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

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

  \[ \langle \text{obj}, t, u, k \rangle \]

  specifying that the user \( u \) annotates the artifact \( \text{obj} \) 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 $\langle r_1, r_2 \rangle$ that associates the relations $r_1$ and $r_2$, 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:

$$\langle \langle r_1, r_2 \rangle, fp, Anhai, match\_correctness \rangle$$

Inconsistencies in Feedback: Example

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

  \[
  \begin{align*}
  uf_1 &= \langle \langle r_1.a, v \rangle, tp, Norman, value\_membership \rangle \\
  uf_2 &= \langle \langle r_2.a', v \rangle, fp, Alvaro, value\_membership \rangle
  \end{align*}
  \]

- Additionally, consider that there is a foreign key that links the attribute \( a \) of \( r_1 \) to the attribute \( a' \) of \( r_2 \).

- 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.

```prolog
1  inconsistent(uf1, uf2) :-
2      -- uf1 and uf2 are of the same kind
3      uf1.type = uf2.type,
4      -- there is a pair of dependency and constraint
5      -- (dep_type, const_type) that are associated with
6      -- the kind of feedback uf1.type
7      \exists (dep_type, const_type) ∈ uf1.type.getDepConstPair(),
8      -- the objects annotated by uf1 and uf2 are
9      -- related using the dep_type dependency
10     dependent(uf1.obj, uf2.obj, dep_type),
11     -- the annotations assigned by the uf1 and uf2
12     -- do not satisfy the const_type constraint
13     ~constraint(uf1.annot, uf2.annot, const_type)
```
Feedback Validity

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

- A feedback instance $uf_1$ is valid at the time point $t$ if it was supplied by the user before $t$, there is no conflicting feedback instance $uf_2$ that is fresher than $uf_1$ and valid at $t$, and the object $uf_1.obj$ on which feedback is given exists at $t$.

- A feedback instance $uf_1$ is invalid at the time point $t$ if there is a valid feedback instance $uf_2$ that was supplied before $t$ and that is inconsistent with and fresher than $uf_1$. 
Feedback Validity (cont.)

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

\begin{verbatim}
holdsAt(valid(uf1), t) :-
  -- uf1 was supplied by the user at a time point
  -- that is equal to or before t
  happens(supplyFeedback(uf1), t1), t1 \leq t,
  -- There is no valid feedback instance uf2 that
  -- is conflicting with and fresher than uf1
  not (happens(supplyFeedback(uf2), t2),
       inconsistent(uf1, uf2),
       holdsAt(valid(uf2), t2)), t1 < t2 \leq t),
  -- The object that uf1 annotates exists at t
  holdsAt(exists(uf1.obj), t).
\end{verbatim}
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.

<table>
<thead>
<tr>
<th>Car</th>
<th>model</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mini</td>
<td>Rover</td>
</tr>
</tbody>
</table>

Updated to

<table>
<thead>
<tr>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>true positive</td>
</tr>
</tbody>
</table>
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

What are the available proteins that are relevant to me?

<table>
<thead>
<tr>
<th>Protein</th>
<th>name</th>
<th>accession</th>
<th>motif</th>
<th>species</th>
</tr>
</thead>
</table>

Scientists

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:

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

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