Semi-Supervised Consensus Clustering: Reducing Human Effort

Tobias Vogel
Hasso Plattner Institute
Potsdam, Germany
E-Mail: tobias.vogel@hpi.de

Felix Naumann
Hasso Plattner Institute
Potsdam, Germany
E-Mail: felix.naumann@hpi.de

***Abstract*—Machine-based clustering yields fuzzy results. For example, when detecting duplicates in a dataset, different tools might end up with different clusterings. Eventually, a decision needs to be made, defining which records are in the same cluster, i. e., are duplicates. Such a definitive result is called a Consensus Clustering and can be created by evaluating the clustering attempts against each other and only resolving the disagreements by human experts.**

Yet, there can be different consensus clusterings, depending on the choice of disagreements presented to the human expert. In particular, they may require a different number of manual inspections. We present a set of strategies to select the smallest set of manual inspections to arrive at a consensus clustering and evaluate their efficiency on a set of real-world and synthetic datasets.

I. Handling Contradictory Clusterings

Clustering a dataset into partitions is a fundamental problem in computer science. It has applications in diverse areas, such as network analysis, business intelligence, or duplicate detection. There is also a plethora of different clustering algorithms that can be tweaked and tuned in different ways. Consequently, different clusterings may arise for the same dataset, created by different parties.

For example, a company might want to launch an advertisement campaign and needs to separate their customers into groups of different revenue. The different enterprise divisions (sales, marketing, claims, research) have different ideas of how to cluster the set of customers. They do not necessarily need to apply computer-based clustering techniques and may use personal experience, instead. Another example could be an information retrieval challenge run by student teams, which competitively try to cluster a given dataset using their preferred algorithms. For further processing, the different clustering results have to be aligned in a way that further decisions (which advertisement campaign to run/whether or not the students' joint clustering quality was better than last year's) can be made. Such an alignment of multiple clustering results is called *consensus clustering* [1].

A third example is the area of duplicate detection, and is the driver of our research: With a vast number of possible algorithms to efficiently and effectively detect multiple, different representations of same real-world entities, it is easily possible to execute multiple duplicate detection runs, each with different similarity measures and other parameter settings. A consensus clustering can merge the different results, and, if done effectively, minimize the manual effort to resolve the conflicts among the different results. We follow a similar goal by merging multiple results to create a near-gold standard [2].

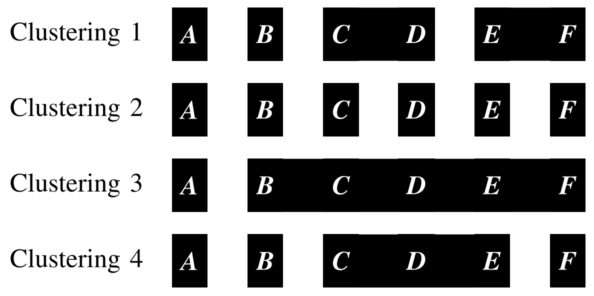


Fig. 1. Different clusterings on the same dataset

Figure 1 shows an example for these different individual clusterings. All four clusterers agree that A is in a cluster of its own (called *singleton*), but have some disagreement on everything else. While B and C are separated by most clusterers, C and D are clustered together by most clusterers. To achieve a consensus clustering, the transitive closure could be applied, ending up in a large cluster $\{B, C, D, E, F\}$. Other automatic clustering approaches might, for example, consider the confidences of the individual clusters and propose $\{A\}$, $\{B\}$, $\{C, D, E\}$, and $\{F\}$ as consensus clustering, when the pair (B, C) has low confidence and (C, D) has high confidence.

We follow a semi-supervised approach to use human experts to decide about the clusters, based on the results of automatic clusterers. In this way, individual clustering decisions are manually reviewed and the resulting consensus clustering has a higher quality than automatic approaches. Because human effort is expensive, we strive for finding strategies that primarily minimize the number of questions to the human expert when creating a consensus clustering. As a secondary goal, we like to have a consensus clustering that resembles the truth as much as possible.

The typical workflow for semi-supervised consensus clustering is depicted in Fig. 2. The set of clusterings $\mathfrak{C}$ is collected from the different clusterers. All disagreed pairs of elements, stored in S, are candidates for a manual inspection. A pair selection component chooses one pair for manual inspection and presents it to the human expert. His/her verdict is then sent to a change propagation component, which in turn updates the clusterings themselves, including inferred changes due to

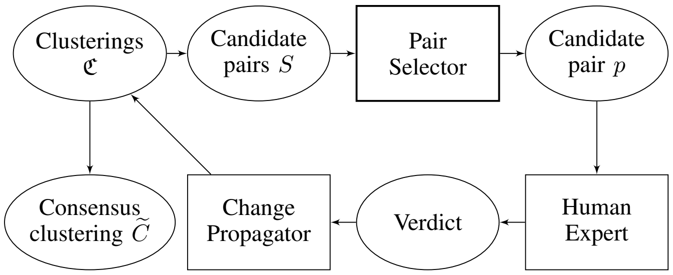


Fig. 2. The workflow for semi-supervised consensus clustering

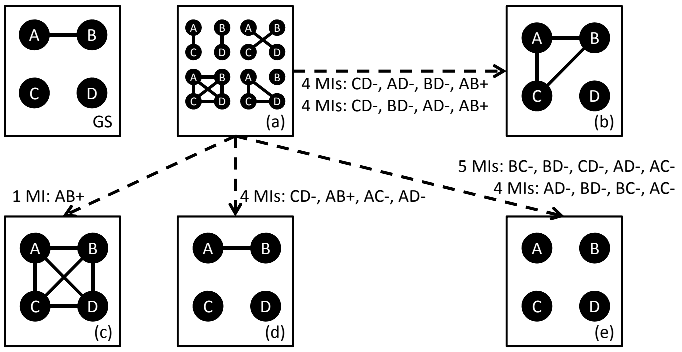

GS
(a)
(b)
(c)
(d)
(e)
4 MIs: CD-, AD-, BD-, AB+
4 MIs: CD-, BD-, AD-, AB+
1 MI: AB+
4 MIs: CD-, AB+, AC-, AD-
5 MIs: BC-, BD-, CD-, AD-, AC-
4 MIs: AD-, BD-, BC-, AC-

Fig. 3. Different consensus clusterings for the same input

transitivity. This loop is repeated until S is empty and thus a consensus clustering is found: the final result.

The selection of pairs for manual inspection and the selection order is crucial for the number of performed manual inspections until a consensus clustering is reached and for the resulting consensus clustering itself. Figure 3 shows an initial set of four clusterings $\mathfrak{C}$ (a) and four consensus clusterings (b) to (e) which mutually differ in the clusters they contain. To come to a consensus clustering, manual inspections are performed, i. e., a human expert is asked for a verdict on the correctness of the queried pair. The expert's verdicts are consistent to a (hidden) gold standard (top left).

There are one or several different sequences of verdicts, leading to each consensus clustering, illustrated with the dashed arrows. The arrow labels are examples of verdict sequences and their lengths. The sequences can differ in both, their lengths and/or the consensus clustering they create. In this example, the following four cases occur.

- It is possible to reach the *same consensus* clustering (b) using different verdict sequences that cause the *same effort*.
- The two *different consensus* clusterings (b) and (e) are the result of same-length sequences representing the *same effort*.
- The *same consensus* clustering (e) can be reached with *different effort* using sequences of 4 or 5 verdicts.
- Finally, *different consensus* clusterings can be reached with *different effort*, for example, by the minimal verdict sequence leading to consensus clustering (c) compared to the much longer sequences leading to consensus clustering (e).

For settings with four clusterings and four elements, clustering configurations can be found that lead to up to seven different consensus clusterings.

In this paper, as the clustering task, we focus on the duplicate detection problem, i. e., identifying multiple representations of same real-world entities (which belong to the same cluster). Each pair of records can either be duplicate or non-duplicate. The notion of *duplicity* is a reflexive, symmetric, and transitive relation. All elements in a cluster are mutually duplicates, elements in different clusters are non-duplicates. The results of this paper are applicable on other clustering tasks as well, as long as they create flat and disjoint clusterings. In summary, we make the following assumptions:

- The input to consensus clustering is a set of clusterings. We treat those clusterings as black boxes. We do not know how they were created.
- Our approach for consensus clustering does not require similarities of the elements nor their attribute values, because we directly start with a set of clusterings.
- The number of clusters is a natural consequence of the algorithm, rather than an a-priori parameter.

Our contributions are in particular:

- A formalization of semi-supervised consensus clustering
- Four different strategies to choose clusters to be examined by a human expert, while minimizing the manual effort
- An extensive evaluation on – and the provisioning of – several artificial and three real-world datasets, clustering results and gold standards.

In Sec. II we show related work concerning consensus clustering. Section III defines *Semi-Supervised Consensus Clustering* formally and Sec. IV presents four strategies for pair selection. For the implementation, several optimizations were developed which are described in Sec. V. The experimental evaluation is presented in Sec. VI. Finally, Sec. VII concludes the paper and discusses future work.

II. Related Work

There is much related work which is presented in the following. Semi-supervised consensus clustering shares characteristics with fuzzy clustering (uncertainty about whether two records belong to the same cluster), with semi-supervised clustering (relying on human judgements to verify or falsify data), graph cutting (a sparse graph structure where some connections are to be refined), and ensemble clustering (create agreed clusterings using (semi-)autonomous means).

Clustering. Aggarwal defines clustering as follows: "Given a set of data points, partition them into a set of groups which are as similar as possible." [3] There is a large variety of clustering methods, flat or hierarchical, distance-based or probabilistic, continuously-spaced or discrete-spaced, unsupervised or semi-supervised, hard or soft. In our case, we already have a set of clusterings and want to solve a meta-clustering problem.

Joint entity resolution [4] by Whang and Garcia-Molina is a duplicate detection approach with the specialty that not one dataset (relation) but several related (via foreign keys) datasets due to a normalization of the relational schema are deduplicated. This brings the possibility to look for duplicates in the different relations in a random access manner. Duplicate decisions might influence the comparison result of records pairs in other relations. To this end, Whang and Garcia-Molina propose a scheduler that creates an execution plan. This is related to the problem of consensus clustering, because human expert verdicts might influence other pairs as well, due to transitivity. However, we do not need to make use of any similarities between the records.

Fuzzyness in clustering. Fuzzy C-means is an extension of the traditional K-means clustering algorithm where an additional membership coefficient is introduced [5]. This coefficient describes how much each element is part of each cluster and thus influences the objective function when re-calculating the centroid positions. Akin to K-means, fuzzy C-means relies on the initialization of centroids and the choice of the number of clusters C. In contrast, we do not to know C and the centroids.

Kaymak and Setner [6] extend fuzzy C-means with an agglomerative approach. They bypass choosing a good value for C by over-estimating the number of clusters and then merging them. Nevertheless, the centroid selection problem remains and it is still not applicable to our problem, because of the lack of pairwise similarities.

(Semi-)supervised clustering. The user can be used to support a clustering process. He can help at various steps, e. g., when selecting an appropriate number of clusters or finding good initial centroids (seeding [7]). Apart from this point-wise supervision, pairwise supervision is performed when individual possible cluster members are compared. Supervising users can merge/split pairs [8] or define must-link/cannot-link constraints [9] on them. Cohn et al. [10] employ user feedback on some pairs to "steer" an already complete clustering into a new direction. Users are invited to criticize arbitrary, self-chosen, suspicious-seeming clustering decisions with a fixed set of statements and these constraints are respected in the next iteration of the clustering. Unfortunately, this approach assumes that re-clustering can be done. Furthermore, our strategies choose which pairs to present to the human expert, rather than the expert himself.

Graph cutting. Another approach on clustering based on a graph is cutting. A cut is a set of removed edges between nodes in a graph such that the resulting graph is separated into k disjoint subgraphs, i. e., clusters. The goal is to find the cut that minimizes the weights of the removed edges. To circumvent trivial, unwanted solutions (separating one node from the remaining nodes), normalized versions are used, the normalized cut [11] or the ratio cut [12]. Shi and Malik propose an adaption for $k > 2$, that relies on k-means and thus, the problem of finding a good k arises, again. Von Luxburg [13] describes several heuristics for coming up with a good k in her section on practical details. In our approach, the number of clusters is a natural consequence of the input data.

Ensemble/Consensus Clustering. Combining a set (ensemble) of clusterings to a joint clustering, all clusterers to some degree agree on, is called Consensus Clustering. There are unsupervised [14] and also semi-supervised approaches, for example, using voting or training a clusterer to perform better upon re-clustering. Stehl and Ghosh [15] use hypergraphs.

However, our approach does not rely on any training, because the clusterings are final and need not to be re-created with improved clusterers. We also do not rely on voting, but instead use a human expert that gives consistent answers. This also means that a single vote is sufficient per pair, reducing the overall manual effort.

III. Formalism for Consensus Clustering

We formally define consensus clustering and the required terms. These formalisms are used in the next section to describe the pair selection strategies which reduce the number of verdicts.

Dataset and set of pairs. A *dataset* D is a set of n records $\{r_1, \dots, r_n\}$. We define the *set of pairs* $P^D \subset D \times D$ as all pairs of records in D where $P^D = \{p_{ij} := \langle r_i, r_j \rangle \mid r_i, r_j \in D, i < j\}$. $|P^D| = \frac{|D| \cdot (|D|-1)}{2}$.

Clusterer, clustering, and clusters. A *clusterer* is a function that partitions D into disjoint subsets, the *clusters* $c_1, \dots, c_k$. We call such a set of clusters a *clustering* $C = \{c_1, \dots, c_k\}$. Each cluster contains at least one record. In a duplicate detection use case, many clusters will contain *exactly* one record, i. e., are singleton clusters.

Since different clusterers use different features of the records or apply different clustering strategies, different clusterings arise that in particular may contain different numbers of clusters. We denote the set of m clusterings $\mathfrak{C} = \{C_1, \dots, C_m\}$.

We further denote the set of pairs contained in a cluster c as $P^c = \{p_{ij} := \langle r_i, r_j \rangle \mid r_i, r_j \in c, i < j\}$, analogously to P^D. For example, a cluster $c = \{r_a, r_b, r_c\}$ would lead to $P^c = \{p_{ab}, p_{ac}, p_{bc}\}$. Similarly, the set of pairs P^C for a clustering C is defined as the union of the set of pairs of its individual clusters: $P^C = \bigcup_{r=1}^{k} P^{c_r}$. Note that in general, $|P^C| \ll |P^D|$, because $|P^C|$ just contains the intra-cluster pairs for clustering C, while P^D contains all possible pairs in the dataset.

Support. The support of a pair is the number of clusters that contain that pair. Pairs with a support of m or 0 are called *agreed*: all or none of the clusterers concordantly declare that pair. We call all other pairs *disagreed* and say that they are in the set of disagreed pairs S which are candidates for a manual inspection.

$$sup(p) = |\{C_w | \, p \in P^{C_w}, C_w \in \mathfrak{C}\}|$$

Manual inspection and clustering modifications. A manual inspection is the review activity by a human expert concerning a pair p. The expert either verifies or falsifies the pair (verification/falsification). The result of a manual inspection is a verdict v: a pair p of which we know whether it is verified or falsified by the human expert. We denote the sequence of all pairs that underwent manual inspection with M. Because a manual inspection directly corresponds to a verdict and vice versa, we use both terms interchangeably.

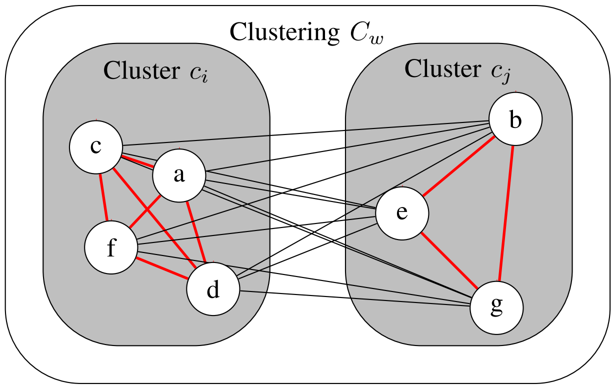


Fig. 4. Example illustration of clusters c_i and c_j of a single clustering C_w with intra-cluster edges (bold) and inter-cluster edges (thin)

Each possible verdict might lead to different modifications of the clusterings. Figure 4 shows two possible clusters c_i and c_j for a clustering C_w. In case of a verification of p_{ab}, the two clusters c_i and c_j containing r_a and r_b, respectively, are merged in C_w and all other clusterings in $\mathfrak{C}$, usually inducing additional pairs due to transitivity.

In case of a falsification of p_{ab}, the cluster containing the inspected pair p_{ab} is split in C_w and all other clusterings in $\mathfrak{C}$, usually discarding additional pairs. Due to the split, r_a and r_b are separated into two fresh clusters c_i and c_j, one containing r_a, the other r_b. All other nodes of the original cluster are placed individually into c_i, c_j, or a third cluster, see below.

Although a merge/split does place the two records from the manual inspection into one/separate clusters, the other records do not necessarily share those cluster(s). For example, assume that p_{ae} in Fig. 4 has already been falsified via a previous manual inspection. When merging c_i and c_j due to the manually verified pair p_{ab}, p_{ae} must not be re-merged, because this would contradict the previous manual inspection. Instead, r_e has to go into a third cluster. The same holds analogously for falsifications/splits.

Regardless of whether or not a clustering has been changed due to a manual inspection, we keep track of each state of a clustering. We denote a clustering after the s-th manual inspection as $C_w^{(s)}$ with $C_w^{(0)} = C_w$ for brevity. Thus, we can refer to the original clusterings.

Impact. When applying a verdict of a pair p to a clustering C, this clustering might be modified. We call the modified clustering C'. The impact $imp_C(v)$ of the verdict v of a pair p on a single clustering C is the number of new or discarded pairs in $P^{C'}$.

$$imp_C(v) = \Big| |P^C| - |P^{C'}| \Big|$$

In the example in Fig. 4 all thin edges represent the new/discarded pairs in case of a verification/falsification, i. e., a merge of/split into clusters c_i and c_j, respectively.

In two special cases, the impact is zero and nothing happens: If the manually inspected pair p was already "known" to the clusterer ($i = j$, $p \in P^C$) and the pair is verified; or the manually inspected pair has never been declared by the clusterer ($i \neq j$, $p \notin P^C$) and the pair is falsified.

If we want to estimate the impact for a *pair*, rather than a *verdict*, we cannot tell the exact impact in advance, because we do not know the expert's verdict. Instead, we simulate the modification of C with a positive verdict (v^+) verifying p and with a negative verdict (v^-) falsifying p and define the impact as being the average of the respective impacts.

$$imp_C(p) = \frac{imp_C(v^+) + imp_C(v^-)}{2}$$

The overall impact $imp_{\mathfrak{C}}(v)$ on all clusterings for a verdict v is simply the sum of all impacts on the individual clusterings. With the sum, we might add the same pair several times. This is intended, because a verdict may have the same effect for several clusterings. The overall impact for a pair is calculated analogously.

$$imp_{\mathfrak{C}}(v) = \sum_{w=1}^{m} imp_{C_w}(v)$$

Table I shows the different impacts on a set of clusterings. For example, let p_{ad} be selected for manual inspection. In case of a verification (a merge), p_{ad} connects the two clusters of C_1 (adding 4 pairs) and creates a triangle for C_3, inducing the two new pairs p_{ab} and p_{bd}. C_3 then resembles C_2, which would not be affected by any clustering modification. The individual impacts are noted in the fourth column. Vice versa, in case of a falsification (a split), C_1 and C_3 are not influenced and C_2 loses two pairs: p_{ad} and either of p_{ab} or p_{bd} (assuming no previous manual inspections). These impacts (0, 2, and 0 for C_1, C_2, and C_3) are shown in the fifth column. In total, the impact $imp_{\mathfrak{C}}(p_{ad})$ is either 6 or 2 and averages in 4. p_{ac} and p_{bc} are no candidates, because all clusterers agree on them. As the table indicates, the merge impact is usually larger than the split impact, due to transitivity.

Agreement. While the number of pairs $|P^D|$ (agreed and disagreed) remains constant during consensus finding, the ratio between agreed and disagreed pairs may change with each manual inspection. Eventually, the number of agreed pairs reaches $|P^D|$, a consensus clustering is found. We call the number of agreed pairs in a set of clusterings $\mathfrak{C}$ the agreement $agree_{\mathfrak{C}}$.

$$agree_{\mathfrak{C}} = |\{p | sup(p) \in \{0, m\}, p \in P^D\}|$$

The agreement of a pair can be calculated by simulating $\mathfrak{C}$ after a verification ($\mathfrak{C}^+$)/falsification ($\mathfrak{C}^-$) of that pair and calculating the average of the agreements on the modified clusterings.

$$agree_{\mathfrak{C}}(p) = \frac{agree_{\mathfrak{C}^+} + agree_{\mathfrak{C}^-}}{2}$$

Table II shows the average agreements for the same example as in Tab. I. There are four records and therefore six pairs. Initially, two pairs are agreed (p_{ac} and p_{bc}), the other pairs are disagreed. For example, let p_{ab} undergo a manual inspection. In case of a verification, p_{ab} achieves a support of m ($= 3$) and p_{ac} and p_{bc} keep their support of 0, summing up to an agreement of 3. The other three pairs stay disagreed. In case of a falsification, p_{ab} is not disagreed anymore as well as p_{ad}. The formerly agreed pairs p_{ac} and p_{bc} are not changed, yielding an agreement of 4. As a result, the average disagreement is 3.5.

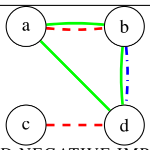
			Impact for		
			Merge	Split	
	Pair	Candidate?	$C_1^+ + C_2^+ + C_3^+$	$C_1^- + C_2^- + C_3^-$	Average Impact
	ab	Candidate!	0+0+2 = 2	1+2+0 = 3	2.5
	ac		4+3+1 = 8	0+0+0 = 0	4
	ad	Candidate!	4+0+2 = 6	0+2+0 = 2	4
	bc		4+3+2 = 9	0+0+0 = 0	4.5
	bd	Candidate!	4+0+0 = 4	0+2+1 = 3	3.5
	cd	Candidate!	0+3+2 = 5	1+0+0 = 1	3

TABLE I. POSITIVE AND NEGATIVE IMPACTS FOR CLUSTERING C_1 (DASHED), CLUSTERING C_2 (SOLID), AND CLUSTERING C_3 (DASH-DOTTED)

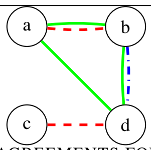
			Agreement for		
	Pair	Candidate?	$\mathfrak{C}^+$	$\mathfrak{C}^-$	Average agreement
	ab	Candidate!	3	4	3.5
	ac		2	2	2
	ad	Candidate!	3	3	3
	bc		3	2	2.5
	bd	Candidate!	1	4	2.5
	cd	Candidate!	1	3	2

TABLE II. AVERAGE AGREEMENTS FOR DIFFERENT PAIRS BASED ON A SET OF CLUSTERINGS WITH AN AGREEMENT OF 2

Clustering difference. We can calculate the difference between two clusterings using the Generalized Merge Distance (GMD) [16]. The GMD is a generalization of different measures (e. g., pairwise precision, pairwise recall, and normalized mutual information) and works similar to the string edit distance, but on clusterings. Instead of character deletions, replacements, and insertions, clusters can be merged and split. The costs for merging or splitting two clusters with sizes $x = |c_i|$ and $y = |c_j|$ are given by two customizable functions $f_m(x, y)$ and $f_s(x, y)$. For the semi-supervised approach intended in this paper, we chose $f_m(x, y) = 1$ and $f_s(x, y) = x \cdot y$, as suggested by Menestrina et al. [16].

Consensus clustering. In the course of manipulating the individual clusterings as a consequence of the manual inspections, they all converge towards the same clustering which is called consensus clustering $\widetilde{C} = C_1^{(|M|)} = \ldots = C_m^{(|M|)}$.

We can now define the overall goal: Given a set of clusterings $\mathfrak{C}$, find a minimal sequence M of manual inspections to modify the clusterings in $\mathfrak{C}$ to achieve a consensus clustering.

Note that the achieved consensus clustering is not necessarily the same clustering that would be generated when manually inspecting all pairs in P^D.

IV. Pair Selection Strategies

The crucial step in getting a quick converging towards the consensus clustering is in having a good selection strategy for the pairs that are presented to the human expert. We propose four different pair selection strategies based on the measures of the previous section.

They all condense to the application of different precedence functions f that assign a precedence value to each candidate pair in S. The pair with the highest precedence value is chosen for the next manual inspection. In case of a tie, we chose a candidate arbitrarily. We have tried different tie resolution strategies, but without significant improvements. Depending on the pair selection strategy, this precedence value is updated after each manual inspection.

MaxSupport. The MaxSupport pair selection strategy prefers pairs that already have a high support, hoping for a verification. The rationale is that highly supported pairs are more likely to be in the same cluster and they also tend to have duplicate pairs in their neighborhood, that are in the impact of the manual inspection and thus do not need to be manually inspected separately. Analogously, pairs with very low support have a good indication for falsification, but as this does not induce pairs, we target highly supported pairs.

$$f_{MaxSupport}(p) = sup(p)$$

MaxAverageImpact. The MaxAverageImpact pair selection strategy prefers pairs with large average impacts, i. e., pairs potentially merging/splitting large or many clusters. In Tab. I, the candidate pair with the largest average impact is p_{ad}.

$$f_{MaxAverageImpact}(p) = imp_{\mathfrak{C}}(p)$$

MaxAverageAgree. The MaxAverageAgree pair selection strategy prefers pairs with a large average number of agreements. In Tab. II, the candidate pair with the highest average agree is p_{ab}.

This is different from MaxSupport because MaxSupport aims for increasing the support, while max-average-agree aims for setting the support of a maximal set of pairs to m or 0, neglecting pairs whose support is only slightly changed and leaving them disagreed.

$$f_{MaxAverageAgree}(p) = agree_{\mathfrak{C}}(p)$$

Random. The Random pair selection strategy serves as a baseline of the other strategies for evaluation. It randomly selects a pair for manual inspection from the set of candidates.

V. Pruning and Splitting

For speedup, we made design decisions that we briefly want to mention.

It is sufficient (and more efficient) to work on individual connected components, instead of the graph as a whole. A connected component contains all pairs between which there is an arbitrary path, ignoring the lineage (i. e., the respective clusterings) of the parts of such a path. With this divide and conquer strategy, the number of candidate pairs to be sorted is heavily reduced and the strategies have to sort much fewer elements.

Proof: The isolation between two connected components would be violated if a manual inspection between them resulted in a positive verdict, a merge. To initiate this manual inspection, at least one clusterer had to declare this pair. In this case, however, the two connected components had not been separated and would belong together: a contradiction.

Splitting the whole graph into connected components has several advantages. First, the whole consensus finding process can be easily parallelized based on connected components. Second, connected components reveal the potential for purging. Usually, many clusters are singletons, especially singletons in all clusterings. Consequently, there are also many connected components containing just a single record. These connected components can be discarded as well as any other connected component that does not contain any disagreed pairs.

Cluster splitting is prone to ties. When there is a choice on how to perform a cluster split (and no alternative contradicts any previous verdicts), we have a tie. We try to resolve this tie by choosing the alternative that is most frequent regarding the other clusterings that have already been modified or that were not affected by the verdict. Ties that still remain are resolved arbitrarily. With that tie resolution heuristics, we promote faster convergence towards a consensus clustering.

VI. Evaluation

In this section we describe the evaluation metrics, the datasets, and finally show how well the different strategies perform. For the experiments, we created an implementation in Java 7. The code and all datasets can be downloaded[1].

A. Evaluation metrics

With the following metrics, we can quantitatively and qualitatively rate a consensus clustering and differentiate between the different pair selection strategies concerning their suitability for finding a consensus clustering with as few manual inspections as possible.

The number of manual inspections $|M|$ issued to the human expert describes the manual effort required to find a consensus for $\mathfrak{C}$.

The average GMD between the original clusterings $C^{(0)}$ and consensus clustering $\widetilde{C}$ indicates how much the clusterings had to be changed to eventually converge. There can be different consensus clusterings that have the same manual effort ($|M|$), but caused cluster operations with a larger impact, i. e., larger GMD. We call this measure C-GMD (for consensus GMD).

We also measure the GMD between the consensus clustering and the gold standard. This indicates the quality of the consensus clustering. We call this measure briefly G-GMD (for gold standard GMD).

B. Datasets

For evaluation, we need a dataset, a gold standard (as a cheaper and faster substitute for a human expert during evaluation), and a set of at least two clusterings. Unfortunately, these three assets together are not available for any real-world dataset we know. Either we have the dataset and a gold standard, but no reasonable clusterings or we have a dataset and a set of sophisticated clusterer results, but no gold standard. Therefore, we generated the missing artifacts, as described below.

We use a modified Chinese Restaurant Process [17] for the generation of gold standards and clusterings, respectively. For each record in the dataset, with a given singleton probability sp, it is placed in a singleton cluster. With the remaining probability $(1 - sp)$, it is placed into an existing cluster. We use exactly this procedure to generate a gold standard.

Where clusterings are not available, we derive them from the (generated or available) gold standard. This is reasonable, because decent clusterings will likely resemble the gold standard to some degree. To create such a derived clustering, we modify the above procedure in the following way: We go over each record in the gold standard and with a (small) cluster change probability ccp, we extract the record from the cluster it is currently in and continue to place it somewhere as described above. We call the singleton probabilities for the gold standard tsp and for the clusterings csp, respectively.

For all our datasets, the actual contents are irrelevant; just the distributions of the records into the clusters are of interest. In the following, we describe the scenarios (combination of dataset, gold standard, and clusterings) that are used for the evaluation.

Freedb. The freedb dataset contains information about 750,000 CDs (artist, title, length, year, genre, etc.). For the clusterings, we have the results of four sophisticated, hand-crafted duplicate detection tools declaring about 127,000 duplicate pairs in connected components up to size 266. The gold standard was created by a crowd using the CrowdFlower platform. The clusterings are sampled from a larger dataset (see freedb-full below) and resemble each other to a high degree.

Freedb-full. The original freedb dataset contains data about several million CD albums. We filtered out all those entries that did not contain readable characters. 1.9 million CDs remain and we have the full results of the duplicate detection tools also on this dataset. They range from 200,000 to 350,000 declared duplicates. As the gold standard, we used the same data as in the freedb scenario above.

NCVoter. This is a voter directory of North Carolina[2]. Ramadan et al. [18] applied extensive duplicate detection techniques on this dataset and provided the results as a gold standard. There are clusters of records up to the size of 5. We used a subset including all 5-, 4-, and some 3-clusters and padded them with the same number of singleton records. Finally, the dataset has a size of about 10,000 records. The cluster change probability ccp is 0.3 and the cluster singleton probability csp is 0.666. We have created 3 clusterings.

Generated. We use a synthetic scenario where the gold standard as well as the clusterings were entirely generated. This generated scenario has the advantage that all parameters can be set deliberately. We vary the respective parameters, but for the default scenario, we chose $tsp = 0.6$, $csp = 0.8$, $ccp = 0.1$, a dataset size ds of 1000 and 3 clusterings ($|\mathfrak{C}|$). We call this scenario Generated-Default. For each of these five different dimensions,

[1]http://tinyurl.com/consensusclustering

[2]http://www.ncsbe.gov/ncsbe/data-statistics

we chose four different values: $tsp \in \{0.5, 0.6, 0.7, 0.8\}$, $csp \in \{0.6, 0.7, 0.8, 0.9\}$, $ccp \in \{0.05, 0.1, 0.15, 0.2\}$, $ds \in \{1,000, 10,000, 50,000, 100,000\}$, and $|\mathfrak{C}| \in \{2, 3, 4, 7\}$. We did not explore every possible combination, but ran experiments with the default values and one changed parameter each. This changed parameter determines the scenario's name. Table III gives an overview on the settings for the synthetic scenarios.

Scenario Name	tsp	csp	ccp	ds	$\|\mathfrak{C}\|$
Generated-D	0.6	0.8	0.1	1,000	3
Generated-tsp-0.5	0.5	0.8	0.1	1,000	3
Generated-tsp-0.7	0.7	0.8	0.1	1,000	3
Generated-tsp-0.8	0.8	0.8	0.1	1,000	3
Generated-csp-0.6	0.6	0.6	0.1	1,000	3
Generated-csp-0.7	0.6	0.7	0.1	1,000	3
Generated-csp-0.9	0.6	0.9	0.1	1,000	3
Generated-ccp-0.05	0.6	0.8	0.05	1,000	3
Generated-ccp-0.15	0.6	0.8	0.15	1,000	3
Generated-ccp-0.2	0.6	0.8	0.2	1,000	3
Generated-ds-10,000	0.6	0.8	0.1	10,000	3
Generated-ds-50,000	0.6	0.8	0.1	50,000	3
Generated-ds-100,000	0.6	0.8	0.1	100,000	3
Generated-$\|\mathfrak{C}\|$-2	0.6	0.8	0.1	1,000	2
Generated-$\|\mathfrak{C}\|$-4	0.6	0.8	0.1	1,000	4
Generated-$\|\mathfrak{C}\|$-7	0.6	0.8	0.1	1,000	7

TABLE III. PROPERTIES OF THE DIFFERENT SYNTHETIC SCENARIOS

To overcome poor random choices when generating the assets, we always state average values over 5 runs. However, the freedb scenarios just had a single run, because there was no random data generation involved.

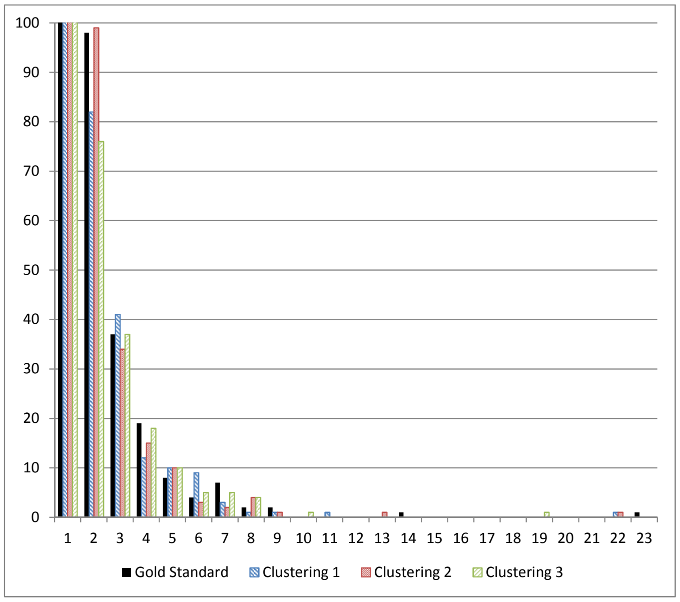


Fig. 5. Histogram of cluster sizes for the default scenario (first instance), cut off at 100, first bars reach 600

For the default generated scenario, we give an overview of the cluster size distribution. Figure 5 contains the histogram of the cluster sizes for the first run of the default scenario (first row in Tab. III). The first bar, printed solid, is the gold standard, the other three bars are the clusterings. One can see that the clusterings differ from each other but adhere quite closely to the gold standard. As in practice, it is a long-tail distribution: The majority of records resides in singleton clusters (in this case about 600 of the 1,000 records) and there are only very few large clusters. The gold standard contains a cluster of size 23, while the clusterings' largest clusters contain 22 or 19 records, respectively. It is highly likely that those clusters have a high overlap, but the third clustering became more diverged in the derived random generation process.

C. Results for the Generated Scenarios

Table IV shows the results for the consensus finding in the generated default scenario. There are four rows, each for a different pair selection strategy. The respective optimal value is printed in bold face. The numbers of manual inspections are comparable, except for the MaxAverageImpact strategy, which stands out and uses the fewest manual inspections. C-GMD and G-GMD are nearly equal for all strategies. The MaxAverageAgree strategy finds consensus clusterings that reflect both, the gold standards and the original clusterings most. This is remarkable, because MaxAverageImpact uses less manual inspections and still finds consensus clusterings that differ more from each other. Therefore, the intuition leading to the MaxAverageImpact strategy (choosing pairs that have a high (average) impact) seems to hold: the (relatively few) manual inspections actually have a higher impact.

Scenario	Strategy	$\|M\|$	C-GMD	G-GMD
Generated-Default	MaxSupport	197.8	115.0	1.4
Generated-Default	MaxAverageImpact	**185.6**	115.0	1.2
Generated-Default	MaxAverageAgree	194.0	**113.4**	**0.8**
Generated-Default	Random	195.6	116.4	1.2

TABLE IV. KEY FIGURES FOR THE DEFAULT GENERATED SCENARIO

Overall, the manual effort caused by the different strategies is quite similar. Unfortunately, the combinatorial complexity prevents the extensive calculation of optimal (i. e., smallest) M's even for small datasets. We therefore empirically determined an approximate optimum by taking the random pair selection strategy and iterating very many times with different random initializations for this strategy. Again, we did this on several instances of the default scenario.

Max Support	Max Average Impact	Max Average Agree	Random after 1 iteration	Min(Random) after 300,000 iterations
202	191	201	197	190
191	179	187	189	180
180	167	176	179	167
214	203	209	211	202
202	188	197	202	191

TABLE V. RESULTS FOR OPTIMUM ESTIMATION

Table V shows $|M|$ for five different instances of the default scenario. The first four columns describe the manual inspection effort generated by the four strategies. Additionally, in the fifth column, we show $|M|$ for the most successful random strategy among 300,000 iterations, i. e., the random run that produced the smallest number of manual inspections. Note that this is just an approximate minimum: In general, other strategies might find even smaller M's, which is indeed the case for the MaxAverageImpact strategy. It is regularly far away from the other strategies and very similar to the anticipated optimum. In other words: there is not much room for the strategies to differ, but the MaxAverageImpact strategy usually hits the (empirically determined) optimum.

Table VI shows the results of many variations on the default scenario parameters. The table is divided in five groups for the five varied parameters, each group having four different scenarios and each scenario contains four rows for the four pair selection strategies. For each of the four strategies, the respective minimum values are printed in bold face. Each group contains the default scenario, aligned according to its parameter value in the numerical order of the varied parameter.

The table allows several observations. The most relevant insight is that the MaxAverageImpact pair selection strategy always yields the smallest number of manual inspections $|M|$ (except for the experiments which timed out). The order of the other strategies, concerning to $|M|$, changes from scenario to scenario. We therefore calculated the normalized[3] rankings over all 14 different scenarios and calculated the average. This is presented in Tab. VII.

Strategy	$\|M\|$ Rank	C-GMD Rank	G-GMD Rank
MaxSupport	3.68	2.29	2.79
MaxAverageImpact	1.00	2.54	2.82
MaxAverageAgree	2.43	1.89	1.57
Random	2.89	3.00	2.68

TABLE VII. NORMALIZED RANKS OF THE STRATEGIES FOR THE 14 DIFFERENT SCENARIOS

The MaxAverageAgree strategy has an average ranking of 2.43 and is closely followed by the Random strategy with an average ranking of 2.89. Finally, the MaxSupport strategy performs worst and achieves an average ranking of 3.68. This strategy usually results in the largest $|M|$. Only with very large clusters ($csp \in \{0.6, 0.7\}$) or very few clusterers ($\mathfrak{C} = 2$), it does not use the most manual inspections. The reason is, that only in those settings the support has a beneficial influence on the pair selection and consequently the number of manual inspections is reduced.

Concerning C-GMD and G-GMD, there is no strategy that finds consensus clusterings with minimal values for these metrics in *all* scenarios. However, MaxAverageAgree *most frequently* finds the smallest C-GMD (7 times) and G-GMD (12 times). This indicates that MaxAverageAgree tends to produce consensus clusterings of (slightly) higher quality in contrast to MaxAverageImpact that rather finds less expensive consensus clusterings.

We calculated Spearman's rank correlation coefficient for the three metrics for each scenario. $|M|$/C-GMD as well as $|M|$/G-GMD have a coefficient of 0.09 and 0.11, respectively, and are thus not very correlated. This confirms the findings described above: strategies which are good regarding $|M|$ (MaxAverageImpact) do not necessarily perform well for C-GMD or G-GMD and vice versa (MaxAverageAgree). Because the correlation is small, the *bare number* of manual inspections used for a consensus clustering has nearly no influence of the GMDs, but the GMDs depend on the *actually selected* candidate pairs. On the other hand, the correlation coefficient for C-GMD/G-GMD is 0.66, again confirming the observation described in the last paragraph.

For the different varied parameters, the effort grows or decreases monotonously. For example, with rising singleton probabilities, the number of (trivial) singleton clusters increases, and consequently, there is less disagreement among the clusterers and the manual effort decreases. Analogously, with increasing cluster change probability, the clusterings diverge more and more, which leads to larger $|M|$'s. Even a small increase of this probability has a high influence on

[3]Ranking ties are resolved by assigning the average rank for all tied strategies.

Scenario	Strategy	$\|M\|$	C-GMD	G-GMD
Generated-tsp-0.5	MaxSupport	219.2	178.4	2.2
Generated-tsp-0.5	MaxAverageImpact	**203.4**	180.8	2.6
Generated-tsp-0.5	MaxAverageAgree	216.6	**178.2**	**1.2**
Generated-tsp-0.5	Random	216.2	181.6	1.8
Generated-Default	MaxSupport	197.8	115.0	1.4
Generated-Default	MaxAverageImpact	**185.6**	115.0	1.2
Generated-Default	MaxAverageAgree	194.0	**113.4**	**0.8**
Generated-Default	Random	195.6	116.4	1.2
Generated-tsp-0.7	MaxSupport	172.8	**81.4**	1.0
Generated-tsp-0.7	MaxAverageImpact	**166.6**	**81.4**	**0.8**
Generated-tsp-0.7	MaxAverageAgree	170.4	81.8	1.0
Generated-tsp-0.7	Random	171.2	81.8	**0.8**
Generated-tsp-0.8	MaxSupport	131.4	58.2	1.2
Generated-tsp-0.8	MaxAverageImpact	**126.4**	**58.0**	**0.8**
Generated-tsp-0.8	MaxAverageAgree	129.2	**58.0**	**0.8**
Generated-tsp-0.8	Random	130.6	58.2	1.0
Generated-csp-0.6	MaxSupport	243.4	**171.8**	**2.6**
Generated-csp-0.6	MaxAverageImpact	**235.4**	173.0	**2.6**
Generated-csp-0.6	MaxAverageAgree	246.4	173.8	2.8
Generated-csp-0.6	Random	244.4	176.2	3.6
Generated-csp-0.7	MaxSupport	225.4	148.0	3.0
Generated-csp-0.7	MaxAverageImpact	**213.8**	147.4	3.6
Generated-csp-0.7	MaxAverageAgree	224.6	147.4	**2.2**
Generated-csp-0.7	Random	226.6	**145.8**	**2.2**
Generated-Default	MaxSupport	197.8	115.0	1.4
Generated-Default	MaxAverageImpact	**185.6**	115.0	1.2
Generated-Default	MaxAverageAgree	194.0	**113.4**	**0.8**
Generated-Default	Random	195.6	116.4	1.2
Generated-csp-0.9	MaxSupport	181.0	90.4	**0.2**
Generated-csp-0.9	MaxAverageImpact	**165.2**	90.4	0.4
Generated-csp-0.9	MaxAverageAgree	169.6	**90.2**	**0.2**
Generated-csp-0.9	Random	177.0	93.0	0.8
Generated-ccp-0.05	MaxSupport	100.0	**55.4**	**0.0**
Generated-ccp-0.05	MaxAverageImpact	**95.0**	**55.4**	**0.0**
Generated-ccp-0.05	MaxAverageAgree	97.6	**55.4**	**0.0**
Generated-ccp-0.05	Random	99.0	58.0	0.2
Generated-Default	MaxSupport	197.8	115.0	1.4
Generated-Default	MaxAverageImpact	**185.6**	115.0	1.2
Generated-Default	MaxAverageAgree	194.0	**113.4**	**0.8**
Generated-Default	Random	195.6	116.4	1.2
Generated-ccp-0.15	MaxSupport	267.4	167.4	3.0
Generated-ccp-0.15	MaxAverageImpact	**250.2**	**166.0**	3.4
Generated-ccp-0.15	MaxAverageAgree	261.4	167.4	**2.4**
Generated-ccp-0.15	Random	263.4	168.0	3.8
Generated-ccp-0.2	MaxSupport	342.4	**216.4**	11.4
Generated-ccp-0.2	MaxAverageImpact	**324.4**	221.8	12.4
Generated-ccp-0.2	MaxAverageAgree	338.8	219.0	**9.2**
Generated-ccp-0.2	Random	339.4	221.2	**9.2**
Generated-Default	MaxSupport	197.8	115.0	1.4
Generated-Default	MaxAverageImpact	**185.6**	115.0	1.2
Generated-Default	MaxAverageAgree	194.0	**113.4**	**0.8**
Generated-Default	Random	195.6	116.4	1.2
Generated-ds-10000	MaxSupport	1,927.0	1,225.2	18.4
Generated-ds-10000	MaxAverageImpact	**1,803.4**	1,228.6	18.4
Generated-ds-10000	MaxAverageAgree	1,884.2	**1,212.0**	**9.6**
Generated-ds-10000	Random	1,908.2	1,223.0	15.4
Generated-ds-50000	MaxSupport	9,830.4	6,707.2	92.6
Generated-ds-50000	MaxAverageImpact	timeout exceeded ($>$ 6 h)		
Generated-ds-50000	MaxAverageAgree	timeout exceeded ($>$ 6 h)		
Generated-ds-50000	Random	9,725.6	6,788.4	75.8
Generated-ds-100000	MaxSupport	19,692.0	13,410.8	200.0
Generated-ds-100000	MaxAverageImpact	timeout exceeded ($>$ 6 h)		
Generated-ds-100000	MaxAverageAgree	timeout exceeded ($>$ 6 h)		
Generated-ds-100000	Random	19,504.2	13,369.8	157.0
Generated-$\|\mathfrak{C}\|$-2	MaxSupport	124.6	128.4	5.2
Generated-$\|\mathfrak{C}\|$-2	MaxAverageImpact	**121.8**	134.4	8.8
Generated-$\|\mathfrak{C}\|$-2	MaxAverageAgree	125.4	128.2	**4.4**
Generated-$\|\mathfrak{C}\|$-2	Random	124.6	**123.4**	5.2
Generated-Default	MaxSupport	197.8	115.0	1.4
Generated-Default	MaxAverageImpact	**185.6**	115.0	1.2
Generated-Default	MaxAverageAgree	194.0	**113.4**	**0.8**
Generated-Default	Random	195.6	116.4	1.2
Generated-$\|\mathfrak{C}\|$-4	MaxSupport	259.4	**115.8**	0.2
Generated-$\|\mathfrak{C}\|$-4	MaxAverageImpact	**239.6**	117.4	0.6
Generated-$\|\mathfrak{C}\|$-4	MaxAverageAgree	253.4	**115.8**	**0.0**
Generated-$\|\mathfrak{C}\|$-4	Random	255.4	116.8	0.2
Generated-$\|\mathfrak{C}\|$-7	MaxSupport	396.0	115.4	**0.0**
Generated-$\|\mathfrak{C}\|$-7	MaxAverageImpact	**353.8**	**115.2**	**0.0**
Generated-$\|\mathfrak{C}\|$-7	MaxAverageAgree	386.0	115.4	**0.0**
Generated-$\|\mathfrak{C}\|$-7	Random	382.0	115.4	**0.0**

TABLE VI. KEY FIGURES FOR ALL GENERATED SCENARIOS

$|M|$. The increase of the dataset size has the largest impact on $|M|$: the number of manual inspections increases proportional to the increase of the dataset size. For the increase of the number of clusterings $|\mathfrak{C}|$, the result is similar: when doubling the number of clusterings, the number of manual inspections nearly exactly doubles, although under-linearly (Generated-$|\mathfrak{C}|$-2 vs. Generated-$|\mathfrak{C}|$-7).

The G-GMD reacts less predictably on the variations of the parameters. In case of Generated-$|\mathfrak{C}|$-7, there are so many clusterings that eventually the exact gold standard is always reached, even in each of the five runs and for each strategy, although the resulting effort strongly differs between the strategies. For Generated-$|\mathfrak{C}|$-7, MaxSupport uses 10% more manual inspections than MaxAverageImpact.

The MaxAverageImpact strategy and the MaxAverage-Agree strategy are the most advanced pair selection strategies. However, because the target is to get a cheap yet sound consensus clustering, the MaxAverageImpact strategy should be selected. It is both more suitable and more reliable in its performance. The degree of differences between the strategies is quite constant 5% between the baseline (Random pair selection strategy) and the MaxAverageImpact strategy.

The performance differences between the strategies were higher, if we would not count the manual inspections from situations, where there was no choice between different pairs to manually inspect. In these cases, all strategies necessarily performed equally. For example, for Generated-$|\mathfrak{C}|$-7, the Max-AverageImpact strategy yielded 300 manual inspections and the Random strategy 330, a difference of 10%. Because we show the actual count of all manual inspections (which is the actual effort), the differences are a bit leveled out.

For both Generated-ds-50000 and Generated-ds-100000, the MaxAverage strategies exceeded a timeout of 6 hours of runtime. However, because $|M|$ for the two simpler strategies follows a linear scale, we expect the missing results to be in the same region. For more comments on the runtime, see the end of this section.

D. Results for Freedb

The results for the freedb scenario are shown in Tab. VIII. In the freedb scenario, the clusterings are very similar. Therefore, the number of manual inspections is relatively low, compared to the results of generated scenarios of the same size. The manually curated gold standard was created inspecting more than 1,000 pairs manually. With our presented strategies, a quarter of the manual inspections could have been saved.

The freedb-full scenario uses much more diverse clusterings. This is why the number of manual inspections is vastly increased. The gold standard does not have a high influence on $|M|$. While it is small, it can still return verdicts for all the selected candidate pairs (being a falsification in most cases). A larger gold standard would have caused more verifications (and inferred pairs), but the number of manual inspections would not have been much smaller. In fact, the number of manual inspections is much smaller than even the smallest clustering: for a large amount of pairs, there were no disagreements among the clusterings and the corresponding connected components could just be adopted for the consensus clustering.

Scenario	Strategy	$\|M\|$	C-GMD	G-GMD
Freedb	MaxSupport	**783**	**1,614**	**379**
Freedb	MaxAverageImpact	**783**	**1,614**	**379**
Freedb	MaxAverageAgree	**783**	**1,614**	**379**
Freedb	Random	784	**1,614**	**379**
Freedb-full	MaxSupport	125,788	159,630	7,087
Freedb-full	MaxAverageImpact	timeout exceeded ($>$ 16 h)		
Freedb-full	MaxAverageAgree	timeout exceeded ($>$ 16 h)		
Freedb-full	Random	136,970	159,419	7,119

TABLE VIII. FREEDB RESULTS

E. Results for NCVoter

In the NCVoter scenario (Tab. IX), many more manual inspections were performed, but eventually, the situation is similar: MaxAverageImpact outperforms the other strategies, even though the differences are not that large. Compared to the Generated-ds-10000 scenario, the number of manual inspection is similarly high, but the G-GMD is much higher in the NCVoter scenario.

Scenario	Strategy	$\|M\|$	C-GMD	G-GMD
NCVoter	MaxSupport	2,631.6	1,207.2	42.4
NCVoter	MaxAverageImpact	**2,594.8**	**1,206.8**	**37.4**
NCVoter	MaxAverageAgree	2,631.4	1,207.8	42.6
NCVoter	Random	2,626.6	1,207.8	42.6

TABLE IX. NCVOTER RESULTS

In this scenario, a relatively high number of records is distributed over a (larger) number of smaller clusters with a size up to 5, which makes it different from the larger generated scenarios. This difference has caused the consensus clusterings to deviate more from the gold standard.

The MaxAverageImpact strategy causes the least G-GMD, while for the Generated Scenarios, the minimal G-GMD was usually achieved by the MaxAverageAgree strategy.

F. General Remarks

It is difficult to obtain real world data. We therefore amended the available real world data with generated data and also used a variety of completely synthetic scenarios. In all performed experiments, MaxAverageImpact outperformed the other strategies. Depending on the five described parameters, the differences may be smaller or larger. The experiments indicate that the MaxAverageImpact strategy is generally recommendable. We also tried to use the maximum instead of the average for the impact and agreement strategies, but the results were slightly worse.

We do not report on the runtime, because the dominant factor is the human expert. Asking the expert takes much longer than virtually any computation. As shown in the workflow (see Fig. 2), expert inquiries, calls of the pair selector, and propagating changes back to the clusterings are alternating.

Nevertheless, there are differences in the runtimes for the strategies. Unsurprisingly, the Random strategy is the fastest, because it does not do any computation besides shuffling the candidate pairs. For the default scenario, consensus finding took 0.1 seconds with the Random strategy and 0.5 seconds with the MaxSupport strategy. MaxAverageImpact and Max-AverageAgree took 12.9 and 11.0 seconds, respectively (all values are averages over the repetitions). Both MaxAverage strategies are considerably more complex and take longer, because for each pair, both possible verdicts have to be simulated. MaxSupport does not need to simulate anything, but still has to calculate the support which causes a slightly larger duration than the Random strategy. The actual durations depend on the parameters, but the order remains similar.

For example, for Generated-ds-10000 the average durations for consensus finding are 75.6 seconds (MaxSupport), 1,462.2 seconds (MaxAverageImpact), 1,362.8 seconds (MaxAverageAgree), and 10.5 seconds (Random). The runtime grows to the square of the growth of the dataset size which is a general, rather than a strategy-specific problem. With more and more records, the clusters get larger, there are more pairs in the clusters, and finally more candidate pairs to check.

Because the duration grows faster than the savings of manual effort regarding a quick and a sophisticated pair selection strategy, there is a break-even point. Beyond that point, choosing a simpler strategy may cause more manual inspections but a smaller overall runtime. However, this break-even point is hard to reach; a connected component has to be quite large to better choose an arbitrary pair instead of waiting for MaxAverageImpact to select a candidate pair. According to our subjective impression, finding M for connected components started to take a significant amount of time when the connected component contained 50 records or more.

Additionally, the pair selection strategy can be changed in each iteration. In particular, a timeout could be set. When it is expired, the proposed pair of a quicker strategy (executed in parallel) could be used as a best guess for the next manual inspection. Moreover in a real application, this timeout could be defined dynamically by the time the human expert is busy with manual inspections of other connected components.

In our experiments, we did the pair selection in parallel for separate connected components. It should be easy to also parallelize the pair selection within a connected component, causing a higher overall speed-up.

VII. Conclusion

Consensus clustering is one way to negotiate between different, contradicting clusterings. Semi-supervised consensus clustering promises a high quality of the resulting clustering, but has high costs due to expensive human effort. Therefore, it is crucial to identify those disputed candidate pairs that serve rapid convergence. We presented four pair selection strategies and evaluated them on different scenarios.

The MaxAverageImpact strategy regularly outperforms the baseline (Random) strategy and saves up to 10% manual inspections. It is close to an empirically determined optimum. Please note that even a 10% drop in the number of manual inspections is already saving a large amount of money in a real-world setting.

Users might still feel the need for a good trade-off between quality and price. Semi- or fully automated consensus processes can create faster and cheaper consensus clusterings, but when the application needs a high-quality consensus clustering, human experts are indispensable. As the evaluation shows, the consensus clusterings are close to the (latent) gold standards.

The strategies provide some room for improvement. For example, initial clusterings are differently close to the later consensus clustering. The closer they are, the better the clustering must have been in the first place. This observation could be utilized by continuously measuring each clustering's confidence. The confidence should be increased if a manual inspection has no effect on a clustering (i. e., the clusterer was correct) and decreased if the impact is high. Having such a confidence score for each clustering allows a weighted support.

Another direction for future research is the restriction of manual inspections to a fixed budget. Which pairs should be selected first, if not necessarily all manual inspections can be afforded? Further, more records could be presented to the human user, be it to give the user more context or even to obtain several verdicts at once.

References

[1] S. Monti, P. Tamayo, J. Mesirov, and T. Golub, "Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data," *Journal of Machine Learning*, 2003.

[2] T. Vogel, A. Heise, U. Draisbach, D. Lange, and F. Naumann, "Reach for gold: An annealing standard to evaluate duplicate detection results," *Journal of Data and Information Quality (JDIQ)*, 2014.

[3] C. C. Aggarwal, "An introduction to cluster analysis," in *Data Clustering: Algorithms and Applications*, C. C. Aggarwal and C. K. Reddy, Eds. CRC Press, 2013.

[4] S. E. Whang and H. Garcia-Molina, "Joint entity resolution," in *Proceedings of the International Conference on Data Engineering (ICDE)*, 2012.

[5] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 1981.

[6] U. Kaymak and M. Setnes, "Extended fuzzy clustering algorithms," Erasmus University Rotterdam, Tech. Rep., 2000.

[7] S. Basu, A. Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2002.

[8] M.-F. Balcan and A. Blum, "Clustering with interactive feedback," in *Internation Conference on Algorithmic Learning Theory*, 2008.

[9] I. Davidson and S. Basu, "A survey of clustering with instance level constraints," *International Conference on Data Mining*, 2005.

[10] D. Cohn, R. Caruana, and A. Mccallum, "Semi-supervised clustering with user feedback," University of Massachusetts Amherst, Tech. Rep., 2003.

[11] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Transactions on Pattern Analysis and Machine Intelligence*, 1997.

[12] L. W. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering." *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1992.

[13] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, 2007.

[14] D. Greene and P. Cunningham, "A matrix factorization approach for integrating multiple data views," in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. Springer, 2009.

[15] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research (JMLR)*, 2003.

[16] D. Menestrina, S. E. Whang, and H. Garcia-Molina, "Evaluating entity resolution results," *Proceedings of the VLDB Endowment (PVLDB)*, 2010.

[17] D. J. Aldous, "Exchangeability and related topics," in *Lecture Notes in Mathematics*. Springer, 1985.

[18] B. Ramadan, P. Christen, H. Liang, R. W. Gayler, and D. Hawking, "Dynamic similarity-aware inverted indexing for real-time entity resolution," in *PAKDD Workshops*, 2013.