A Semantic-Aided Designer for Knowledge Discovery

Claudia Diamantini, Domenico Potena, Emanuele Storti

e.storti@univpm.it
Organizations need methods and technologies to analyze huge amounts of data, to support decisional processes.

Knowledge Discovery in Databases (KDD) is the process of identifying valid, novel, potentially useful patterns in data:

- many steps, iterations
- interaction
- user knowledge
1\textsuperscript{st} generation of IDAs (Intelligent Data Analysis systems):
- local frameworks
- single-user
- predefined set of tools (little extensibility)

2\textsuperscript{nd} generation: distribution of tools & computational aspects

Evolution of organizations: distribution of user, collaboration

How to support the design of a KDD project in an open, distributed and collaborative scenario?
Heterogeneity & tool distribution

Many KDD and Data Mining tools available for any domain/task, many possible combinations

- Heterogeneous interfaces programming languages, OSs, transfer protocols,..

- Complex to use process design, data preparation, precondition satisfaction, I/O interpretation

- tools should be easily and dynamically added in the platform
- they should be accessible, searchable, executable via standard API
- suggestions about the best tool sequences
- support for tool setup and process execution
Issues

User distribution

Distributed organizations:
- multiple branch enterprises
- E-Science project

Collaboration:
- source of complexity
- distributed computation: several users can succeed where a single user is likely to fail

- collaborative design of KDD processes
- tool/process sharing and annotation
- easy join of new partners in Virtual Teams
Basic Services

Services for any KDD task: every KDD tool is wrapped as a Web Service, deployed on the publisher's server, and published in a common repository.

Support Services

Back-end services:
- access control
- data transfer
- service publishing
- UDDI registry

High-level functionalities:
- service discovery
- interface matchmaking
- process composition
Semantic descriptors for Basic Services

Separation of information in 3 abstraction layers

Tools/services are annotated through XML descriptors: details about interfaces and QoS

Algorithms are formally described in a KDD ontology, which contains an algorithm taxonomy and high level information about their tasks, methods and functionalities.
Benefits: loose-coupling, reusability

Support services rely on such layers:
- service discovery
- interface matchmaking
- process composition
Benefits: loose-coupling, reusability

Support services rely on such layers:
- service discovery
- interface matchmaking
- process composition
A web-based tool aimed at supporting users in collaborative KDD process design
Service discovery

Retrieval of KDD services satisfying user requirements
Service discovery

Retrieval of KDD services satisfying user requirements
Service discovery

Retrieval of KDD services satisfying user requirements
Process design

A Semantic-Aided Designer for KD
Interface matchmaking

Verification of data compatibility in an I/O connection

A Semantic-Aided Designer for KD
Interface matchmaking

**Matchmaker** service checks the validity of the match

- *syntactic compatibility*
  comparison between service descriptors
  (I/O primitive datatype and syntax)

- Output: *cost of match*
Interface matchmaking

**Matchmaker** service checks the validity of the match

- **syntactic** compatibility
  comparison between service descriptors
  (I/O primitive datatype and syntax)

- **semantic** compatibility
  comparison between ontological annotations of the services
  (kind of match between I/O, preconditions/postconditions... and many more)

<table>
<thead>
<tr>
<th>KDD ontology</th>
<th>x</th>
<th>same concept?</th>
<th>subconcept?</th>
<th>part-of concept?</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDD services</td>
<td></td>
<td>abc</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Output: *cost* of match
Semi-automatic composition

KDDComposer: advanced service for composition

Input
• user dataset
• a set of requirements (max num algorithms, computational complexity, max cost of match)
• user goal (classification, regression, ...)

Output
A ranked list of abstract processes (suggestions about processes useful to solve the user problem)
Collaboration

- collaborative process edit/annotation (wiki-style)
- versioning system
- team management and add of new users
- manual parameter setting
Conclusion

SOA for KDD
- Basic Services and Support Services
- KDD Designer: a semantic-aided designer for KDD

Open environment and heterogeneous tools
- different interfaces: need of a common representation (service)
- abstraction for an high-level description of tools (algorithm)
- semantics for interoperability and high-level functionalities

Future work
- extension with new support services
- process export in more workflow languages
- more collaborative features (real-time editor)