2010-10

New Trends in Automatic Assessment: Ontology Matching

Maria Mitina
Technological University Dublin, mmitina@itnet.ie

Patricia Magee
Technological University Dublin, patricia.magee@tudublin.ie

John Cardiff
Institute of Technology, Tallaght

Recommended Citation
Mitina, M., Magee, P. and Cardiff, J. New trends in automatic assessment: Ontology matching. The IT&T 10th International Conference on Information Technology and Telecommunication 2010 RESEARCH—Driving the Knowledge Economy. Letterkenny

Funder: Institute of Technology Tallaght

New Trends in Automatic Assessment: Ontology Matching

Maria Mitina[1], Patricia Magee[2], John Cardiff[2]

Department of Computing
Institute of Technology Tallaght Dublin
Dublin, Ireland
[1] mmitina@itnet.ie
[2] {patricia.magee, john.cardiff}@ittdublin.ie

Abstract

Instant individual feedback represents a result of assessment which allows for considerable improvements in both teaching and learning. In this paper we present the application of ontology matching techniques in automatic correction of students' answers for SQL tests, which will provide teachers with instant feedback to facilitate manual correction and marking and which they can pass to the students. Students experience many problems learning SQL due to the necessity to memorise database schemas, unclear feedback from the database engine on the execution of the query, etc. The program environment utilising the described approach is designed to solve the abovementioned problems in learning SQL and is currently under development; it will be used to generate and deliver automatic correction capabilities to teachers through detailed feedback.

Keywords: Assessment, SQL, Adaptive Feedback, Ontology, Ontology Matching.

1 Introduction

As one of the key components in the learning process, assessment allows for the improvement of both teaching and learning through feedback. There are two types of feedback – summative and formative. Summative feedback is the result of summative assessment – assessment for grading – and is usually provided at the end of the learning course, whilst formative feedback represents the result of formative assessment targeted on the improvement of learning outcomes during the learning course. Feedback on success or failure of the teaching objectives helps teachers in the improvement of the teaching strategy. Feedback allows students to improve the comprehension of the learning material [1] and motivates them for future learning [2].

The traditional approach with manual assessment and subsequent feedback from the teacher can be quite time consuming with a large number of students [3]. This paper presents the application of ontology matching techniques in order to provide students with instantaneous individual feedback.

Following [4], an ontology is a formal structure, which includes "a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms." Ontologies are applied in a wide range of areas from medical applications to learning systems. One of the most popular environments to construct and manipulate ontologies is the ontology editor Protégé [5], developed at Stanford University, CA, was used to create these ontologies.

Ontology matching is the process of determining correspondences between concepts, which are typically expressed as labels for data. One of the most critical applications of ontology matching and numerous ontology matching techniques is the interpolation of heterogeneous resources [6].

Structured Query Language (SQL) is the dominant language for database manipulation today. Although SQL is a simple and highly structured language, students experience many problems learning it. The causes of these problems include the necessity to memorise database schemas, unclear feedback from the database engine on the execution of the query, etc [7]. In this paper the authors describe the

utilisation of SQL as an application domain for ontology matching. Ontology matching techniques can help to provide students with valuable instant and detailed feedback during or after an assessment by revealing correspondences and discrepancies between ontological representations of the student and the correct answer. Three tools for matching ontologies – S-Match, FOAM, and PROMPT – were explored and the latter was selected to solve the described task.

2 Background

Several systems for assessment of SQL skills are described in the literature related to this subject and these will be discussed in this section.

AsseSQL [8], developed by the researchers from the University of Technology, Sydney, executes the student's answer on a test database and compares with the teacher's results. This system provides simple binary feedback indicating if the answer is correct or incorrect.

The SQLify system [9] created at the Department of Mathematics & Computing at the University of Southern Queensland provides rich feedback to students in a semi-automated fashion and uses seven levels of correctness to check the student's answer. The system utilises automatic, peer and teacher review and provides feedback to the test-taker only after all the reviews are completed, which may take several hours.

The SQL-KnoT [10], developed in the School of Information Sciences at the University of Pittsburgh, employs the use of ontologies to describe the domain knowledge. This allows for better shareability, re-usability and enrichment of the domain data. The system provides the student with a task and a schema of a working database, the answer is executed on a test database and simple feedback from the database engine is presented to the student instantly.

SQL-Tutor [11], developed at Computer Science Department of University of Canterbury in New Zealand, provides students with the instant, quality, and rich feedback utilising a constraint-based model. The system checks the student's answer against a set of constraints on the SQL query, the total number of which reaches 700. When a constraint is violated, the student receives a feedback message assigned to this constraint. Here a constraint-based model written in LISP is used to describe the domain.

A prototype of the tool for automatic correction of SQL queries SQL-CT (SQL Correction Tool) is developed within the framework of the current research. The main purpose of the tool is the provision of error correction and detailed feedback to the teacher and the student. Unlike AsseSQL, SQLify and SQL-KnoT, the tool will be looking at both syntax and semantics of the SQL query and the correction and feedback will be delivered to the teacher and the student instantly. Despite the outcomes of SQL-CT are similar to the ones of SQL-Tutor, it utilises ontologies to build the constraint base instead of LISP and ontology matching techniques to verify the constraint violation. The authors propose that the utilisation of an ontology to describe the domain and constraints could provide for better portability and re-usability of the domain data along with a more convenient way to expand the constraint base.

3 Utilisation of Ontologies

SQL-CT will provide a problem-solving environment to complement SQL lectures in a classroom and can be used for grading purposes. The current research focuses on correction and the provision of feedback to the teacher. Ontology matching is used to check the syntax and semantics of the students' answers and retrieve the feedback message. The architecture of the SQL-CT is illustrated in Fig. 1.

In this section we describe the modules which create and utilise ontologies – the Ontology Generator and Ontology Matching modules. SQL-CT leverages three types of ontologies – the main ontology of the SQL domain, the ontology of the correct answer, and the ontology of the student's answer.

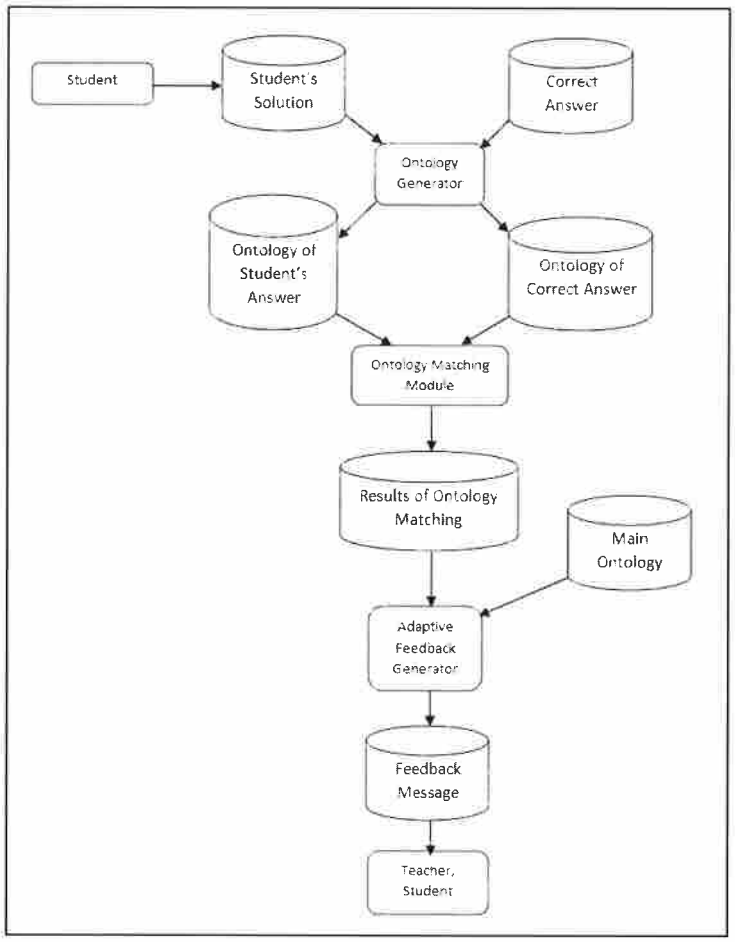

Figure 1. **The Architecture of the SQL-CT**

The main ontology of the SQL domain was developed in order to provide and organize the information about the concepts which were taught within the learning course. These concepts of SQL include the main operators and keywords of SELECT, FROM and WHERE clauses. Fig. 2 outlines an example of the application of the WHERE clause concepts used to create the main ontology.

"WHERE" Clause Operators

Arithmetic and Text Operators

=	Equal to
<>	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

SQL Operators

BETWEEN.. AND..	Between two values (inclusive)
IN (list)	Any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Figure 2. **Examples of SQL concepts.**

The SQL concepts are represented with classes, subclasses and individuals in the main ontology to reflect the logical structure of SQL. For example, every aggregate function in SQL such as AVG, SUM, MIN, MAX, and COUNT will be represented with a corresponding individual in the main ontology. At

the same time these functions belong to a set of Aggregate functions and will be individuals of the class AggregateFunctions of the main ontology. Aggregate functions can be utilized in both SELECT and WHERE clauses, thus the class AggregateFunctions will be a subclass of two other classes – SELECTClause and WHEREClause.

Typically individuals have various properties assigned to them. As defined in [12] "properties are binary relations on individuals - i.e. properties link two individuals together" and in the main ontology of the SQL domain they serve a number of purposes. For example, the property "isUsedIn" specifies which class or subclass an individual belongs to and thus helps to refine the representation of the logical structure of SQL. According to the specification of SQL two concepts – RIGHT OUTER JOIN and RIGHT JOIN are used equally and the "hasAllowedSynonym" property defines this relation (this property says that students are allowed to replace one of the concepts with another one in their answer). The "hasNotAllowedSynonym" property links "=" and LIKE functions, which means that substitution of "=" with LIKE and vice versa is not allowed as it will lead to a narrower set of results from the database. Properties generate corresponding constraints on the use of the concepts in the student's answer. These constraints are question independent and can be applied to any task of a test. Fig. 3 provides examples of properties and individuals linked with their help.

<sql:WhereOperator> <sql:isUsedIn> <sql:SelectStatement>

<sql:LIKEfunction> <sql:isUsedIn> <sql:WhereClause>

<sql:RIGHT_JOINOperator> <sql:hasAllowedSynonym> <sql:RIGHT_OUTER_JOINOperator>

<sql:LIKEfunction> <sql:hasNotAllowedSynonym> <sql:Equalsfunction>

Figure 3. **Examples of properties in the main ontology of the SQL domain.**

The ontology of the correct answer and the ontology of the student's answer – are programmatically generated by the Ontology Generator in real time. The correct answer is sourced from a file which is prepared by the teacher and contains correct solutions for all questions in the test, whilst the student's answer is obtained at runtime. Both answers are parsed in the Ontology Generator module and SQL concepts are retrieved from these two strings. Java API and OWL API are used to generate the two ontologies in real time.

Fig. 4 shows the ontology of the correct answer and the ontology of the student's answer opened in Protégé. The two ontologies are different in the use of SQL concepts – the student used "Max" instead of "Min", did not use "Avg" in the Select clause, provided "=" ("Equals_To" in the picture) instead of the "Like" in the Where clause and did not Order the result in the answer.

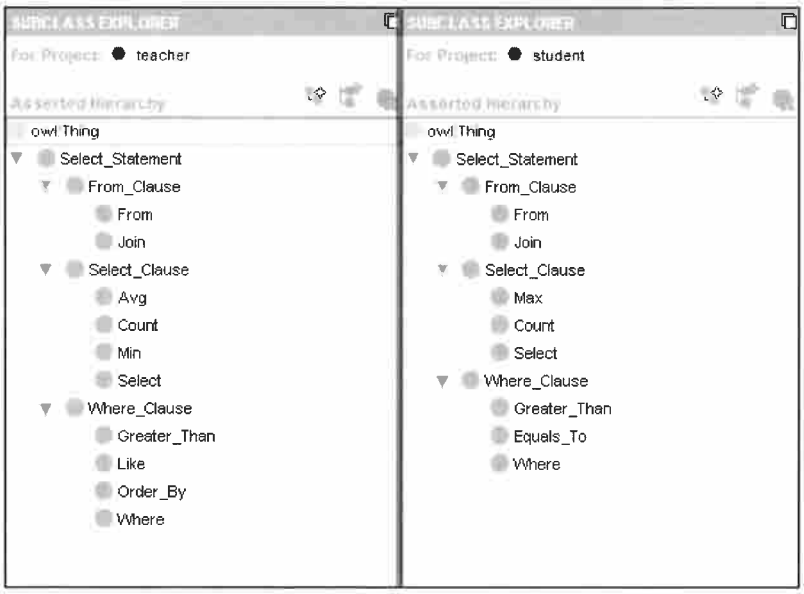

Figure 4. **Ontologies of the teacher's and student's answers.**

These ontologies contain only those SQL concepts, that were retrieved from the correct solution and the student's solution. The concepts become individuals of the classes of the ontologies. Classes and subclasses are appended according to the known logical structure of the SQL statement, which always includes the SELECT, and the FROM clauses and may contain the WHERE clause.
Properties are not assigned to the individuals at runtime as it is a time consuming process, instead, they are contained in the main ontology and used at a later stage.

4 Ontology Matching Technique

Each answer provided by the student requires verification of syntax and semantics. Whilst verification of the syntax is provided by the majority of database management systems, almost none of them allows for the verification of the semantics. SQL-CT utilises detection of the constraint violation for this purpose.
The selected approach to trigger the check for the constraint violation involves the comparison of the two runtime ontologies. The discrepancies between them are revealed with the help of the ontology matching algorithm described below.
Ontology matching is currently an area of development and research and it aims to find correspondences between semantically related entities of different ontologies. These correspondences may signify equivalence as well as other relations, such as consequence, subsumption, or disjointness, between ontology entities [13].
Three of the existing solutions in the area of ontology matching were investigated – S-Match [14], FOAM [15], and PROMPT [16]. S-Match is a semantic matching framework developed by the researchers from the University of Trento, Italy. This tool provides several semantic matching algorithms and facilities for developing new ones. FOAM (Framework for ontology alignment and mapping) is a tool to fully or semi-automatically align two or more OWL ontologies based on heuristics (similarity) of the individual entities (concepts, relations, and instances). PROMPT is a plug-in for the ontology editor Protégé which allows for comparison of different versions of the same ontology, the merging of two ontologies into one and extraction of a part of an ontology.
S-Match was investigated on the two test ontologies – the initial and resulting one. The resultant data contained the comparison of each concept of the initial ontology to each concept of the resulting ontology. As the structure of the ontology is not taken into consideration the tool provides an excessive dataset, which is difficult to interpret.
The FOAM tool is designed for programmatic use. It is easy to integrate and launch from a program. The result of this tool is a log file with the percentage of similarity between the two concepts in both ontologies. This data requires extensive data analysis and conversion before it can be used in the current task.
PROMPT proved to be the most relevant to the aforementioned task both in terms of the quality and the type of feedback and the possibility to interpret it. This tool was selected for the implementation of the selected ontology matching approach for automatic correction.

5 Background and Utilisation of PROMPT

A brief description of the PROMPT plug-in for Protégé along with the detailed description of its comparison algorithm and feedback will be provided in this section.
The PROMPT plug-in was developed by researchers from Stanford University, CA, as a part of the ontology editor Protégé. The first and the current stable version of the tool is 3.0 and it is compatible with many versions of the Protégé editor including the most widely used 3.4.4. PROMPT and Protégé are open-source projects and their code is free to download from the Stanford repository. All these advantages allow for PROMPT to become a convenient and available solution for development of custom tools and programs.
PROMPT is represented as a tab in Protégé and provides users with a comprehensive graphical user interface shown in Fig. 5.

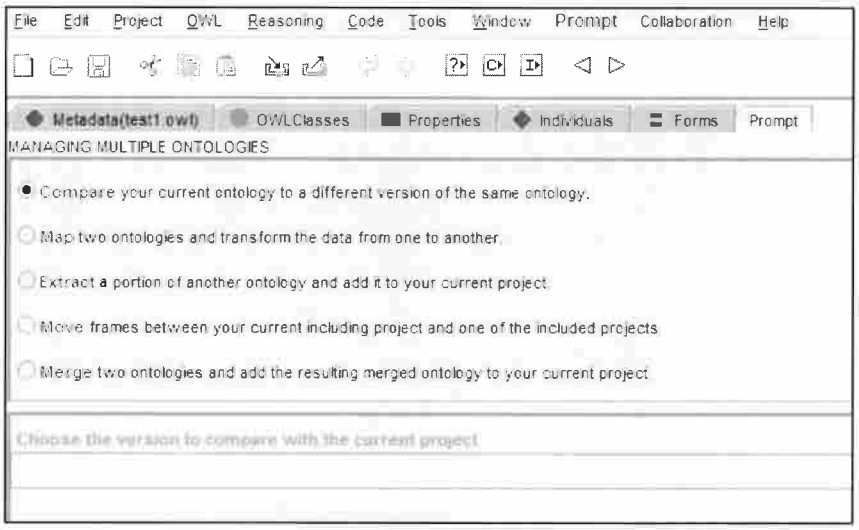

Figure 5. **PROMPT GUI in Protégé 3.4.4.**

SQL-CT launches the PROMPT tool without the GUI. The ontologies of the student's and the correct answers represent a source and a target ontology respectively. They are required to accomplish any of the five operations offered by PROMPT, including mapping of two ontologies and transformation of the data from one to another; extraction of a portion of the source ontology and addition to the target project; moving frames between the ontologies; merging two ontologies and addition of the resulting merged ontology to the current project; and finally comparison of the source and target ontologies, which is used as a base for the proposed technique for automatic correction.

A specifically designed PROMPT DIFF algorithm is used for the comparison, while the source ontology is treated as a newer version of the target ontology.

When PROMPT DIFF is launched by SQL-CT, the two runtime ontologies are loaded into memory, their elements are compared step by step and the structure is taken into account. You can see a detailed runtime log produced by PROMPT DIFF in Fig. 6.

```
Diff started:  Mon Aug 23 14:42:46 BST 2010
Loaded versions:  Mon Aug 23 14:42:46 BST 2010
Using id slot:  null; checking id slot only: false
********** start CompareFrameNamesAndTypes ************
********** start CompareOWLAnonymousIndividuals ************
********** start UnmatchedInverseSlot ************
********** start SingleUnmatchedDomain ************
********** start CompareOWLIntersectionAndUnionClasses ************
********** start SlotsWithSameAllowedClass ************
********** start SingleUnmatchedAllowedClass ************
********** start SameRestrictionTypeForSamePropertyAtSameClass ************
********** start LoneUnmatchedRestrictionOfSameType ************
********** start SplitClasses ************
```

Figure 6. **Step by step comparison conducted by PROMPT DIFF.**

The result of PROMPT DIFF's work is a short log shown in Fig. 7.

```
Unmatched entries from ontology 1: 2
Unmatched entries from ontology 2: 3
Rows without rename in the table: 292
Rows with rename in the table: 6
Unchanged rows in the table: 292
Isomorphic rows in the table: 0
Changed rows in the table: 0
```

Figure 7. **Final short log by PROMPT DIFF.**

More detailed feedback which can be used for the task of automatic correction and assessment is retrieved from an array of PROMPT DIFF results. This feedback contains hundreds of lines with information not only about the target entities but about all minor changes (and their absence) in the background structure, which is used to describe the ontologies. That is why this feedback requires further processing and interpretation.
When the ontologies of the correct and student's answer are compared, two types of operations are observed – "Add" and "Delete" – indicating that the student added or missed some SQL concepts in his answer. An example of the detailed feedback from PROMPT DIFF is shown in Fig. 8.

```
f1: null, f2: DefaultOWLNamedClass(http://test1.owl#GreaterThan), r: No, operation: Add
f1: null, f2: DefaultOWLNamedClass(http://test1.owl#Where), r: No, operation: Add
f1: null, f2: DefaultOWLNamedClass(http://test1.owl#WhereClause), r: No, operation: Add
f1: DefaultOWLNamedClass(http://test13.owl#Join), f2: null, r: No, operation: Delete
f1: DefaultOWLNamedClass(http://test13.owl#On), f2: null, r: No, operation: Delete
```

Figure 8. **Detailed feedback of PROMPT DIFF.**

This information from the detailed feedback of PROMPT DIFF will be applied to the properties of the main ontology further in order to check the violation of constraints.

6 Conclusions and Future Work

The selected approach to automatic correction of students' answers in SQL tests leverages the ontology of the domain knowledge, which allows for better portability, shareability and re-use of the domain data. Ontology editors provide a fast and easy way to add new classes, individuals and properties, thus refining the ontology and enriching the set of constraints, which results in more relevant and precise feedback.
New properties have been added to the main ontology to enrich the constraint base and the PROMPT tool has been integrated into the main module of the program. Assessment of SQL skills was conducted in the Institute of Technology of Tallaght Dublin where 36 students took part. The participants were given a test containing 13 questions and had to provide SQL queries to satisfy the given task. The students were allowed to use database management system to verify their results. Later the anonymous answer queries were analysed and a set of standard errors formed. This data is being used in ongoing tests of PROMPT to establish typical feedback for standard errors as part of development of the module that is responsible for the interpretation of PROMPT's log. The next module utilised in this pipeline is Adaptive Feedback Generator also shown in Fig. 1. The Adaptive Feedback Generator will detect violated constraints relying on the output of the ontology matching module and the main ontology of the SQL domain. Each property in the main ontology has a message assigned to it, which is presented to the student when the constraint is violated. The implementation of the Adaptive Feedback Generator is the next and the final step in the development of the first prototype of the tool for automatic correction of students' answers.
This research set out to demonstrate the benefits that an automated assessment tool can have in improving the current ways to correct and mark student's results in assessment of SQL skills and in follow-up research surveys will be undertaken to assess this impact and new ways will be investigated to expand the functionality of the tool.

References

[1] Morris, M., Porter, A., Griffiths, D., (2003). Assessment as a tool for learning. A Snapshot of a Work-in-Progress. University of Wollongong.

[2] Black, P., Wiliam, D., (1998). *Assessment and Classroom Learning*. Assessment in Education, March, p7-74.

[3] Black, P., (1999). *Assessment, learning theories and testing systems*. In Patricia Murphy (ed.), Learners, learning and assessment. London, Paul Chapman publishing, pp. 118-134.

[4] Uschold, M., (1998). Knowledge level modelling: Concepts and terminology. Knowledge Engineering Review, 13(1).
[5] Erikssonc, H., Noy, N., Tu, S., (2003). The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies.* Volume 58, Issue 1, January 2003, Pages 89-123.
[6] Ontology alignment, (2010), Wikipedia. [Internet] Wikipedia contributors. Available from: <http://en.wikipedia.org/wiki/Ontology_alignment>. [Accessed 27 September 2010].
[7] Mitrovic, A., (1997). SQL-Tutor: a preliminary report. Computer Science Department, University of Canterbury.
[8] Prior, J., Lister, R., (2004). The backwash effect on SQL skills grading. *In proceedings of ITiCSE'04*, pages 32-36, Leeds, UK.
[9] Raadt, M., Dekeyser, S., Lee, T., (2007). A system employing peer review and enhanced computer assisted assessment of querying skills. *Informatics in Education*, 2007, Vol. 6, No. 1, 163–178.
[10] Brusilovsky, P., et al, (2008). An Open Integrated Exploratorium for Database Courses. *ITiCSE'08*, June 30 - July 2, 2008, Madrid, Spain.
[11] Mitrovic, A., (1998). A Knowledge-Based Teaching System for SQL. Computer Science Department, University of Canterbury.
[12] Horridge, M. et al, (2004). A practical guide to building OWL ontologies using the Prot´eg´e-OWL plugin and CO-ODE tools edition 1.0. The University of Manchester.
[13] Euzenat, J., Shvaiko, P., (2007). Ontology matching. Springer.
[14] S-Match – Semantic Matching, (2010), S-Match. [Internet]. Giunchiglia, F., et al. Available from: < http://semanticmatching.org/s-match.html>. [Accessed 27 September 2010].
[15] Ehrig, M., Sure Y., (2005). FOAM – Framework for Ontology Alignment and Mapping Results of the Ontology Alignment Evaluation Initiative. *Proceedings of the Workshop on Integrating Ontologies*. Volume 156., CEUR-WS.org.
[16] Prompt, (2009), Protégé Wiki. [Internet]. Noy, N., et al. Available from: < http://protege.cim3.net/cgi-bin/wiki.pl?Prompt >. [Accessed 27 September 2010].