) is the process of identifying all pairs of records that represent the same real-world entity. The recent explosion in both volume and velocity of data within scientific and industrial settings has made ER one of the critical data pre-processing techniques utilized before data analysis takes a place. A significant number of work has addressed these needs by attempting to develop machine-based techniques of entity resolution (*e.g.*, [2, 5, 6, 11, 18]). Despite their success, some ER tasks are still inherently challenging for machines to obtain high accuracy. To illustrate this, we explain the following example scenario.

Example 1. There are four images in Figure 1 and suppose "matching" images need to be identified. Note that the precise definition of the "matching" can be determined differently in various applications. This example shows a scenario where image matching is determined in a more semantic sense (*i.e.*, identifying images of the same sport type). This can be a challenging task, even when using the latest techniques in image processing. Not only it can be computationally challenging, diverse features such as color, shape, texture, or position can lead to different results. However, if the same set of images is given to humans, it would be relatively easy to identify that images (1) and (3) include cycling as the same activity.

(1) (2) (3) (4)

Figure 1: Description of four sport images for ER

Recently while addressing this challenge, an array of new attempts (*e.g.*, [7, 15, 16, 17]) have been made to incorporate a *crowdsourcing* framework into the ER process. The underlying premise of such attempts is that some ER tasks (*e.g.*, Example 1) are inherently challenging for machines to perform well (using any contemporary state-of-the-art techniques), though much easier for humans. Therefore, the new methods first identify the subset of ER tasks that are challenging for machines, and have them solved through human workers in crowdsourcing marketplaces such as Amazon Mechanical Turk (AMT)[1] or CrowdFlower[2].

Leveraging on these work, this paper investigates a core technique, called *blocking* in the ER literature that reduces the quadratic nature of all pair-wise comparisons of records. By grouping records that are more likely to match into the same "block" and run pairwise comparisons only within blocks, the time of the ER process on n records would be reduced from $O(n^2)$ to $O(n + k\tilde{b}^2)$, where k is the number of blocks and $\tilde{b}$ is the average number of records in a block. Because k and $\tilde{b}$ are often much smaller than n, this adoption of blocking can decrease the running time of the ER process dramatically.

[1] `http://www.mturk.com/`
[2] `http://www.crowdflower.com/`

In particular, when crowdsourcing is involved in the ER process, the number of tasks assigned to human workers, referred to the *Human Intelligence Task* (*HIT*), is proportional to the cost paid to human workers. That is, it is important to design *human-powered ER* methods that achieve high accuracy while keeping the expenses low. With the adoption of blocking, if the number of comparisons is reduced from $O(n^2)$ to $O(n + kb^2)$, it would essentially amount a reduction in HITs, thereby reducing the cost.

In this paper, the objective of the feasibility study is to investigate whether the blocking technique can be successfully incorporated into human-powered ER. Toward this goal, we first propose two human-powered blocking methods. Then, each method is validated in its effectiveness in reducing overall cost through both simulated crowdsourcing and AMT-based experiments in synthetic and real-life datasets, respectively.

To summarize, this paper makes the following main contributions:

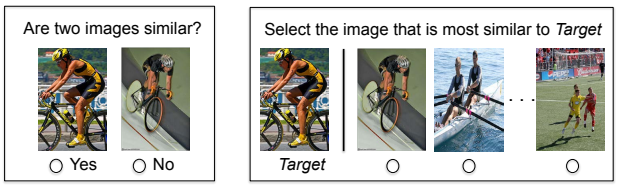
- We identify fundamental human-powered operations that are needed to perform human-powered ER tasks.
- We develop two methods for human-powered blocking building upon the fundamental human-powered operations.
- We validate that our two proposed human-powered blocking methods significantly reduce the cost of human-powered ER while maintaining reasonable ER accuracy.

This paper is organized as follows. In Section 2, we overview our research background and related work. In Section 3, we illustrate fundamental human-powered operations. In Section 4 two variations of human-powered blocking are proposed using the fundamental operations. In Section 5, we report and analyze the experimental results of our proposed human-powered ER blocking methods. In Section 6 we finally conclude our work.

2. BACKGROUND

In recent years, crowdsourcing has developed rapidly and has made tremendous contributions to the database community [13], data mining algorithms [1], and data integration techniques [4, 10, 14, 20]. In addition, crowdsourcing is newly applied to entity resolution. Wang et al. [15] first incorporated crowdsourcing in ER. To minimize the number of crowd-labeled pairs, a threshold of the machine-based similarity between records [15] and transitive relations [16, 17] are applied. Moreover, more sophisticated algorithms are extended for ER such as approximation approaches [15] and probabilistic networks and machine learning [7, 17]. In general, most work uses crowd-human computation by crowdsourcing only in parts of the ER workflow, (*i.e.*, a matcher), but it needs developers in the remaining parts. Corleone [7] first introduced the notion of hands-off crowdsourcing, where crowdsourcing rather than machine-based techniques is used in the entire workflow of ER. Similarly, in this paper, we also take advantage of crowdsourcing in the multiple steps of ER.

As studied in existing work, the need for ER tasks is growing rapidly. Becuase the real-life datasets are usually large, ER tasks are costly. The common approach to enhancing the performance of ER is to reduce the number of pairs to be matched in the *blocking* step, which is the first step of the ER workflow. Many studies [2, 3, 18] have demonstrated the importance of blocking, and applied it in ER. Specifically, an entire dataset is first partitioned into smaller multiple partitions (so-called blocks) with similar records by blocking. In this case, matching is only used within the same block, and pair-wise comparisons are made on a smaller number of candidate records. However, these works on large-scale ER [2, 3, 18] do not employ *crowdsourcing* in the blocking step in ER.



(a) hp_match (b) hp_most_similar

Figure 2: Basic human-powered operations as HITs.

So far, we found that Corleone [7] first attempted the combination of crowdsourcing and blocking in a naïve manner. Similarly, crowd clustering was proposed for leveraging crowdsourcing to assess the similarity of records [8, 19]. In the clustering problem, the selection of centroids is a key part to perform clustering algorithms. To address this problem, the crowd-median algorithm [9] developed the crowdsourced median selection, and applied it to the k-means clustering algorithm (including assign, update steps). Nevertheless, there is an urgent need for more advances in crowdsourcing blocking.

In this paper, we study how to extend crowdsourcing to the blocking methods in ER. Toward this goal, we first introduce basic human-powered operations used in the blocking methods. We then develop two blocking methods: (1) an extension of the crowd-median method [9] for blocking and (2) a *hierarchical blocking* method. Different from other crowdsourced techniques, our blocking methods for ER are built upon fundamental operations, and therefore are intuitive and easy to implement.

3. HUMAN-POWERED OPERATIONS AND COMPONENTS

In this section, we first introduce two fundamental human-powered operations. Using these operations, questions are asked to human workers and their answers are used in the human-powered ER. We then describe the human-powered components that are composed of the fundamental operations, which are used for our proposed human-powered blocking methods. Specifically, we first explain the following two human-powered operations:

- hp_match(r, r'): This operation presents two records, r and r' to humans, as illustrated in Figure 2(a). It asks human workers whether two records are matching or not. If workers answer "yes", two records are matched. Otherwise, they are unmatched. As mention in Example 1, the meaning of matching two records can be ambiguous. Therefore, the definition of matching should be mentioned precisely in this operation.
- hp_most_similar(r_t, C): This operation asks workers to choose a single record that is most similar (per whatever given definition) to the target record r_t among a set of candidates C, as illustrated in Figure 2(b).

Algorithm 1 FindCentroids

Input: D: a set of records
Output: C: a set of block centroids

1: Shuffle D
2: $C \leftarrow \{D[0]\}$ ▷ Set the first centroid
3: **for each** $r \in D \setminus C$ **do**
4: **if** r is dissimilar to all current centroids $c \in C$ **then**
5: $C \leftarrow C \cup \{r\}$ ▷ Add a new centroid
6: **end if**
7: **end for**
8: **return** C

Algorithm 2 Assign

Input: D: a set of records
C: a set of block centroids
Output: B: a set of blocks of records

1: **for each** centroid $c_i \in C$ **do**
2: $B_i \leftarrow \{c_i\}$
3: **end for**
4: $B \leftarrow \{B_1, B_2, \ldots, B_k\}$ ▷ Initialize blocks
5: **for each** record $r \in D \setminus C$ **do**
6: c_i = hp_most_similar(r, C)
7: $B_i \leftarrow B_i \cup \{r\}$
8: **end for**
9: **return** B

Algorithm 3 PairwiseMatch

Input: D: a set of records
Output: P: a set of matching record pairs

1: $P \leftarrow \emptyset$ ▷ Initialization
2: **for each** $r \in D$ **do**
3: **for each** $r' \in D \setminus \{r\}$ **do**
4: **if** hp_match(r, r') **then**
5: $P \leftarrow P \cup \{(r, r')\}$
6: **end if**
7: **end for**
8: **end for**
9: **return** P

We next describe key components used in human-powered blocking. Human-powered blocking groups a set of records, via human workers, into multiple blocks so that matching records are most likely to be clustered into the same block. To perform this process, our human-powered blocking consists of three components: **FindCentroids**, **Assign** and **PairwiseMatch**.

We explain how to execute each component using fundamental operations as follows.

1. **FindCentroids**: Let human workers select a set of most representative records, called *centroids*, where the number of centroids can be explicitly specified or set arbitrarily. Algorithm 1 illustrates the overall procedure of **FindCentroids**, which is based on the aforementioned human-powered operation, hp_match(r, c), for all existing centroids $c \in C$ (see line 4). That is, we scan over the dataset D, compare every record $r \in D$ with the current block centroids founded. If r is different with respect to every block centroid, then r is a newly founded block centroid added to the set of block centroids.

2. **Assign**: Let human workers assign each remaining record to the most similar centroid among K centroids (thus forming K blocks). Algorithm 2 illustrates the implementation of the **Assign** component based on hp_most_similar(r, C), that determines to which block a current record r should be assigned to.

3. **PairwiseMatch**: For every pair of records within each block, let human workers determine whether or not two records are matching. Algorithm 3 illustrates the implementation of the **PairwiseMatch** component that again utilizes hp_match(r, r') for every pair (r, r') in a block.

4. HUMAN-POWERED BLOCKING

In this section, we present two human-powered blocking methods using the three components introduced in Section 3. First, Algorithm 4 illustrates the conceptual workflow of the blocking-based ER process. Within each block, we employ pair-wise comparisons. As the possible implementation of **Blocking**() module in Algorithm 4 (line 2), we propose two variations: (1) a flat blocking with crowd-median method in [9], and (2) a recursive hierarchical blocking.

4.1 Median-based Human-Powered Blocking

Heikinheimo and Ukkonen [9] proposed a crowd-based method to identify the median of a collection via asking a sample of triplet questions (*i.e.*, "Out of three items pick an outlier that appears to be different from the two others") to human workers, and then demonstrated its usage by simulating the k-means clustering algorithm. The main idea is abstracted as **UpdateCentroids**(B) in Algorithm 5, where B is a set of blocks. Because centroids directly affects the quality of clustering, we optimizes the centroid selection by finding the median of each individual cluster ($B_i \in B$) that minimizes the sum of the distance to every other record in the cluster, respectively. The distance can be implicitly induced by the workers through making decisions on the similarity between records.

Specifically, **UpdateCentroids**(B) takes in a set of blocks as input. Within each block, it first computes a penalty score for each record, defined as the number of times the record is chosen to be "different" in triplets (a set of three records chosen from the block) by the crowd. It then returns the record with the lowest penalty score, which is the block centroid. To reduce the number of HITs, the scheme in [9] also allows the inputs of two sampling parameters, the number of pruning cycles (L) and the number of passes (H) over the database to obtain sampling of triplets. Here, for the sake of simpler presentation, we omit such inputs in Algorithm 5. We can refer to more detailed implementation for **UpdateCentroids**(B) in [9].

Algorithm 5 illustrates the overall procedure of median-based blocking method. This blocking methods essentially starts with the random centroid-based initial blocking (using **FindCentroids** and **Assign**) and subsequently updates on the quality of blocking via the crowd-based median method (using **UpdateCentroids**). **UpdateCentroids** and **Assign** are performed recursively until the number of iterations reaches the maximum number or the algorithm converges when the assignments no longer change.

4.2 Hierarchical Human-Powered Blocking

Algorithm 4 Entity Resolution (ER)

Input: D: a set of records
Output: P: a set of matching record pairs

1: $P \leftarrow \emptyset$ ▷ Initialization
2: $B \leftarrow$ **Blocking**(D) ▷ Create blocks
3: **for each** block $B_i \in B$ **do** ▷ matching within each block
4: $P \leftarrow P \cup$ **PairwiseMatch**(B_i)
5: **end for**
6: **return** P

Algorithm 5 Median-based Human-Powered Blocking

Input: D: a set of records
$maxIter$: the # of maximum iterations
Output: B: a set of blocks of records

1: $iter \leftarrow 1$
2: $C \leftarrow$ **FindCentroids**(D) ▷ Initial centroids
3: $B \leftarrow$ **Assign**(D, C)
4: $iter \leftarrow iter + 1$
5: $changed \leftarrow true$
6: **while** $iter \leq maxIter$ and $changed = true$ **do**
7: $oldB \leftarrow B$ ▷ existing blocks
8: $C \leftarrow$ **UpdateCentroids**(B) ▷ updated centroids
9: $B \leftarrow$ **Assign**(D, C)
10: **if** $oldB = B$ **then**
11: $changed \leftarrow false$
12: **end if**
13: $iter \leftarrow iter + 1$
14: **end while**
15: **return** B

This method incorporates the hierarchical blocking idea to reduce the number of comparisons and improve the accuracy. To improve the overall accuracy, the algorithm runs multiple iterations, mitigating the randomness effect of initial centroids. In each iteration, the algorithm builds a K-ary tree of blocks in a top-down fashion outlined in the **Hierarchy** procedure (Lines 9-23) in Algorithm 6. Splits are performed recursively by applying the human-powered **FindCentroids** and **Assign** components when moving down the hierarchical tree. The final blocks are the union of blocks through all iterations (Line 4 in Algorithm 6). In the end, pair-wise matching will be implemented within each final block, and duplicated matching will be avoided.

One issue to address is deciding when to stop the splitting of blocks. In this paper, we experimented the following two heuristics:

1. the number of current blocks exceeds the maximum number of blocks, and
2. the number of HITs due to further blocking exceeds that from direct matching. (abstracted as $split\_costs\_more$ in Line 14 in Algorithm 6.)

Algorithm 6 illustrates the overall procedure of hierarchical blocking method. The blocking is recursively partitioned and two splitting criterion are used in line 14. Direct matching involves pairwise comparisons, therefore needs $n * (n-1)/2$ HITs, where n is the current size of the partition. It is not easy to estimate the number of HITs needed for further blocking and matching within blocks. However, we can estimate the minimum value that will be explained in Section 5.

Algorithm 6 Hierarchical Human-Powered Blocking

Input: D: a set of records
S: a block-size threshold
K: a K-ary tree
$maxIter$: the # of maximum iterations
Output: B: a set of blocks of records

1: **procedure** HIERARCHICALBLOCKING$(D, S, K, maxIter)$
2: $B \leftarrow \emptyset$, $iter \leftarrow 1$ ▷ Initialization
3: **while** $iter \leq maxIter$ **do**
4: $B \leftarrow B \cup$ **Hierarchy**(D, S, K)
5: $iter \leftarrow iter + 1$
6: **end while**
7: **return** B
8: **end procedure**
9: **procedure HIERARCHY**(D, S, K)
10: $B \leftarrow \emptyset$
11: **if** $|D| > S$ **then**
12: $C \leftarrow$ **FindCentroids**(D, K)
13: $IB \leftarrow$ **Assign**(D, C) ▷ IB: Intermediate blocks
14: **for each** $IB_i \in IB$ **do**
15: **if** $|IB_i| > S$ && $split\_costs\_more$ **then**
16: **Hierarchy**(IB_i, S, K)
17: **else**
18: $B_i \leftarrow IB_i$
19: $B \leftarrow B \cup \{B_i\}$
20: **end if**
21: **end for**
22: **else**
23: $B \leftarrow \{D\}$
24: **end if**
25: **return** B
26: **end procedure**

5. EXPERIMENTS

In this section, we evaluate our proposed methods in both synthetic and real-life datasets. In particular, we employ two different HIT designs, *binary* and *n-ary* HIT designs, utilized for the human-powered operations. Since displaying more records can provide crowd workers with more information as discussed in [12], we empirically validate the differences between the two HIT designs in terms of accuracy and the number of HITs. Specifically, we describe the two HIT designs for **FindCentroids** and **Assign** components. As only one pair of records is displayed at a time for the **PairwiseMatching** component, it is simply represented by the binary HIT.

- **binary HIT**: This only processes two images at a time. In order to generate K block centroids using **FindCentroids**, it requires us to generate at least $1 + 2 + \cdots + (K-1) = K \times (K-1)/2$ HITs. In addition, in order to assign a non-centroid record into one of K blocks using **Assign**, we need $(|D| - K) \times (K-1)$ HITs for $(|D| - K)$ non-centroid records, where $|D|$ is the number of records in D.

- **n-ary HIT**: Suppose that all centroids can be displayed to the workers at once. In order to find K block centroids using **FindCentroids**, it thus requires us to generate at least $(K-1) * 1 = K - 1$ HITs, *i.e.*, each n-ary HIT generates a new centroid in the best case. In addition, in order to assign a non-centroid record into one of K blocks using **Assign**, we need $(|D| - K)$ HITs. Compared to the binary HIT, n-

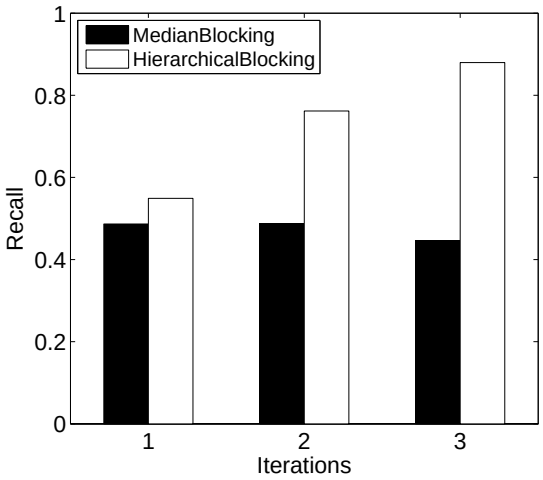


Figure 3: Recall comparison between the median-based blocking and hierarchical blocking over various iterations.

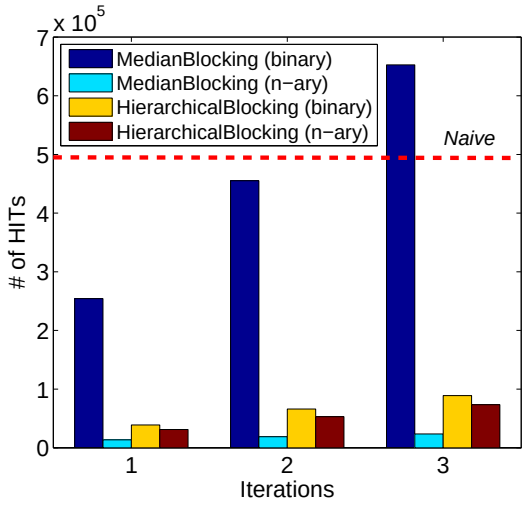


Figure 4: The number of HITs for median-based blocking and hierarchical blocking over various iterations. $Naive$ indicates the baseline method by all pair-wise matchings.

ary HIT can significantly reduce the number of HITs in both **FindCentroids** and **Assign** components.

5.1 Evaluation on Synthetic Data

In order to simulate crowdsourcing, we randomly generate 1000 two-dimensional points consisting of 10 clusters[3] by Weka[4]. To measure the similarity between two points p and q, Euclidean distance $d(p, q)$ is used. Depending on a distance threshold δ, a pair of points can be matched or non-matched. Specifically, if $d(p, q) \leq \delta$, points are matched. Otherwise, they are non-matched. We use the same distance threshold to determine whether two points are similar or not. The distance comparison can be used to simulate the fundamental human-powered operations, hp_match and hp_most_similar. In our simulation, we set δ as 1.41, where 3% of point pairs (*i.e.*, 14,985 out of 499,500 pairs) are matched.

[3] `www.personal.psu.edu/~wul135/CrowdSens14/1k`
[4] `www.cs.waikato.ac.nz/ml/weka`

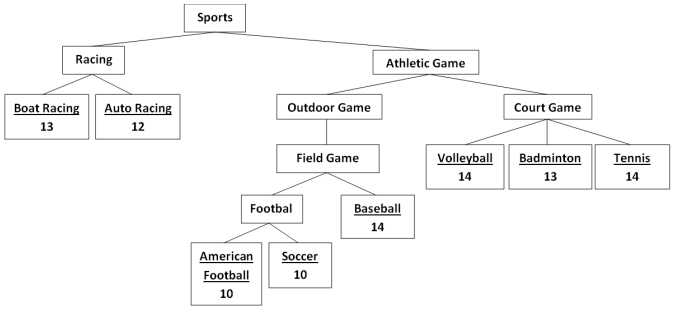


Figure 5: Hierarchy of the images dataset. The number in the leaf node indicates the number of images in the ground truth.

We next explain the detailed settings of our proposed blocking methods: median-based blocking and hierarchical blocking. First, in the median-based blocking method, we use a sampling method proposed in [9]. By default, two parameters for sampling are set as $L = 1$ and $H = 5$ to obtain sampling of triplets, where L and H denote the number of pruning cycles and the number of passes respectively. Next, the hierarchical blocking has two parameters , *i.e.*, the block size threshold S and the arity of the tree K. By default, we set $S = 100$ and $K = 5$. For simplicity, we assume that human workers always give the correct answers for given HITs. In this case, the outcomes for all human-powered operations (*i.e.*, **FindCentroids**, **Assign** and **PairwiseMatch**) are correct for either binary or n-ary HIT designs, thus the precision is always 1.0. In both binary and n-ary designs, we also assume the order of records for every operation are the same as input. In this case, because the same results could be achieved from two different HIT designs, the recall can be equal in both binary and n-ary designs. However, such two assumptions do not hold in our AMT-based experiments.

We next report our experimental results for the number of HITs and recall to quantify the cost and accuracy, respectively. The recall of the baseline method is 1.0 when we do all pair-wise matchings. Figure 3 demonstrates the significant improvement for the recall, as hierarchical blocking is iterated. After 3 iterations, the recall reaches about 0.9. Meanwhile, the recall for median-based block method stays constant regardless of iterations, *i.e.*, it is about 0.5 for all the iterations. This implies that the updated centroids do not help to improve the recall. Next,

Figure 4 shows that our proposed blocking methods can help to reduce the number of HITs, compared our baseline method that generates all pair-wise matching. That is, the naïve method needs 1000*(1000-1)/2 = 499,500 HITs (see the red dash in Figure 4). As expected, we observed that the n-ary HIT design can further reduce the number of HITs than the binary HIT design in both blocking methods. As the number of iterations increases, the number of HITs increases as well. Nevertheless, hierarchical blocking still generates a small number of HITs for both HIT designs. The simulation study shows that, when hierarchical blocking is employed in ER, we can achieve high recall with a low cost for binary and n-ary HIT designs.

5.2 Evaluation on Real-life Data

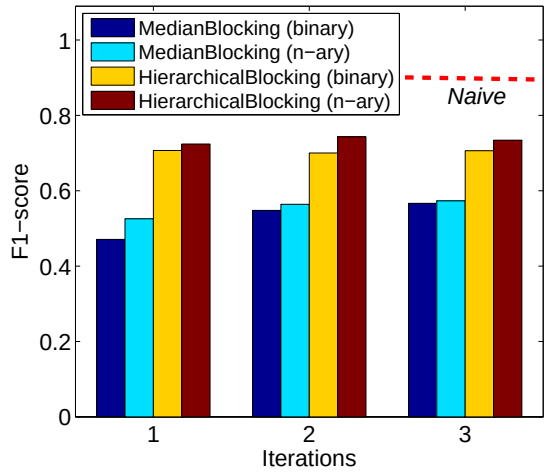


Figure 6: F1-score comparison between median-based blocking and hierarchical blocking over various iterations. $Naive$ **indicates the baseline method by all pair-wise matchings.**

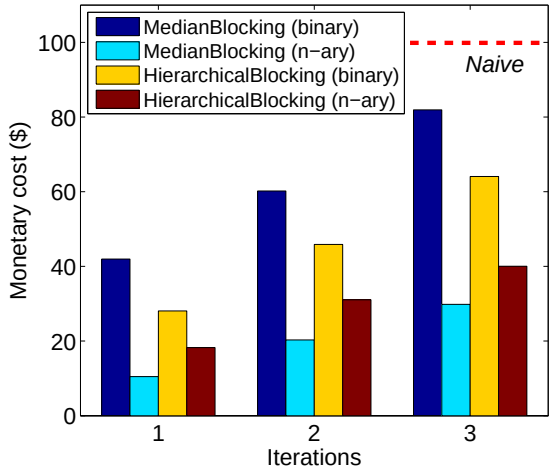


Figure 7: Monetary cost in AMT for median-based blocking and hierarchical blocking over various iterations. $Naive$ **indicates the baseline method by all pair-wise matchings.**

We next evaluate our proposed algorithms in a real-life dataset. We collected 100 images [5] from ImageNet[6]. Specifically, ImageNet is an image database organized according to the WordNet hierarchy, where each node of the hierarchy includes hundreds of thousands of corresponding images. If two images share the same parent node in the hierarchy, they are matched. The hierarchy of our dataset is shown in Figure 5. This data set has 585 pairs of matching records in eight leaf nodes. Therefore, our human-powered ER task is to identify the images in the same categories via crowdsourcing.

We also implemented two human-powered blocking methods with both n-ary and binary HIT designs. For median-based blocking, two parameters for sampling are set as $L = 1$ and $H = 5$. For

[5] `www.personal.psu.edu/~wul135/CrowdSens14/100imgs`
[6] `www.image-net.org/`

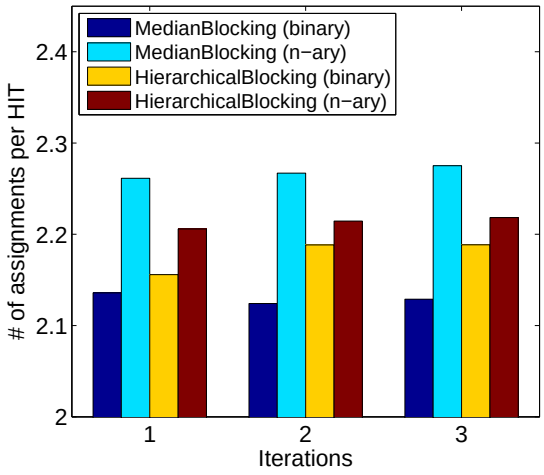


Figure 8: The average number of assignments per HIT in AMT for the median-based blokcing and hierarchical blocking over various iterations

hierarchical blocking, we set $S = 15$ and $K = 4$. In both blocking methods, the maximum number of iterations are 3, *i.e.*, $maxIters = 3$. We created HITs on Amazon Mechanical Turk (AMT), which is the most famous crowdsourcing platform. For each question, we paid \$0.01 to crowd workers. While a binary HIT is assigned to three different workers, an n-ary HIT requires $(n + 1)$ workers. The final answers of HITs are then decided by *majority voting*. To reduce the number of assignments (*i.e.*, the replicated times a HIT is assigned to different workers), we simply optimize HIT assignments for both HIT designs: Initially, each HIT is assigned to two workers. If the answers of the two workers are different, the HITs are iteratively assigned to the other worker until we obtain the answers with majority votes.

However, the F1-score remains low after iterations. The accuracy for ER with hierarchical blocking is not improved much after iterations. However, the F1 score is slightly higher than 0.7 even in the first iteration.

When measuring the accuracy, we adopted *F1-score*, which is widely used in the IR literature. This is computed by the harmonic mean of precision and recall. The closer the F1-score is to 1, the more accurate the ER solution is. In addition, the costs for different iterations have been analyzed. The baseline of accuracy is 0.9, and the cost is \$101.3 when the naïve method is only run one time (see red dashes in Figures 6 and 7).

Figure 6 illustrates that hierarchical blocking can achieve higher accuracy for F1-score than the median-based blocking. As the number of iteration increases, the F1-score for both blocking method becomes slightly higher. For instance, after 3 iterations, the F1-score for hierarchical blocking reaches about 0.74. The F1-score is improved by 5% in the n-ary HIT design and 9% in the binary HIT design. In Figure 6, it is clear that blocking methods with n-ary HIT design help to improve the accuracy. This implies that the n-ary HITs give more information than the binary HITs.

Figure 7 shows the comparisons results between two blocking methods in terms of monetary cost. As observed in the simulation study,

the binary HIT design requires more cost than n-ary HIT design in both blocking. Specifically, the median-based blocking generated 6 blocks by the binary HIT design and 23 blocks by the n-ary HIT design at each iteration. Meanwhile, the hierarchical blocking using both HIT designs generated 10-13 blocks at each iteration. Although in the n-ary HIT design, the cost for hierarchical blocking is higher than that for median-based blocking by around $10, it still much lower than the baseline.

Lastly, Figure 8 depicts an n-ary HIT design requires slightly larger number of assignments per HIT than the binary HIT design. For both HIT designs, less than 2.3 assignments are needed for each HIT. This suggests that, because crowd workers usually return consistent answers, our simple optimization for assignment can help to reduce the overall cost.

6. CONCLUSION AND FUTURE WORK

We have proposed two variations of human-powered blocking for ER, *i.e.*, the median-based and the hierarchical human-powered blocking algorithms, based upon the fundamental human-powered operations that are simple and easy for crowd workers to answer. The feasibility study indicates our proposals can reduce the cost and maintain a relative high accuracy. Our human-powered blocking methods can also be applied to large-scale entity resolution problems (*i.e.*, big data), and extended to other data integration tasks to enhance the overall performance as well as accuracy. As the future work, we will apply machine learning algorithms for constructing blocking functions to reduce uncertainty and hence improve the accuracy while maintaining a low cost.

7. REFERENCES

[1] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart. Crowd mining. In *Proceedings of the 2013 international conference on Management of data*, pages 241–252. ACM, 2013.

[2] P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *Knowledge and Data Engineering, IEEE Transactions on*, 24(9):1537–1555, 2012.

[3] A. Das Sarma, A. Jain, A. Machanavajjhala, and P. Bohannon. An automatic blocking mechanism for large-scale de-duplication tasks. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1055–1064. ACM, 2012.

[4] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st international conference on World Wide Web*, pages 469–478. ACM, 2012.

[5] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):1–16, 2007.

[6] L. Getoor and A. Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.

[7] C. Gokhale, S. Das, A. Doan, J. F. Naughton, R. Rampalli, J. Shavlik, and X. Zhu. Corleone: Hands-off crowdsourcing for entity matching.

[8] R. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. 2011.

[9] H. Heikinheimo and A. Ukkonen. The crowd-median algorithm. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[10] S. R. Jeffery, L. Sun, M. DeLand, N. Pendar, R. Barber, and A. Galdi. Arnold: Declarative crowd-machine data integration. In *CIDR*, 2013.

[11] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197–210, 2010.

[12] A. Marcus, D. Karger, S. Madden, R. Miller, and S. Oh. Counting with the crowd. In *Proceedings of the 39th international conference on Very Large Data Bases*, pages 109–120. VLDB Endowment, 2012.

[13] B. Mozafari, P. Sarkar, M. J. Franklin, M. I. Jordan, and S. Madden. Active learning for crowd-sourced databases. *arXiv preprint arXiv:1209.3686*, 2012.

[14] N. W. Paton and A. A. Fernandes. Crowdsourcing feedback for pay-as-you-go data integration. *DBCrowd 2013*, page 32, 2013.

[15] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012.

[16] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *Proceedings of the 2013 international conference on Management of data*, pages 229–240. ACM, 2013.

[17] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. *Proceedings of the VLDB Endowment*, 6(6):349–360, 2013.

[18] S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 219–232. ACM, 2009.

[19] J. Yi, R. Jin, A. K. Jain, and S. Jain. Crowdclustering with sparse pairwise labels: A matrix completion approach. In *AAAI Workshop on Human Computation*, volume 2, 2012.

[20] C. J. Zhang, L. Chen, H. Jagadish, and C. C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *Proceedings of the VLDB Endowment*, 6(9):757–768, 2013.