Distrib Parallel Databases (2015) 33:33–67
DOI 10.1007/s10619-014-7160-z

Enabling community-driven information integration through clustering

Khalid Belhajjame · Norman W. Paton · Cornelia Hedeler · Alvaro A. A. Fernandes

Published online: 25 October 2014

Abstract It has become widely recognized that user feedback can play a fundamental role in facilitating information integration tasks, e.g., the construction of integration schema and the specification of schema mappings. While promising, existing proposals make the assumption that the users providing feedback expect the same results from the integration system. In practice, however, different users may anticipate different results, due, e.g., to their preferences or application of interest, in which case the feedback they provide may be conflicting, thereby deteriorating the quality of the services provided by the integration system. In this paper, we present clustering strategies for grouping information integration users into groups of users with similar expectations as to the results delivered by the integration system. As well as grouping information integration users, we show that clustering results can be used as inputs to a wide range of functionalities that are relevant in the context of crowd-driven information integration. Specifically, we show that clustering can be used to identify feedback of relevance to a given user by exploiting the feedback provided by other users in the same cluster. We report on evaluation exercises that assess the effectiveness of the clustering strategies we propose, and showcase the benefits community- and crowd-driven information integration can derive from clustering.

K. Belhajjame (✉)
PSL, Paris Dauphine University, LAMSADE, Paris, France
e-mail: Khalid.Belhajjame@dauphine.fr

N. W. Paton · C. Hedeler · A. A. A. Fernandes
School of Computer Science, University of Manchester, Manchester, UK
e-mail: npaton@manchester.ac.uk

C. Hedeler
e-mail: cornelia.hedeler@cs.man.ac.uk

A. A. A. Fernandes
e-mail: a.fernandes@manchester.ac.uk

Keywords User feedback · Information integration · Clustering · Crowd · Dataspaces

1 Introduction

Information integration is difficult. Essentially, the difficulty lies in devising schema mappings that populate the elements of the integration schema using given data sources. To face this challenge, there has recently been growing interest in guiding the specification of schema mappings using feedback solicited from end users [4,8,9,19,28]. For example, Jeffery et al. [19] developed a decision-theoretic framework for specifying the order in which candidate mappings can be confirmed by soliciting feedback from users, with the objective of providing the *most benefit* to a data integration system.

As existing proposals demonstrate [4,8,9,19,22,28], feedback can play a key role in facilitating the construction of information integration systems. However, a user may need to provide a sizable number of feedback instances before the quality of the services provided by the integration system reaches a satisfactory level. To address the problem of feedback scarcity, a handful of researchers investigated the use of feedback solicited from communities of users [22], or more generally through the crowd [19,22, 31,32]. For example, McCann et al. [22] developed a community-based approach that solicits feedback on matches from the multitude of community members. The idea is to use collectively the feedback provided by the members of given community, thereby ensuring that the quality of the services provided by the information integration system quickly reaches an acceptable level. Wang et al. [31] and Whang et al. [32] explored the use of crowdsourcing techniques for leveraging entity resolution. Specifically, the crowd members are presented with a set of record pairs with the objective of identifying the pairs of records that are semantically equivalent.

While promising, the above proposals make the assumption that the users providing feedback have the same expectations as to the results that should be delivered by the integration system. In practice, however, different users may have different expectations from the integration, in which case the quality of the services provided by the integration system may deteriorate due to conflicts in feedback. By *user expectations*, we mean the results that the user anticipates from the integration system. The difference in expectations between users may be due to differences in their preferences, or to their domain and application of interest.

The difference in users' expectations can be captured by analyzing metadata that describe, e.g., their fields and applications of interest. However, such metadata are not always available, and when they are, they may be misleading. For example, two scientists who work on the same field, e.g., proteomics, may be judged as having the same expectations. Yet, one may be interested in using the integration system to profile proteins functionalities of a given species, whereas the other is interested in identifying protein interactions of a different species. In these cases, user feedback seems to be a better source of information. For example, to deal with the low precision of wrappers of web resources that are generated based on metadata, Crescenzi et al. [10] showed that the quality of the wrappers

FavoriteCity

	name	province	country	User 1	User 2
t_1	Bolzano	South Tyrol	IT	✔	✖
t_2	Cancún	Quintana Roo	MX	✖	✔
t_3	Grenoble	Rhône-Alpes	FR	✔	✖
t_4	Manchester	Greater Manchester	UK	✖	✖
t_5	Melbourne	Victoria	AU	✖	✔
t_6	Salt Lake City	Salt Lake County	USA	✔	✖

Fig. 1 Examples of tuples of the $FavoriteCity$ relation annotated with feedback from two users

generated using crowd feedback is better than the wrappers generated based on generic metadata.

To illustrate the case of users with different expectations, consider an information integration system providing integrated access to information about geographical areas. In particular, the integration system contains a relation, $FavoriteCity$, that provides information about favorite cities in the world. Figure 1 illustrates example tuples of the $FavoriteCity$ relation. Different users have different preferences as to the criteria a city must meet to be listed among his/her favorites. For example a given user, $user_1$, may prefer cities that are located near ski resorts, whereas another user, $user_2$, would prefer cities located near a beach. Because of this, some of the feedback instances supplied by the two users (see Fig. 1) are conflicting: the tuples t_1, t_3 and t_6 are true positives for $user_1$ and are false positives for $user_2$, and the tuples t_2 and t_5 are true positives for $user_2$ and false positives for $user_1$. Tuple t_4 refers to the city of Manchester, which is a false positive for both users as it is not located near a ski resort or a beach. Using the feedback instances provided by the two users, e.g., to identify the mapping to be used to populate the $FavoriteCity$ relation, may deteriorate the quality of the integration system over time, e.g., the mappings constructed or selected in the light of the feedback provided by the two users may miss a large number of true positives for both users. This simple example shows the need to identify groups of users with similar expectations as to the results that should be provided by the integration system, e.g., $user_1$ and $user_2$ should belong to different groups. Grouping users ensures that if used collectively, the feedback instances the users provide can improve the quality of the integration system.

We consider for the purposes of this paper feedback of the above form, that comments on the membership of tuples to relations in the integration schema. We have shown in previous work [3,25], that schema mappings can be selected, refined and annotated with metrics specifying their fitness based on this kind of feedback. We have also showed through empirical exercises that feedback scarcity impacts negatively on the quality of the data integration system. For example, we have shown that using a few feedback instances, e.g. 10, commenting on tuple membership to a relation in the integration schema, the precision estimates computed for assessing the quality of the schema mappings that are candidates to populate such a relation, suffer from a high error [3].

In this paper, we show how clustering [18] can be used to identify groups of users with similar expectations as to the results delivered by the integration system.

Specifically, given a set of users, we use clustering to identify groups of users with similar expectations. Users can be clustered by comparing the feedback instances they provide, i.e., by comparing the feedback given by different users on the same tuples. The more overlapping and less conflicting the feedback provided by two users, the more likely they are to belong to the same cluster.

Because users provide feedback only on a subset of the tuples presented to them, and because the number of tuples that can be used to populate a given relation in the integration schema can be large, the chance that the same tuple is annotated by more than one user may be small. As a result the above clustering strategy performs poorly, as the experimental results that we will report on later in the paper show, due to the sparsity of the feedback provided by users. To overcome this problem, we devised a second clustering strategy that measures similarity between users as to the results they expect from the integration by using mapping precision estimates that are computed based on the feedback provided by users. Such annotations are estimates that assess the fitness of schema mappings in the information integration system to the expectations of a given user based on the feedback supplied by that user. Given a schema mapping, the closer the respective precision estimates computed based on the feedback supplied by two users, the more similar are their expectations from the integration. Unlike the first strategy, users do not have to provide feedback annotating the same tuples. We show that this clustering strategy produces good quality clustering with only a small amount of feedback.

Clustering results can be used to leverage the process by which user expectations from the integration are ascertained. In particular, the expectations of individual users from the integration system can be learned using the feedback supplied by the cluster to which they belong. We empirically investigate the amount of feedback gained by using clustering results to learn the expectations of individual users from the integration system. We also present a method to accelerate the process by which the expectations of new users of the integration system are ascertained. To do so, we cast this problem as a classification problem whereby new users are assigned to existing clusters.

Clustering can be used to improve the quality of information integration systems. For example, it can be used to improve the quality of schema mappings by refining them to meet the expectations of the users within a cluster [3]. In this paper, we report on the results of two applications of clustering. The first shows that the feedback supplied by a cluster can be used to select the schema mappings that fit best the expectations of the users within the cluster. We also show that clustering can be used to identify malicious users, i.e., users who provide misleading feedback with the objective of hampering the process by which the quality of the information integration system is improved.

The contributions we make in this paper enable crowd-based information integration using clustering. While in a general setting the workers in the crowd are different from the users who exploit the integration system, in our setting, the users of the information integration system are enlisted to be the workers who, through their feedback, improve the quality of the information integration system. Specifically, we make the following contributions:

1. *Strategies for clustering information integration users*: we present and examine distance functions that can be used to group information integration users (in Sect. 3).
2. *Methods for learning feedback for users, as to their expectations from the integration system, based on clustering*: we estimate the benefit in terms of feedback that individual users gain, i.e. do not have to provide, by using clustering results (in Sect. 4). We also present a method whereby new users are assigned to the (existing) cluster that best meets their expectations (in Sect. 5).
3. *Clustering applications*: we show that clustering can be used to improve the quality of an information integration system. Specifically, we show that it can be used to identify the mappings that best meet the expectations of users, and to identify malicious users (in Sect. 6).

Additionally, we present the feedback model we consider in this paper, and show how user feedback can be used to annotate schema mappings with precision estimates in Sect. 2. We analyze and compare existing works to ours in Sect. 7, and conclude the paper in Sect. 8.

2 Background: user feedback and schema mappings

Schema mappings are fundamental elements to any data integration system. They are used to specify the extent of an integration schema using data retrieved from existing sources [15]. Without loss of generality in terms of the results presented on clustering, we consider *global-as-view* mappings [20], which relate one element in the integration schema to a query over the source schemas. Specifically, given a relation r_i in the integration schema, a mapping m that is used to populate r_i is specified by the pair $m = \langle r_i, q_s \rangle$, where q_s is a relational query over the source schemas. We use $m.integration$ to refer to r_i, and $m.source$ to refer to q_s.

The specification of schema mappings is difficult and time consuming. It requires deep knowledge of the contents of the sources to be integrated, as well as the expectations of end users as to the results that should be delivered by the integration system. To overcome these difficulties and to reduce the upfront costs required to set up the data integration system, existing schema matching techniques can be used to produce the input for algorithms capable of automatically generating the mappings between the integration schema and the source schemas (e.g., [12,23,24,27]). Multiple matching mechanisms can be used, each of which could give rise to multiple mapping candidates for populating the elements of the integration schema. Given a relation r in the integration schema, we refer to the mappings that are output by mapping generation techniques to populate r using the term *candidate mappings*.

To assess the quality of the candidate mappings of a given relation r, such mappings can be labeled by scores that are computed based on the confidence of the matches used as input for their generation (e.g., [13]). This approach does not guarantee, however, that the candidate mapping with the highest score best fits the expectations of end users.

This is because the confidences of matches are generated based on heuristics [6,17], that typically have little information on domain semantics.

In this paper, we use a different source of information for assessing the fitness of candidate mappings, namely user feedback. In doing so, the user is not provided with a set of (possibly complex) mapping expressions; rather, s/he is given a set of answers to a query issued against the integration schema that was answered using one or more of the candidate mappings. To further illustrate the kinds of feedback that can be supplied, assume that the user issued a query to retrieve the tuples of the relation r_i in the integration schema, that was evaluated using one or more mappings that are candidates for populating r_i, and that the query results were displayed to the user. The user then examines and comments on the results displayed using the following kinds of feedback: (i) that a given tuple was expected in the answer (true positive), (ii) that a given tuple was not expected in the answer (false positive), and (iii) that an expected tuple was not retrieved (false negative).

In what follows, given a mapping candidate m for populating a relation r in the integration schema, and given a set of feedback instances UF supplied by the user, we use $tp(m, UF)$, $fp(m, UF)$, $fn(m, UF)$, respectively, to denote the true positives, false positives and false negatives of m given the feedback instances in UF.

The quality of mapping candidates to populate a relation r in the integration schema can be quantified using precision [30], i.e., the fraction of the tuples retrieved by the mapping candidate that meet user expectation from the integration. Of course, we cannot compute this metric since it requires the availability of the extent of r, i.e., the set of tuples that belong to r in the users' conceptualization of the world. Notice, however, that the feedback instances supplied by users provide *partial* information about the extent of r. Specifically, they allow the identification of (some of the) true positives, false positives and false negatives of a given candidate mapping. Using this information, we can compute the precision *relative* to (the extent of r identified through) the feedback supplied by the user. Specifically, we define the precision estimate of a mapping m relative to the feedback instances in UF as the ratio of the number of true positives of m given UF to the sum of true positives and false positives of m given the feedback instances in UF. That is:

$$prec(m, UF) = \frac{|tp(m, UF)|}{|tp(m, UF)| + |fp(m, UF)|} \quad (1)$$

where $|s|$ denotes the magnitude of the set s.

We use the following notation in the rest of the paper.

- Given a user u, we use UF_u to refer to the set of feedback instances provided by u, and given a cluster c, UF_c is the union of the feedback provided by the users within the cluster c.
- $expected(UF)$ is the set of tuples that are annotated either as true positive or false negative given the feedback in UF.
- $unexpected(UF)$ is the set of tuples that are annotated as false positives given the feedback in UF.

3 Clustering users

This section presents three strategies that we devised for clustering information integration users, and reports on evaluation exercises for assessing their effectiveness.

3.1 Clustering strategies

There are many algorithms for clustering that are readily available in the literature [18]. To make use of such algorithms, however, we need to define a function for measuring the distance between user expectations from the integration system. It is worth stressing here that we do not aim to develop yet another clustering algorithm or to identify the best existing clustering algorithm. Instead, our objective is to devise a distance functions, that can be used by such algorithms, to effectively group information integration users.

The distance between two users can be computed by comparing the feedback they supply. Specifically, the more overlapping and less conflicting their respective feedback instances, the smaller the distance between users. The functions $distance_{overlap}$ and $distance_{conflict}$, defined below, can be used to measure the distance between two users in terms of overlap and conflict in feedback.

$$distance_{overlap}(u_i, u_j) = 1 - \frac{|overlap(u_i, u_j)|}{|union(u_i, u_j)|}$$

$$distance_{conflict}(u_i, u_j) = \frac{|conflict(u_i, u_j)|}{|union(u_i, u_j)|}$$

where:

$$union(u_i, u_j) = expected(UF_{u_i}) \cup unexpected(UF_{u_i}) \cup expected(UF_{u_j}) \cup unexpected(UF_{u_j})$$

$$overlap(u_i, u_j) = (expected(UF_{u_i}) \cap expected(UF_{u_j})) \cup (unexpected(UF_{u_i}) \cap unexpected(UF_{u_j}))$$

$$conflict(u_i, u_j) = (expected(UF_{u_i}) \cap unexpected(UF_{u_j})) \cup (unexpected(UF_{u_i}) \cap expected(UF_{u_j}))$$

The functions $distance_{overlap}$ and $distance_{conflict}$ are likely to require a large number of feedback instances before the clusterings obtained are similar to the ground truth clustering, i.e., the clustering obtained when all the tuples retrieved by the mappings are annotated by every user. This is because the chances that the same tuple is annotated by different users is small, when the number of feedback instances given by the users is small. (This observation will be confirmed in the empirical exercises that we report on Sect. 3.2). We, therefore, explored a second clustering strategy. As described in Sect. 2, feedback can be used to annotate schema mappings with precision estimates. Given a mapping m, users with similar expectations as to the results delivered by

the integration system are likely to be associated with similar precision estimates, and, conversely, users with different expectations are likely to be associated with disparate precision estimates. Therefore, we can measure the distance between users by comparing precision estimates, computed based on the feedback they provide, for annotating schema mappings. The function $distance_{precision}$, defined below, can be used for this purpose.

$$distance_{precision}(u_i, u_j) = \frac{\sum_{k=1}^{n} |prec(m_k, UF_{u_i}) - prec(m_k, UF_{u_j})|}{n} \quad (2)$$

where $prec(m, UF)$ is the precision estimate computed for the mapping m given the feedback instances in UF as defined in Sect. 2.

The distance in terms of precision estimates for candidate mappings calculated using Eq. (2) ensures that two users who expect the same results from the integration are grouped within the same cluster. However, two users who have different expectations may be grouped in the same cluster. This is because the precision of a given mapping can be the same for two users who have different expectations: different sets of expected and unexpected results may yield the same precision estimates. Note, however, that the likelihood that such a phenomenon occurs diminishes when there are multiple mapping candidates that retrieve different (and not necessarily disjoint) result sets. Specifically, the likelihood that two users who have different expectations have similar ground truth precision for multiple mapping candidates is low when the mapping candidates retrieve different data sets. That said, we explored a source of information that can be used to overcome the above problem. Specifically, in addition to using mapping precision estimates, we explored the use of conflicts in feedback that users provide. As we mentioned earlier, the likelihood that two users provide feedback annotating the same tuple is small. Because of this, we explored an approach whereby a proportion of the feedback given by users is provided on the same tuples. To illustrate this, consider that each user provides n feedback instances. Of these n feedback instances, users need to provide p feedback instances annotating the same p tuples, which we refer to as *common tuples*. The difference in expectations between two users is then estimated based on the feedback given on the p common tuples using the following distance:

$$distance_{commonTuples}(u_i, u_j) = \frac{|conflict_{commonTuples}(u_i, u_j)|}{|commonTuples|} \quad (3)$$

where $commonTuples$ denotes the set of tuples that received feedback from both u_i and u_j, and $conflict_{commonTuples}(u_i, u_j)$ refers to the tuples in $commonTuples$ that were given conflicting feedback by the users u_i and u_j. Using the Eqs. (1) and (2), the following weighted distance can be used to estimate the distance between two users:

$$distance(u_i, u_j) = w_i \times distance_{precision} + w_j \times distance_{commonTuples} \quad (4)$$

where $0 \leq w_i, w_j \leq 1$, and $w_i + w_j = 1$.

Before assessing the effectiveness of the functions presented so far, it is worth noting that all of them satisfy the positive definiteness [1]. On the other hand, they do not satisfy the triangular inequality, which is required in a metric space [5]. That said, several clustering algorithms, for example the hierarchical clustering algorithm that we will use in the next section, do not require this last condition. In other words, there are several clustering algorithms that can use the distance functions we devised to group information integration users.

3.2 Evaluation

The distance functions presented in the previous section can be used to cluster information integration users into groups. In general, one would expect users to provide a small number of feedback instances relative to the size of an extent. Consider, for instance, our earlier example regarding the information integration system providing information about geographical areas. It is unrealistic to expect a user to provide feedback instances identifying every tuple that should or should not be used to populate the $FavoriteCity$ relation. Instead, a user is likely to provide some examples of tuples that are expected to belong to the $FavoriteCity$ relation and others that are not. This raises the following question regarding the quality of clusterings computed based on the above distances: *is the clustering obtained using a small amount of feedback close to the ground truth clustering, i.e., the clustering obtained with complete knowledge of users expectations?* To answer this question, we ran an experiment, with the following objectives in mind.

3.2.1 Experiment objectives

1. To assess and compare the effectiveness of the distance functions devised in the previous section. The smaller the amount of feedback necessary, the more effective is the function.
2. To examine how sensitive clustering results are to the population size, to the number of ground truth clusters, and to the ground truth distances between the clusters. In other words, do these parameters influence the amount of feedback necessary to converge to the ground truth clustering?

3.2.2 Experiment setup

We clustered users of an information integration system using $distance_{overlap}$ and $distance_{conflict}$ by varying the number of feedback instances they provide. Specifically, we considered the following setting. We defined the relation *FavoriteCity*(*name*, *country*, *province*) in the integration schema, and manually created 5 mapping candidates, $m_1, \ldots, m_5$ for populating such a relation using data from the Mondial geographical database.[1] The mappings are of the following form:

[1] http://www.dbis.informatik.uni-goettingen.de/Mondial.

$$m_i = \langle FavoriteCity, \Pi_{name,country,province} City_i \rangle$$

Where $City_1, \ldots, City_5$ are relations that we created by taking different subsets of the tuples that compose the relation $City$ in the Mondial geographical database. For example, $City_1$ contains Cities that are located in North America. We took this approach to simulate the situation where the integration relation, in this case $FavoriteCity$, is populated using data coming from different sources. Note that the relations $City_1, \ldots, City_5$ are not disjoint. Together, the candidate mappings return 2,523 tuples to populate the $FavoriteCity$ relation. We then created 15 *synthetic* users identified by the integers 1,..., 15.

To specify the ground truth expectations of users as to the tuples that they are expecting to be used to populate the integration relation $FavoriteCity$, we defined three groups of users {1, 2, 3, 4}, {5, 6, 7, 8, 9} and {10, 11, 12, 13, 14, 15}. Then, for each group (and therefore the users within the group), we randomly selected a subset of the 2,523 tuples, returned by the candidate mappings $m_1, \ldots, m_5$, and set it to be the set of ground truth tuples expected by the users in the group. When selecting the tuples that are expected by each group of users, we made sure that the three user groups have both overlapping and conflicting expectations.

To cluster users, we use the *hclust* algorithm provided by the `R` statistical framework.[2] for performing hierarchical clustering. We chose hierarchical clustering mainly because, unlike most clustering algorithms, it does not require as input the number of clusters to be identified. We note here again that our objective is not to evaluate the pros and cons of a specific clustering algorithm, rather, we aim to evaluate the effectiveness of the functions used for measuring the distance between users in the light of feedback provided by users.

To assess the quality of the clustering obtained based on feedback, we started our experiment by clustering users based on ground truth expectations. Specifically, we computed the distance, $distance_{overlap}$, in terms of overlap between every pair of users based on complete knowledge of expectations. Using the distances computed, we clustered users using the *hclust* algorithm. The ground truth clustering obtained is depicted by the dendrogram illustrated in Fig. 2. A dendrogram is a tree that visualizes hierarchical clustering results. Leaf nodes represent users, which are identified by integers. The leaf nodes are spaced evenly along the x-axis. Inner nodes in the dendrogram are used to specify that at a given distance, which is specified by the vertical axis, two clusters (users) are merged in one cluster. Consider, for example, the dendrogram illustrated in Fig. 2. It specifies that the two clusters {5, 6, 7, 8, 9} and {10, 11, 12, 13, 14, 15} are merged into one cluster at a distance of 0.43, and that the clusters {1, 2, 3, 4} and {5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15} are merged into one cluster at a distance of 0.74.

Note that the ground truth clustering using the distance in terms of conflict, $distance_{conflict}$, is the same as that obtained using the distance in terms of overlap, $distance_{overlap}$. The distances in term of overlap and conflict are equal when the ground truth expectations of users are known. This is because the following

[2] http://www.r-project.org.

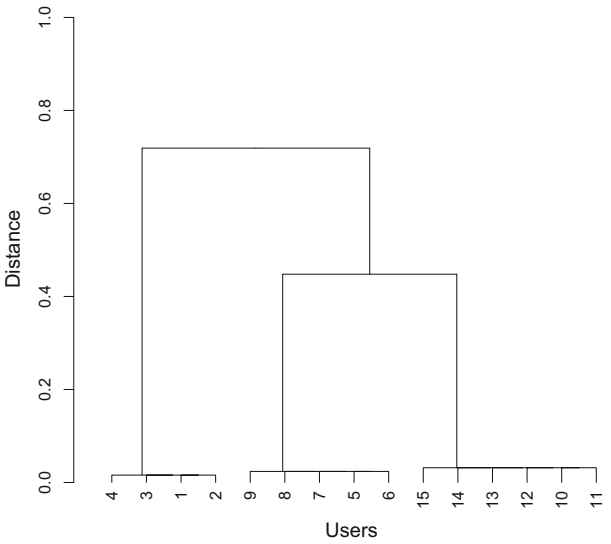

Fig. 2 Ground truth clustering obtained using $distance_{overlap}$

equality holds when the ground truth expectations of two users u_i and u_j are known: $|union(u_i, u_j)| = |overlap(u_i, u_j)| + |conflict(u_i, u_j)|$.

We then clustered the above users based on feedback. Specifically, we ran the following procedure multiple times by varying the value of the variable n, which represents the number of feedback instances supplied by each user, from 10 to 500. To generate a feedback instance for a given user u_i, we randomly select a tuple t in the set of tuples generated by the candidate mappings. If the tuple t belongs (resp. does not belong) to the ground truth expectations of the user u_i, then it is annotated as an expected (resp. unexpected) tuple for the user u_i and added to the set of feedback instances for u_i.

1. For each user
2. Generate randomly n feedback instances
3. Cluster users based on distance in overlap
4. Cluster users based on distance in conflict

We then re-ran the above procedure, but this time using mapping precision estimates to form clusters, i.e., using the function $distance_{precision}$ (see (2)).

To assess the effectiveness of the third clustering strategy that uses the distance function in Eq. (4) in situations where mapping precision is the same for users who have different expectations as to the results delivered by the integration, we ran an experiment with the following setting: $w_i = w_j = \frac{1}{2}$, and the fraction of feedback that is given on common tuples is $\frac{1}{10}$. We also modified the ground truth expectations of the users $1, \ldots, 15$, so that some of the users who belong to different clusters have the same precision estimates for candidate mappings. Specifically the following pairs of users, who belong to different clusters, have the same precision estimates: $\langle 1, 5\rangle$, $\langle 2, 11\rangle$, and $\langle 6, 13\rangle$. To do so, we had to specify mapping candidates that allow

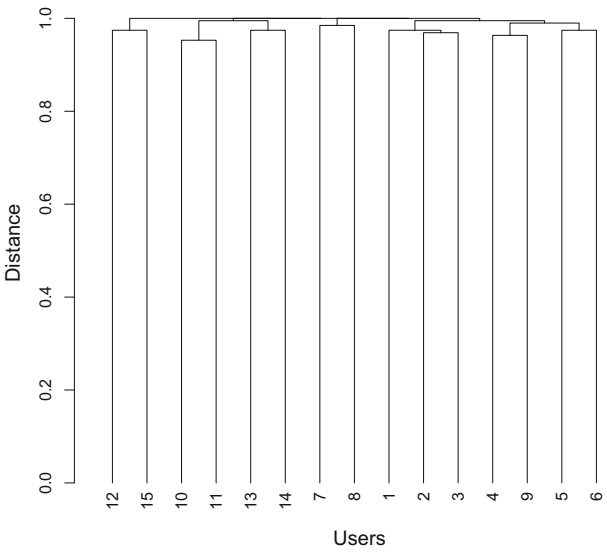

Fig. 3 Clustering obtained using $distance_{overlap}$ based on 100 feedback instances

for this. Using the initial mapping candidates, it was difficult to define ground truth expectations for the users such that the above pairs of users have the same precision estimates.[3]

To assess the sensitivity of the clustering results, we re-ran the above experiment by varying the number of users and that of the numbers of clusters in the ground truth clustering. Specifically, we varied the number of users to be clustered from 15 to 1,000 (specifically, we considered the following population sizes: 15, 50, 100, 500 and 1,000), and varied the number of ground truth clusters to be found from 3 to 15. We also varied the ground truth distance between clusters from 0.005 to 0.5. For each distance, we clustered users using different amounts of feedback to identify the minimum number of feedback instances required to distinguish between the two clusters.

3.2.3 Experiment results

The results of this experiment revealed that the distances in terms of overlap and conflict are not effective in the sense that they yield clusterings that are different from the ground truth clustering using (even) large amounts of feedback. For example, Figs. 3, 4 and 5 show the clusterings obtained using $distance_{overlap}$ when the number of feedback instances provided by each user is 100, 200 and 500, respectively, and Figs. 6, 7 and 8 show the clusterings obtained using $distance_{conflict}$ when the number of feedback instances provided by each user is 100, 200 and 500, respectively. These results

[3] This enforces the observation we made earlier: the likelihood that two users who have different expectations have similar ground truth precision for multiple mapping candidates is low when the mapping candidates retrieve different data sets.

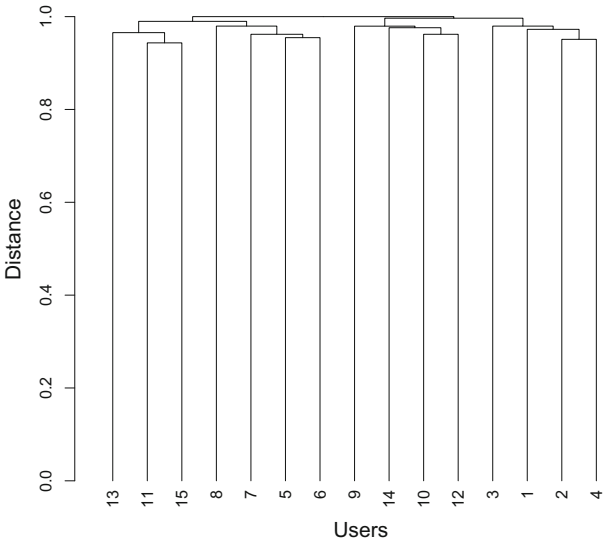

Fig. 4 Clustering obtained using $distance_{overlap}$ based on 200 feedback instances

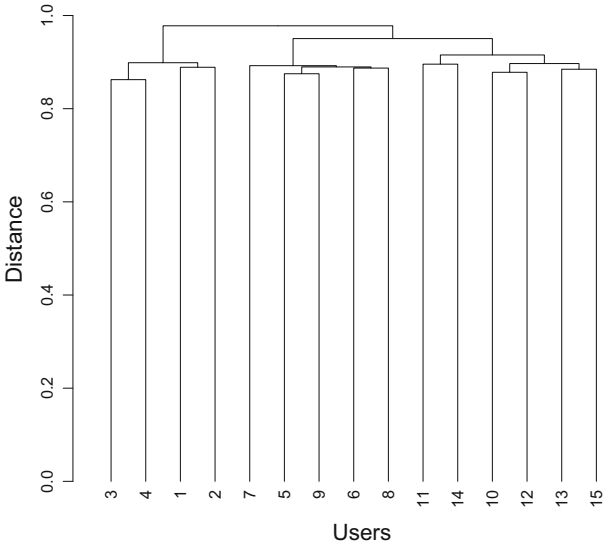

Fig. 5 Clustering obtained using $distance_{overlap}$ based on 500 feedback instances

show that using 100 feedback instances, the structure of the clustering obtained using both $distance_{overlap}$ and $distance_{conflict}$ is different from the ground truth clustering. Using 200 feedback instances, the structure of the obtained clustering resembles the structure of the ground truth clusterings. However, the error in the distances between clusters is high. This error remains high even when each user contributes 500 feedback

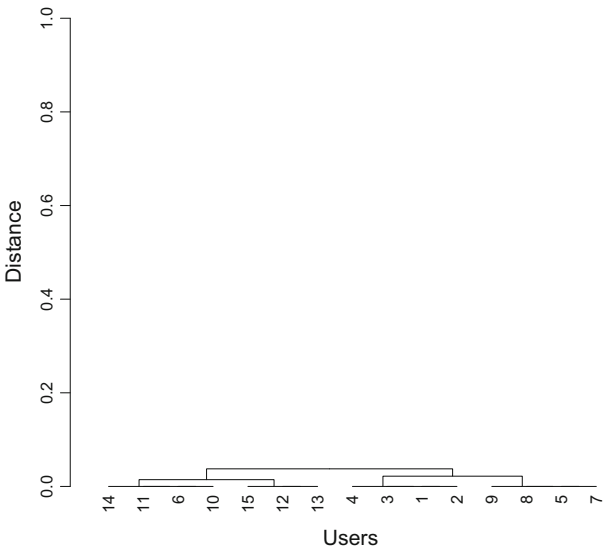

Fig. 6 Clustering obtained using $distance_{conflict}$ based on 100 feedback instances

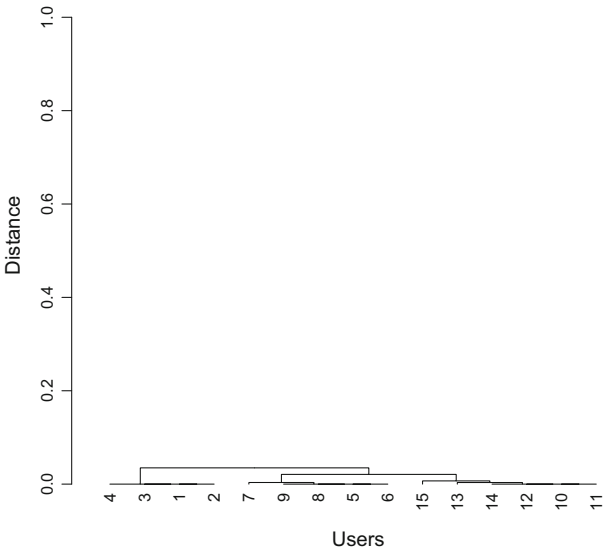

Fig. 7 Clustering obtained using $distance_{conflict}$ based on 200 feedback instances

instances, i.e., $\simeq$20 % of the total number of tuples returned by the candidate mappings (Figs. 5, 8).

This disappointing result is due to the sparsity of the feedback supplied by users. Indeed, the likelihood that the same tuple is annotated by two users is small. To illustrate this, consider that two users u_1 and u_2 have the same expectations. Specifically, of 2,523 returned by the candidate mappings, 200 tuples are expected and 2,323 are unexpected

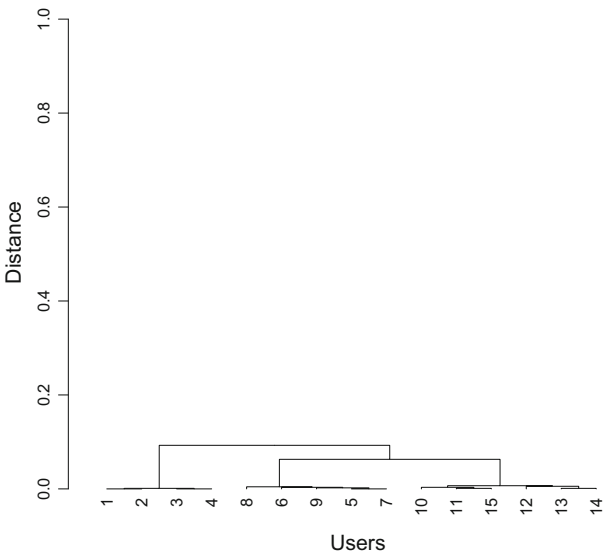

Fig. 8 Clustering obtained using $distance_{conflict}$ based on 500 feedback instances

according to u_1 and u_2. And, consider that u_1 and u_2 provided 100 feedback instances each. The probability that the same tuple is annotated by both u_1 and u_2 is very small, 0.0013. Indeed, The probability that a given tuple is annotated as expected by both u_1 and u_2 is $(\frac{100}{2{,}523} \times \frac{200}{2{,}523})^2$, and the probability that a given tuple is annotated as unexpected by both u_1 and u_2 is $(\frac{100}{2{,}523} \times \frac{2323}{2{,}523})^2$. Therefore, the probability that a tuple is annotated by both u_1 and u_2 as expected or unexpected is $(\frac{100}{2{,}523} \times \frac{200}{2{,}523})^2 + (\frac{100}{2{,}523} \times \frac{2{,}323}{2{,}523})^2 = 0.0013$.

On the other hand, the experiment showed that that the clusterings obtained using the distance based on mapping precision estimates (see Eq. (2)) are close to the ground truth clustering using small amounts of feedback. For example, Figs. 10, 11 and 12 show the clusterings obtained when the number of feedback instances provided by each user is 50, 100 and 200, respectively. Using 50 feedback instances, which is relatively small as it represents 2 % of the tuples returned by the candidate mappings, the structure of the clustering obtained (Fig. 10) is close to the structure of the ground truth clustering (Fig. 9). Moreover, the distances between users and clusters are close to the ground truth distances. As the amount of feedback increases, the structure of the clustering and the distances improve in the sense that they approach the ground truth clustering (see Figs. 11 and 12). This positive results can be explained by the facts that: (i) the expectation of users from the integration can be compared even when the feedback instances they provide annotate different tuples, and (ii) the error in the precision estimates computed for mapping candidates decreases substantially in the first feedback iterations [3].

Regarding the third clustering strategy, which can be used in situations where mapping precision is the same for users who have different expectations as to the results delivered by the integration. The results of the experiment showed the effectiveness

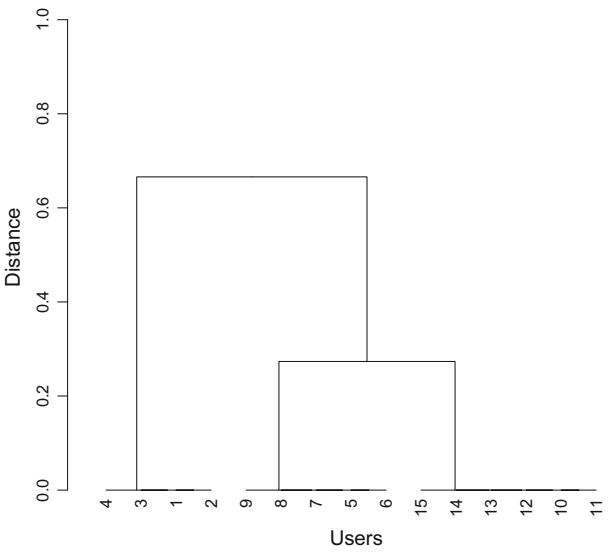

Fig. 9 Ground truth clustering obtained using the distance function in (2)

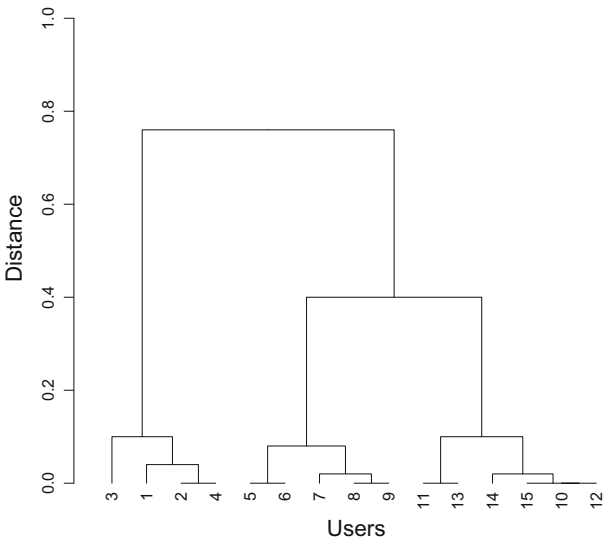

Fig. 10 Clustering obtained using the distance function in (2) based on 50 feedback instances

of the distance in Eq. (4): the quality of the clustering is not affected by the presence of users who have different expectations and have the same precision estimates for mapping candidates. On the other hand, we recorded an increase in the cost, i.e., the amount of feedback, required to reach a clustering that is close to the ground truth clustering. For example, each user had to provide 160 feedback instances to obtain a clustering similar to the ground truth clustering.

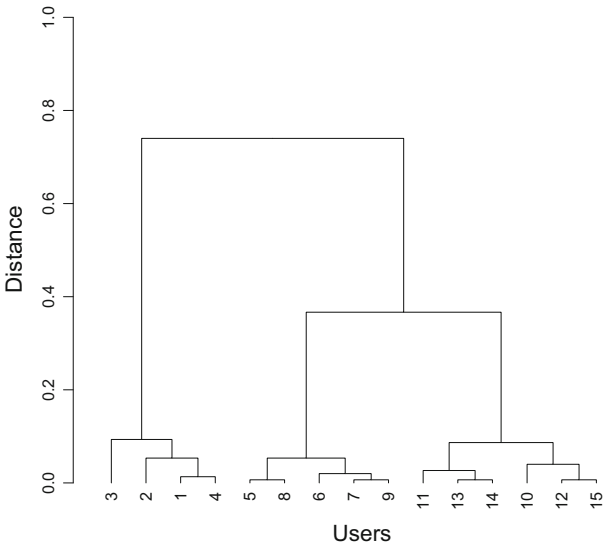

Fig. 11 Clustering obtained using the distance function in (2) based on 100 feedback instances

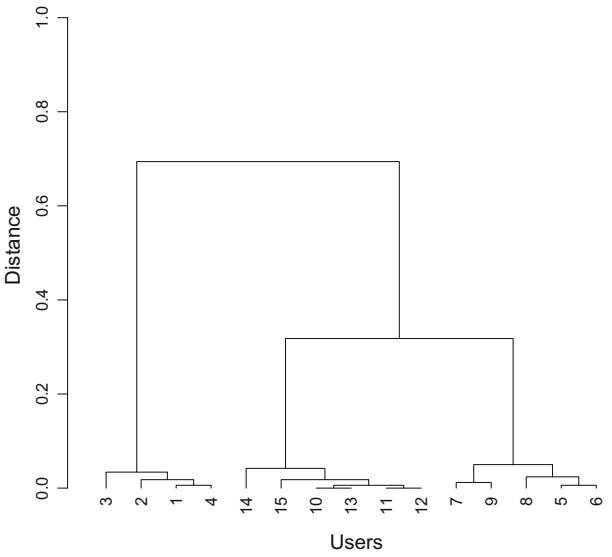

Fig. 12 Clustering obtained using the distance function in (2) based on 200 feedback instances

That amount of feedback can be decreased by increasing the fraction of feedback given on common tuples. For example, the number of feedback instances required to reach a clustering similar to the ground truth clustering decreases from 160 to 110 when the fraction of feedback given on common tuples is 0.5.

Note, however, that by increasing the fraction of feedback given on common tuples, the benefit that can be derived from clustering decreases. In particular, when all users

Table 1 Sensitivity to distance between clusters

Distance between clusters	No. of feedback instances necessary to identify clusters
0.05	670
0.1	150
0.15	110
0.25	70
0.5	20

provide feedback annotating the same tuples, i.e., when the fraction of feedback given on common tuples is 1, we cannot ascertain new feedback for a given user by utilizing the feedback supplied by other users in the same cluster. To illustrate this, consider a cluster c that contains two users u_i and u_j, and consider that u_i (resp. u_j) supplied feedback annotating the set of tuples T_i (resp. T_j). If $T_i \neq T_j$, then we can use the feedback supplied by u_i to learn the feedback that u_j would give to annotate the set of tuples in $T_i - T_j$. On the other hand, if $T_i = T_j$, i.e., u_i and u_j supplied feedback annotating the same tuples, then we will not be able to infer new feedback for u_i nor u_j.

Given the poor results of the clustering based on user feedback, i.e., using the distance functions $distance_{overlap}$ and $distance_{conflict}$, we focused in our analysis of the sensitivity of clustering results on the clustering based on mapping precision estimates, i.e., the distance function $distance_{precision}$. In this respect, the experiment showed that clustering is not sensitive to population size or to the number of clusters: the number of feedback instances required from each user to obtain a clustering similar to the ground truth clustering remains more or less the same, i.e., $\simeq$50. However, the experiment showed that the clustering results were sensitive to the ground truth distance between clusters. Specifically, as illustrated in Table 1, as the distance between the clusters decreases, the minimum number of feedback instances required to identify the two clusters increases. Specifically, the clusters can be identified with a small amount of feedback, i.e. 20, when the distance between the clusters is 0.5. On the other hand, when the distance between the clusters is small, 0.05, a large amount of feedback, 670 feedback instances, is needed. This is a understandable result, since the task is harder when the clusters are similar.

3.3 Feedback sampling

In the experiment that we reported on in the previous section, we have used a uniform random generator to select the tuples on which feedback is given. That is, tuples have the same chance of being picked by the users to be annotated. This raises the question as to whether other random generators, whereby tuples do not have the same chance of being picked for annotation, would yield different clustering results.

To answer the above question, we re-ran the experiment presented in the previous section and used a normal random generator, instead of a uniform one. Specifically, the tuples, on which feedback is given, are randomly selected using a normal (gaussian)

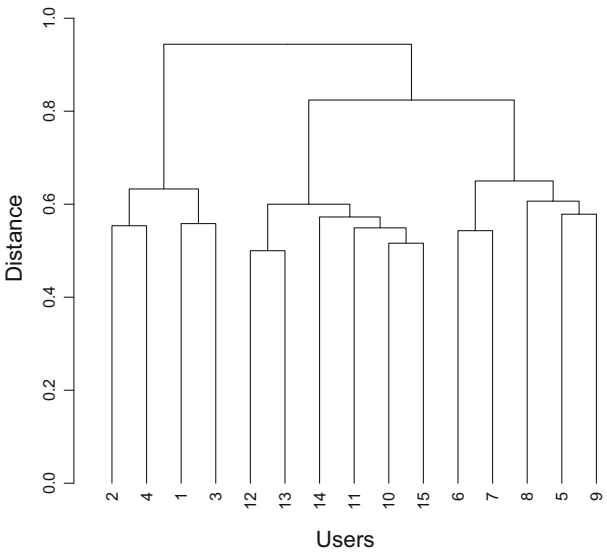

Fig. 13 Clustering obtained based on 100 feedback instances generated using a normal random generator with a standard deviation of 0.02

distribution. In doing so, we changed the following parameters of the generator: the median of the distribution, and the standard deviation.

We clustered users using the three distance functions presented in the paper: based on overlap in feedback instances, $distance_{overlap}$, based on conflict in feedback instances, $distance_{conflict}$, and based on the precision estimates computed for the schema mappings, $distance_{precision}$.

The results of this experiment showed that for clustering based overlap and conflict in feedback instances, the use of a normal distribution yields clustering of better quality compared with uniform distribution. More importantly, the experiment showed that the smaller is the standard deviation of the distribution, the better is the quality of the clustering. For example, in the case of $distance_{overlap}$, using 100 feedback instances that are generated using a distribution with a standard deviation of 0.02 (Fig. 13) we obtain a clustering of better quality than that obtained with a standard deviation of 0.4 (Fig. 14). The ground truth clustering is illustrated in Fig. 2.

This can be explained by the fact that for distributions with small standard deviations, the chances that different users choose to annotate the same tuple, which then contribute in the computation of $distance_{overlap}$ and $distance_{conflict}$, is high. Inversely, for distributions that have a large standard deviation, the chances that different users choose to annotate the same tuple are low. Therefore, different users provide feedback instances that do not contribute in the computation of the distances $distance_{overlap}$ and $distance_{conflict}$.

Regarding clustering based on the $distance_{precision}$, we did not observe any change in the clustering obtained using uniform or normal distribution. Moreover, changing the standard deviation of the distribution did not have an effect on the clustering quality. This can be explained by the fact that, unlike $distance_{overlap}$ and $distance_{conflict}$,

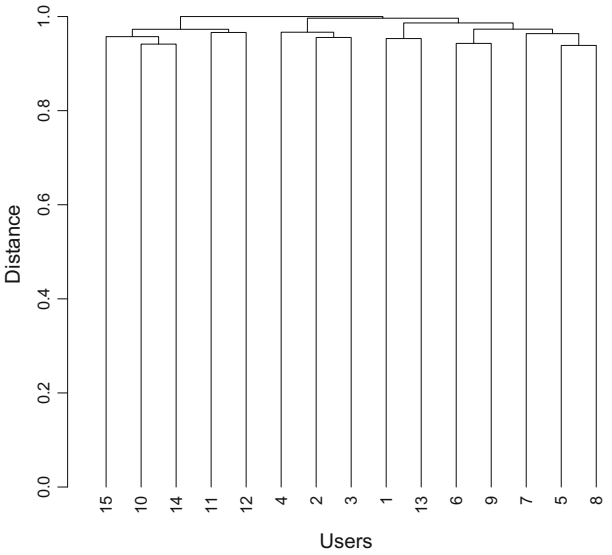

Fig. 14 Clustering obtained based on 100 feedback instances generated using a normal random generator with a standard deviation of 0.4

$distance_{precision}$ does not require the same tuple to be annotated by different users for the feedback to contribute to the computation of the distance. This is because this distance uses precision estimates of the mappings.

4 Estimating the benefit users derive from clustering

In essence, the benefit that a given user u derives from a cluster c can be defined in terms of the feedback instances that are supplied within the cluster c, which can be used to learn *more* about the expectations of u. When the expectations captured by the cluster meet the exact expectations of the user from the integration system, i.e., the feedback instances provided within the cluster c are consistent with the expectations of the user u, the benefit that u derives from c can be defined as the amount of feedback that u gains by using the feedback instances within c. That is: $benefit(u, c) = |UF_c - UF_u|$, i.e., the magnitude of the set $UF_c - UF_u$.

Without loss of generality, in what follows, we will confine ourselves to the case in which the user did not provide any feedback. In other words, we focus on estimating the value of the feedback supplied within a given cluster to the user. The benefit can, therefore, be defined as:

$$benefit(u, c) = |UF_c|$$

In practice, the expectations of the user and those captured by the cluster may not be exactly the same, in which case, the above formula cannot be used because some of the feedback instances supplied within the cluster may be inconsistent with the

ground truth expectations of the user. This raises the question as to how the benefit can be estimated when the user expectations, although similar to, are not the same as the expectations captured by the cluster. In what follows, we present a method that we propose to estimate the benefit by using the error in precision estimates of schema mappings.

Consider a user u, a cluster c and the mapping candidates $m_1, \ldots, m_n$ to populate relation r, and consider that the feedback instances in UF_c are provided within the cluster c to annotate tuples of the r relation. The feedback instances in UF_c can be used to compute the precision estimates for the mappings $m_1, \ldots, m_n$. Let err_c be the average error in precision estimates computed based on the feedback in UF_c given the expectations of the user u. We define the benefit that a user u derives from the cluster c as the number of feedback instances that the user u needs to provide in order that the average error in precision estimates computed based on the feedback supplied by u, err_u, is equal to err_c.

Note that the benefit that a user gains depends on the amount of feedback supplied within the cluster, but also on the distance between the user expectations and the expectations captured by the cluster. In particular, one would expect that the benefit increases as the distance between user and cluster decreases. The distance between user and cluster expectations can be defined as follows:

$$distance^{gt}(u, c) = \frac{|\bigcup_{u_i \in c} conflict^{gt}(u, u_i)|}{|extent(r)|}$$

where $extent(r)$ denotes the extent of the r relation, and $conflict^{gt}(u, u_i)$ is the set of tuples in $extent(r)$ that are expected according to the ground truth expectations of u and are unexpected according to the ground truth expectations of u_i, or vice–versa.

4.1 Evaluation

4.1.1 Experiment objective

To identify the situations in which a user u derives benefit from a cluster c considering the following parameters: the amount of feedback provided within the cluster and the distance between the user and the cluster.

4.1.2 Experiment setup

We computed the benefit that u derives by varying the values of the following parameters: (i) the number of feedback instances supplied by the cluster c, and (ii) the distance between the expectations of u and the expectations captured by the cluster c. Specifically, the number of feedback instances supplied within a cluster varies between 0 and 1,000, and the distance between the user and the cluster varies between 0.05 and 0.3. We chose 0.3 as a maximum distance value, because sensitivity analysis showed that for distances greater than 0.3 the user does not gain any benefit regardless of the amount of feedback supplied by the cluster.

We used the data from the experiment reported in Sect. 3. Specifically, we considered the $FavoriteCity$ relation and the schema mapping candidates for populating that relation using data from the Mondial Geographical database.

The experiment consisted of two stages. In a first stage, we computed err_u, the average error in precision estimates for the candidate mappings given the expectations of the user u and given the feedback supplied by the user u. To do that, we applied the following procedure iteratively by varying n_u, the number of feedback instances supplied by the user u, from 10 to 1,000 feedback instances.

1. Generate randomly n_u feedback instances for the user u
2. Compute the precision estimates for the candidate mappings
3. Compute the average error in precision estimates err_u

In a second stage, we computed err_c, the average error in precision estimates for the candidate mappings given the expectations of the user u and given the feedback supplied by the cluster c that is distant by d from the expectations of the user u. To do that, we applied the following procedure by varying n_u, the amount of feedback supplied by the user u, from 10 to 1,000 feedback instances. To do that, we ran the following procedure iteratively by varying n_c and d.

1. Generate randomly n_c feedback instances for the cluster c that is distant by d from the user u
2. Compute the precision estimates for the candidate mappings
3. Compute the average error in precision estimates err_c

The results of the above two procedures extentionally define the following two functions: $|UF_u| \rightarrow err_u$ and $|UF_n|, d \rightarrow err_c$. The first returns the average error given the amount of feedback supplied by the user u, and the second returns the average error given the amount of feedback supplied by a cluster that is distant by d from the expectations of u. Given these two functions, we extensionally defined a third function that computes the benefit derived by the user: $|UF_c|, d \rightarrow |UF_u|$. Given $|UF_c|$ feedback instances supplied by a cluster that is distant by d from the expectations of the user u, this last function returns the corresponding amount of feedback $|UF_u|$ that the user u would need to provide in order that the average error in precision computed based on the feedback supplied by u, $error_u$, is equal to the average error computed based on the feedback supplied by the cluster, i.e., $error_u = error_c$.

4.1.3 Experiment results

The benefit function obtained is illustrated in Fig. 15. It shows that the greater the amount of feedback provided within the cluster and the smaller the distance between the user and the cluster, the better is the benefit gained by the user. Consider, for example, the case where the distance between the user and the cluster is small, $d = 0.05$. The amount of feedback gained by the user is close to that provided within the cluster, e.g., the user gains 110 feedback instances when the number of feedback instances supplied by the cluster is 150. As the distance between the user and the cluster increases, the benefit decreases. When the distance reaches 0.3, the benefit gained by the user does not exceed = 70 feedback instances, even when the number of feedback instances supplied by the cluster is large, i.e., 1,000 feedback instances.

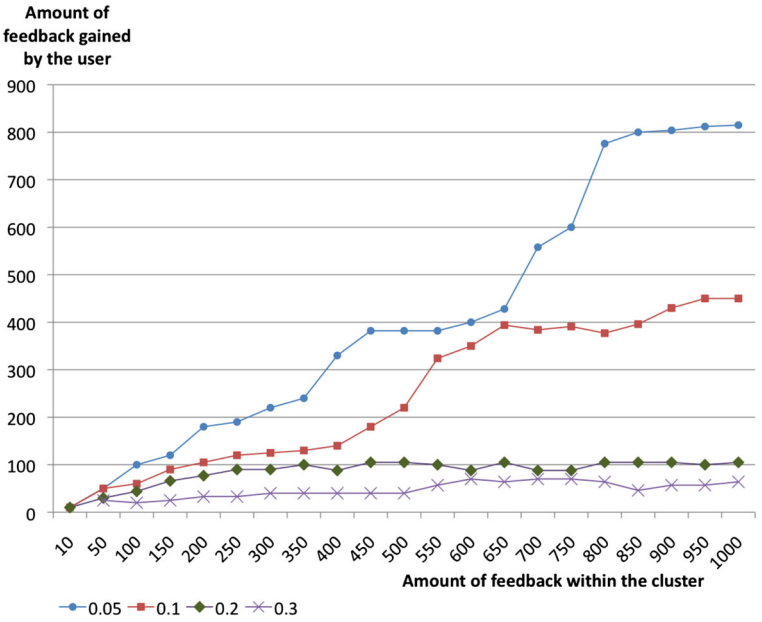

Fig. 15 Benefit in terms of feedback instances

5 Efficiently identifying the suitable cluster for new users

The process of clustering that we described in Sect. 3 is incremental. Users are re-clustered when (a subset of) users provide new feedback instances, thereby (hopefully) improving the quality of the clustering. When a new user joins in, we can, in principle, adopt a similar approach: we can ask the new user to provide feedback, and based on the newly acquired feedback we can re-apply the clustering operation. However, because initially the new user is likely to provide a small number of feedback instances, such an approach may yield the following undesirable outcomes: (i) the new user may be associated with a cluster that does not fit his/her expectations, and (ii) the quality of existing clustering may be degraded: previous clustering results that were identified without considering the new user may be of better quality since existing users have already provided sizable numbers of feedback instances. In this section, we present a method that supports the association of new users with clusters that fit their expectations.

5.1 Identifying the suitable cluster for a new user as a classification problem

The method we explore can be summarized as follows: instead of re-applying clustering when a new user joins in, given the existing clustering, we seek to assign the new user to the cluster that meets his/her expectations the best. In other words, we define the problem of identifying the expectations of new users as a classification problem. To do so, instead of asking new users to provide feedback on tuples of their

choice, we present them with a small set of tuples that, if annotated, will allow them to be associated with the cluster that best fits their expectations. The key problem is, of course, to identify the set of tuples that a new user should provide feedback on. Before proceeding to the details of how such a set is generated, we will describe the overall process whereby the cluster that meets the expectations of a given new user is identified. Such a process is iterative. In every iteration, a tuple is used to discriminate between two sets of clusters. For example, consider that, initially, there are n clusters, $n \geq 2$, and consider that t is a tuple that is annotated as expected within the clusters $c_1, \ldots, c_k$, $k < n$, and is annotated as unexpected within the clusters $c_{k+1}, \ldots, c_n$, or vice-versa. That is, t belongs to the following set:

$$\begin{array}{l} \left(\left(\bigcap_{j=1}^{k} expected(c_j)\right) \cap \left(\bigcap_{j=k+1}^{n} unexpected(c_j)\right)\right) \\ \cup \\ \left(\left(\bigcap_{j=1}^{k} unexpected(c_j)\right) \cap \left(\bigcap_{j=k+1}^{n} expected(c_j)\right)\right) \end{array}$$

where $expected(c)$ and $unexpected(c)$ are the sets of tuples that are respectively expected and unexpected given the feedback instances supplied by the members of the c cluster. Consider for example that t is annotated as expected within the clusters $c_1, \ldots, c_k$, and is annotated as unexpected within the clusters $c_{k+1}, \ldots, c_n$, and consider that a new user u annotates t as expected. Given this, we can rule out the clusters in $\{c_{k+1}, \ldots, c_n\}$, and focus on identifying the cluster in $\{c_1, \ldots, c_k\}$ that meets the expectations of user u. This process is iterative. In every iteration the set that contains the clusters that are candidates to meet the expectations of the user u is reduced. This process is applied iteratively until a singleton set, that contains the cluster that best meets the expectations of the new user, is identified. Note that we say the cluster that "best" meets the expectations. This is because the exact expectations of the new user may not be captured by any of the existing clusters, in which case, the best cluster is the one within the smallest distance from the expectations of the new user.

5.2 Generating samples of tuples for assigning new users to existing clusters

The process we have just described can be guided using a binary coupling tree [14], a binary tree in which the leaf nodes are given distinct labels. To illustrate this, Fig. 16 shows a binary coupling tree that can be used for assigning new users to a clustering composed of 3 clusters. The leaf nodes of the tree represent the clusters. An internal node is labeled by a tuple t that is used to discriminate between the two (sets of) clusters specified by its children nodes. The edges connecting the internal node to its children nodes are labeled as either "expected" or "unexpected". If the user annotates a tuple t as expected (resp. unexpected), the search in the next iteration focuses on the set of clusters designated by the children node that the edge labeled "expected" (resp. "unexpected") points to.

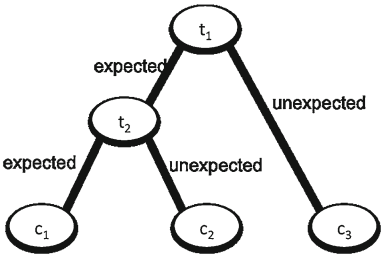

Fig. 16 Example of a binary coupling tree used to guide the assignment of a new user to a cluster

Table 2 Number of binary coupling trees for given numbers of clusters

No. of clusters	No. of binary coupling trees
2	1
3	3
7	135,135
10	654,729,075

The problem of generating a sample of tuples for assigning new users to clusters can, therefore, be mapped to that of generating a binary coupling tree. This raises the question as to how such a tree can be generated.

Given a set of clusters $C = \{c_1, \ldots, c_n\}$ and their respective feedback instances, we may not be able to generate all possible binary coupling trees with the properties stated above. Indeed, to be able to do so, we need to be able to discriminate between every two disjoint subsets S_i and S_j of C. That is:

$$\forall S_i, S_j \subseteq \{c_1, \ldots, c_n\}\ s.t.\ S_i \cap S_j = \emptyset,$$
$$\exists\, t,\ s.t.\ t \in$$
$$\left(\left(\bigcap\nolimits_{c_i \in S_i} expected(c_i)\right) \cap \left(\bigcap\nolimits_{c_j \in S_j} unexpected(c_j)\right)\right)$$
$$\cup$$
$$\left(\left(\bigcap\nolimits_{c_i \in S_i} unexpected(c_i)\right) \cap \left(\bigcap\nolimits_{c_j \in S_j} expected(c_j)\right)\right)$$

We can use a greedy algorithm that iterates over possible binary coupling trees, until finding one that can be defined based on the feedback the users have provided. Such an approach may turn out to be expensive. The number of possible binary coupling trees grows exponentially w.r.t. the number of clusters [14]. Specifically, given n + 1 clusters, the number of possible binary coupling trees is: $(2n-1) \times (2n-3) \ldots 3 \times 1$. Table 2 shows that the number of possible binary coupling trees increases dramatically even for a small number of clusters.

Fortunately, a source of information that can be used for generating a binary coupling tree for assigning new users is readily available. Indeed, we can make use of the dendrogram generated by the clustering based on the distance in terms of conflicts in feedback (Eq. (3)) to do so. Such a dendrogram discriminates between two (sets of)

clusters based on conflicts in feedback. Therefore, if two (sets of) clusters are found to be disparate based on such a dendrogram, i.e., they are separated within a distance in the dendrogram, then it is guaranteed that we can specify tuples that can be used to discriminate between the two (sets of) clusters. Furthermore, such a dendrogram can be used to identify the cases in which it is not possible to identify a tuple that can be used to discriminate between two clusters. This is the case, if the two clusters are not identified as distinct clusters in such a dendrogram, e.g., because the users in existing clusters provided small amounts of feedback that do not allow detecting conflicts.

Notice that we do not make use of the dendrogram obtained using the distance between mapping precision estimates (2), but rather use the dendrogram obtained based on the distance in terms of conflicts in feedback. This is because, in principle, using mapping precision estimates two clusters may be found to be disparate without necessarily having conflicts in terms of feedback.

5.3 Evaluation

To assess the effectiveness of the classification method we have just presented, we ran an experiment to answer the following questions.

1. Does the method proposed assign a new user to the cluster that meets the exact expectation of the user, if such a cluster exists?
2. Is the method more effective compared to the case where users are assigned to a cluster based on feedback they provide on tuples of their choice, as opposed to tuples that were selected?
3. Does the method proposed assign the new user to the cluster that meets best the user expectations, i.e., with a smallest distance from the user, when there is no cluster that meets the exact expectations of the user?
4. How sensitive is the method proposed to the number of clusters?

5.3.1 Experiment setup

We ran an experiment using the clustering obtained in Sect. 3. Specifically, we consider the existence of the clusters $c_1 = \{1, 2, 3, 4\}$, $c_2 = \{5, 6, 7, 8, 9\}$ and $c_3 = \{10, 11, 12, 13, 14, 15\}$, and considered that the users $1, \ldots, 15$ have provided enough feedback for the quality of the clustering to be good and for conflicts in feedback to occur between the clusters, namely 150 feedback instances. We then considered 3 new users 16,17 and 18, whose ground truth expectations are captured by the clusters c_1, c_2 and c_3, respectively. Using the method presented in this section, we generated a binary coupling tree from the dendrogram obtained using $distance_{conflict}$. Such a binary tree is of the same form as the one illustrated in Fig. 16. The tuples t_1 and t_2 referred to in the binary tree are specified in Table 3.

To compare the method presented in this section with the case in which users are assigned to clusters based on feedback they provide on tuples of their choice, we generated for each of the new users, i.e., the users 16, 17 and 18, n feedback instances that were randomly picked. We varied the amount of feedback n from 1 to 100. Given

Table 3 Tuples used to discriminate between the clusters c_1, c_2 and c_3

Tuple id	Name	Province	Country
t_1	Muenchen	Bayern	D
t_2	Swansea	Swansea	GB

Table 4 Distances between the new users and the clusters c_1, c_2 and c_3

User	Distance from c_1	Distance from c_2	Distance from c_3
16	0.08719778	0.305192231	0.578676179
17	0.16646849	0.225921522	0.49940547
18	0.325009909	0.067380103	0.340864051
19	0.509710662	0.11732065	0.156163298
20	0.721363456	0.328973444	0.055489497

the feedback generated we then computed the distances between the new user and the clusters c_1, c_2 and c_3. We then assigned the new user to the cluster with the smallest distance.

To assess the effectiveness of the method in the situations where there is no cluster that meets the exact expectations of the new user, we created four users identified by the integers 16, 17, 18, 19 and 20. The ground truth expectations of these users do not meet the exact the expectations captured by the clusters c_1, c_2 or c_3. Specifically, the distances that separate the new users from the clusters are presented in Table 4. Based on the dendrogram obtained using the distance in terms of conflict in feedback, we generated 10 binary coupling trees of the form shown in Fig. 16: the 10 trees generated differ in term of the tuples used to discriminate between clusters. To ensure that there are sufficient conflicts between user feedback to generate 10 different coupling trees, each existing user contributed 350 feedback instances. Note, however, that we needed to generate 10 trees for experimentation purposes, and that, in practice, one coupling tree should be enough. We then assigned the new users 16, 17, 18, 19 and 20 to clusters using each of the trees generated.

Finally, to examine how sensitive the results obtained by our method to the number of clusters, we repeated the above experiment by varying the number of clusters. Specifically, we considered the following numbers: 10, 15 and 20. We constructed in each case a binary coupling tree, and assigned a new user, the expectations of which were specified by randomly selecting the tuples that should be used to populate the $FavoriteCity$ relation.

5.3.2 Experiment results

The analysis of the results of the experiment showed that the three users, 16, 17 and 18, were assigned to the correct cluster, that is, the user 16 was assigned to c_1, 17 to c_2, and 18 to c_3.

The experiment also showed that the method proposed outperforms the naive approach, whereby a new users is assigned to a cluster based on feedback provided by

Table 5 The results of assigning new users to clusters

	c_1	c_2	c_3
User 16	10	0	0
User 17	6	4	0
User 18	0	9	1
User 19	1	5	4
User 20	0	0	10

the user on tuples of his/her choice. Indeed, using the second approach, the new users were assigned to the wrong cluster even when they provided 33 feedback instances. Moreover, the classification results were not stable: using the same number of feedback instances, $n \leq 33$, a new user may be assigned to different clusters. This result is in favor of the method we presented in this section: 33 feedback instances is a relatively large number compared with the 2 feedback instances that the coupling tree shown in Fig. 16 requires.

The experiment also showed that new users are likely to be assigned to the cluster that best meets their expectations, when there is no cluster that meets their exact requirement, as illustrated in Table 5. A cell in the table shows the number of times out of 10 a given user was assigned to a given cluster.

Regarding the sensitivity of the results obtained by our method, when the number of clusters is large, the experiment showed that in certain cases, the new user was not assigned to the best cluster, i.e., the cluster with the closest expectations. For example, when the number of clusters is 15, the new user was assigned to a cluster other than the best cluster 6 times out of 10. That said, the analysis of the results showed that although the cluster to which the new user was assigned was not the best cluster, the distance between the user and that cluster was small, less than 0.04.

It is worth underlining that the size of the binary coupling tree, and therefore the number of tuples to be annotated by new users, increases when the number of clusters does. In the worst case, a user may have to annotate $n-1$ tuples, where n is the number of clusters. Note, however, that for certain kinds of binary coupling trees, the number of tuples to be annotated remains low even for a large number of clusters. This is the case, for example, when the binary coupling tree is a perfect binary tree, in which all leaf nodes are at the same depth. For trees of this kind, new users annotate $log_2(n)$ tuples to be assigned to a cluster [7], e.g., when the number of clusters is 128, new users will have to annotate only 7 tuples.

6 Clustering applications

In this section, we empirically investigate two clustering applications. The first shows that feedback supplied within a cluster can be used to identify, for a given user that belongs to the cluster, the best schema mapping among a set of candidate mappings. The second application shows that clustering can be used to detect malicious users, i.e., users who provide non consistent feedback with the objective of hampering the improvement of the information integration system utilizing such feedback.

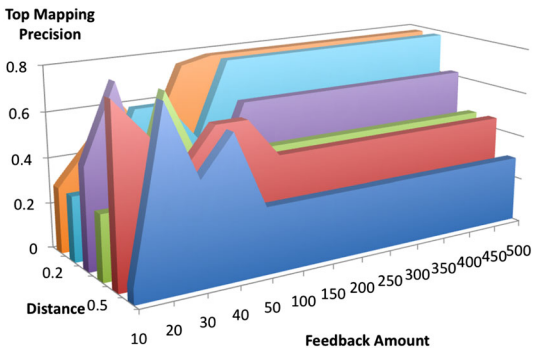

Fig. 17 The ground truth precision of the top mapping using feedback supplied by a cluster

6.1 Experiment on selecting mappings

6.1.1 Experiment objective

To assess if clustering helps in identifying the top mapping, in terms of precision, for a given user among a set of candidate mapping.

6.1.2 Experiment setup

We ran an experiment to identify the top mapping, among a set of candidate mappings, for a given user by using feedback supplied by a cluster. Specifically, we used the following setting. There are 5 candidate mappings for populating the $FavoriteCity$ relation, which according to the ground truth expectations of the user have, respectively, the following precision values: 0.07, 0.29, 0.45, 0.62 and 0.78. The experiment involved selecting the mapping with the best precision using the feedback supplied by the cluster, which is of a given distance from the user. The parameters of this experiment are, therefore, the amount of feedback supplied by the cluster and the distance between the user and the cluster.

6.1.3 Experiment results

Figure 17 illustrates the results of this experiment. It shows that when the number of feedback instances is smaller than 50 the top mapping is not stable, which can be explained by the fact that at this stage, given the feedback acquired, the precision estimates computed for the mappings are not stable. After 50 feedback instances, the choice of the top mapping is stable for most clusters. The top mapping given the feedback supplied by clusters which are of distance equal to or less than 0.2, is the top mapping according to the ground truth expectations of the user. When the distance is greater than 0.2, the top mapping selected given the feedback supplied by the cluster is different from the top mapping identified given the ground truth expectations of the user. This is a positive result, since it shows that even clusters that capture expectations that are, although similar, different from the expectations of a user can improve the experience of the user of the information integration system.

6.2 Experiment on identifying malicious users

Some users may be malicious, in that they provide feedback with the objective of hindering the improvement of the information integration system. For example, they may seek to provide feedback with the objective of increasing the precision estimates for "bad" mappings and decreasing the precision estimates for "good" mappings. We show in this section that such users can be identified through clustering. The intuition is that, using clustering, malicious users will not be grouped together with "genuine" users who provide feedback that aims to improve the quality of the information integration system. To empirically verify the validity of this hypothesis, we ran an experiment with the following objective in mind.

6.2.1 Experiment objective

To assess if malicious users can be identified by analyzing clustering results over time.

6.2.2 Experiment setup

We considered the dataset and mappings used in Sect. 3. The integration relation $FavoriteCity(name, country, province)$ and the candidate mappings that populate such a relation using data from the Mondial geographical database. We then created 6 synthetic users identified by the integers $1, \ldots, 6$ who have the same expectations. With this in mind, we specified the ground truth expectations of the users, i.e., the set of tuples that should be used to populate $FavoriteCity$, for the users $1, \ldots, 6$. The users $1, \ldots, 5$ are genuine in that they provide feedback instances that conform with their ground truth expectations, whereas the user 6 is malicious in that s/he provides feedback instances that are inconsistent with the expectations. If a tuple is expected according to the expectations, then the malicious user provides feedback specifying that such a tuple is unexpected, and vice-versa.

We then ran the following procedure multiple times by varying the value of the variable n, which represents the number of feedback instances supplied by each user, from 50 to 500.

1. For each user in $1, \ldots, 6$
2. Generate n feedback instances
3. Cluster users.

6.2.3 Experiment results

Figures 18 and 19 show the dendrograms obtained when each user provides 50 and 100 feedback instances, respectively. These dendrograms confirm our hypothesis. They show that the malicious user was not clustered with other users; furthermore, the distance between the malicious user and other users is large even when users provide relatively small amount of feedback, e.g., 50 which is less than 2 % of the dataset.

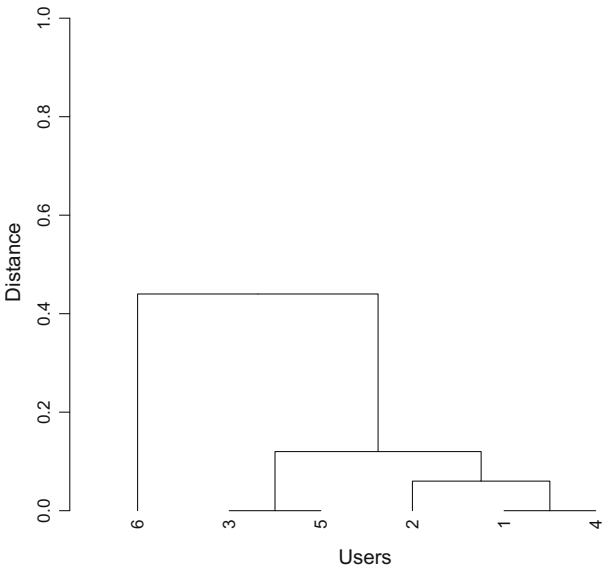

Fig. 18 Clustering obtained when each user provides 50 feedback instances

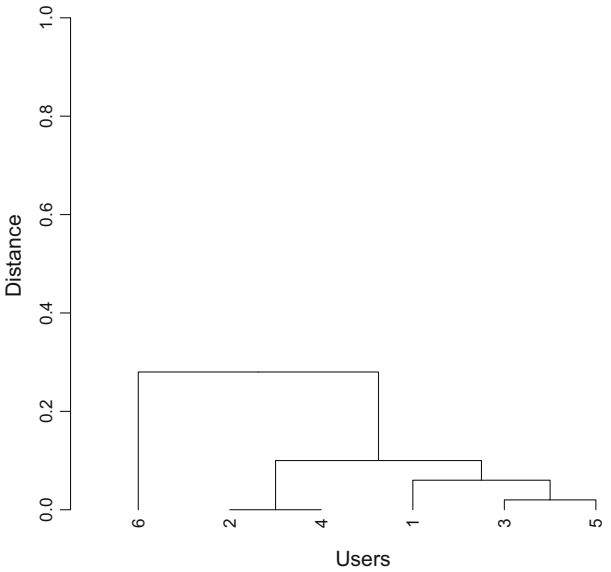

Fig. 19 Clustering obtained when each user provides 100 feedback instances

7 Related work

In this section, we compare and analyze existing work to ours. In doing so, we review proposals that tackle the problem of clustering of users, and those exploiting feedback

given by the crowd (e.g., a user community) to drive the construction and improve the quality of information integration systems.

7.1 Clustering users

The problem of clustering web users into communities has attracted a great deal of attention recently. For example, Zhou et al. [34] proposed a method for graph clustering that can be used for identifying communities in social networks. The clustering algorithm they propose considers two criteria: the graph topology and vertex properties. In a social network setting, the graph topology provides information about the relationships between individuals, whereas vertex properties describe the role of individuals. Papadimitriou et al. [26] also show that clustering can be used to identify a hierarchy of user communities in large graphs. Specifically, they developed an algorithm that constructs recursive community structures.

Clustering has also been used as a means for leveraging information integration tasks. For example, Wu et al. [33] proposes a method for matching query interfaces in the deep web. Specifically, query interfaces, typically HTML forms, are modeled as trees in which the nodes refer to field in the HTML forms. A field is characterized by a name, a label and a domain. Interfaces are matched using an agglomerative hierarchical clustering algorithm that groups together fields that are semantically similar. Barbosa et al. [2] used clustering techniques in order to organize databases in the deep web. Specifically, given a set of HTML forms, the clustering algorithm they propose groups together forms that correspond to similar databases. Similarly, Mahmoud et al. [21] employ clustering techniques to organize schemas of web databases according to their domains.

While, as the above proposals illustrate, clustering has been used to facilitate information integration tasks, the work reported in this paper is, to our knowledge, the first study that investigates the use of clustering to group information integration users according to their expectations, and to examine the benefits that can be derived from the obtained clusterings. Our work is particularly of interest to the proposals that solicit human inputs to drive information integration tasks [16,19,22,28,29]. Take, for example, the proposal by Talkudar et al. [28,29], which assists users in designing schema mappings by seeking feedback on results of alternative mappings. Our work can be of benefit in this setting to identify communities of users who have similar expectations with respect to the schema mappings, thereby reducing the workload of individual users, and to identify the expectations of new users based on a small number of feedback instances.

7.2 Crowd-based information integration systems

A handful of researchers investigated the use of crowdsourcing techniques for tackling data integration tasks. For example, Dermartini et al. [11] investigated the use of crowdsourcing for inter-linking entities. In particular, they showed how entities that are extracted from HTML documents can be connected to similar entities on the Linked Open Data Cloud using the crowd.

Similarly, Wang et al. [31] and Whang et al. [32] explored the use of crowdsourcing techniques for record linkage. Specifically, Wang et al. proposed a clustering based algorithm for generating batches of record pairs to be examined by the crowd, whereas Whang et al. focused on identifying the order in which records are to be compared.

McCann et al. [22] developed a community-based approach that solicits feedback from the community members to inform the schema matching operation. In doing so, the feedback is used to assess the matches between attributes in two schemas. For example, user feedback can be used to verify the data type of an attribute (e.g., month), or the validity of a domain constraint (e.g., the value of an attribute is always less that the value of another).

Crowd feedback has also been solicited in a way that maximizes the benefits the user can draw. For example, Jeffery et al. [19] developed a decision-theoretic framework for specifying the order in which candidate matches can be confirmed through feedback solicited from users.

While the above proposals show that crowdsourcing can be used for effectively leveraging information integration tasks, such as matching, entity resolution, and schema mapping, they assume that users of the integration system all have the same needs. In our proposal, we showed how differences in expectations between the users of an integration system can be identified through clustering. Furthermore, we presented applications, i.e., mapping selection and malicious user identification, that can benefit from the resulting clustering.

8 Conclusions

Crowd computing provides access to a new and extensive resource for use in a wide range of applications. In information management, the crowd has been used to support a range of tasks, for example in information extraction [10], matching [22], mapping [25] and entity resolution [31,32], demonstrating its potentially wide utility. However, by their nature, crowds are populated by individuals, who may have different perspectives and requirements. As such, it is potentially important when obtaining information from crowds to identify different groups of crowd users, if maximum use is to be of the information.

In this paper, we have presented clustering strategies that can be used to organize information integration users into groups with similar expectations based on the feedback they provide. Revisiting the contributions claimed in the introduction:

1. *Strategies for clustering information integration users*: we have described and evaluated distance functions that have been used with hierarchical clustering to support dependable clustering even when there is little or no feedback on identical data items. This enables users to provide feedback on whichever data items they choose, both allowing the use of unobtrusive information gathering and reducing the requirement for users to answer rather specific questions for which they may not have the necessary knowledge.
2. *Methods for sharing feedback between users:* given the clusters of users from (1), we have: (i) explored the circumstances in which, and the extent to which, benefits can be derived from sharing feedback within clusters; and (ii) described

and evaluated a technique for that for assigning new users to existing clusters efficiently in terms of the amount of feedback required.

3. *Clustering applications:* given the clusters of users from (1), we have: (i) described and evaluated how feedback associated with the users in a cluster can be used to select schema mappings that return results of relevance to individual cluster members; and (ii) shown how the clustering process can itself be used to identify users who maliciously provide incorrect information.

As such, we complement existing work on crowdsourcing for information management, which generally assumes that information obtained from the crowd is agnostic to the perspectives and requirements of individuals, by showing how clustering can be used to identify similarities between groups of users that can be built upon to support the sharing of feedback between similar users.

Acknowledgments The work reported in this paper was supported by a grant from the UK Engineering and Physical Sciences Research Council.

References

1. Albert, A.: Conditions for positive and nonnegative definiteness in terms of pseudoinverses. SIAM J. Appl. Math. **17**(2), 434–440 (1969)
2. Barbosa, L., Freire, J., da Silva, A.S.: Organizing hidden-web databases by clustering visible web documents. In: ICDE, pp. 326–335. IEEE, Istanbul (2007)
3. Belhajjame, K., Paton, N.W., Embury, S.M., Fernandes, A.A.A., Hedeler, C.: Incrementally improving dataspaces based on user feedback. Inf. Syst. **38**(5), 656–687 (2013)
4. Belhajjame, K., Paton, N.W., Fernandes, A.A.A., Hedeler, C., Embury, S.M.: User feedback as a first class citizen in information integration systems. In: CIDR. www.crdrdb.org (2011)
5. Blumenthal, L.M.: Distance geometry. In: A study of the development of abstract metrics. With an introduction by Karl Menger, Univ. Missouri Stud. **13**, 235–241 (1953)
6. Bonifati, A., Mecca, G., Pappalardo, A., Raunich, S., Summa, G.: Schema mapping verification: the spicy way. In: EDBT, pp. 85–96. ACM, New York (2008)
7. Brodal, G.S., Träff, J.L., Zaroliagis, C.D.: A parallel priority data structure with applications. In: IPPS, pp. 689–693. IEEE Computer Society, Los Alamitos (1997)
8. Cao, H., Qi, Y., Candan, K.S., Sapino, M.L.: Feedback-driven result ranking and query refinement for exploring semi-structured data collections. In: EDBT, pp. 3–14. ACM, New York (2010)
9. Chai, X., Vuong, B.-Q., Doan, A., Naughton, J. F.: Efficiently incorporating user feedback into information extraction and integration programs. In: SIGMOD Conference, pp. 87–100. ACM,New York (2009)
10. Crescenzi, V., Merialdo, P., Qiu, D.: A framework for learning web wrappers from the crowd. In: WWW, pp. 261–272 (2013)
11. Demartini, G., Difallah, D.E., Cudré-Mauroux, P.: Large-scale linked data integration using probabilistic reasoning and crowdsourcing. VLDB J. **22**(5), 665–687 (2013)
12. Dhamankar, R., Lee, Y., Doan, A., Halevy, A.Y., Domingos, P.: imap: Discovering complex mappings between database schemas. In: SIGMOD, pp. 383–394 (2004)
13. Dong, X.L., Halevy, A.Y., Yu, C.: Data integration with uncertainty. VLDB J. **18**(2), 469–500 (2009)
14. Fack, V., Lievens, S., der Jeugt, J.V.: On the diameter of the rotation graph of binary coupling trees. Discrete Math. **245**(1–3), 1–18 (2002)
15. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. **336**(1), 89–124 (2005)

16. Franklin, M.J., Halevy, A.Y., Maier, D.: From databases to dataspaces: a new abstraction for information management. SIGMOD Record **34**(4), 27–33 (2005)
17. Gal, A.: Why is schema matching tough and what can we do about it? SIGMOD Record **35**(4), 2–5 (2006)
18. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
19. Jeffery, S.R., Franklin, M.J., Halevy, A.Y.: Pay-as-you-go user feedback for dataspace systems. In: SIGMOD Conference, pp. 847–860. ACM, New York (2008)
20. Lenzerini, M.: Data integration: a theoretical perspective. In: PODS, pp: 233–246. ACM, New York (2002)
21. Mahmoud, H.A., Aboulnaga, A.: Schema clustering and retrieval for multi-domain pay-as-you-go data integration systems. In: SIGMOD Conference, pp. 411–422. ACM, New York (2010)
22. McCann, R., Shen, W., Doan, A.: Matching schemas in online communities: A web 2.0 approach. In: ICDE, pages 110–119. IEEE, Cancun (2008)
23. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: ICDE, pp. 117–128. IEEE CS, San Jose (2002)
24. Miller, R.J., Haas, L.M., Hernández, M.A.: Schema mapping as query discovery. In: VLDB, pp. 77–88 (2000)
25. Osorno-Gutierrez, F., Paton, N.W., Fernandes, A.A.A.: Crowdsourcing feedback for payasyougo data integration. In: DBCrowd, pp. 32–37 (2013)
26. Papadimitriou, S., Sun, J., Faloutsos, C., Yu, P.S.: Hierarchical, parameter-free community discovery. In: ECML/PKDD (2), pp. 170–187. Springer, New York (2008)
27. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001)
28. Talukdar, P.P., Ives, Z.G., Pereira, F.: Automatically incorporating new sources in keyword search-based data integration. In: SIGMOD Conference, pp. 387–398. ACM, New York (2010)
29. Talukdar, P.P., Jacob, M., Mehmood, M.S., Crammer, K., Ives, Z.G., Pereira, F., Guha, S.: Learning to create data-integrating queries. PVLDB **1**(1), 785–796 (2008)
30. van Rijsbergen, C.J.: Information Retrieval. Butterworth, London (1979)
31. Wang, J., Kraska, T., Franklin, M.J., Feng, J.: Crowder: crowdsourcing entity resolution. PVLDB **5**(11), 1483–1494 (2012)
32. Whang, S.E., Marmaros, D., Garcia-Molina, H.: Pay-as-you-go entity resolution. IEEE Trans. Knowl. Data Eng. **25**(5), 1111–1124 (2013)
33. Wu, W., Yu, C.T., Doan, A., Meng, W.: An interactive clustering-based approach to integrating source query interfaces on the deep web. In: SIGMOD Conference, pp. 95–106. ACM, New York (2004)
34. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. PVLDB **2**(1), 718–729 (2009)