MANCHESTER 1824

User Feedback as a First Class Citizen in Information Integration Systems

Khalid Belhajjame, Norman W. Paton, Alvaro A. A. Fernandes,
Cornelia Hedeler, and Suzanne M. Embury

School of Computer Science,
University of Manchester,
Manchester, UK

Feedback In Information Integration Systems

- Information integration is a difficult task.
  - Essentially, the difficulty lies in understanding user requirements as to what the relevant data sources are, and the way their contents should be combined and structured.

- User feedback can be used to facilitate the construction or improvement of information integration systems.
  - The Q system assists users in creating integration queries using as input feedback on query results (Talkudar *et al.*, VLDB 2008).
  - User feedback is also fundamental to dataspaces (Franklin et al., Sigmod Record 2005).

Examples of proposals that Use Feedback in the Information Integration Literature

Proposal	Objects on which feedback is given	Set of terms used for annotating objects
Alexe *et al.*	an instance of a given schema and the instance obtained by its transformation into another schema	{'yes', 'no'} // used to comment on schema transformation
Belhajjame *et al.*	a result tuple	{'true positive', 'false positive', 'false negative'}
	an attribute and its value	{'true positive', 'false positive', 'false negative'}
Cao *et al.*	a candidate query	{'true positive', 'false positive'}
	a pair of candidate queries	{'before', 'after'} // used for ordering queries
Chai *et al.*	a view result tuple	{'insert', 'delete', 'update'}
Jeffery *et al.*	a mapping	{'true positive', 'false positive'}
McCann *et al.*	a relation attribute	set of attribute data types
	two attributes of a given relation	set of constraints
	a match	{'true positive', 'false positive'}
Talkudar *et al.*	a result tuple	{'true positive', 'false positive'}
	a pair of result tuples	{'before', 'after'} // used for ordering results

Feedback In Information Integration Systems (cont.)

- While existing proposals showcase the key role user feedback can play within information integration systems, they are
  - confined to the use of feedback given on a specific artifact, e.g., query results, to tackle a specific information integration sub-problem, e.g., selecting a mapping.
  - they make assumptions that do not necessarily hold in practice, e.g., that user requirements are not subject to change over time.

- User feedback should be considered and managed as a first class citizen:
  - to foster the semantic cohesion of the feedback provided by users
  - to increase the value and the benefits that can be drawn from acquired feedback

Outline

- What is Feedback?
- Feedback Consistency and Validity
- Clustering users
- Conclusions

What is User Feedback

- Feedback can be seen as annotations that a user provides to comment on artifacts of an information integration system with the objective of informing the construction of the system and/or improving the quality of the services it provides.

- We define a feedback instance by the tuple:

$$\langle \texttt{obj}, \texttt{t}, \texttt{u}, \texttt{k} \rangle$$

  specifying that the user `u` annotates the artifact `obj` using the term `t`. the term `k` specifies the kind of feedback given.

Example

- McCann et al. developed a system that informs schema matching by soliciting feedback from users.
- As an example, consider a binary match $\langle \mathtt{r}_1, \mathtt{r}_2 \rangle$ that associates the relations $\mathtt{r}_1$ and $\mathtt{r}_2$, and consider that the user `Anhai` specified that such a match is incorrect.
- Using the model presented earlier, such feedback can be specified as follows:

$$\langle \langle \mathtt{r}_1, \mathtt{r}_2 \rangle, \mathtt{fp}, \mathtt{Anhai}, \mathtt{match\_correctness} \rangle$$

R. McCann, W. Shen, and A. Doan. Matching schemas in online communities: A web 2.0 approach. In Proceedings of the 24th International Conference on Data Engineering, pages 110–119. IEEE, 2008.

Inconsistencies in Feedback: Example

- Consider the following feedback instances which annotate values of relation attributes as true positives or false positives:

$$\mathtt{uf}_1 = \langle\langle \mathtt{r}_1.\mathtt{a}, \mathtt{v}\rangle, \mathtt{tp}, \mathtt{Norman}, \mathtt{value\_membership}\rangle$$
$$\mathtt{uf}_2 = \langle\langle \mathtt{r}_2.\mathtt{a}', \mathtt{v}\rangle, \mathtt{fp}, \mathtt{Alvaro}, \mathtt{value\_membership}\rangle$$

- Additionally, consider that there is a foreign key that links the attribute $\mathtt{a}$ of $\mathtt{r}_1$ to the attribute $\mathtt{a}'$ of $\mathtt{r}_2$.

- Given that the above constraint is not satisfied, we can deduce that uf_1 and uf_2 are inconsistent.

Inconsistencies in Feedback

- The artifacts that constitute an information integration system can be dependent on each other. Such dependencies may give rise to constraints between feedback instances, that if not satisfied results in inconsistencies between the feedback instances in question.

```
1  inconsistent(uf_1, uf_2) :−
2      -- uf_1 and uf_2 are of the same kind
3      uf_1.type = uf_2.type,
4      -- there is a pair of dependency and constraint
5      -- ⟨dep_type, const_type⟩ that are associated with
6      -- the kind of feedback uf_1.type
7      ∃ ⟨dep_type, const_type⟩ ∈ uf_1.type.getDepConstPair(),
8      -- the objects annotated by uf_1 and uf_2 are
9      -- related using the dep_type dependency
10     dependent(uf_1.obj, uf_2.obj, dep_type),
11     -- the annotations assigned by the uf_1 and uf_2
12     -- do not satisfy the const_type constraint
13     ¬constraint(uf_1.annot, uf_2.annot, const_type)
```

Feedback Validity

User requirements may change over time in which case previously acquired feedback may become invalid.

- A feedback instance $\texttt{uf}_1$ is valid at the time point `t` if it was supplied by the user before `t`, there is no conflicting feedback instance $\texttt{uf}_2$ that is fresher than $\texttt{uf}_1$ and valid at `t`, and the object $\texttt{uf}_1\texttt{.obj}$ on which feedback is given exists at $_t$.

- A feedback instance $\texttt{uf}_1$ is invalid at the time point t if there is a valid feedback instance $\texttt{uf}_2$ that was supplied before `t` and that is inconsistent with and fresher than $\texttt{uf}_1$.

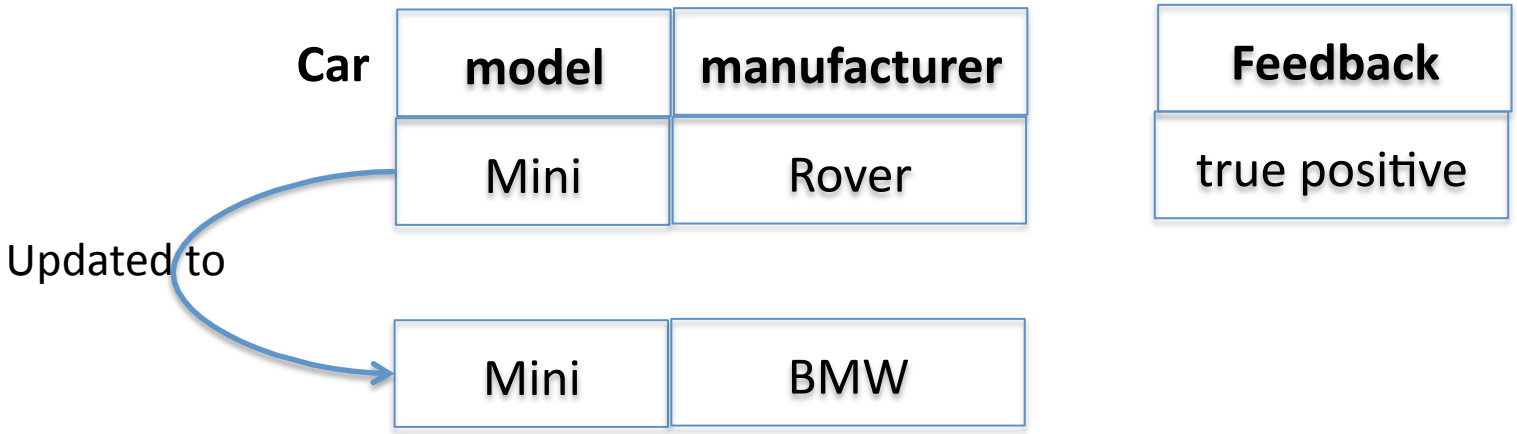
Feedback Validity (cont.)

Below is a rule defined using event calculus to determine whether a given feedback instances is valid at a certain point in time.

```
1 holdsAt(valid(uf_1), t) :-
2   -- uf_1 was supplied by the user at a time point
3   -- that is equal to or before t
4   happens(supplyFeedback(uf_1), t_1), t_1 ≤ t,
5   -- There is no valid feedback instance uf_2 that
6   -- is conflicting with and fresher than uf_1
7   not (happens(supplyFeedback(uf_2), t_2),
8        inconsistent(uf_1, uf_2),
9        holdsAt(valid(uf_2), t)), t_1 < t_2 ≤ t),
10  -- The object that uf_1 annotates exists at t
11  holdsAt(exists(uf_1.obj), t).
```

Feedback Validity (cont.)

- We do not consider feedback that was given on an object that is no longer available.
- This is because this may mean that the annotation given by the feedback is no longer valid.



Clustering Users Based on Feedback

Clustering presents the following potential benefits:

- The feedback provided by the users within a cluster can be used collectively, thereby improving the quality of the integration system.
- Clustering may reveal that the users of an existing information system have different requirements, and therefore point out the need to create multiple information integration systems to replace the existing one.
- Conversely, clustering may identify opportunities for grouping the users of two or more information integration systems into a single group, if it turns out that they have similar requirements.
- Clustering can also be used to identify outlier users within an information integration system.

Clustering: Example

scientists

What are the available proteins that are relevant to me?

Protein | name | accession | motif | species

Mappings

PedroDB

PepSeeker

Pride

GPMDB

Clustering: Example (cont.)

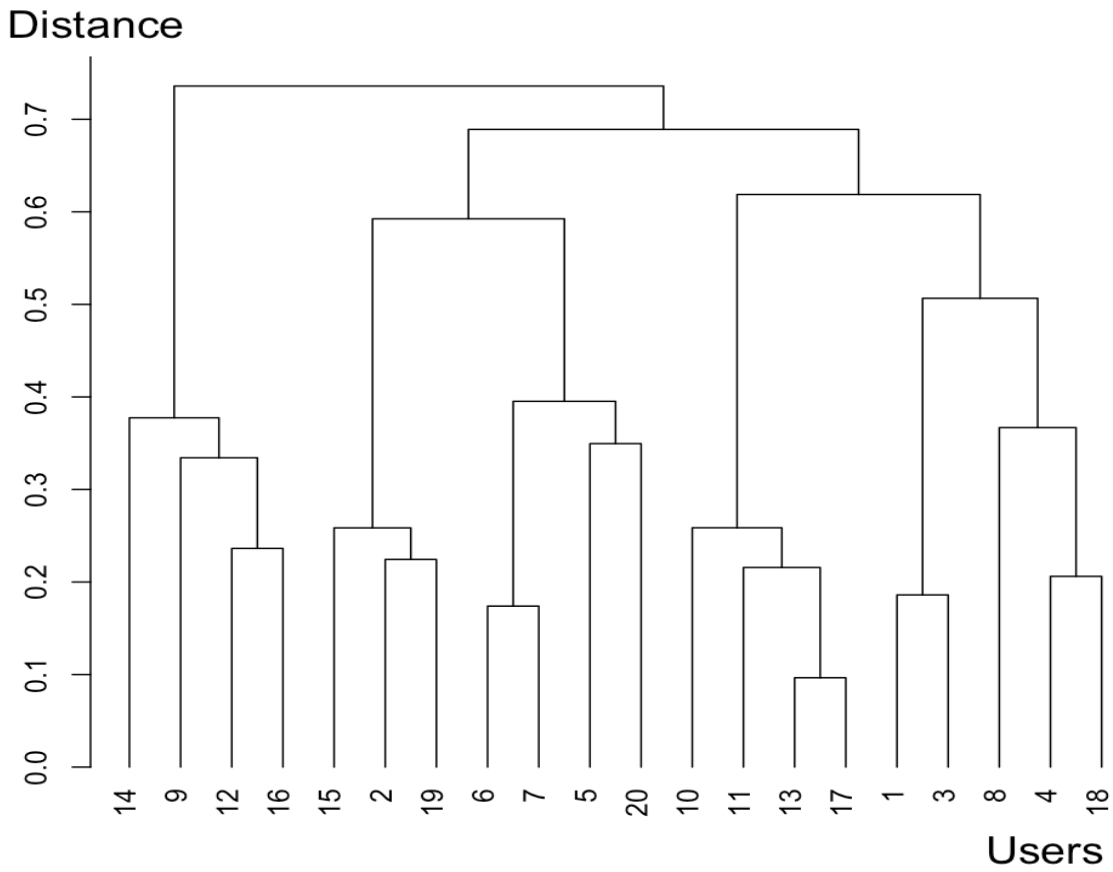
- Pattern Representation: a user's requirements can be captured using the pair ⟨E,UE⟩, where E is the set of tuples that are expected by the user, and UE is the set of tuples that are not expected by the user.

- Distance between user requirements:

$$\texttt{distance}(u_1, u_2) = 1 - w_e \times \frac{|E_1 \cap E_2|}{|E_1 \cup E_2|} + w_{ue} \times \frac{|UE_1 \cap UE_2|}{|UE_1 \cup UE_2|}$$

where w_e and w_{ue} are weights such that $w_e + w_{ue} = 1$.

Clustering: Example (cont.)



The clusters obtained can be used to annotate and select the mappings to be used to populate the protein relation.

Conclusions

On the basis of the examples provided, we have shown the benefits that can be drawn from managing user feedback as a first class citizen in information integration systems.

- Single consistent representation of feedback.
- Maximum use can be made of the limited feedback available.

Open Issues

- Feedback provides partial knowledge on users requirements
  - Uncertainty in the results produced by the operations given the feedback used as input.
- Feedback can be scarce
  - Make it difficult to check the validity of feedback, or to effectively cluster users.
- Feedback can be given on artifacts of different kinds
  - Need a means for comparing and using feedback of different kinds.