RubricAce ™: A Case-based Feedback Recommender for Coursework Assessment*

Nirmalie Wiratunga[1], Ibrahim Adeyanju[1], Paul Coghill[1], and Clemence Pera[2]

[1] School of Computing, IDEAS Research Institute,
The Robert Gordon University, Aberdeen, United Kingdom
[n.wiratunga|i.adeyanju|p.j.coghill]@.rgu.ac.uk
[2] Computing Department, Institute of Technology,
Universite d'Auvergne, Clermont-Ferrand, France
clemence.pera@gmail.com

Abstract. Coursework based student assessment is a core task in schools and higher education institutions. Grading coursework is commonly guided by means of a pre-defined marking scheme commonly known as a Rubric. The aim of the Rubric is two-fold: to help students understand the expectation for achieving each of the grades along one or more specific judging criterion; and to help assessors provide contextualised individual feedback. Accordingly the assessor would normally be expected not only to assign grades but also to explain this assignment through individual feedback text. Aim of RubricAce is to support assessors by providing feedback recommendations. It does this by adopting the CBR paradigm of reusing feedback allocated to previous similarly graded students for a given coursework. Whilst this allows RubricAce to mimic human behaviour, in doing so it helps to maintain consistency and fairness; hence, improving overall feedback quality. RubricAce also includes management tools to visualise similarly graded students, charts to view the distribution of grades and functionality to import and export student records. Initial demonstrations of RubricAce functionality to academic staff at the Robert Gordon University has been well received. However a more comprehensive user study is planned for the future.

Keyword. Text Reuse, CBR and education, Text generation, Rubric assessment

1 Introduction

The task of authoring documents that include pre-defined attributes along with some textual content is common to several applications. Such documents typically include reviews, student feedback, medical notes and incident reports. Authoring feedback is easiest when responding to structured content such as assigning a percentage score, ticking checked boxes, rating things on a qualitative scale or just picking from a list rather than responding to a feedback box. However it is precisely this form of open-ended textual content that helps to individualise student feedback albeit at the expense of assessor time.

Textual case base reasoning (TCBR) [27] is a research area that deals with solving new problems by reusing previous similar experiences documented as text. Such problem-solving experiences are reused to assist new users by adapting previous solutions whose problems are similar to the current problem. TCBR is particularly well placed to support authoring of textual content particularly when it can be guided by other structured attributes. For instance, it can recommend relevant text provided that there is sufficient similarity in the structured content between previous cases and the current student being assessed.

Evaluation and feedback is a core mechanism for assessing a student's ability to perform a skill [10]. A student's satisfaction can be adversely affected by the quality of feedback they receive since dis-satisfied students can be demoralised and lose confidence in their ability, possibly leading them to discontinue the course. Equally useful feedback facilitates learning by helping student's to understand the cause of failure [25]. Rubrics provide a means of structured feedback by communicating expectations both in terms of what will be graded and level of quality needed to achieve a specific grade [14, 20]. It is a popular form of assessment that is being adopted both in schools and

* This software is currently undergoing rigorous testing for commercialisation purposes

higher education institutions across the UK. Whilst Rubrics are able to improve quality and save assessment time there is still the need to generate some individualised feedback text.

A rubric divides the coursework into a limited number of criteria and summarises what is expected from the student to obtain a specific grade in each criterion. Different weights might also be assigned to each criterion and an average (possibly weighted) gives a student's final grade in the coursework. A sample rubric is shown in Figure 1. The coursework was given for a module on Interface Design in the Computing department of a UK university (The Robert Gordon University, Aberdeen [1]). Each submission is graded based on three criteria with each criterion assigned a specific weight. In this example, the highlighted text across different cells indicate the grades assigned to this particular submission ('CBB') which can also be seen at the top left corner of the figure. Here, 'CBB' implies that a grade 'C' was obtained by the student based on the first criterion ('Cognitive Walkthrough') while the student scored 'B' in the other two criteria. It is also interesting to note that the highlighted text for the first criterion was from the rubric cells corresponding to 'B' and 'D' with no text highlighted from cell 'C' for this criterion.

COURSEWORK ASSESSMENT GRID 2010

Grade CBB Course **CM2006: Interface Design** _____ Assessment **Coursework** _____

FINAL GRADE	A	B	C	D	E	F	NS
DEFINITION	EXCELLENT Outstanding Performance	COMMENDABLE Meritorious Performance	GOOD Highly Competent Performance	SATISFACTORY Competent Performance	BORDERLINE FAIL Open To Compensation	FAIL Unsatisfactory	NON-SUBMISSION
COGNITIVE WALKTHROUGH (15%)	Walkthrough actions mostly complete with actions properly and validly assessed. Walkthrough is being used critically.	Walkthrough actions almost complete, with most actions properly and validly assessed. Walkthrough is being used critically.	Walkthrough actions almost complete, with most actions properly and validly assessed.	Walkthrough actions mostly complete, with some actions and correctly properly assessed.	Limited walkthrough only within the assignment.	No proper walkthrough conducted within the assignment.	
PROTOTYPE IMPLEMENTATION (30%)	Prototype communicates the user's experience of using the system in a clear manner, and adheres to the principles of good interface design.	Prototype communicates the user's experience of using most of the system in a clear manner, and adheres to many of the principles of good interface design.	Prototype communicates the user's experience of using much of the system in a clear manner, and adheres to the principles of good interface design.	Prototype communicates the user's experience of using the core of the system in a clear manner, and adheres to most the principles of good interface design.	Prototype communicates the user's experience of using the essence of the system, and breaks some of the most important principles of good interface design.	No proper prototype submitted, or the prototype totally fails to communicate the user's experience of using the system.	
COMMUNICATION & PRESENTATION (10%)	Clarity of expression excellent, consistently accurate use of grammar and spelling with fluent professional/academic writing /speaking style.	Thoughts and ideas clearly expressed. Grammar and spelling accurate and language fluent.	Language mainly fluent. Grammar and spelling mainly accurate. Communication of thoughts and ideas beginning to be affected.	Meaning apparent in most instances, but language not always fluent, grammar and spelling poor/moderate.	Often ambiguous, leading to meaning being barely apparent. Language, grammar and spelling poor.	Purpose and meaning of assignment unclear. Language, grammar and spelling poor.	

Fig. 1. A sample rubric

We introduce RubricAce, a TCBR system that can recommend textual feedback for student assessment using the Rubric marking scheme. Based on the assumption that students with similar grades should be given similar feedback, we support coursework assessment tasks by proposing feedback text given to previously assessed students with similar assigned grades. Since students are likely to make similar mistakes (although not usually the same subset of mistakes) in a given coursework, we are able to exploit this regularity and reuse relevant feedback text from multiple cases. Our expectation is that RubricAce can improve consistency, fairness and objectivity of assessment grading and improve efficiency as increasing numbers of students are assessed.

Section 2 presents related work in TCBR, while details of RubricAce's design and implementation appear in Sections 3 and 4. Proposed evaluation is outlined in Section 5 followed by conclusions in Section 6.

2 Related Work

A textual case has at least one of its attributes (either problem, solution or both components) in free text form. The retrieval stage of the CBR cycle [1] deals mainly with the problem component of cases to determine their similarity to a new problem while reuse deals with the solution components. Text reuse is applicable when the solution is in free text form.

Text reuse was proposed for report writing applied to the air travel incidents investigation [24]; we refer to this technique as jCOLIBRI-Reuse. Here, incident reports are semi-structured in that text is organised into a set of pre-defined sections. Text reuse is facilitated by presenting clusters

[1] www.comp.rgu.ac.uk

of similar text from other documents for each section while a user modifies the best match case's solution. This supports manual reuse from several similar documents whereby overall content of each section is modified independently. Though an intuitive form of text reuse, this approach is restrictive since it can only be used when common sectional headings are present in all cases.

Lamontagne and Lapalme (2004) demonstrated Case Grouping (CG), a form of structural text reuse on a semi-automated email response application. This involves the reuse of previous email messages to synthesize new responses to incoming requests. Sentences in a retrieved solution are labelled as reusable or not depending on whether there is sufficient evidence that previous similar problems contain such sentences. Reuse evidence for each sentence is computed by comparing the centroid of two clusters (support and reject) to the query. Only cases that have a similar sentence in their solution belong to the support cluster while the reject cluster contains the rest. Although the use of similarity knowledge to guide text reuse is novel, CG uses the entire casebase to determine if a sentence can be reused. This will be computationally expensive and seems counter-intuitive since cases with no similarity to the query nor retrieved solution will contribute to reuse evidence. However, such an approach is likely to guide reuse towards generic solutions. Some of the drawbacks of CG were addressed by the Case Retrieval Reuse Net (CR2N) applied to the weather forecast and incident reporting domains [2, 3]. CR2N utilised the average similarity of cases in each of the support or reject clusters to compute reuse evidence rather than the similarity of the centroid (average case). Efficiency was improved in the CR2N by computing localised reuse evidence unlike CG which uses all cases. CG and CR2N are more generic than jCOLIBRI-Reuse, since their clusters are not restricted to domains with a common template structure.

The GhostWriter systems [11, 16] aid text reuse by suggesting features and values or phrases during authoring of a product description for trading [11] or its review after purchase [16]. Features, values or noun phrase suggestions are iteratively extracted from top previous similar cases using manually-defined regular expressions or shallow NLP techniques. The list of suggestions is limited using a set of criteria such as ensuring their absence in the solution being authored at that point. Another criteria ensures that the length of the phrases are not too short. Like jCOLIBRI-Reuse, text recommendations are generated dynamically as the new solution is being incrementally authored. However, users start with an empty solution in the GhostWriter systems rather than a retrieved solution text. Another commonality between GhostWriter and jCOLIBRI-Reuse is that the problem and solution share a common vocabulary. This is unlike CG and CR2N where the problem and solution vocabularies are separate though they might share some common terms.

Transformational and compositional approaches to text reuse were proposed and applied in the hotel reviews domain [4]. The idea is to align particular structured attribute and values to specific sentences in a solution text using seed words. Reuse is then carried out by improving the best match's solution with better aligned sentences from other similar cases (transformational) or constructing a new solution from prototypes generated by aggregating similarly aligned sentences across several cases (compositional). Other forms of text reuse have involved the use of translation models for word alignment in incident reports [18], diagrammatic representation for legal texts [8] and formal concept analysis for adapting recipes [12]. In RubricAce, we adopt a localised approach to feedback generation which like GhostWriter and jCOLIBRI-Reuse is dynamic but exploits the alignment between structured and textual content to identify and rank suitable text snippets as with CR2N. This ranking is guided by a sentence level centroid which is computed locally unlike CG whose centroids are global.

The increased use of E-learning and E-assessment in education has paved the way to the introduction of several software tools. AdaLearn[5] is an adaptive e-learning environment that can guide the learning process by recommending new content based on a learner's past performance. Whilst this recommendation can be viewed as similar to RubricAce's ability to recommend feedback (sentence recommendation), they clearly differ on the types of knowledge used; AdaLearn is reliant on user profiles whilst RubricAce exploits similarity knowledge. Another difference between RubricAce and AdaLearn is that one is designed for helping students to learn (AdaLearn) while the other is designed to support academics when assessing students' submissions after learning. Accordingly this can be seen as complementary functionality. E-ASSISTment system [23, 26, 13] is more similar to RubricAce in that they both focus on assessment. However unlike RubricAce, they do not use rubrics in their asessment, this is because not only does the assessment take into acount the final solution but involves an iterative process that revisits assessments until they are com-

pleted satsifactorily. Other E-learning and assessment tools include EduComponents[7, 6], APLE (Adaptable Personal Learning Environment)[15] and eLGORM (e-Learning Governance Reference Model)[9]. We intend to incorporate publicly open source components from these current e-learning environments into RubricAce.

3 System Design

In this section, we examine the underlying CBR logic on which the application is built, which includes: the case representation (casebase), similarity mechanism (retrieval), reuse, revision and retain stages. We explain the design of these components in RubricAce.

3.1 Case Representation

A case is created for each student to be assessed. This implies that the casebase might contain cases created by multiple users when several academics are involved in the grading of submissions for a single coursework. In RubricAce we maintain five case attributes:

- $A1$ Criteria grades: This is the set of grades for each criterion in the Rubric (see sample in Figure 2). Each criterion is graded using 'A' to 'F' where 'A' represents the highest grade (excellent) and 'F' the lowest (fail). This attribute has a flexible size since the number of criteria can vary from one coursework to another.
- $A2$ Final grade: A final grade is an aggregation of the criteria grades. However, the method of aggregation can vary: computed as a weighted average or based on the profile of criteria grades. Currently the weighted aggregation is supported by RubricAce, however this can be overridden by the user.
- $A3$ Highlighted Rubric texts: Since the rubric describes the expected quality for possible grades in each criteria, these descriptions are often utilised by the assessor when generating feedback text. RubricAce further facilitates this activity by allowing users to highlight relevant phrases across the rubric for each student being graded. Users can also easily copy these highlighted text and use it when authoring the feedack text (with $A4$ and $A5$). Highlighted text for each criteria are maintained seperately by RubricAce.
- $A4$ Criteria feedback: Optional textual feedback provided with reference to any of the criteria grades (i.e. $A1$).
- $A5$ Overall feedback: This forms the overall textual feedback which should ideally explain the assigned grades.

We consider attributes $A1$, $A2$ and $A3$ as the case's problem component because these must be completed by the user before RubricAce can retrieve similar cases from which feedback text can be recommended. Whilst $A1$ and $A2$ are structured, $A3$'s content is textual as this relates to highlighted text from criteria expectation descriptions. The solution part of a case consists of the feedback text (i.e. $A4$ and $A5$). The idea is to recommend sentences from feedback text authored for other students with similar grading and highlights. Such system recommendations can be ignored or reused (with or without editing) by the assessor.

The casebase is generated incrementally with a case being added for each new student being assessed and marked. Therefore, RubricAce is initially unable to recommend any feedback text but the recommendation component of the tool becomes active once a few submissions have been marked. Clearly, its effectiveness improves as the casebase grows.

3.2 Similarity and Retrieval

Given a query consisiting of the problem components (i.e. $A1$, $A2$ and $A3$), RubricAce retrieves and ranks the top k similar cases in the casebase. Similarity between grading attributes ($A1$, $A2$) are based on the user-defined similarity matrix detailed in Table 1. This primarily ensures that two grades only have a similarity when the absolute difference between their ordinality is less than 2. Here, ordinality means the numeric value that can be assigned to a grade. For example, grades 'A' to 'F' can be viewed as having ordinal values '6' to '1'. Hence, we do not want any similarity

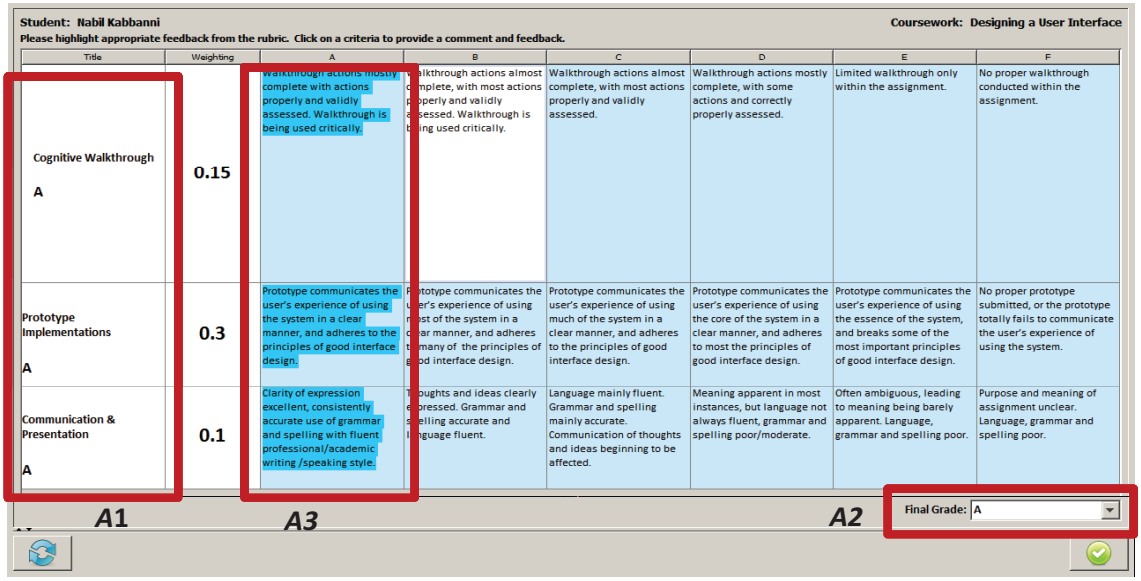

Fig. 2. A sample rubric from RubricAce

Table 1. Similarity of Grades

	A	**B**	**C**	**D**	**E**	**F**
A	1.0	0.5	0.0	0.0	0.0	0.0
B	0.5	1.0	0.5	0.0	0.0	0.0
C	0.0	0.5	1.0	0.5	0.0	0.0
D	0.0	0.0	0.5	1.0	0.5	0.0
E	0.0	0.0	0.0	0.5	1.0	0.5
F	0.0	0.0	0.0	0.0	0.5	1.0

between a grade 'A' and 'C' nor between 'D' and 'F'. Equal weights are assumed if no weights are given in the rubric.

Similarity between highlighted text ($A3$) is computed using the standard cosine similarity metric. For each criterion, any highlighted texts are combined together as space separated phrases which allows us to treat them as a single textual attribute. Apart from tokenisation of this text, we found that standard text pre-processing (stop word removal or stemming) was unnecessary because with the typically restrictive and small vocabulary in rubrics, stemming and stop word removal will have little or no effect. From an implementation point, all highlighted text is captured by its cell position in relation to the criterion and grade and relative text start and end positions within a cell.

We found that the similarity values obtained from comparing just the highlights across grades were not sufficient to differentiate case relevance. This is explained by the fact that keywords across different grades for a single criterion tend to be mostly identical with differences influenced by just the use of adjectives such as 'excellent', 'very good', 'good' or 'poor'. The non-highlighted cells therefore provide a way to minimise this effect. Accordingly, we also compute similarity to capture the non-highlighted cells that are common between cases. For each criterion, a set of non-highlighted cell IDs is formed (where the ID correspond to the grade alphabet). For instance, the set {A, D, E, F} implies that no text was highlighted from cells representing grades 'A', 'D', 'E' and 'F'. Similarity between such sets are computed using the dice coefficient. Whilst linguistic analysis of adjectives may provide a similar solution, it will require further parsing to capture and associate adjectives to other linguistic entities. Therefore we expect our approach without any language parsing to be a good enough approximation that is also efficient.

Aggregation of similarity values for the highlighted text attribute ($A3$) is carried out in two stages. For each criterion, the cosine coefficient similarity of the highlighted text and dice coefficient

similarity of the non-highlighted cells are first combined with a weighted average using equal weights. This is then followed by an aggregation across all criteria as a weighted average based on the assigned weights of each criterion.

The final global similarity combines the local similarity between the problem components ($A1$, $A2$ and $A3$) using a weighted average. The default weights are 0.4 for $A1$, 0.3 for $A2$ and 0.3 for $A3$, but RubricAce provides the option to change these values before similarity computation.

3.3 Text Reuse in RubricAce

The text reuse component of RubricAce uses a method based on the compositional approach proposed by Adeyanju et al. [4]. While a user is authoring feedback, the idea is to suggest sentences from the solutions of the nearest neighbours. This functionality is available for both feedback attributes ($A4$ and $A5$). Reuse is contextualised such that sentence recommendations for $A4$ are biased by neighbouring cases with greater similarity at the criterion level whilst with $A5$ recommendations are formed from cases with overall similarity.

Algorithm 1 shows the pseudo codes for the method used for recommending sentences in RubricAce. The aim here is to recommend a ranked list of relevant sentences extracted from the k nearest neighbours (line 2 of algorithm); here we chose $k \leq 5$. Given the set of sentences from the k neighbours, they are ranked based on their similarity to the centroid (average) sentence. In order to ensure that recommendations are not repeated we refine this centroid by subtracting from it any feedback content that is being incrementally entered by the user. Essentially the refined centroid will be the difference between the original centroid and the query (see Lines 3-5).

Algorithm 1 Sentence recommendation algorithm in RubricAce

Require: CB= $\{C_1, \ldots, C_n\}$, set of cases in the case base
Require: $CR = \{cr_1, \ldots, cr_p\}$, set of criteria attributes in rubric
Require: $V = \{v_1, \ldots, v_q\}$, set of possible grade values for each criterion
Require: IE= information entity consisting of a criterion and its assigned grade,
where $(attributeType(IE) \in CR) \wedge (attributeValue(IE) \in V)$
Require: $C_i = \{\text{IE}_{i1}, \ldots, \text{IE}_{ip}, \text{FinalGrade}_i, \text{HighlightedTexts}_i, \text{FeedbackText}_i\}$,
where $(i \in \{1 \ldots n\})$ i.e. a case consists of p criteria grades, a final grade, rubric highlighted texts and a feedback text
Require: $Q = \{\text{IE}_1, \ldots, \text{IE}_p, \text{FinalGrade}, \text{HighlightedTexts}\}$,
a query with p criteria grades, a final grade, and rubric highlighted texts
1: $NN \leftarrow RET(\text{CB}, Q, k)$
retrieve k similar cases;
2: **for each** $cr_j \in$ CR **do**
Require: S_q= Current feedback text for criterion j
3: $S \leftarrow getCriterionSentences(cr_j, NN)$
where $S = \{s_1, \ldots, s_\alpha\}$, all sentences related to criterion j in k neighbours
4: $avg \leftarrow getCentroidVector(S) - getVector(S_q)$
5: Propose $s_1, \ldots, s_\alpha$ ranked by similarity of their vectors to avg
i.e. $rank(s_a) > rank(s_b)$, if $SIM(s_a, avg) > SIM(s_b, avg)$
6: **end for**

In addition to suggesting sentences for reuse based on similar grades and highlighted text, RubricAce also provides an auto-completion tool tip to aid authoring of feedback. This is done per sentence and activates when a user types in two or more words but before a sentence marker (e.g. full stop, question mark, exclamation etc). The auto-completion method compares the current words in a partially completed sentence to sentences from the retrieved neighbours. It then suggests a completion if it finds a sentence with identical words in spelling and sequence at its beginning.

The reuse component of RubricAce is generic because it had earlier been applied to hotel reviews authoring domain [4]. We therefore expect it to be applicable to several other domains such as scientific reviews and analysing medical or psychological reports with very little modifications to cater for domain dependent issues.

3.4 Revise and Retain

The revision stage of the CBR cycle is completely left to the user in RubricAce. A fully marked submission is then retained in the casebase (an XML file) for comparison with other unmarked submissions. We do not envisage any maintenance issues such as duplicates and noisy cases if all submissions are for the same coursework in the same academic semester. However, if a coursework in its entirety was repeated in another session and the casebase was formed from graded submissions in the previous academic session, maintenance techniques [22, 17, 21] can be applied before using the cases for assisting student evaluation in the new session.

4 Implementation

RubricAce is a coursework management tool-kit which also incorporates CBR methods for supporting feedback generation to improve consistency and fairness. Its main features are:

1. Store and organize students, coursework submissions and their marking data.
2. Allow evaluation of students with marking against a pre-defined rubric.
3. Recommendation of similar submissions.
4. Support users by proposing useful suggestions when authoring textual feedback.
5. Suggest grade prediction for each criterion based on highlighted rubric texts.
6. Provide marking statistics and useful charts, e.g. visualization of similar submissions.
7. Output completely marked submissions in a printable format (e.g. PDF) for each student.

In this section, we discuss some of the implementation issues as regards the development of RubricAce with respect to the application data flow, its graphical user interfaces and required third party libraries.

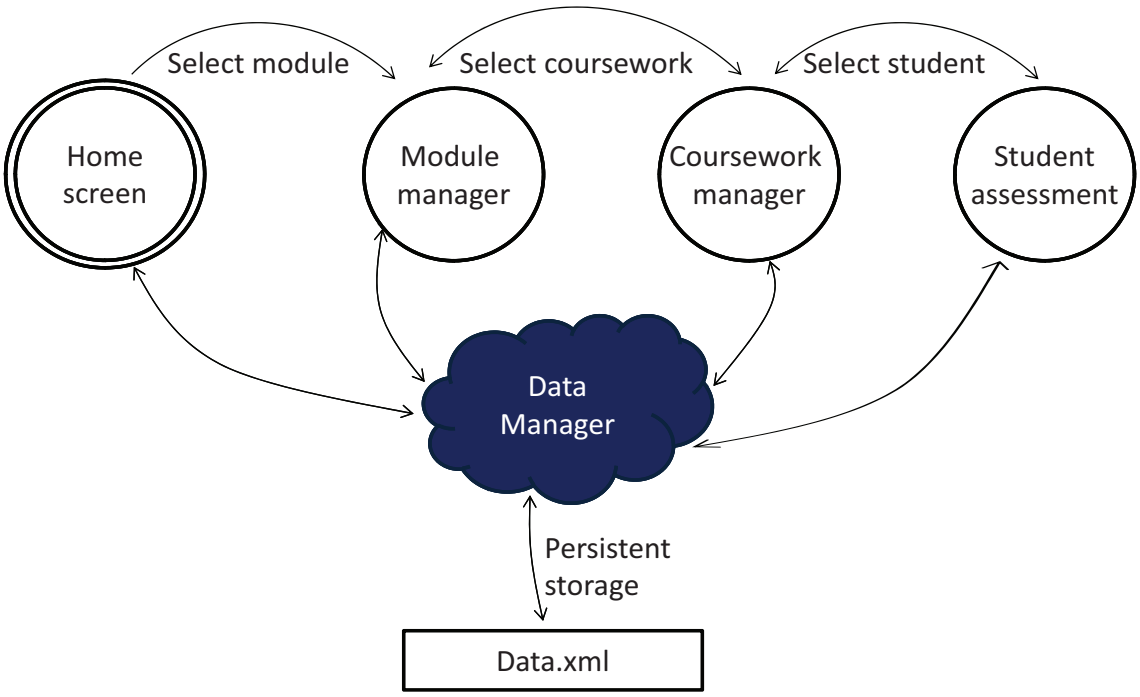

Fig. 3. Normal Data Flow of the application

4.1 Application data flow

The application has four main components: module manager, coursework manager, student evaluation manager and data manager. Each component exchanges information with one or more of the other components mostly in a bi-directional form as shown on data flow diagram in Figure 3.

Module Manager: The module manager screen (see Figure 4) allows the assigning of coursework to the selected module as well as enrolling students. Students can be manually entered or imported from other modules or from a CSV (comma separated values) file with the fixed structure of: Matriculation Number, First Name, Surname.

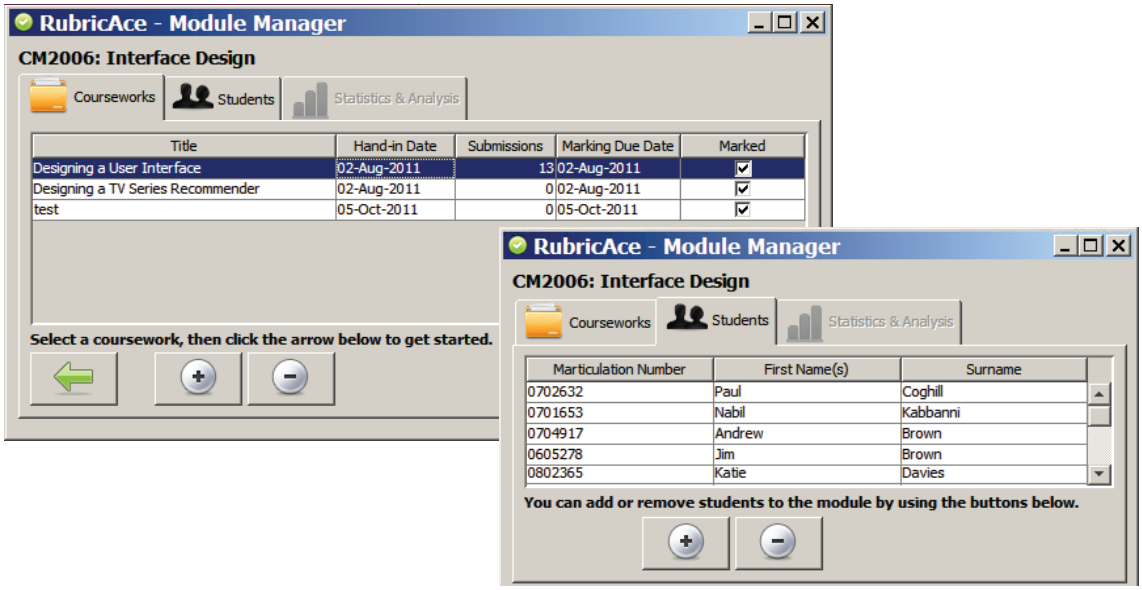

Fig. 4. Module Manager screen shots

Coursework Manager: This enables the creation of a coursework rubric and recording of submissions for enrolled students as shown in Figure 5. Each criterion should be given a weighting (float) value and the sum of all criteria weightings should add up to 1. However, we normalise the weights if they do not. Criteria can also be imported from .doc (Microsoft Word) documents. This component also allows the user to flag each submitted coursework as completely graded or not.

Student Evaluation: There are two main aspects in the evaluation component.

1. Marking: The marking pane consists of a rubric whose textual contents can be highlighted. When a criterion is clicked on a rubric, a context menu is provided. This context menu provides an opportunity to give feedback for an individual criterion and give a grade for the criteria. The system will give a grade prediction, based on how much has been highlighted from each cell. Once graded, the row is then colour coded. There is also space provided to give an overall comment and a final grade awarded.
2. Similarity: The similarity pane can provide a list of similar students. Once some marking for the student has been completed, the system can recommend similar submissions to the marker. The marker can select a submission and view the comments and grades given to the submission. Additionally, the highlights that the selected similar submission received will be shown on the rubric in an alternate style so the marker can compare.

This component also contains the statistics and analysis tool which currently shows two metrics:

1. Grade Overlap: A spring model which displays the distance between submissions, which are colour-coded by grade. Such a visualisation is useful to re-grade borderline submissions, as well as ensure fair marking. The submissions should be clustered by grade if marking is consistent.
2. Grade Distribution: This is a histogram of grade values to identify anomalies such as skewed distributions.

Data Manager: This component manages the storage and retrieval of all data (students and markings) from the persistent file (data.xml).

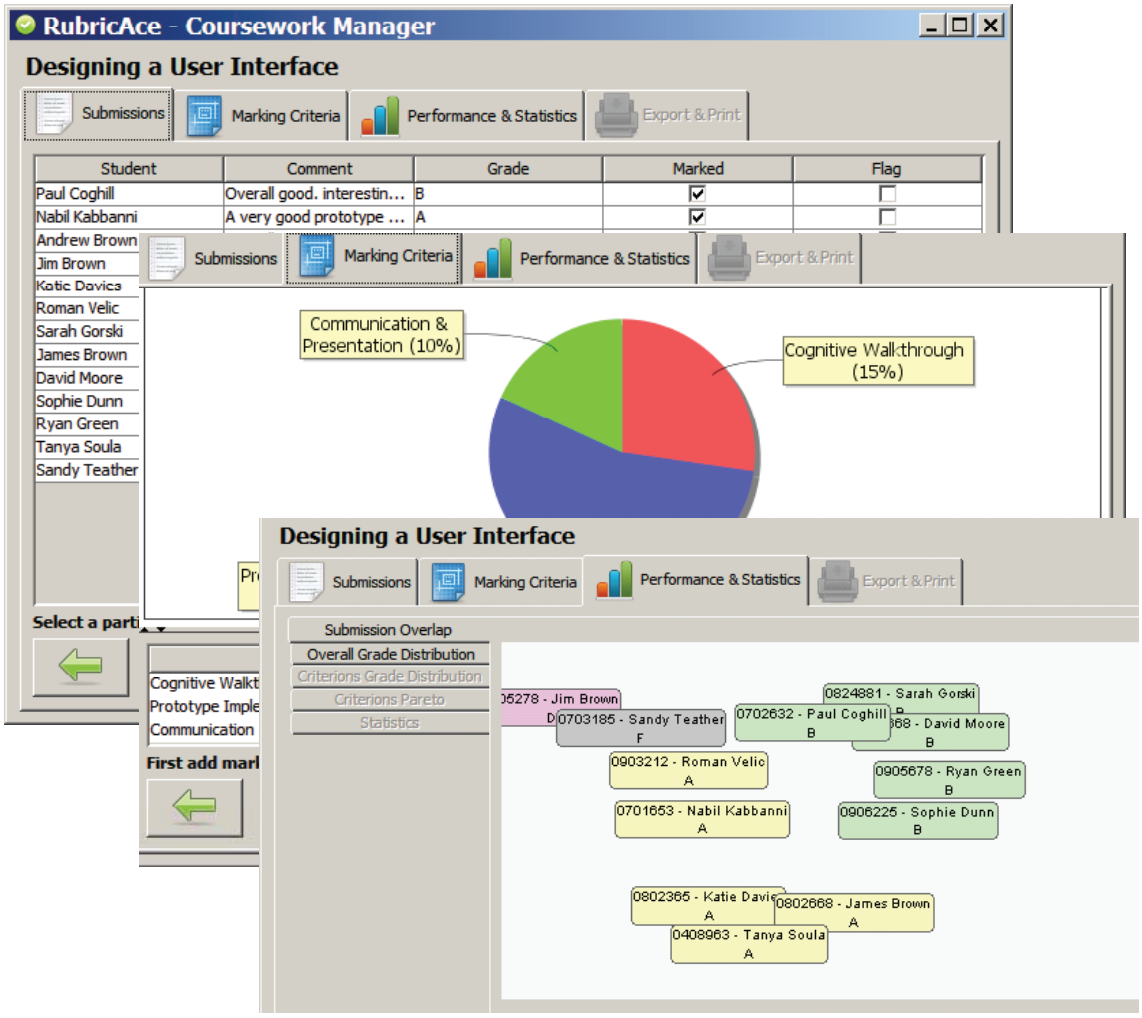

Fig. 5. Coursework Manager screen shots

4.2 Third party Libraries used

RubricAce is built using Java JDK 1.6 and utilizes the following additional libraries:

- jCOLIBRI 2 (Textual Processing & CBR framework)
  http://gaia.fdi.ucm.es/projects/jcolibri/#Download
- xStream (XML Persistent Data Store)
  http://xstream.codehaus.org/download.html
- iText (PDF Generation)
  http://www.itextpdf.com/download.php
- Apache POI (Office Document Processing)
  http://poi.apache.org/

5 Proposed System Evaluation

Preliminary evaluation of RubricAce by potential users has generally been very positive and the system seem to give relevant and useful suggestions. However, a comprehensive evaluation of RubricAce is yet to be carried out. We intend to do a user study by requesting some academic staff to use the system for evaluation of their students' coursework submissions. The system will be programmed to automatically collect the following information which will be used for our system evaluation analysis.

1. Marking times with dates to show if the marking time reduces as more submissions are marked.
2. To what extent users select the auto completion phrases.
3. To what extent users picked a suggestion from our reuse method.
4. Take feedback from users once all submissions have been completely marked.

Our proposed evaluation will also be carried out with multiple users with different settings. We would attempt to simulate scenarios where all submissions for the same coursework are graded by a single user or shared among multiple users. The evaluation also has to take care of scenarios where each submission is graded by more than one user and the average grade is given as final grade. How do we combine the free comments from the users without loss of content but avoiding unnecessary repetitions? We then intend to improve the system based on our evaluation results and ultimately make it available for use by interested academic staff.

6 Conclusion

We have introduced RubricAce, a coursework management tool-kit which also incorporates feedback generation based on the CBR paradigm. The software supports academic staff to achieve professional best practices during grading of courseworks by proposing feedback text from previous similarly graded students. This makes it easier to maintain consistency and fairness thereby improving the quality of the feedback given to students. RubricAce also provides other tools that helps to identify similar students based on grades, charts to view the distribution of students' grades and facilities to import and export student records. Our primary task in the future is to carry out a comprehensive user evaluation of the system. The application can also be easily integrated into web learning platforms such as moodle (http:/moodle.org) although this will require changing the persistent storage from its current form in a XML file to a proper database which will significantly improve performance with an increase in the data stored by RubricAce.

Acknowledgements This research work is supported by a UK-India Education and Research Initiative (UKIERI) standard award.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. Artificial Intelligence Communications (AICom) 7(1), 39–59 (1994)
2. Adeyanju, I., Wiratunga, N., Lothian, R., Sripada, S., Craw, S.: Solution reuse for textual cases. In: Petridis, M. (ed.) Proceedings of the Thirteenth UK Workshop on Case-Based Reasoning (UKCBR'08). pp. 54–62. CMS Press, University of Greenwich (2008)
3. Adeyanju, I., Wiratunga, N., Lothian, R., Sripada, S., Lamontagne, L.: Case retrieval reuse net (CR2N): Architecture for reuse of textual solutions. In: McGinty, L., Wilson, D. (eds.) Proceedings of the Eighth International Conference on Case Based Reasoning (ICCBR'09). pp. 14–28. LNAI 5650, Springer, Heidelberg (2009)
4. Adeyanju, I., Wiratunga, N., Recio-Garcia, J.A., Lothian, R.: Learning to author text with textual CBR. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) Proceedings of the Nineteenth European Conference on Artificial Intelligence (ECAI'10). pp. 777–782. IOS press, Amsterdam (2010)
5. Alian, M., Al-Akhras, M.: Adalearn: an adaptive e-learning environment. In: Proceedings of the 1st International Conference on Intelligent Semantic Web-Services and Applications (ISWSA '10). ACM, New York, NY, USA (2010)
6. Amelung, M., Forbrig, P., Rösner, D.: Towards generic and flexible web services for e-assessment. In: Proceedings of the 13th annual conference on Innovation and technology in computer science education (ITiCSE '08). pp. 219–224. ACM, New York, NY, USA (2008)
7. Amelung, M., Piotrowski, M., Rösner, D.: Educomponents: experiences in e-assessment in computer science education. SIGCSE Bulletin 38, 88–92 (June 2006)
8. Ashley, K., Lynch, C., Pinkwart, N., Aleven, V.: Towards modeling and teaching legal case-based adaptation with expert examples. In: McGinty, L., Wilson, D.C. (eds.) Proceedings of the Eighth International Conference on Case Based Reasoning (ICCBR'09). pp. 45–59. LNAI 5650, Springer, Berlin Heidelberg (2009)

9. Baruque, L.B., Baruque, C.B., Melo, R.N.: Towards a framework for corporate e-learning evaluation. In: Proceedings of the 2007 Euro American conference on Telematics and information systems (EATIS '07). pp. 58:1–58:5. ACM, New York, NY, USA (2007)
10. Boehler, M.L., Rogers, D.A., Schwind, C.J., Mayforth, R., Quin, J., Williams, R.G., Dunnington, G.: An investigation of medical student reactions to feedback: a randomised controlled trial. Medical Education 40, 746 –749 (2006)
11. Bridge, D., Waugh, A.: Using experience on the read/write web: The ghostwriter system. In: ICCBR'09 workshop proceedings. pp. 15–24 (2009)
12. Dufour-Lussier, V., Lieber, J., Nauer, E., Toussaint, Y.: Text adaptation using formal concept analysis. In: Bichindaritz, I., Montani, S. (eds.) Proceedings of the Eighteenth International Conference on Case Based Reasoning (ICCBR'10). pp. 96–110. LNAI 6176, Springer-Verlag, Berlin Heidelberg (2010)
13. Feng, M., Heffernan, N.T., Koedinger, K.R.: Addressing the testing challenge with a web-based e-assessment system that tutors as it assesses. In: Proceedings of the 15th international conference on World Wide Web (WWW '06). pp. 307–316. ACM, New York, NY, USA (2006)
14. Goodrich, H.: Understanding rubrics. Educational Leadership 54(4), 14–18 (1996)
15. Green, S., Nacheva-Skopalik, L., Pearson, E.: An adaptable personal learning environment for e-learning and e-assessment. In: Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing (CompSysTech '08). ACM, New York, NY, USA (2008)
16. Healy, P., Bridge, D.: The ghostwriter-2.0 system: Creating a virtuous circle in web 2.0 product reviewing. In: ICCBR'10 Workshop Proceedings. pp. 121–130 (2010)
17. Iglezakis, I., Reinartz, T., Roth-Berghofer, T.: Maintenance memories: Beyond concepts and techniques for case base maintenance. In: Proceedings of the European Conference on Case Based Reasoning (ECCBR'04). pp. 227–241. LNAI 3155, Springer-Verlag, Berlin-Heidelberg (2004)
18. Lamontagne, L., Bentebibel, R., Miry, E., Despres, S.: Finding lexical relationships for the reuse of investigation reports. In: ICCBR'07 workshop proceedings (2007)
19. Lamontagne, L., Lapalme, G.: Textual reuse for email response. In: Proceedings of the European Conference on Case based Reasoning (ECCBR'04). pp. 234–246. LNAI 3155, Springer-Verlag (2004)
20. Mansilla, V., Duraisingh, E., Wolfe, C.R., Haynes, C.: Targeted assessment rubric: An empirically grounded rubric for interdisciplinary writing. Journal of Higher Education 80(3), 334–353 (2009)
21. Massie, S.: Complexity Modelling for Case Knowledge Maintenance in Case-Based Reasoning. Ph.D. thesis, The Robert Gordon University, Aberdeen (2006)
22. Patterson, D., Anand, S., Dubitzky, D., Hughes, J.: A knowledge light approach to similarity maintenance for improving case-based competence. In: ECAI workshop on Flexible Strategies for Maintaining Knowledge Containers. pp. 65–77 (2000)
23. Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N.T., Koedinger, K.R., Junker, B., Ritter, S., Knight, A., Aniszczyk, C., Choksey, S., Livak, T., Mercado, E., Turner, T.E., Upalekar, R., Walonoski, J.A., Macasek, M.A., Rasmussen, K.P.: The assistment project: Blending assessment and assisting. In: Proceedings of the 12th Annual Conference on Artificial Intelligence in Education. pp. 555–562 (2005)
24. Recio-García, J., Díaz-Agudo, B., González-Calero, P.: Textual cbr in jCOLIBRI: From retrieval to reuse. In: ICCBR'07 workshop proceedings. pp. 217–226 (2007)
25. Schank, R.: Dynamic memory revisited. Cambridge University Press (1999)
26. Turner, T.E., Macasek, M.A., Nuzzo-Jones, G., Heffernan, N.T.: The assistment builder: A rapid development tool for its. In: Proceedings of the 12th Annual Conference on Artificial Intelligence in Education. pp. 929–931 (2005)
27. Weber, R.O., Ashley, K.D., Bruninghaus, S.: Textual case-based reasoning. Knowledge Engineering Review 20(3), 255–260 (2006)