Managing Information Quality in e-science: the Qurator workbench

Paolo Missier, Suzanne Embury,
Mark Greenwood
School of Computer Science
University of Manchester, UK
{pmissier,sembury,markg}@cs.man.ac.uk

Alun Preece, Binling Jin
Dept. of Computing Science
University of Aberdeen, UK
{apreece,bjin}@csd.abdn.ac.uk

ABSTRACT

Data-intensive e-science applications often rely on third-party data found in public repositories, whose quality is largely unknown. Although scientists are aware that this uncertainty may lead to incorrect scientific conclusions, in the absence of a quantitative characterization of data quality properties they find it difficult to formulate precise data acceptability criteria. We present an Information Quality management workbench, called *Qurator*, that supports data experts in the specification of personal quality models, and lets them derive effective criteria for data acceptability. The demo of our working prototype will illustrate our approach on a real e-science workflow for a bioinformatics application.

Categories and Subject Descriptors: H.3.3 Information Search and Retrieval: Information filtering

General Terms: Management, Measurement, Experimentation.

Keywords: Information Quality Management, Semantic modelling of Information Quality.

1. INTRODUCTION AND MOTIVATION

A long-standing problem in e-science is that the quality of public data used in *in silico* experiments is largely unknown; this may contribute to data-intensive experiments producing misleading results, often after substantial resources have been invested. In addition to traditional data quality dimensions, such as completeness, consistency, and others that are often cited in the literature [2], e-scientists are interested in data accuracy, i.e., a quantitative measure of confidence that a given data value can be relied upon – for instance, that the published outcome of some experiment is indeed the result of a sound and correct procedure. Estimating quality levels is problematic, however, given the broad variety of potential experimental errors that may occur even on restricted experimental domains [3], combined with a lack of effective quality control procedures prior to data submission to public sites. Furthermore, few established quality estimation procedures are available for e-science data that is often itself the result of cutting-edge research conducted using new and experimental techniques. These issues make quality estimation an experimental process in its own right; its goal is to devise data acceptability criteria that are grounded in measurable evidence, for instance a detailed record of the experiment that produced the data.

As part of the ongoing Qurator project,[1] we have designed and implemented a software framework and user workbench to facilitate the definition and testing of user-defined quality models for e-science data, in a way that encourages their reuse across data-intensive processing applications with common data characteristics. A detailed description of this work can be found in [5].

The workbench functionalities are best described using the real-life case study addressed in the proposed demonstration. A typical problem in qualitative proteomics [1] concerns the identification and functional characterization of proteins from a cell sample. The experiment includes an *in vitro* portion whereby a mass spectrometer is used to quantify the peptide masses in the sample, i.e., by computing "peptide mass fingerprints" (PMF); followed by an *in silico* portion where the observed masses are matched against theoretically computed masses for a large collection of known proteins. The matching algorithm produces a list of protein hits, which are further described by a number of parameters that can be used to determine a confidence level in the match. The workflow then retrieves functional annotations for the matched proteins, to be further analyzed by the scientist. This *in silico* portion of the experiment consists of a workflow, shown in Figure 1(a), implemented in the Taverna workflow environment [4][2].

A number of potential quality issues may affect the experiment outcome, including the mis-handling of the sample in the lab, the erroneous calibration of the experimental equipment, and the incorrect matching of the peptide masses. For the purpose of our demo, we focus exclusively on the latter, and assume that the scientist would like to rank the results of the match according to personal scoring criteria to express the probability of false positives. Based on the ranking, data acceptability decisions can be made by choosing suitable thresholds on the ordered set.

Such a score model for protein hits has indeed been proposed by a group of collaborators at the Aberdeen Proteomics Facility [6]. The model has two interesting properties, namely (i) it shows excellent true positive-to-false positive ratio performance on extensive test sets; and (ii) it is computed using parameters that are common to most available PMF matching algorithms, regardless of their proprietary nature. These properties make the model an important piece of reusable quality knowledge, suggesting that it can be applied to all e-science datasets that represent protein identification by PMF, one of the most common technologies in use today.

The example just given illustrates a paradigmatic case of personal definition of a quality model that (i) is experimentally justified; (ii) it provides a quantitative measure of data reliability; and (iii) is grounded in some underlying quantities that are readily available. The goal of the Qurator framework is to facilitate the definition, adoption and re-use of such quality knowledge, by

[1]Funded by the EPSRC Programme Fundamental Computer Science for e-Science: GR/S67593 & GR/S67609 — *Describing the Quality of Curated e-Science Information Resources*

[2]http://taverna.sourceforge.net/

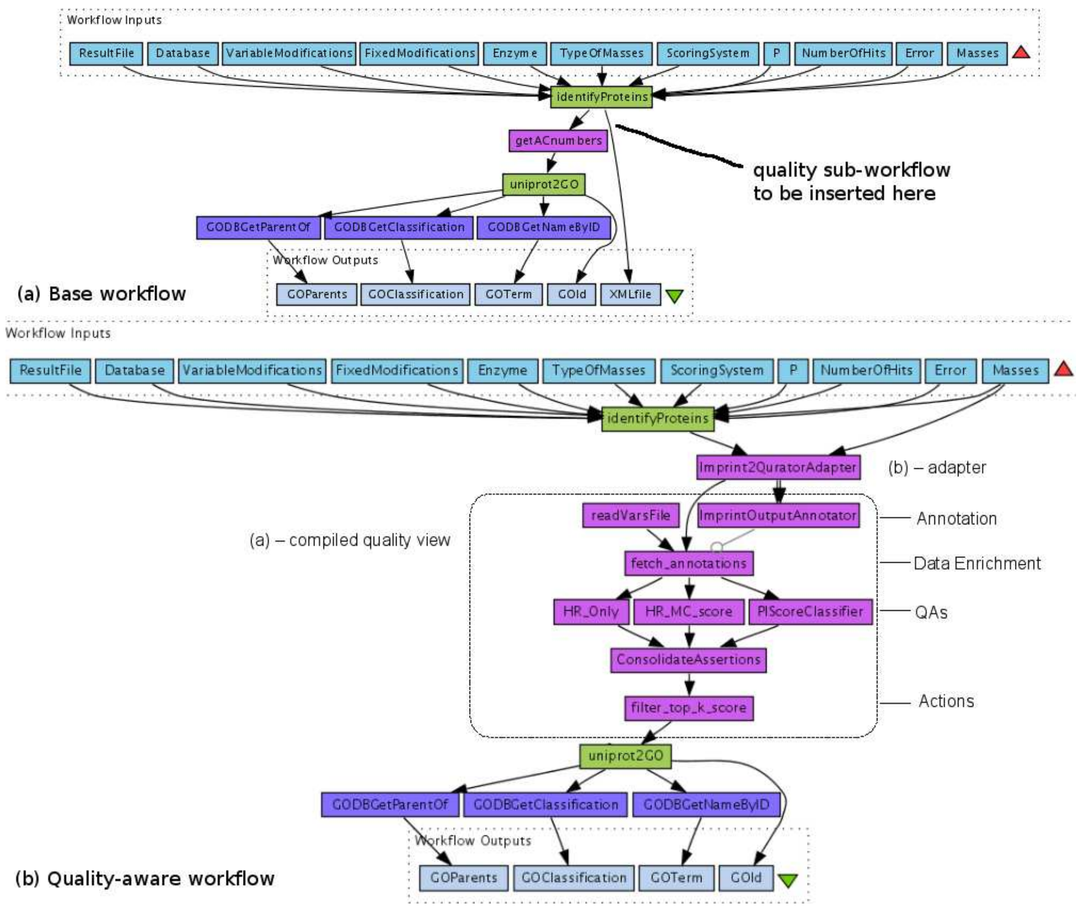

Figure 1: Baseline and quality-aware workflows for the proteomics analysis example

letting users specify, in a declarative way, its use as part of existing data processing applications. Thus, the main research question addressed in the project is whether data-intensive applications can be made quality-aware in a cost-effective way, by encoding explicit quality knowledge and by automating its processing, so that users can be presented with simple quality-based options during application execution.

Our approach can be summarized as follows. Firstly, we define a common conceptual model for Information Quality, so that personal quality models can be expressed in a formal, uniform and shareable way, independent of any implementation and data processing environment. Secondly, we provide a compilation of abstract quality models to software "quality components" that can be embedded into specific data processing environments such as a query processor or a data-driven workflow, making them quality-aware. For example, given a workflow activity for protein identification, a new quality component is compiled from a high-level specification of a quality model for protein identification data. This component provides quality control functionality on the output of protein identification, and can be deployed in combination with the original workflow. Finally, we provide a runtime environment for the execution of quality components.

One benefit of this approach is the incremental growth of a library of community-endorsed quality components, automatically obtained from shared high-level definitions, that can be used as commodity quality controls that can be added to pre-existing data processing applications.

2. THE QUALITY VIEW MODEL

The conceptual model for Information Quality (IQ) formalizes the elements presented in our example. Specifically, we assume that any dataset D can be annotated with a tuple of metadata elements, called *Quality Evidence* (QE), by way of some suitable *annotation function*, and that quantitative quality measures for D, called *Quality Assertions* (QA), can be computed as functions of the QE. Using this model, one may state that data for some domain-specific type, for instance "PMF identification hit", can be annotated with particular types of quality evidence, such as `Hit Ratio`, `Coverage`, and `ELDP`,[3] using an annotation function that computes the evidence values using information derived from the output of the protein matching algorithm.

Furthermore, the input requirements of QAs, i.e., the quality evidence types, and the type of their output are also defined. As we have seen, an example of QA for a dataset D is a score model; in general, a QA is any function of evidence that defines a partial order on D. Such an ordering can be obtained for instance by partitioning D into equivalence classes, and by asserting an order among the classes, e.g. $\texttt{reject} \leq \texttt{investigate} \leq \texttt{accept}$. In this case, the QA is a classifier that associates class labels to the elements of D, taking as input an attribute vector of associated evidence.

Once a partial order is defined, acceptability criteria on elements of D can be specified as a set of conditions/actions pairs on the val-

[3]The exact definition of these quantities is irrelevant here. We refer the interested reader to [6] for details.

ues of the QA, such that when a condition evaluates to true, the associated action is performed. For a classifier QA, for example, one may simply associate an action to each class. While simple types of actions like the acceptance / rejection of a data element could be embedded in a query processor, more specific actions depend on the data processing environment in which the quality model is evaluated. In the case of a workflow application, for example, actions may be specified as workflow tasks, so that data in class `investigate` could be sent to a task that performs some additional and possibly interactive analysis on the data, to determine its suitability.

We use the term *Quality View* (QV) to denote a specification of one or more QAs for a certain data type, along with the QAs' input evidence requirements, the annotation functions that compute the annotations, and the condition/actions pairs that define data acceptability. The Quality View specification used in the demo includes the score model described in our example (as well as two additional score models, for comparison), based on the three evidence types mentioned earlier.

2.1 Semantic modelling of Quality Views

The conceptual IQ model is expressed as an ontology in the OWL-DL Semantic Web language.[4] By adopting this very expressive representation model, the Description Logic operators available in OWL can be used to express constraints among the elements of a Quality View, for example the input type requirements and the evidence types computed by annotation functions, and the input requirements for Quality Assertions. Based on these constraints, we can define *consistent* Quality Views, as those views that satisfy all axioms defined in the ontology. We can then exploit standard Description Logic reasoning capabilities, like those of Pellet[5], to validate a user-defined Quality View, as well as to issue recommendations to the user while constructing a QV. This functionality is important to support users in correctly adopting third-party QAs that are defined in the ontology.

2.2 Compiling abstract quality views

Quality Views, expressed in Qurator using an XML syntax, are abstract, in the sense that their specification is based exclusively on the IQ model, regardless of the implementation of the functions involved, and of the software environment in which the data is processed. As mentioned, Qurator provides functionality for compiling abstract Quality Views to software components for particular target data processing environments. In the demo, we show compilation into "quality workflows" that can be managed from within the Taverna workflow definition environment mentioned earlier.

The compilation step involves mapping abstract Quality View elements to concrete implementations, by binding each QE and QA defined in the ontology to a Web service. Qurator defines a single WSDL interface that all QA functions must implement, and maintains a registry of available QE and QA service implementations.

The result of compiling our example Quality View is shown in Figure 1(b), where the quality flow has been embedded within the original workflow (the embedding is described by a deployment descriptor, which specifies the necessary connections between the original tasks and the quality tasks). The workflow is an instance of a generic quality process pattern defined within Qurator, and described in detail in [5]. In particular, invocations of QE and QA Web services are mapped to specific workflow tasks. Note that QE services store computed evidence annotations into a database, where the *Fetch Annotations* task can retrieve them.

[4]OWL – http://www.w3.org/TR/owl-guide/

[5]Pellet – http://www.mindswap.org/2003/pellet/

3. QURATOR DEMONSTRATION

Creating an effective Quality View is an experimental process involving a number of steps, from the original definition or the discovery of third-party Quality Assertions, to selecting data acceptability conditions and actions, to testing their effect on application data. Our current Qurator implementation supports the specification of Quality Views and their compilation into quality flows, and provides runtime services to interactively test various data acceptability conditions while executing the flow. The following specific functionalities are demonstrated, using our proteomics experiment as an example:

• A GUI for the specification of consistent Quality Views. In addition to providing user-friendly access to the ontology classes that define the quality concepts, the GUI exploits the reasoning functionality of Pellet to (i) validate a user specification, and (ii) offer suggestions to users for creating a complete consistent QV, starting from any user-defined partial QV configuration;

• The transparent compilation of the Quality View into a Taverna workflow;

• The specification of a deployment descriptor, and its use to embed the quality flow into the original flow;

• The execution of the quality-enhanced workflow on a test dataset. This includes an interactive decision step for testing the practical impact of the QV on the dataset. In this step, users may write and evaluate conditions on the computed Quality Assertions for the dataset, triggering the associated actions. Note that our current implementation does not, at present, provide support for the specification of Quality Assertions, i.e., for the definition of new classifiers or scoring models – a task that is best supported by existing machine learning environments, e.g. Weka [7] among others.

4. REFERENCES

[1] R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422:198–207, March 2003.

[2] C. Batini and M. Scannapieco. *Data Quality – Concepts, Methodologies and Techniques*, volume XX of *Data-Centric Systems and Applications*. Springer, 2006.

[3] C. Hedeler and P. Missier. *Database Modeling in Biology: Practices and Challenges*, chapter Quality management challenges in the post-genomic era. Artech House, 2007. In print.

[4] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34:W729–W732, 2006.

[5] P. Missier, S. M. Embury, M. Greenwood, A. D. Preece, and B. Jin. Quality views: Capturing and exploiting the user perspective on data quality. In *VLDB*, pages 977–988, Seoul, Korea, September 2006.

[6] D. A. Stead, A. Preece, and A. J. Brown. Universal metrics for quality assessment of protein identifications by mass spectrometry. *Molecular & Cellular Proteomics*, 5(7):1205–1211, 2006. Also available at http://www.mcponline.org/papbyrecent.shtml.

[7] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.