A Rule-Based System for Test Quality Improvement

Gennaro Costagliola, Vittorio Fuccella
Dipartimento di Matematica e Informatica, Università degli Studi di Salerno, Fisciano (SA), Italy.

ABSTRACT

To correctly evaluate learners' knowledge, it is important to administer tests composed of good quality question items. By the term "quality" we intend the potential of an item in effectively discriminating between skilled and untrained students and in obtaining tutor's desired difficulty level. This paper presents a rule-based e-testing system which assists tutors in obtaining better question items through subsequent test sessions. After each test session, the system automatically detects items' quality and provides the tutors with advice about what to do with each of them: good items can be re-used for future tests; among items with lower performances, instead, some should be discarded, while some can be modified and then re-used. The proposed system has been experimented in a course at the University of Salerno.

Keywords: e-Testing, Computer Aided Assessment, CAA, item, item quality, questions, eWorkbook, Item Response Theory, IRT, Item Analysis, online testing, multiple choice test.

1. INTRODUCTION

E-testing, also known as *Computer Assisted Assessment* (*CAA*), is a sector of e-learning aimed at assessing learner's knowledge through computers. Through *e-testing*, tests composed of several question types can be presented to the students in order to assess their knowledge. *Multiple choice* question type is frequently employed, since, among other advantages, a large number of tests based on it can be easily corrected automatically.

The experience gained by educators and the results obtained through several experiments (Woodford & Bancroft, 2005) provide some guidelines for writing good *multiple choice* questions (*items*, in the sequel), such as: "use the right language", "avoid a big number of unlikely *distractors* for an item", etc.

It is also possible to evaluate the effectiveness of the items, through the use of several statistical models, such as *Item Analysis* (IA, 2008) and *Item Response theory* (*IRT*). Both of them are based on the interpretation of statistical indicators calculated on test outcomes. The most important indicators are the *difficulty* indicator, which measures the difficulty of an item, and the *discrimination* indicator, which represents the information of how effectively an item discriminates between skilled and untrained students. More statistical indicators are related to the *distractors* (wrong options) of an item. A good quality item has a high discrimination potential and a difficulty level close to tutor's desired one.

Despite the availability of guidelines for writing good items and statistical models to analyze their quality, only a few tutors are aware of the guidelines and even fewer are used with statistics. The result is that the quality of the tests used for exams or admissions is sometimes poor and in some cases could be improved.

The most common Web-based e-learning platforms, such as *Moodle* (Moodle, 2008), *Blackboard* (Blackboard, 2008), and *Questionmark* (Questionmark, 2008) evaluate item quality by generating and showing item statistics. Nevertheless, their interpretation is left to the tutors: these systems do not advise or help the tutor in improving items.

In this paper we propose an approach and a system for improving items: we provide tutors with feedback on their quality and suggest them the opportune action to undertake for improving it. To elaborate, the approach consists of administering tests to learners through a suitable *rule-based system*. The system obtains item quality improvement by analyzing the test outcomes. After the analysis, the system provides the tutor with one of the following suggestions:

- "Keep on using the item" in future test sessions, for good items;
- "Discard the item", for poor items;
- "Modify the item", for poor items whose defect is originated by a well-known cause. In this case, the system also provides the tutor with suggestions on how to modify the item.

Though item quality can be improved after the first test session in which it is used, the system can be used for subsequent test sessions, obtaining further improvements.

Rule-based systems are generally composed of an *inferential engine*, a *knowledge-base* and a *user interface*. Our system follows this model. The *inferential engine* works by exploiting *fuzzy classification*: the items are classified on the basis of the values of some parameters calculated on test outcomes. *Fuzzy classification* has been successfully employed in technological applications in several sectors, from weather forecast (Bradley et al.; 1982) to medical diagnosis (Exarchos et al.; 2007). In our system, it has been preferred over other frequently used classification methods, such as *decision trees* and *Bayesian classifier* due to the following reasons:

- *Knowledge availability*. Most of the knowledge is already available, as witnessed by the presence of numerous theories and manuals on *psychometrics*.
- *Lack of data*. Other types of classification based on data would require the availability of large data sets. Once they have been gathered, in such a way to have statistically significant classes to perform data analysis, such methods might be exploited.

The *knowledge-base* of the system has been inferred from *IA* and other statistical models for the evaluation of the items.

The system has been given a Web-based *interface.* Rather than developing it from scratch, we have preferred to integrate the system in an existing Web-based e-testing platform: *eWorkbook* (Costagliola et al.; 2007), developed at *University of Salerno*.

An experiment on system's performances has been carried out in a course at the *University of Salerno*. As shown in the experiment, we can obtain items which better discriminate between skilled and untrained students and better match the difficulty estimated by the tutor.

The paper is organized as follows: section 2 presents a brief survey on fuzzy classification; section 3 describes the statistical models for evaluating the effectiveness of the items; the approach for item quality improvement is presented in section 4. In section 5, we describe the system: its architecture and its instantiation in the existing *e-testing* platform; section 6 presents an experiment and a discussion on its results; section 7 contains a comparison with work related to ours; lastly, several final remarks and a discussion on future work conclude the paper.

2. FUZZY CLASSIFICATION

The approach presented in this paper employs a *fuzzy classification* method. *Classification* is one of the most widespread *Data Mining* techniques (Roiger & Geatz; 2004). It lies in grouping *n* entities of a given knowledge domain into *m* knowledge containers, often called *classes*, *sections*, *categories*, etc. To perform a *classification*, several attributes of the entities must be analyzed. These are called *input attributes*. The class in which the entity will be inserted is an *output attribute*. A good *classification* consists of classes with high *internal cohesion* and *external separation*. *Classification* differs from *clustering*. The difference lies in the final classes, which are *predefined* only for the former problem. In *clustering*, instead, the classes (*clusters*) are discovered during the process. For this reason, we say that classification is a *supervised* process.

Classification has been employed in several fields for solving real problems, such as:

- In medicine, for medical diagnosis;
- In pattern discovery, for fraud detection. E.g., the FALCON system (Brachman et al.; 1996), created by the HNC Inc. is used for detecting possible transaction with false credit cards;
- In economy and financing, for risk management, for classifying the credit risk of a person who has requested funds.

Several methods can be used for *classification*. Some of them, such as *decision trees*, use *machine learning* for extracting knowledge from data. The most frequently used *machine learning* approaches divide

data in two sets: the *training set* and the *test set*. The former is used to produce the knowledge, the latter to test the effectiveness of the approach. The *decision tree* lends itself to be used in classification, but it gives just one output categorical attribute. Furthermore, the *decision tree* produces particularly easy to explain results and can be suitable in the case of unknown data distribution. Nevertheless, it can be advisable to employ other methods, such as the *Bayesian classifier*, when all or most of the input attributes are numerical: the tree could have too many conditional tests to satisfy to be informative.

When data are missing, and the knowledge is already available, a *rule-based* system is a suitable solution for classification. A *rule-based* systems is a system whose knowledge-base is expressed under the form of *production rules*. *Rule-based* systems have been employed in many applications for decision making. Such systems can also be used for classification. The *production rules* can be inferred directly from the expertise or obtained through *machine learning* methods. In general, the rules are in the following form:

IF <antecedent conditions>
THEN <consequent conditions>

The *antecedent conditions* define the values or the value intervals for one or more input attributes. The *consequent conditions* define the values or the value intervals for one or more output attributes. In the case of *classification*, the *consequent conditions* determine if a given entity belongs to a class. In *rule-based* systems, it is often necessary to deal with *uncertainty*. To this aim, f*uzzy logic* is often employed, e.g. it has been used for economic performance analysis (Zhou et al.; 2005).

Fuzzy logic is derived from *fuzzy sets* theory. *Fuzzy sets* were first introduced by Zadeh (1997), and have been applied in various fields, such as decision making and control (Bardossy & Dukstein; 1995). *Fuzzy set* theory deals with reasoning that is approximate rather than precisely deduced from classical predicate logic. A *fuzzy set* is characterized by a *membership function* which maps a value that might be a member of the set to a number between zero and one indicating its actual *degree of membership*. The *triangular membership function* is the most frequently used function and the most practical, but other shapes, continuous or discrete, are also used.

A variable used in a *fuzzy production rule* is also called a *linguistic variable* and is associated to a *linguistic value* (*term*). Each *linguistic value* is associated to a *fuzzy set*.

A *fuzzy system* is a set of *fuzzy rules* connecting fuzzy input and fuzzy output in the form of IF-

THEN sentences. Once we have the rules and the fuzzy sets for defining the values of the *linguistic variables*, the *fuzzy inference* can be applied. The most commonly applied method is the 4-phases procedure introduced by Mamdani & Assilian (1999). The four steps are the following:

- *Fuzzyfication*: conversion of the input values in the corresponding membership levels in each fuzzy set;
- *Inference*: the membership levels are combined in order to obtain a degree of fulfillment for each rule;
- *Combination*: combination of all the values obtained for the rules to obtain a unique fuzzy set;
- *Defuzzyfication*: conversion of the fuzzy set obtained at the previous phase into a value.

A *fuzzy classifier* is a function that at each entity associates a set of Boolean functions defining the possibility (*Degree of Fulfillment, DoF* briefly) that an instance belongs to the output classes. The *fuzzy classifier* produces a categorical value as final output. Often, the classification is performed by selecting the class for which the *DoF* is the highest. This method corresponds to the case of *maximum* method for *combination* and *maximum* method for *defuzzyfication*.

3. ITEM QUALITY: ITEM AND DISTRACTOR ANALYSIS

This section describes the main statistical models on which the knowledge-base of our system is based. In particular, it focuses on *IA*, whose statistical indicators are used in our system's rules. The tests administered through our system make use of *multiple choice* items for the assessment of learners' knowledge. Those items are composed of a *stem* and a list of *options*. The *stem* is the text that states the question. The only

correct answer is called the *key*, whilst the incorrect answers are called *distractors* (Woodford & Bancroft, 2005).

As mentioned in the introduction, two main statistical models are available for evaluating item quality: *IA* and *IRT*. Although today *IRT* is the pre-dominant measurement model, *IA* is still frequently employed by psychometricians, test developers, and tutors for a number of reasons. First, concepts of *IA* are simpler than those of their *IRT* counterpart: even the tutors without a strong statistical background could easily interpret the results without going through a steep learning curve. Second, *IA* could be computed by many popular statistical software programs, including *SAS*, while *IRT* necessitates use of specialized software packages such as *Bilog*, *Winsteps*, *Multilog*, *RUMM* (Yu and Wong, 2003; Yu, 2005). One great advantage of *IRT* is the invariance of ability and item parameters: it is the cornerstone of IRT and the major distinction between *IRT* and *IA* (Hambleton & Swaminathan, 1985). One drawback, however, of *IRT* is that a big sample size is necessary for the estimation of parameters. Nevertheless, empirical studies, examining and/or comparing the invariance characteristics of item statistics from the two measurement frameworks, have observed that it is difficult to find a great invariance or any other obvious advantage in the *IRT* based item indicators (Stage, 1999).

For our study, *IA* has been preferred over *IRT* due to the following main reasons: it needs a smaller sample size for obtaining statistically significant indicators; it is easier to use *IA* indicators to compose rule conditions. The following statistical indicators are available from *IA* and other models, such as *distractor analysis*:

- *difficulty*: a real number between 0 and 1 which expresses a measure of the difficulty of the item, intended as the proportion of learners who get the item correct.
- *discrimination*: a real number between -1 and 1 which expresses a measure of how well the item discriminates between good and bad learners. *Discrimination* is calculated as the *point biserial* correlation coefficient between the score obtained on the item and the total score obtained on the test. The *point biserial* is a measure of association between a continuous variable (e.g. the score on the test) and a binary variable (e.g. the score on a *multiple choice* item).
- *frequency(i)*: a real number between 0 and 1 which expresses the frequency of the i-th option of the item. Its value is calculated as the percentage of learners who choose the i-th option.
- *discrimination(i)*: a real number between -1 and 1 which expresses the discrimination of the i-th option. Its value is calculated as the *point biserial* correlation coefficient between the result obtained by the learner on the whole test and a dichotomous variable that says whether the i-th option was chosen (yes=1, no=0) by the learner or not.
- *abstained_freq*: a real number between 0 and 1 which expresses the frequency of the abstention (no answers given) on the item. Its value is calculated as the percentage of learners who did not give any answer to the item, where allowed.
- *abstained_discr*: a real number between -1 and 1 which expresses the discrimination of the abstention on the item. Its value is calculated as the *point biserial* correlation coefficient between the result obtained by the learner on the whole test and a dichotomous variable that says whether the learner refrained or not (yes=1, no=0) on the item.

Discrimination and *difficulty* are the most important indicators. They can be used for both determining item quality and choosing advice for tutors. As experts suggest (Massey, 2007), a good value for *discrimination* is about 0.5. A positive value lower than 0.2 indicates that the item does not discriminate well. This can be due to several reasons, including: the question does not assess learners on the desired knowledge; the stem or the options are badly/ambiguously expressed; etc. It is usually difficult to understand what is wrong with these items and more difficult to provide a suggestion to improve them, so, if the tutor cannot understand the problem her(him)self, the suggestion is to discard the item. A negative value for *discrimination*, especially if joined with a positive value for the discrimination of a *distractor*, is a sign of a possible mistake in choosing the key (a data entry error occurred). In this case it is easy to recover the item by changing the key.

If *difficulty* is too high (>0.85) or too low (<0.15), there is the possibility that the item does not correctly evaluate the learners on the desired knowledge or subject. This is particularly true when such values for

difficulty are sought together with medium-low values for *discrimination*. Furthermore, our system allows the tutor to define the foreseen difficulty for an item.

In a test, in order to better assess a heterogeneous class with different levels of knowledge, it is important to balance the difficulty of the items: for example, in the preparation of the *Michigan Educational Assessment Program* (*MEAP*, 2007), "easy" and "difficult" items are used in every form to balance the difficulty level of the items. Having a precise estimation of item's difficulty allows the tutor to correctly assign it to a test section on the basis of its difficulty, when composing tests. Thus, the closer a tutor's estimation of item *difficulty* is to the actual calculated difficulty for that item, the more reliable that item is considered to be.

When *difficulty* is too high or underestimated, this can be due to the presence of a *distractor* (noticed for its high frequency) which is too plausible (it tends to mislead a lot of students, even skilled ones). Removing or substituting that *distractor* can help in obtaining a better item. Sometimes, the item has its intrinsic difficulty and it can be difficult to adjust it, so the suggestion can be to modify the tutor's estimation.

As for *distractors*, they can contribute to form a good item when they are selected by a significant number of students. When the frequency of the *distractor* is too high, there could be an ambiguity in the formulation of the stem or of the *distractor*. A good indicator of *distractors*' quality is their discrimination, which should be negative, denoting that the *distractor* was selected by untrained students. In conclusion, a good *distractor* is the one which is selected by a small but significant number of untrained students.

High abstention is always a symptom of high difficulty for the item. When it is accompanied by a high (not negative or next to 0) value for its discrimination and a low value for item *discrimination*, it can tell that the question has a bad quality and it is difficult to improve it.

4. THE APPROACH

The approach consists of administering tests to the learners through a suitable e-testing *system*. On the basis of test outcomes, the system evaluates the items and suggests the tutor the most suitable action to undertake on each of them. This is possible after a test has been administered to a statistically significant number of learners. In general, the quality improvement is obtained in two ways:

- through the increment of item *discrimination*. This objective is pursued by both eliminating and opportunely modifying items with low *discrimination.*
- by having the tutor's estimation of the *difficulty* closer to the calculated difficulty for the item. In the most desirable cases (when possible), the system suggests how to modify the item. Otherwise, the estimation must be modified.

Though item quality can be improved after the first test session in which it is used, the items can be evaluated by the system through subsequent test sessions, following the lifecycle shown in Figure 1. The figure shows a *UML activity diagram*, in which the role of the tutor and the role of the system are specified in two different swimlines.

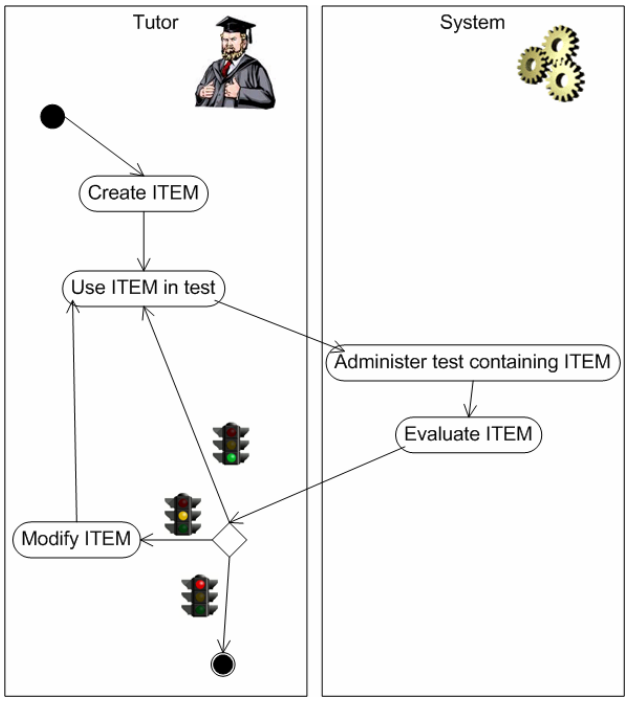


Figure 1. Item Lifecycle

The item starts its lifecycle when it is created by the tutor. Then, the tutor selects the item for a test session. The test is administered to the learners through the system in a test. At the end of the session, the system stores learners' outcomes. Such outcomes are used to calculate statistical indicators, which are used in the *production rules* for item evaluation. The output of the evaluation is the *state* of the item, whose value is expressed through a traffic light. Later on, according to the system output, the tutor decides the destiny of the item as follows:

- **State = Green** : the item has good performances and can be re-used for future test sessions.
- **State = Red**: the item has bad performances and should be discarded.
- **State = Yellow**: the item has bad performances, but its quality can be improved. The system suggests how. The item is modified by the tutor and can be re-used for future test sessions.

It is worth noting that the system just suggests the tutor the most suitable action. Figure 1 shows the case in which the tutor follows the suggestion of the system. Nevertheless, the tutor can choose not to follow the system's suggestion if s/he thinks it is opportune.

5. THE SYSTEM

Typically, rule-based systems are composed of an *inferential engine*, a *knowledge-base* and a *user interface* (Momoh et al.; 2000). Our system follows this model.

The *knowledge-base* has been mostly inferred by translating into rules the verbal knowledge presented in section 3. Since such knowledge does not completely cover all of the aspects considered in our system, it has been integrated with knowledge extracted from data.

The *inferential engine* works by performing a *classification* of the items. Several classes of items have been identified, and each class is associated to a *production rule*. Fuzzy sets have been used in order to cope with *linguistic uncertainty* contained in the rules: sources of uncertainty in our system are associated to both the conditions in the antecedents of the rules and to the combination of the rules themselves. The *DoF* of a rule tells the membership of the item to the corresponding class. The classification is performed by selecting the class for which the *DoF* is the highest. This model fits well our question item classification problem, since, in most cases (except for Class 1, see Table 3), the belonging to a class indicates the presence of a defect affecting the item. By choosing to classify the item to the class to which the item belongs with the maximum degree, a decision is taken according to the heaviest problem affecting the item.

The system has been equipped with a Web-based user interface. Rather than developing it from scratch, we have preferred to integrate the system in an existing Web-based e-testing platform. To elaborate, the system has been implemented as a Java *Object Oriented* framework, called *Item Quality Framework*, which can be instantiated in any Java-based e-testing platform. Our choice is fallen on *eWorkbook*, already in use at our faculty.

The Knowledge-Base

This section describes the process for obtaining the *fuzzy production rules* from the knowledge. As already pointed out, the rules have been mostly inferred from the verbal knowledge presented in section 3 and integrated with knowledge extracted from data. The integration has only been necessary for modeling a few *membership functions*.

Variables and Fuzzyfication

The set of variables used are reported, together with an explanation of their meaning and the set of possible values they can assume (*terms*), in Table 1. These variables are directly chosen from the statistical indicators presented in section 3 or derived from them.

The *discrimination* and *difficulty* variables are the same indicators for item *discrimination* and *difficulty* defined in section 3. The same discourse is valid for the variables related to the abstention, *abst_frequency* and *abst_discrimination*. *difficulty_gap* is a variable representing the error in tutor's estimation of item *difficulty*: through the system interface, the tutor can assign one out of three difficulty levels to an item (easy = 0.3; medium = 0.5; difficult = 0.7). *difficulty_gap* is calculated as the difference between the tutor estimation and the actual difficulty calculated by the system.

Three variables representing the frequency of the *distractors* for an item have been considered: *max_distr_freq*, *min_distr_freq*, *distr_freq*. Their value is not an absolute frequency, but relative to the frequency of the other *distractors*: it is obtained by dividing the absolute frequency by the mean frequency of the *distractors* of the item. In the case of items with five options, as our system has been tested, their value is a real number varying from 0 to 4.

Table 1. Variables and Terms

Variable	Explanation	Terms
discrimination	Item's discrimination (see sec. 3)	Negative, low, high
difficulty	Item's difficulty (see sec. 3)	Very_low, medium, very_high
difficulty_gap	The difference between the tutor's estimation of item's difficulty and the difficulty calculated by the system	Underestimated, correct, overestimated
max_distr_discr	The maximum discrimination for the distractors of an item	Negative, positive
max_distr_freq	The maximum (relative) frequency for the distractors of an item.	Low, high
min_distr_freq	The minimum (relative) frequency for the distractors of an item	Low, high
distr_freq	The (relative) frequency of the distractor with maximum discrimination for an item	Low, high
abst_frequency	The frequency of the abstentions for an item	Low, high
abst_discrimination	The discrimination of the abstentions for an item	Negative, positive

Membership Functions

As for the *membership functions* of fuzzy sets associated to each *term*, *triangular* and *trapezoidal* shapes have been used. Most of the values for the *bases* and the *peaks* have been established using the expertise. Only for some variables, the *membership functions* have been defined on an experimental basis. While we already had clear ideas on how to define most of them, we did not have enough information from the knowledge on how to model *membership functions* for the variables related to abstention (*abst_frequency* and *abst_discrimination*). A calibration phase was required in order to refine the values for the bases and peaks of their *membership functions*. As a calibration set, test results from the *Science Faculty Admission Test* of the 2006 year were used. The *calibration set* was composed of 64 items with 5 options each. For each item, about one thousand records (students answers) were available, even if only a smaller random sample was considered. Test items and their results were inspected by a human expert who identified items which should have been discarded due to low *discrimination* and anomalous values for the variables related to abstention. We have found 5 items satisfying the conditions above: the mean values for *abst_discrimination* and *abst_frequency* were, respectively, 0.12 and 0.39, as shown in Table 2.

Due to the limited size of the calibration set, the simple method of choosing the peaks of the functions at the mean value, as shown in (Bardossy & Duckstein; 1995), has been used. When more data will be available, a more sophisticated method will be used for the definition of membership functions, such as the one proposed in (Civanlar & Trussel; 1986). Charts for the membership functions are shown in Figure 2.

Table 2. Anomalous values forvariables related to abstention.

Question Id	abst_discrimination	abst_frequency
23	0.03	0.26
29	0.10	0.53
33	0.14	0.42
34	0.18	0.32
61	0.17	0.42
Mean	**0.12**	**0.39**

Rules

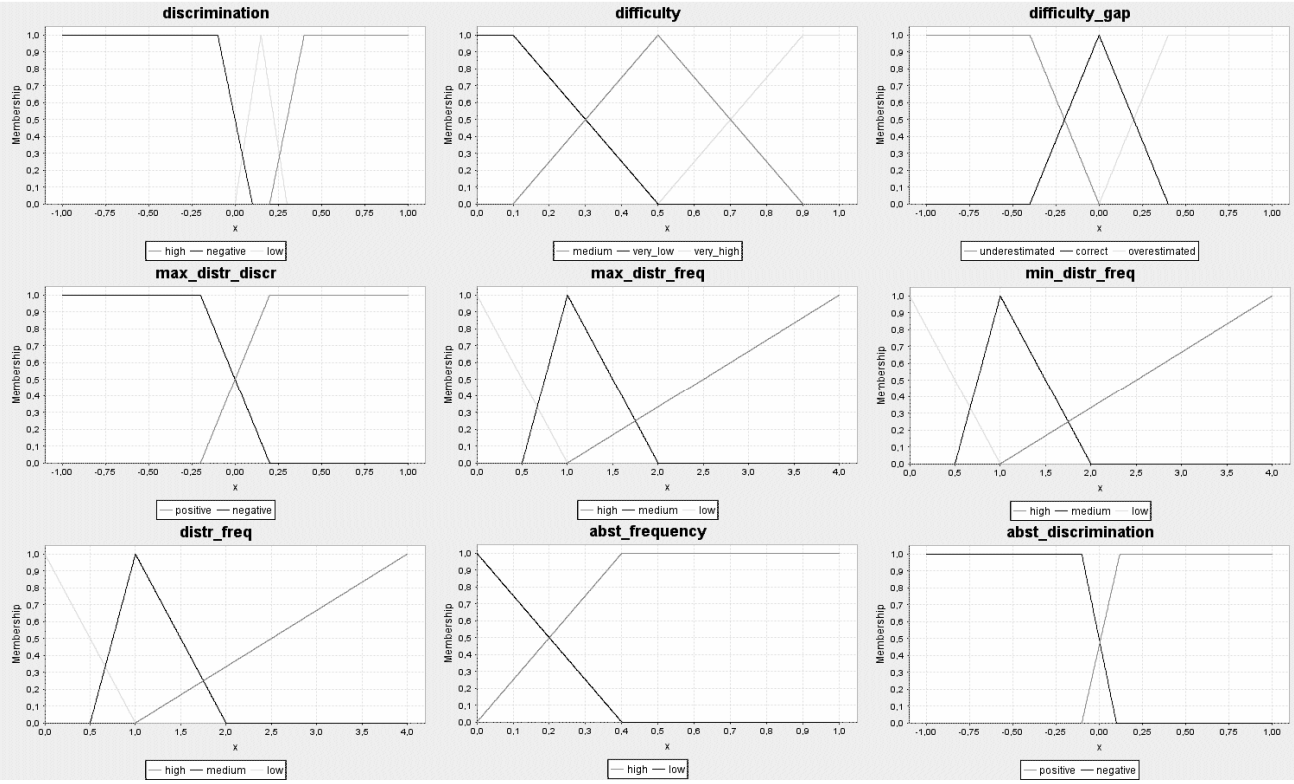


Figure 2. Membership Functions

Table 3. Rules

Class	Rule	State	Problem and Suggestion
1	`discrimination IS high AND abst_discrimination IS negative WITH 0.9`	Green	/
2	`discrimination IS low AND abst_frequency IS high AND abst_discrimination IS positive`	Red	/
3	`difficulty IS very_low AND discrimination IS low`	Red	/
4	`difficulty IS very_high AND discrimination IS low AND max_distr_freq IS high`	Yellow	Item too difficult due to a too plausible distractor, delete or substitute distractor *x*.
5	`difficulty_gap IS overestimated AND discrimination IS low`	Yellow	Item difficulty overestimated, avoid too plausible distractors and too obvious answers.
6	`difficulty_gap IS overestimated AND discrimination IS NOT low`	Yellow	Item difficulty overestimated, modify the estimated difficulty.
7	`difficulty_gap IS underestimated AND max_distr_freq IS high`	Yellow	Item difficulty underestimated due to a too plausible distractor, delete or substitute distractor *x*.
8	`difficulty_gap IS underestimated AND max_distr_freq IS NOT high`	Yellow	Item difficulty underestimated, modify the estimated difficulty.
9	`max_distr_discr IS positive AND discrimination IS negative`	Yellow	Wrong key (data entry error), select option *x* as the correct answer.
10	`discrimination IS high AND max_distr_discr IS positive AND distr_freq IS NOT low`	Yellow	Too plausible distractor, delete or substitute distractor *x*.

From the verbal description of the knowledge presented in section 3, the rules summarized in Table 3 have been inferred. The first three columns in the table contain, respectively, the class of the item, the rule used for classification and the item state. For items whose state is yellow, the fourth column contains the problem affecting the item and the suggestion to improve its quality.

Conditions in the rules are connected using AND and OR logic operators. The commonly-used *min-max* inference method has been used to establish the *degree of fulfillment* of the rules. All the rules were given the default *weight* (1.0), except for the first one (0.9). By modifying the weight of the first rule, we can tune the sensitivity of the system: the lower this value, the higher the probability that anomalies will be detected in the items. Some suggestions in the last column advise to perform an operation on a *distractor*. The *distractor* to modify or eliminate (in case of rules 4, 7 and 10) or to select as correct answer (rule 9) is signaled by the system. An output variable *x* has been added to the system to keep the identifier of the *distractor*.

It is worth noting that the most important *IA* statistical indicators have been employed more frequently than other indicators. For example, the *discrimination*, which is an good indicator for the overall quality of an item, is present in 8 rules out of 10, while a more specific indicator, such as *distractor discrimination* has only been employed in 2 rules.

The Inferential Engine

The *inferential engine* performs a process composed of the following steps:

1. Obtaining input data from the e-testing platform;
2. Construction of the *item data matrix*;
3. Item classification;
4. Giving output to the e-testing platform.

In step 1, data are obtained from the e-testing platform in which the *Item Quality Framework* is instantiated. This operation required the development of a wrapper to access the e-testing platform database.

The input data obtained at the previous step, are used in step 2 for the construction of the *item data matrix* which reports, for each item, the value of the following attributes:

- *N*: number of options;
- *key*: the index of the right option;
- *discrimination*: item discrimination;
- *difficulty*: item difficulty;
- *tutor_difficulty*: tutor's estimation for item difficulty;
- *discrimination (1); … ; discrimination(N)*: N columns containing the discrimination of each option.
- *difficulty (1); … ; difficulty(N)*: N columns containing the difficulty of each option;
- *abstained_discr*: discrimination of the abstention on the item;
- *abstained_freq*: frequency of the abstention on the item.

Item classification is performed, at step 3, by firing the rules. Before the rules can be fired, their variables must be assigned to values directly taken from the *item data matrix* (e.g. *discrimination*, *difficulty*, etc.) or derived from them (e.g. *difficulty_gap*, *max_distr_freq*, etc.). Then, the rules are fired and a new matrix containing the *DoF* for each item and for each class is obtained. As stated before, the item is classified in the class with the maximum *DoF*.

Lastly, at step 4, the output with item state, problem and suggestion, is passed to the *e-testing* platform.

System Implementation and Interface

The system was implemented in two phases:

1. Development of the *Item Quality Framework*;
2. Its instantiation in an existing Web-based e-testing platform, called *eWorkbook*.

The Item Quality Framework

The system has been implemented as a Java Object Oriented framework. In this way, it would have been easily integrated in any e-testing java-based platform. The *Item Quality Framework* offers the following functionalities:

- Implements the inferential engine;
- Provides an *Application Programming Interface* (*API*) for both the construction of the *item data matrix* and the access to output data.

For the development of the *inferential engine*, a free java library implementing a complete Fuzzy inference system, called *jFuzzyLogic* (jFuzzyLogic, 2008), has been used. The system variables, fuzzyfication, inference methods and the rules have been defined using *Fuzzy Control Language* (FCL, 1997) , supported by the *jFuzzyLogic* library. The advantage of this approach, compared to a hard-coded solution, is that *membership functions* and rules can be simply changed by editing a configuration file, thus avoiding to build the system again. Data can be imported from various sources and exported to several formats, such as spreadsheets or relational databases. The data matrix and the results can be saved in persistent tables, in order to avoid to perform calculations every time they must be visualized.

The *API* is composed of two different Java classes, which allow to perform input and output to the *Inferential Engine*, respectively. The former contains methods for adding rows to the *item data matrix*. The latter contains methods for obtaining the *state* of an item (*green*, *yellow*, *red*) and, in case of *yellow state*, the *suggestion* for improving the item quality. It is worth noting that suggestions can be internationalized, that is, they can easily be translated into any language by editing a text file.

Instantiation in eWorkbook

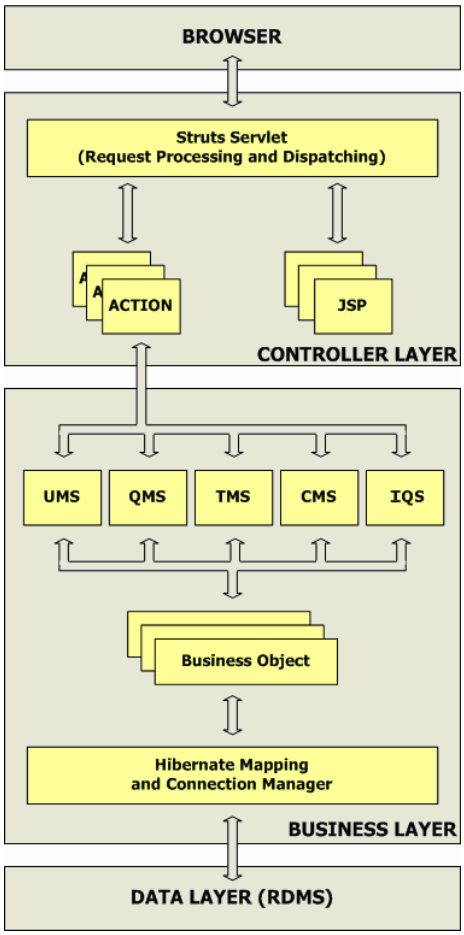


Figure 3. eWorkbook Architecture (after the instantiation of the Item Quality Framework)

eWorkbook is a Web-based e-testing platform that can be used for evaluating learner's knowledge by creating (the tutor) and taking (the learner) on-line tests based on *multiple choice* question type. The questions are kept in a *hierarchical* repository. The tests are composed of one or more sections. There are two kinds of sections: *static* and *dynamic*. The difference between them is in the way they allow question selection: for a static section, the questions are chosen by the tutor. For a dynamic section, some selection parameters must be specified, such as the difficulty, leaving the platform to choose the questions randomly whenever a learner takes a test. In this way, it is possible with *eWorkbook* to make a test with banks of items of different difficulties, thus balancing test difficulty, in order to better assess a heterogeneous set of students.

As shown in Figure 3, *eWorkbook* has a layered architecture. The *Jakarta Struts framework* (Struts, 2008) has been used to support the *Model 2* design paradigm, a variation of the classic *Model View Controller* (*MVC*) approach. In our design choice, *Struts* works with *JSP*, for the *View*, while it interacts with *Hibernate* (Hibernate, 2008), a powerful framework for *object/relational persistence* and query service for Java, for the *Model*. The application is fully accessible with a Web browser. No browser plug-in installations are needed, since its pages are composed of standard HTML and *ECMAScript* (EcmaScript, 2008) code. The Web browser interacts with the *Struts Servlet*, at the *Controller Layer*, that processes the request and dispatches it to the *Action Class*, responsible for serving it, according to the predefined configuration. It is worth noting that the *Struts Servlet* uses the JSP pages to implement the user interfaces. The *Action Classes* interact with the modules of the *Business Layer*, responsible for the logic of the application. At this layer, the functionalities of the system are implemented in four main sub-systems:

- *User Management Subsystem* (UMS), responsible for user management. In particular, it provides insert, update and delete facilities.

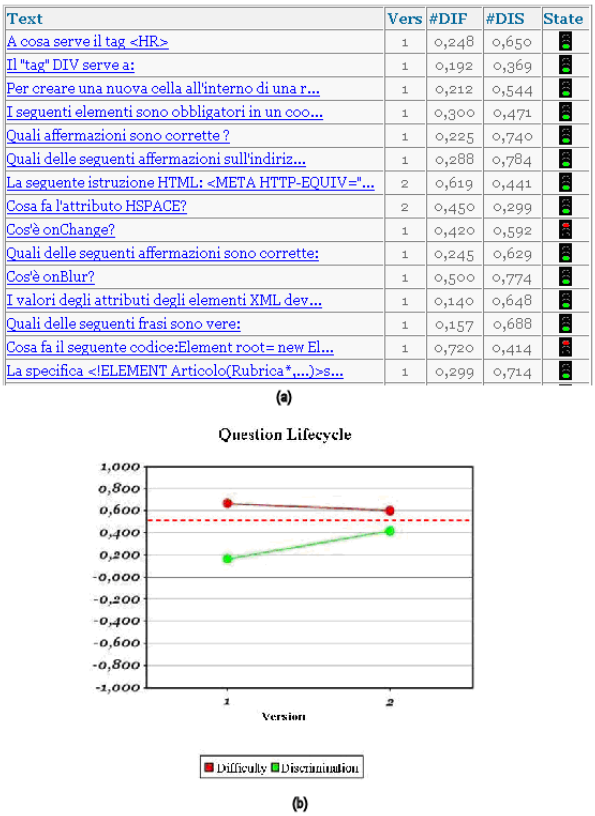
- *Question Management Subsystem* (QMS), which manages *eWorkbook*'s question repository and controls access to it.
- *Test Management Subsystem* (TMS), which manages *eWorkbook*'s test repository.
- *Course Management Subsystem* (CMS), responsible for course management. In particular, it allows the insertion, update and deletion of a course.

The *Business Layer* accesses to the *Data Layer*, implemented through a Relational Data Base Management System (RDBMS), to persist the data across the functionalities provided by *Hibernate* framework.

The integration of the new functionalities in *eWorkbook* has required the development and integration in the platform of new modules at all the layers. In particular, a new sub-system, called *Item Quality Sub-System* (IQS), responsible for instantiating the framework and providing input, output and visualization functionalities, has been added at the *Business Layer*.

Further minor modules have been added at the other layers: input of data is performed by a wrapper module that reads data from *eWorkbook*'s database and calls the *API* to fill the data matrix of the framework; the interface for browsing the item repository in *eWorkbook* has been modified in order to show item's performances (*difficulty* and *discrimination*) and state (*green*, *yellow* or *red*). In this way, defective items are immediately visible to the tutor, who can undertake the opportune actions (delete or modify). A screenshot of the item report is shown in figure 4a.

Furthermore, the platform has been given a *versioning functionality*: once an item is modified, a newer version of it is generated, keeping the old data in the question repository. Through this functionality, the tutor can analyze the entire lifecycle of an item, thus having a feedback on the trend of statistical indicators over time. In this way he/she can verify that the changes he/she made to the items positively affected their quality. Figure 4b shows the chart of an item improved across two sessions of tests. The improvement is visible both from the increase in the item *discrimination* (the green line), and in the convergence of the calculated difficulty with the tutor's estimation of the difficulty (the continuous and dashed red lines, respectively).



Text	Vers	#DIF	#DIS	State
A cosa serve il tag <HR>	1	0,248	0,650	
Il "tag" DIV serve a:	1	0,192	0,369	
Per creare una nuova cella all'interno di una r...	1	0,212	0,544	
I seguenti elementi sono obbligatori in un coo...	1	0,300	0,471	
Quali affermazioni sono corrette ?	1	0,225	0,740	
Quali delle seguenti affermazioni sull'indiriz...	1	0,288	0,784	
La seguente istruzione HTML: <META HTTP-EQUIV="...	2	0,619	0,441	
Cosa fa l'attributo HSPACE?	2	0,450	0,299	
Cos'è onChange?	1	0,420	0,592	
Quali delle seguenti affermazioni sono corrette:	1	0,245	0,629	
Cos'è onBlur?	1	0,500	0,774	
I valori degli attributi degli elementi XML dev...	1	0,140	0,648	
Quali delle seguenti frasi sono vere:	1	0,157	0,688	
Cosa fa il seguente codice:Element root= new El...	1	0,720	0,414	
La specifica <!ELEMENT Articolo(Rubrica*,...)>s...	1	0,299	0,714	

(a)

(b)

Figure 4. eWorkbook Interface

6. EXPERIMENTAL RESULTS

We experimented the system by using it across two test sessions in a university course, and measuring the overall improvement of the items in terms of discrimination capacity and matching to the tutor's desired difficulty. A database of 50 items was arranged for the experiment. In the first session, an on-line test, containing a set of 25 randomly chosen items, was administered to 60 students. After, items were inspected through the system interface in order to check those to substitute or modify. Once the substitutions and modifications were performed, the modified test was administered to 60 other students.

Figure 7a shows a table, exported in a spreadsheet, containing a report of the items presented in the first test session and their performances. The item to eliminate are highlighted in red, while those to modify are highlighted in yellow. According to the system analysis, 5 out of 25 items must be discarded, while 4 of them must be modified.

Actually, among the items to modify, for two of them (those with id 1-F-4 and 1-E-1) the *difficulty* was underestimated due to a *distractor* that was too plausible (class 7), whose text was opportunely modified. In another case (1-B-16), the difficulty was different from that estimated by the tutor, due to the intrinsic difficulty of the item (class 8). The action undertaken was to adjust the tutor's estimation of the difficulty. Lastly, the item with id 1-F-1, with a negative *discrimination*, presented a suspect error in the choice of the key (class 9).

To give the reader a more precise idea, two modified items (opportunely translated from Italian) have been reported in Figure 5 and Figure 6. In item 1-F-4 (Figure 5a), a *distractor* (option D) was sought to be "too plausible". Since the *distractor* was chosen by too many learners (26 out of 60 = 0.43%), the item was much more difficult (*difficulty* = 0.79) than expected (medium = 0.5). The tutor modified the distractor by changing the text from "Refreshes the content of the page http://www.expedia.it/info.htm in 20 seconds" to "Refreshes 20 times the content of the page http://www.expedia.it/info.htm" (Figure 5b). Such a modification significantly decreased the distractor plausibility, thus obtaining a difficulty level (0.43) for the item closer to the desired one, in the second session.

1-F-4. The following HTML statement: <META HTTP-EQUIV="refresh" CONTENT="20;URL=http://www.expedia.it/info.htm">
A. Includes the first 20 rows of the page http://www.expedia.it/info.htm into the one containing the statement
B. Includes the page http://www.expedia.it/info.htm into the one containing the statement in 20 seconds
***C. Loads the specified page http://www.expedia.it/info.htm* in 20 seconds**
D. Refreshes the content of the page http://www.expedia.it/info.htm in 20 seconds
E. Refreshes the content of the current page in 20 seconds

(a)

1-F-4. The following HTML statement: <META HTTP-EQUIV="refresh" CONTENT="20;URL=http://www.expedia.it/info.htm">
C. Includes the first 20 rows of the page http://www.expedia.it/info.htm into the one containing the statement
B. Includes the page http://www.expedia.it/info.htm into the one containing the statement in 20 seconds
***C. Loads the specified page http://www.expedia.it/info.htm* in 20 seconds**
D. Refreshes 20 times the content of the page http://www.expedia.it/info.htm
E. Refreshes the content of the current page in 20 seconds

(b)

Figure 5. The versions of item 1-F-4 used for the first (a) and the second (b) test session.

By inspecting item 1-F-1 (Figure 6a), the tutor verified that the chosen *key* was not correct, even though the *distractor* labeled as *correct* by the system was not the right answer: simply, the item did not have any correct answer. The text of the key was modified to provide the right answer to the stem (Figure 6b).

1-F-1. What is the role of the HSPACE attribute?
A. *It sets the horizontal space among images*
B. It inserts a Java Applet in the page
C. It sets the horizontal space between text attributes
D. It sets the vertical space among images
E. It sets the horizontal space between images and page borders

(a)

1-F-1. What is the role of the HSPACE attribute?
A. *It sets the horizontal space between the image and surrounding text*
B. It inserts a Java Applet in the page
C. It sets the horizontal space between text attributes
D. It sets the vertical space among images
E. It sets the horizontal space between images and page borders

(b)

Figure 6. The versions of item 1-F-1 used for the first (a) and the second (b) test session.

A new test was prepared, containing the same items, except for the 5 discarded ones, substituted by 5 unused items, and for the 4 modified ones, which were substituted by newer versions. A new set of sixty students participated in this test. In the analysis of test outcomes, our attention was more focused on the eventual improvement obtained than on the discovery of new defective items.

Question Id	Options	Correct	Discrimination	Tutor Diff	Difficulty
1-F-1	5	1	-0,04	0,7	0,76
1-B-10	5	3	0,51	0,5	0,24
1-B-6	5	4	0,6	0,5	0,58
1-A-19	5	4	0,22	0,5	0,29
1-D-2	5	2	0,7	0,5	0,82
1-B-4	5	5	0,42	0,5	0,34
1-A-13	5	1	0,33	0,3	0,08
1-F-4	5	3	0,32	0,5	0,79
1-B-18	5	4	0,55	0,3	0,53
1-A-15	5	5	0,37	0,5	0,66
1-B-2	5	5	0,59	0,5	0,45
1-E-4	5	4	0,4	0,5	0,79
1-C-1	5	1	0,07	0,5	0,39
1-B-12	5	4	0,48	0,3	0,74
1-A-24	5	2	0,36	0,3	0,37
1-D-3	5	5	0,15	0,5	0,53
1-C-4	5	1	0,16	0,5	0,74
1-B-16	5	3	0,41	0,5	0,76
1-A-9	5	3	0,57	0,3	0,53
1-B-20	5	5	0,38	0,3	0,47
1-A-2	5	2	0,21	0,3	0,34
1-B-8	5	5	0,49	0,5	0,29
1-C-5	5	1	0,17	0,7	0,87
1-B-15	5	4	0,52	0,5	0,42
1-E-1	5	4	0,44	0,5	0,87
Average Discrimination			0,3752		
Average Difficulty Gap			0,19		

(a)

Question Id	Options	Correct	Discrimination	Tutor Diff	Difficulty
1-F-1	5	1	0,48	0,7	0,67
1-B-10	5	3	0,54	0,5	0,20
1-B-6	5	4	0,58	0,5	0,57
1-D-4	5	3	0,08	0,5	0,76
1-D-2	5	2	0,62	0,5	0,78
1-B-4	5	5	0,42	0,5	0,30
1-A-13	5	1	0,39	0,3	0,07
1-F-4	5	3	0,47	0,5	0,43
1-B-18	5	4	0,63	0,3	0,45
1-A-15	5	5	0,40	0,5	0,71
1-B-2	5	5	0,56	0,5	0,39
1-E-4	5	4	0,40	0,5	0,88
1-A-17	5	5	0,44	0,3	0,45
1-B-12	5	4	0,46	0,3	0,65
1-A-24	5	2	0,38	0,3	0,29
1-D-3	5	5	0,13	0,5	0,44
1-A-23	5	3	0,38	0,3	0,32
1-B-16	5	3	0,40	0,7	0,84
1-A-9	5	3	0,50	0,3	0,52
1-B-20	5	5	0,45	0,3	0,46
1-F-3	5	5	0,76	0,7	0,72
1-B-8	5	5	0,55	0,5	0,25
1-B-5	5	5	0,66	0,3	0,50
1-B-15	5	4	0,59	0,5	0,49
1-E-1	5	4	0,37	0,5	0,58
Average Discrimination			0,4660		
Average Difficulty Gap			0,1573		

(b)

Figure 7. Results after test sessions

Figure 7b shows the report of the second test session. To measure the overall improvement of the new test, compared to the previous one, the following parameters were calculated for each of the two tests:

- the average *discrimination* of the items;
- the average of the differences |*tutor_difficulty – difficulty*| for the items of the tests;

As for parameter 1, we have observed an improvement from a value of 0.375, obtained in the first session, to a value of 0.466, obtained in the second session. As for parameter 2, we had a decrement in the mean difference between the difficulty estimated by the tutor and the one calculated by the system, passing from a value of 0.19 to 0.157 across the two sessions.

It is worth noting that, in our experiment, the tests have been administered to learners enrolled to the same university course, even if across different exam sessions. The results can be considered valid with respect to

the above requirement. Due to the dependency of *IA* results on the learners' ability, there is no warranty that the system behaves in the expected way when radically changing the context between different sessions.

7. RELATED WORK

Several different applications supporting e-testing, such as the most common Web-based e-learning platforms, such as *Moodle*, *Blackboard*, and *Questionmark*, evaluate item quality by generating and showing item statistics. Nevertheless, in most cases, the interpretation of their results is left to the tutors: these systems do not advise or help the tutor in improving items.

Several commercial stand-alone applications are available for improving test quality through *IA* (Integrity, 2008; Berk & Griesemer, 1976; Lertap, 2008) or *IRT* (RASCAL, 2008; Gierl & Ackerman, 1996). These can import test data from e-testing systems through a text file. Some of them are Web-based applications, such as *Integrity*. It can perform a detailed test analysis which also identifies problem areas and includes relevant recommendations for addressing them. Differently from our system, parameters are not combined in rules: a recommendation is given when for a given parameter an anomalous value is sought.

Some other systems run under specific platforms (OS or spreadsheets). A program running under MS Windows is *ITEMAN* (Berk & Griesemer, 1976). *ITEMAN* analyzes data files (ASCII format) of test item responses produced by optical mark readers (scanners) or by manual data entry to compute conventional item analysis statistics. *ITEMAN* offers a multiple-keying option that allows items to have more than one correct answer (e.g., for a poorly-written item), and will flag those answers which appear to function better than the keyed answer. Our system does something similar by firing rule 9.

An application running in a spreadsheet is *Lertap*, an Excel-based classical item and test analysis program. A nice feature of this program is the so called *Visual Item Analysis*, suggesting an ocular approach to item analysis, and exemplifying some of the graphics made by *Lertap*.

A model for presenting test statistics, analysis, and to collect students' learning behaviors for generating analysis result and feedback to tutors is described in (Hsieh et al., 2003).

In other approaches, the qualitative characteristics of the items are considered for different aims: *IRT* has been applied in some systems (Ho & Yen, 2005) and experiments (Chen et al., 2004; Sun, 2000) to select the most appropriate items for examinees based on individual ability. In (Chen et al., 2004), the fuzzy sets theory is combined with the original *IRT* to model uncertainly learning response. The result of this combination is called *Fuzzy Item Response Theory*. Winters et al. (2005), mining the data of their educational institutions, found some scores that could be analyzed with the purpose of identifying those items that were particularly good or particularly bad, giving instructors feedback that will hopefully train them to ask better questions more consistently.

A work closely related to ours is presented in (Hung et al., 2004). It proposes an e-testing system, where rules can detect defective items, which are signaled using traffic lights. It proposes an analysis model based on *IA*. Statistics are calculated by the system both on the items and on the whole test. Unfortunately, the four rules on which the system is based seem to be insufficient to cover all of the possible defects which can affect an item. Moreover, these rules are not inferred from consolidated statistical models and use crisp values (i.e., one of them, states that an option must be discarded if its frequency is 0, independently from the size of the sample). Furthermore, it does not contain any experiment which demonstrates the effectiveness of the system in improving assessment. Nevertheless, this work has given us many ideas, and our work can be considered a continuation of it. To elaborate, our system improves the above cited one in the following aspects:

- it broadens and improves the rules used to check the items;
- it gives advice to tutors to improve item quality;
- it manages rules uncertainty (using fuzzy logic);
- it has been evaluated in an experiment.

Lastly, most of the scientific literature about *e-testing* and structured tests focuses on item generation with automatic (Mitkov & Ha, 2003; Brown et al., 2005) or semi-automatic (Wang et al., 2007; Hoshino &

Nakagawa, 2007; Chen et al., 2006) processes based on *Natural Language Processing* (*NLP*) techniques, performed on instructional documents in an electronic format. The automatic systems generate the items, while the semi-automatic ones assist the user in their generation. In general, the human intervention is anyhow necessary for verifying the good sense of the items before using them in a test. Only in a few cases, the quality of the generated items is verified through statistical model such as *IA* or *IRT*. In most cases they are inspected and eventually modified by the tutor. The evaluation of the whole system is performed by checking the percentage of the reliable items out of the number of generated ones.

In conclusion, we believe that tools that automatically generate items or assist the tutor in their creation, as those described before, can be very useful, since they permit to reduce the times of the onerous item construction phase. Nevertheless, they are still far from offering optimal performances and many of the analyzed systems are tailored for a specific educational subject, mostly foreign language teaching. Our approach is more general and can be applied to any subject. Furthermore, many tutors will keep on using their own items and our system is still applicable to generated items, for further improving their quality. Our system, compared to automatic or semi-automatic ones, requires a longer time for item construction, but allows us to obtain better quality items on the following aspects:

- a better discrimination capacity;
- evaluation of the learners on tutor's desired knowledge;
- a difficulty level closer to tutor's desired one.

8. CONCLUSION

In this paper we have presented a *rule-based* system, capable of improving item quality. Our system's knowledge-base is mostly taken from several statistical models for item evaluation and partly extracted from data.

The system detects anomalies on the items and gives tutors advise for their improvement. Obviously, the system can only detect defects which are visible by analyzing results of *item* and *distractor analysis*.

The strength of our system is in the possibility for all the tutors, and not only experts of assessment or statistics, to improve test quality, by discarding or, when possible, by modifying defective items. An initial experiment carried out at the University of Salerno has produced encouraging results, showing that the system can effectively help the tutors to obtain items which better discriminate between skilled and untrained students and better match the difficulty estimated by the tutor. More accurate experiments, involving a larger set of items and students, are necessary to better measure the system capabilities.

Our system performs a classification of items, carried out by evaluating fuzzy rules. At present, we are collecting data on test outcomes. Even though fuzzy classification has proven itself to perform well, we intend to investigate also other classification methods, such as *decision trees* and *Bayesian classifiers*, once a large database of items and learner's answers will be available.

REFERENCES

Bardossy A., Duckstein L. (1995). Fuzzy Rule-Based Modeling with Applications to Geophysical, Biological, and Engineering Systems. CRC Press, Boca Raton, USA.

Berk, R. A., Griesemer, H. A. (1976). Software Review: ITEMAN: An Item Analysis Program for Tests, Questionnaires, and Scales. *Educational and Psychological Measurement, 36 (1)* (pp. 189-191).

Blackboard (2008). Available at http://www.blackboard.com.

Brachman R.J., Khabaza T., Kloesgen W., Pieatetsky-Shapiro G., Simoudis E. (1996). Mining Business Databases. *Communications of the ACM*, 39 (11). (pp 42-48).

Bradley R.S., Barry R.G., Kiladis G. (1982). Climatic fluctuations of the western United States during the period of instrumental records. *Final report to the National Science Foundation*. University of Massachussets, Amherst.

Brown J.C., Frishkoff G.A., Eskenazi M. (2005). Automatic Question Generation for Vocabulary Assessment. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp 819-826).

Chen C.M., Duh L.J., Liu C.Y. (2004). A Personalized Courseware Recommendation System Based on Fuzzy Item Response Theory. *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service*, Taipei, Taiwan (pp. 305—308).

Chen C-Y, Liou H-C, Chang J.S. (2006). FAST: an automatic generation system for grammar tests. *Proceedings of the COLING/ACL on Interactive Presentation Sessions* (pp 1-4).

Civanlar M.R., Trussel H.J. (1986). Constructing membership functions using statistical data. *Fuzzy Sets and Systems*, 18 (pp. 1—14).

Costagliola G., Ferrucci F., Fuccella V., Oliveto R. (2007). eWorkbook: a Computer Aided Assessment System. *International Journal of Distance Education Technology*, 5 (3), (pp. 24—41).

ECMAScript (2008). Standard ECMA-262, ECMAScript Language Specification, available at http://www.ecma-international.org /publications/files/ECMA-ST/Ecma-262.pdf.

Exarchos T. P., Tsipouras M. G., Exarchos C. P., Papaloukas C., Fotiadis D. I., Michalis L. K. (2007). A methodology for the automated creation of fuzzy expert systems for ischaemic and arrhythmic beat classification based on a set of rules obtained by a decision tree. *Arificial Intelligence in Medicine*, 40 (3), (pp. 187-200).

FCL (1997): Fuzzy Control Prog. Committee Draft CD 1.0 (Rel. 19 Jan 97), http://www.fuzzytech.com/binaries/ieccd1.pdf.

Gierl, M. J., Ackerman, T. (1996). Software Review: XCALIBRE — Marginal Maximum-Likelihood Estimation Program, Windows Version 1.10. *Applied Psychological Measuremen,t 20 (3)* (pp. 303-307).

Hambleton R. K., Swaminathan H. (1985). *Item Response Theory - Principles und Applications*. Netherlands: Kluwer Academic Publishers Group.

Hibernate (2008), available at http://www.hibernate.org.

Ho R.G., Yen Y.C. (2005). Design and Evaluation of an XML-Based Platform-Independent Computerized Adaptive TestingSystem. *IEEE Transactions on Education*, Vol. 48, No. 2 (pp. 230—237).

Hoshino A., Nakagawa H. (2007). A Cloze Test Authoring System and Its Automation. *Proceedings of 6th International Conference on Web-Based Learning* (pp. 174- 181).

Hsieh C.T., Shih T.K, Chang W.C., Ko W.C. (2003). Feedback and Analysis from Assessment Metadata in E-learning. *Proceedings of the 17th International Conference on Advanced Information Networking and Applications*, Xi'an, China, (pp. 155—158).

Hung J.C., Lin L.J., Chang W.C., Shih T.K., Hsu H.H., Chang H.B., Chang H.P., Huang K.H. (2004). A Cognition Assessment Authoring System for E-Learning. *Proceedings of the 24th Int. Conf. on Distributed Computing Systems Workshops* (pp. 262—267).

IA (2008). *Item Analysis*. Available at http://www.washington.edu/oea/pdfs/resources/item_analysis.pdf.

Integrity (2008). Integrity - Item analysis and collusion detection tools. http://integrity.castlerockresearch.com/

jFuzzyLogic (2008): Open Source Fuzzy Logic (Java). Available at http://jfuzzylogic.sourceforge.net/html/index.html.

Lertap (2008). Lertap 5! http://www.lertap.curtin.edu.au/

Mamdani E.H., Assilian S. (1999). An experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Human-Computer Studies*, 51 (2), (pp. 135-147).

Massey (2007). The Relationship Between the Popularity of Questions and Their Difficulty Level in Examinations Which Allow a Choice of Question. *Occasional Publication of The Test Dev. and Res. Unit*, Cambridge.

MEAP (2007). State of Michigan – Department of Education. *Design and Validity of the MEAP Test*. Available at http://www.michigan.gov/mde/0,1607,7-140-22709_31168-94522--,00.html.

Mitkov R., Ha L.A. (2003). Computer-Aided Generation of Multiple-Choice Tests. *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing - Volume 2* (pp. 17-22).

Momoh J., Srinivasan D., Tomsovic K., Baer D. (2000). Chapter 5: Expert Systems Applications, in K. Tomsovic, M.Y. Chov (eds.), *Tutorial on Fuzzy Logic Applications in Power Systems*.

Moodle (2008). Available at http://moodle.org.

Questionmark (2008). Available at http://www.questionmark.com.

RASCAL (2008). RASCAL - Rasch Analysis Program. http://www.assess.com/xcart/product.php?productid=253&cat=29&page=1

Roiger R.J., Geatz M.W. (2007). *Introduzione al Data Mining* (in Italian), McGraw-Hill.

Stage C. (1999). A Comparison Between Item Analysis Based on Item Response Theory and Classical Test Theory. A *Study of the SweSAT Subtest READ*. Available at http:// www. umu. se/ edmeas/ publikationer/ pdf/ enr3098sec.pdf.

Struts (2008), The Apache Struts Web Application Framework, http://struts.apache.org

Sun K.T. (2000). An Effective Item Selection Method for Educational Measurement. *Proceedings of the International Workshop on Advanced Learning Technologies* (pp. 105—106).

Wang W., Hao T., Liu W. (2007). Automatic Question Generation for Learning Generation in Medicine. *Proceedings of 6th International Conference on Web-Based Learning* (pp. 198-203).

Winters T., Payne T. (2005). What Do Students Know? An Outcomes Based Assessment System. *Proceedings of the 2005 international workshop on Computing education research* (pp. 165-172).

Woodford K., Bancroft P. (2005). Multiple Choice Items Not Considered Harmful. *Proceedings of 7th Australian Conference on Computing Education* (pp. 109—116).

Yu, C. H. (2005). A Simple Guide to the Item Response Theory (IRT) http://seamonkey.ed.asu.edu/~alex/computer/sas/IRT.pdf

Yu, C. H., Wong, J. W. (2003). Using SAS for classical item analysis and option analysis. *Proceedings of 2003 Western Users of SAS Software Conference*. http://www.lexjansen.com/wuss/2003/DataAnalysis/c-using_sas_for_classical_item_analysis.pdf

Zadeh L. A. (1977). *Fuzzy Sets and Their Applications to Pattern Classification and Clustering*, World Scientific Publishing Co. Inc., River Edge, NJ, USA.

Zhou J., Li Q., Xu D., Chen Y., Xiao T. (2005). Fuzzy Rule-based Integrated System Multi-indicators Economic Performance Evaluation and Decision Making Support Framework. *Proceedings of International Conference on Computational Intelligence for Modelling, Control and Automation/ International Conference on Intelligent Agents, Web Technologies and Internet Commerce* (pp. 714-720).