Automatic summary of teachers’ error feedback based on taxonomy

Hakim Mokeddem*
Ecole nationale Supérieure d’Informatique ESI,
BP 68M Oued-Smar,
16309 Algiers, Algeria
Email: h_mokeddem@esi.dz
*Corresponding author

Cyrille Desmoulins and Nadine Mandran
Université Grenoble Alpes, LIG, CNRS,
F-38000 Grenoble, France
Email: cyrille.desmoulins@imag.fr
Email: nadine.mandran@imag.fr

Rachid Chalal
Ecole nationale Supérieure d’Informatique ESI,
BP 68M Oued-Smar,
16309 Algiers, Algeria
Email: r_chalal@esi.dz

Abstract: This paper presents an algorithm that computes the most common error types reported by teachers on students’ lab reports in LabBook system. These common error types aim to improve students’ learning by helping teachers to take necessary corrective actions. Computing the most common error types is not obvious because of their taxonomic structure. Thus, using frequencies of error types leads to select the uppermost type as the most common one. For this reason, two other parameters are taken into account in addition to the type frequency: a type generality level and a number of type subtypes. To define a computing algorithm, the most common error types are formalised with three rules where each rule uses one parameter. The algorithm proposed is based on ranking function that respects the three rules. It assigns a score to an error type by multiplying its frequency with a weight function based on information content. The feature that provides common error types to teachers was implemented using semantic web technologies. The results of a qualitative study conducted with teachers showed that experienced teachers used and combined the algorithm rules to select the most common error types that can help them take corrective actions efficiently.

Keywords: error types; formative assessment; semantic web; taxonomy.

Biographical notes: Hakim Mokeddem received master’s degree in computer science from the National Computer science Engineering School (ESI, Algiers, Algeria). He is currently a PhD candidate and an assistant professor at the National Computer science Engineering School. His research interests include issues related to formative assessment tools based on annotations, ontologies and semantic web reasoning for technology enhanced learning systems.

Dr Cyrille Desmoulins is an Assistant Professor in Computer Science at Grenoble Alpes University (France). He graduated in Computer Science in 1989 at this university where he received PhD in computer science in 1994. His main research interests are in computerised learning and knowledge representation, particularly focused on ontologies for education. He worked on various projects in this area, at European level (IMAT, InterGeo, and national level (Cartographie des Savoirs)). On the academic side, he leads the Master of Business Informatics at Grenoble Alpes University.

Nadine Mandran is an Engineer in Human and Social Sciences at the CNRS. Her area of expertise focus on methodologies for production and analysis of data. She is working at Grenoble Informatics Laboratory (LIG). Her mission is to provide support and advice to research teams for the design of studies and experiments where the human dimension must be taken into account. For each study, she is in charge for the complete process, from production to analyse data. For several years, she contributes to improve production methods and data analysis in the TEL, HMI and MDI. She has developed an interest for the sequential data analysis and the data visualisation analysis. She is also in charge of ‘quality process’ to manage support activities.

Rachid Chalal is a Professor at the National Computer science Engineering School (ESI, Algiers, Algeria). He is responsible for a team of researchers in the field of decision support at the LMCS Laboratory. His main research concerns cover information systems, competitive intelligence and decision support.

1 Introduction

Formative assessment or the assessment for learning supports teaching and learning (Oosterhof and Ely, 2007; Shute and Kim, 2014) and provides information to adjust teaching and to improve student learning (Leighton et al., 2013). Using annotations in online learning systems can help teachers send formative feedback to students on their works (Wolfe, 2002).

In this context, a qualitative study based on user-centred design methodology was conducted with teachers who use LabBook learning system in experimental sciences (Wajeman et al., 2015). The study aimed at getting teachers’ requirements regarding the use of MemoNote annotation tool to provide formative feedback annotations to students on their lab reports. The main study outcome is that providing the annotation facility to teachers is useful for giving formative feedback to students. More precisely, the teachers find useful adding formal semantics to annotations in order to get advanced features relying on this semantics.

To meet teachers’ requirements, a first version of MemoNote has been embodied in LabBook system with basic features. MemoNote enables teachers to add formal semantics
to their formative feedback annotations, thanks to an ontology which structures students’ error types and enables reporting errors on tasks that students are expected to master.

The main feature that interested teachers is to get the most common error types that were reported on students’ lab reports. Because these error types are reported on a lab report tasks, this feature could help a teacher take corrective actions in order to improve students’ learning outcomes. For instance, a teacher could explain in a different way an abstract concept that students had difficulties to apply in a lab session, or he could give students other exercises that deal with the same issue.

The feature that provides the most common error types has not been implemented in the first version of MemoNote because its specification and its computing algorithm are not obvious. Indeed, error types in MemoNote are structured into a taxonomy. Thus, an error can be reported at various level of detail and the most common error types to be provided to a teacher can be themselves at various levels. In this case, occurrences of an error type belong also to its supertypes and then to the uppermost type. A simple count leads choosing the uppermost type as most common error type, which is not the result expected. Consequently, a more elaborated count is needed to compute them.

The purpose of this paper is to develop and evaluate the algorithm that computes the most common error types that have a taxonomic structure.

This paper is structured as follows. In Section 2, we study related works on formative assessment systems. Section 3 presents the feedback ontology and the basic features of MemoNote in LabBook. Section 4 explains the computational problem of the most common error types in a taxonomy and the methodology used to study how to solve this problem. Section 5 defines formally the most common error types in a taxonomy. Section 6 details the algorithm that computes the most common error types. Section 7 describes its implementation in MemoNote. Section 8 presents the evaluation of the algorithm with teachers.

2 Related works

Formative assessment activities aim to monitor learning and assess learners’ understanding. This assessment intends to modify learning activities in order to enable students to achieve the desired level of knowledge (Gikandi et al., 2011). A study conducted in the context of an online learning system (Wolsey, 2008) demonstrates that formative feedback helps students identify their strengths and weaknesses and improve their learning continuously by reviewing provided feedback.

In this context, systems were developed to support the formative assessment activity in online learning. These systems can be divided into two categories.

The first category includes simple systems that enable teachers to provide feedback to students with comments written in natural language (Ahren, 2005). Using this category of systems, teachers can provide feedback to students easily. However, these feedback cannot be exploited to provide useful features, such as summarising common errors made by students to identify their difficulties.

The second category comprises systems that enable providing formative feedback to students using concepts (Yeh and Lo, 2009; Gouli et al., 2004; Mar Sánchez-Vera, 2012).
The annotation tool proposed in Yeh and Lo (2009) provides teachers with facilities that help them report errors on students’ essays. A list of error types organised into five categories is used to assign a type for each reported error. An error analyser feature provides the distribution of error types on students’ essays. This feature can help teachers identify students’ difficulties and improve students’ writing. However, error types made by students can be general, whereas the system provides only five specific error types that cannot cover all students’ errors.

COMPASS tool (Gouli et al., 2004) is a web-based concept map assessment system that supports assessment and learning activities. To learn using COMPASS, a learner must select a learning goal. The system provides a set of concepts that correspond to the selected learning goal. The learner is asked to build a concept map using these concepts. COMPASS then uses this concept map to automatically assess the learner understanding by providing qualitative feedback and a quantitative estimation of the learner’s knowledge level. The formative feedback provided to the learner has one of the three types: unknown concept, incomplete understanding and false beliefs. These feedback types depend on the category of learners’ errors that are organised in a simple list of error types.

OeLE platform (Mar Sánchez-Vera, 2012) uses ontologies and semantic links to generate formative feedback and quantitative marks in assessment tests based on open questions in accounting domain. Teachers semantically link questions to course concepts using a course ontology. The OeLE system semantically links students’ answers (free texts in this case) to this ontology, using natural language processing technologies. It then compares these student answer links to teachers ones using semantic similarity. Formative feedbacks provided are the best acquired concepts (near to required ones) and the worst acquired concepts (far from required ones). This system does not provide a range of error types but only one on a unique and very general error type, that is “a concept is not acquired”.

To summarise, these two systems can enable teachers to identify common students’ error types. However, these errors are very general and cannot help teachers take corrective actions easily.

To find systems that compute the most common types that have a taxonomic structure, we have explored other domains. For example, in social networks, the system proposed in Michelson and Macskassy (2010) helps find categories that cover the users’ topics of interest in twitter. These categories are organised into a taxonomy and are used to annotate users’ tweets. The category taxonomy is inverted, therefore the root of the taxonomy is the most specific category.

To compute the users’ topics of interest, the frequency, category level and average branching factor of the taxonomy are used as parameters to rank each category. The ranking function multiplies the frequency with the weight function \( W(c) = 1/b^d \), where \( b \) is the branching factor of the taxonomy and \( d \) is the depth of the category \( c \). This ranking function assigns very small weights to general categories and selects only specific categories as topics of interest. In the case of MemoNote, not only specific error types can be selected as the most common ones because teachers can report errors at different levels of generality and teachers can take a general corrective action such as explaining a method to achieve a task.
3 Research context

This section presents the research background which is divided into two parts. The first part presents the feedback ontology. The second part presents basic features of MemoNote in LabBook.

3.1 The feedback ontology

A feedback annotation ontology, called $\text{OntoNoteF}$ was developed to represent the formal semantics of feedback to be used by teachers in LabBook. $\text{OntoNoteF}$ extends the MemoNote annotation ontology.

In this annotation ontology, an annotation has a type represented by the $\text{Annotation}$ class. This $\text{Annotation}$ class is structured into subtypes. The three main subtypes are:

1. $\text{DocumentAnnotation}$: a teacher provides an editing opinion on the textual description of a learning activity.
2. $\text{DomainAnnotation}$: a teacher provides a technical opinion on a learning domain element of the document.
3. $\text{PedagogicalAnnotation}$: a teacher provides a pedagogical opinion on a learning objective of the document.

Additionally, the $\text{Annotation}$ subclasses are related to an $\text{ObjectiveItem}$ ontology that provides more details about the teacher’s opinion. For instance, in the chemistry domain, a teacher creates an annotation of type recall a course concept where the course concept (objective item) is dissolution rate.

To represent teachers’ feedback, types were designed under $\text{DomainAnnotation}$ class because teachers’ feedback relate to the results of a lab report and not its form neither its pedagogical objective. As a result, a feedback is an instance of the $\text{FeedbackAnnotation}$ class, which is a subclass of $\text{DomainAnnotation}$ class (see Figure 1).

Figure 1 The MemoNote error ontology
OntoNoteF is not specific to a particular learning domain but the domain is taken into account using a specific OntoPrax ontology for a particular lab domain. OntoPrax has two main classes: Task and Method. The Task class represents task types on which teachers report feedback, for example, resolve a quadratic equation. The instances of this class represent tasks that students are expected to perform, for example “resolve the equation $x^2 + 2x + 1 = 0$”. The Method class describes the different methods that can achieve task types, for example, using the discriminant or by completing the square.

Teachers’ feedbacks are positive and negative comments provided to students. However, in this paper, only negative comments that represent students’ errors are considered because the purpose is to help teachers take corrective actions in order to improve students’ learning outcomes.

OntoNoteF uses OntoPrax ontology as a model for learning domain items, linking annotations to OntoPrax items via hasFeedbackItem property (see Figure 1). The classes and properties of OntoPrax describe learning outcomes, i.e. tasks students are expected to master after a course and methods used to achieve these tasks.

3.2 MemoNote basic features in LabBook

MemoNote was embodied in LabBook with two basic features:

1. creating an annotation
2. creating and using patterns.

To provide a feedback to a student on its lab report, a teacher selects the text she wants to annotate. MemoNote immediately highlights it and displays a popup to be filled in with annotation properties coming from the annotation ontology: colour, textual comment, feedback type, task type and importance (see Figure 2). Session elements are automatically captured from LabBook. Previous textual comments reported on the same feedback type are displayed and the teacher can select one of these textual comments, without rewriting it.

MemoNote provides annotation pattern facility that assists teachers to specify rapidly and fluently the annotation semantics. An annotation pattern represents annotation habits of annotators, in this case teachers. In the context of MemoNote, the annotation pattern facility deduces partly or entirely the annotation semantics from its graphical form, colour and context elements. By checking use as template MemoNote creates the annotation pattern that corresponds to the current annotation colour, feedback type, and annotation session elements. If the annotation colour and session correspond to an existing annotation pattern, MemoNote prefills automatically the feedback type.

The main feature that interests teachers is to provide the most common error types. Its specification and implementation are presented in the next sections.
4 Research issue and methodology

This section aims first to define a research question for computing the most common error types and second to present the methodology used to address this research question.

4.1 Research issue

Using a simple count of occurrences to compute the most common error types leads choosing the uppermost type as the most common. This is because occurrences of a given error type belong recursively to its supertypes.

In the example of Figure 3, the number over a rectangle is the number of type occurrences. Providing a teacher the upper class `FeedbackAnnotation` as the most common error type cannot help her identify students’ errors. This error type is obviously purposeless and cannot help the teachers decide which corrective action to take. Consequently, computing the most common error types cannot consider only the number of errors reported on each type, but also other parameters.

The error type `TooGeneralMethod` is the most common error type because it has been reported to students frequently and is the most precise one on which a teacher can take remedial actions.

The research problem is how to compute the most common error types that have a taxonomic structure in order to help teachers improve students’ learning by taking corrective actions.
Figure 3  Computing the most common error type using a simple count (see online version for colours)

4.2 Research methodology

Five steps are followed in order to solve the research problem:

1. Illustrative examples are used to explain informally how to select common error types and why these types are the common ones.
2. Rules are defined to formalise the most common error types.
3. Algorithms are developed and evaluated according to these rules in order to select the relevant algorithm.
4. The feature that retrieves the most common error types is implemented and deployed in MemoNote embodied in LabBook.
5. An evaluation is conducted with teachers to assess the algorithm and its rules.

5 What are the most common error types?

The aim of this section is to define the most common error types that have a taxonomic structure. To achieve this goal, the common error types are first defined informally with examples and then formalised with rules.

5.1 Illustrative examples

The two following examples illustrate the selection of the most common error types reported on MemoNote taxonomy. For each example, we explain why error types are selected.
To simplify the explanation, we focus on a unique task, as if it was the only task on which errors were reported and we generalise then the explanation on multiple tasks.

Because the most common error types are calculated depending on a unique task type, the number over a rectangle is the number of type occurrences. Error types on green rectangles are the most common ones that can help teachers take necessary corrective actions.

**Example 1:** *MethodProblem* is not the most common error type even if it has a high number of reported errors because most of these errors are reported on one of its subtypes *TooSpecificMethod* (see Figure 4). Thus, *MethodProblem* is not the best error type on which a teacher can take corrective actions because this would require taking a general corrective action on the method or taking unnecessary corrective actions on two of its subtypes. For the same reason, *IrrelevantMethod* and *TaskProblem* types are not selected.

Consequently, *TooSpecificMethod* and *UnfulfilledTask* are the most common error types. *TooSpecificMethod* is selected first because it is more specific than *UnfulfilledTask* and therefore leads to more precise corrective actions.

**Example 2:** *MethodProblem* and *TaskProblem* are not selected even if they have a high number of reported errors (see Figure 4). *UnfulfilledTask* and *IncorrectMethod* are selected. *IncorrectMethod* is selected because errors on *IncorrectMethod* are reported equally on all its subtypes. Therefore, taking corrective actions on this error type would need to take a general corrective action or corrective actions on each of *IncorrectMethod* subtypes.

**Figure 4** The error taxonomy with instances (see online version for colours)

The error type *UnfulfilledTask* is the first selected because *UnfulfilledTask* does not have subtypes, which does not require taking corrective actions on different cases, one for each subtype, as for *IncorrectMethod*. 
5.2 The algorithm rules

The first criterion that should be considered is the number of type occurrences. However, selecting the most common error types is not only based on this criterion. The generality level of types, their number of subtypes should also be considered. These criteria are used to define the following rules.

**Frequency rule.** The frequency of an error type for a given task type is the ratio between the number of its occurrences reported on the task type and the number of occurrences of all error types reported on all task types.

The most common error type must have a high frequency. Therefore, the frequency rule can be expressed by “The higher is the error type frequency, the more the error type is common”. The frequency function is the main parameter of this rule.

**Generality level rule.** The depth of an error type is its generality level in the taxonomy. The generality level of an error type is an important parameter that should be taken into account. A specific error type gives teachers precise information about students’ difficulties. Consequently, the generality level rule can be expressed by “The higher is the error type depth, the more the error type is common”. The depth function is the main parameter of this rule.

**Number of subtype rule.** The number of subtypes of an error type is the number of all its subtypes. Subtypes of a given error type are more precise than this error type. If the most common error type has a lot of subtypes, then teachers need to find a general corrective action for this error, which could not be possible, or they should take corrective actions for each specific error subtype, which is also difficult. The rule can be expressed by “The fewer the error type has subtypes, the more the error type is common”. The number of subtype function is the main parameter of this rule.

To conclude, the purpose is to find a combination of three parameters in order to verify the three rules together. Consequently, the most common error types are the error types that cover most of the errors reported by a teacher, being as specific as possible in order to give the teacher precise information about students’ difficulties and having few number of subtypes, enabling the teacher to take few corrective actions.

6 The computing algorithm

The aim of this part is to specify an algorithm that computes most common error types. This algorithm ranks each error type according to a ranking function.

To achieve this goal, different ranking functions are evaluated in order to select the most relevant one. As specified in the definition of the most common error types, a ranking function should respect three rules together. For this reason, its evaluation consists to check that each rule is respected.

To define the ranking function, the information content IC (Pedersen, 2010; Zhou et al., 2008) could be used because it assigns a weight to an error type in a taxonomy by combining its depth and its number of subtypes. To be able to compare IC to other ranking functions, we define a weight function that combines these two parameters. This function must respect together the generality level and the number of subtype rules. This weight function is then combined with the frequency parameter.
6.1 The weight function

The purpose of this part is to define a weight function that combines for a given error type, its number of subtypes and its generality level. This ranking function should be evaluated according to the generality level and the number of subtype rules.

As explained above, information content (IC) is evaluated as a weight function in order to verify if it respects the two rules. Information content (IC) is a specificity measure for a concept. IC higher values are associated with more specific concepts, while lower values correspond to more general ones (Pedersen, 2010).

In a taxonomy structure, an IC computation formula for a given concept that takes into account its generality level and its number of subtypes was proposed by (Zhou et al., 2008). In this case, higher IC values correspond to specific concepts that have few subtypes.

This IC function is the following:

\[
IC(e) = k \left( 1 - \frac{\log(hypo(e) + 1)}{\log(Wn)} \right) + (1 - k) \frac{\log(depth(e))}{\log(max(depth))}
\]

where \( e \) is concept within a taxonomy, \( hypo(e) \) is its number of subtypes, \( depth(e) \) is its depth in the taxonomy, \( Wn \) is the number of concepts in the taxonomy and \( max(depth) \) is the depth of the taxonomy.

This formula combines two functions. The first function \( F_1 = \frac{\log(hypo(e) + 1)}{\log(Wn)} \) measures the number of subtypes of \( e \). It assigns more weight to types that have a few number of subtypes. The second function \( F_2 = \frac{\log(depth(e))}{\log(max(depth))} \) measures the generality level of \( e \). It assigns more weight to specific types in the taxonomy. \( k \) is a tuning factor that adjusts the weights of \( F_1 \) and \( F_2 \) in the final weight.

Even if the function \( F_1 \) respects the number of subtype rule, in the case of a taxonomy with a high number of types, this function does not assign a small weight to a type that has a high number of subtypes. This is because the denominator of \( F_1 \) does not have a high value. For the same reason, in a taxonomy with a large depth, the function \( F_2 \) does not assign a small weight to a general type. Consequently, the functions \( F_1 \) and \( F_2 \) should be adapted.

To solve these issues, we re define another weight function by using the IC function as a basis, i.e. we redefine the two functions \( F_1 \) and \( F_2 \).

We need for \( F_1 \) a positive and decreasing function of the number of type subtypes. The function \( \frac{1}{hypo(e)} \) respects the number of subtype rule even in a taxonomy with a high number of types. However, in the case of leaves, its denominator is equal to 0. Thus, \( hypo(e) \) is replaced by \( hypo(e) + 1 \). Finally the function \( F_1 = \frac{1}{hypo(e) + 1} \) is selected.

We need for \( F_2 \) a positive and increasing function of a type generality level. The function \( \frac{depth(e)}{max(depth)} \) respects the generality level rule even if in a taxonomy with large depth.

As a result, the weight function that respects the two rules is the following:

\[
We = k \left( \frac{1}{hypo(e) + 1} \right) + (1 - k) \frac{depth(e)}{max(depth)}
\]

We choose \( k = 0.5 \) to consider with the same weight the depth and the number of subtypes.
6.2 The ranking function

As explained above, ranking an error type requires using its weight function and its frequency. For this purpose, we propose and evaluate functions that combine these two parameters in order to select the most relevant one.

6.2.1 The minimum function

Using the minimum function, the ranking function of a given error type \( e \) is:

\[
\text{Rank}(e) = \min(Freq(e), We)
\]  

As can be seen in Figure 5, the application of the minimum function does not give the result expected which is \textit{MethodProblem}. The problem is that the minimum function considers only the frequency for leaves and only the weight for the most general types. The leaves have the highest value of \( We \) (\( We = 1 \) then \( \text{Rank}(e) = Freq(e) \)) and the most general types have the highest value of frequency (\( \text{Rank}(e) = We \)).

\textbf{Figure 5} Application of the minimum function

Consequently, the minimum function verifies the frequency rule in some cases only, and it verifies the generality level and the number of subtype rules in other cases only.

6.2.2 The maximum function

Using the maximum function, the ranking function of a given error type \( e \) is:

\[
\text{Rank}(e) = \max(Freq(e), We)
\]  

Consequently, the minimum function verifies the frequency rule in some cases only, and it verifies the generality level and the number of subtype rules in other cases only.
Using the maximum function, the most common error types are the most specific types because they have the highest value of the weight \( W_e \) \( (W_e = 1) \). Consequently, the maximum function does not verify the frequency rule for leaves.

### 6.2.3 The average function

Using the average function, the ranking function of a given error type \( e \) is:

\[
\text{Rank}(e) = \frac{\text{Freq}(e) + W_e}{2}
\]  

The example of Figure 6 illustrates the application of the average function. It shows that the average function does not give the result expected which is \text{IrrelevantMethod}.

**Figure 6** Application of the average function

![Diagram](image)

The problem of the average function is that it gives high scores to leaves even if they have low frequencies because these types have high \( W_e \) values \( (W_e = 1) \). Therefore, the average function does not verify the frequency rule in this case.

### 6.2.4 The multiplication function

Using the multiplication function, the ranking function of a given error type \( e \) is:

\[
\text{Rank}(e) = \text{Freq}(e) \times W_e
\]  


The Figure 7 illustrates the application of the multiplication function. It shows that the multiplication function gives the expected result which is *IrrelevantMethod* as the most common error type.

**Figure 7** Application of the multiplication function

The multiplication function is the most suitable function to rank error types because it considers the frequency and the weight \( W \) whereas other functions favour one of these two parameters in some cases.

As a result, the ranking function is the following:

\[
Rank(e) = Freq(e) \times \left( \frac{1}{hypo(e) + 1} \right) + \left( 1 - k \right) \left( \frac{depth(e)}{\max(depth)} \right)
\]  

(7)

### 6.3 Generalisation: providing feedback on multiple task types

Teachers provide formative feedback to students on multiple task types. To generalise the computing algorithm of the most common error types on multiple task types, the initial ranking function on one task type \( T \) is redefined.

\[
Rank(e) = Freq(e) \times \left( \frac{1}{hypo(e) + 1} \right) + \left( 1 - k \right) \left( \frac{depth(e)}{\max(depth)} \right)
\]

(8)
This ranking function multiplies the frequency of the error type $e$ with its weight. The frequency is the ratio between the number of errors of the type $e$ reported on the task type $T$ and the number of all reported errors, and the weight of the error type $e$ is computed according to the weight function.

In the case of multiple task types, the frequency of an error type $e$ cannot be counted only on a given type, but on all ones. In contrast, the weight of the error $e$ is the same because it depends only on its generality level and its number of subtypes within the taxonomy.

Therefore, the frequency function $Freq(e, T)$ must compute the frequency of each ordered pair $(e, T)$ where $e$ is an error type and $T$ is its corresponding task type:

$$Freq(e, T) = \left(\frac{Occ(e, T)}{OccGlob}\right)$$

(9)

$Occ(e, T)$ is the number of occurrences of the error type $e$ on the task type $T$
$OccGlob$ is the number of all reported errors.

This frequency function ensures assigning frequencies to all ordered pairs equitably because a frequency is computed relatively to all errors and not only to ones reported on the same task type $T$. As a result, the final ranking function assigns a score to each ordered pair $(e, T)$. Its formula is the following:

$$Rank(e, T) = Freq(e, T) * \left( k \left( \frac{1}{hypo(e) + 1} \right) + (1 - k) \frac{depth(e)}{\max(depth)} \right)$$

(10)

7 Implementation in MemoNote

The feature that computes the most common error types has been implemented and deployed in MemoNote. This section describes the architecture of MemoNote, the computing steps required to compute the most common error types. It describes also a scenario of using the feature.

7.1 MemoNote architecture

MemoNote has a web architecture with three sides (see Figure 8). The first side is a web client, developed with HTML and JavaScript. The second side is an application server deployed on an Apache Tomcat. The third side is a MySQL Database to store triples using SDB Jena (McBride, 2001).

MemoNote annotation ontology is deployed on the application server. Jena SDB is used for ontology persistent storage through the MySQL database. Therefore, triples are stored in a persistence way, which allows improving performances. In addition, transactions are performed in a safety way ensuring the ACID properties.
7.2 Computing steps

To compute the most common error types, five steps are followed (see Figure 9).

1. The ranking function requests the semantic web reasoner Parsia and Sirin (2004) to compute the inferred ontology.
2. Pellet reasoner requests the ontology stored in the database.
3. Pellet receives the ontology and performs classification and realisation services.
4. Pellet sends the inferred ontology to the ranking function.
5. The ranking function ranks each error type on the inferred ontology.

It uses a SPARQL query to compute the frequency of each error type related to each task type.
7.3 Scenario

To display error types of a given lab work mission, a teacher clicks on the Report button. MemoNote displays a popup that contains a list of lab work missions. She then selects the lab mission on which she wants to display common error types made by students.

The teacher’s request is immediately sent to the server to be processed. After processing the request, MemoNote displays a table of the most common error types for the selected lab mission. The table has two columns: the error type and its corresponding task type (see Figure 10).

8 Evaluation

This section aims at evaluating the algorithm that computes the most common error types. This evaluation consists of assessing with teachers the rules that the algorithm uses to enable teachers to take corrective actions efficiently.

To achieve this purpose, we tried to understand with teachers’ strategies, they use to select the most common error types in order to take corrective actions and improve students’ learning. These strategies were then compared with the algorithm rules.

8.1 Evaluation method and design

In this evaluation, teachers were asked to describe in their own way relevant strategies to select the most common error types. In addition, we needed to understand why these
strategies were useful for teachers to take corrective actions. In this case, evaluation questions are very open-ended. Consequently, a qualitative methodology is the more appropriate method to address these issues (Sofaer, 1999).

The interview is one of the most common data collection methods in qualitative studies. It aims at exploring views and beliefs of individual participants (Gill et al., 2008). Since the purpose of this study is to understand each teacher’s strategies to select the most common error types, we adopt interviews as a data collection method.

The study conducted is based on semi-structured interviews. Each interview took approximately 45 minutes and started by presenting to teachers the aim of the study. We then explained to teachers the error types, the differences between specific and general error types and how errors are reported on the taxonomy.

To be able to compare teachers’ strategies with the algorithm rules, we presented to teachers seven examples of the taxonomy with errors reported. Each example was created by the combination of one or more algorithm rules. For instance, the first example takes into account the frequency rule, the second example takes into account the generality level rule and so on. For each example, teachers were asked to answer two questions:

1. What useful error type do you select in order to take necessary corrective actions?
2. What criteria did you take into account to select this error type?

8.2 Participants

In qualitative studies, the number of participants depends on attaining a saturation point. The saturation point is when there is no new relevant information about study issues after increasing the number of participants (Galvin, 2015).

In our study, the saturation point was reached after interviewing 20 university teachers from three disciplines: computer science, chemistry and biology. These teachers were divided into two categories. First, 12 teachers are experienced and have more than 5 years in supervising students’ lab works. Second, 8 teachers are novice and have less than 1 year in supervising students in laboratories.

8.3 Data analysis

In order to ensure data reliability, detailed notes were taken during interviews. The interviews were audiotape-recorded and transcribed verbatim. In addition, each transcript was read and reviewed many times in order to identify important text passages.

To analyse data, we used code and coding method (Miles and Huberman, 1985). Thus, important text passages were marked with label codes. When this step was completed, labels that represent similar ideas were grouped into themes.

8.4 Findings

After analysing and coding data, five themes were identified. These themes represent strategies that teachers use to select error types on which they should take necessary corrective actions. In the following, we discuss and illustrate these themes with interview excerpts.
Theme 1: Number of students.
The number of students that made an error type was discussed by all teachers. The teachers preferred taking correction actions on a frequent error type. Indeed, according to these teachers, a frequent error type enables them to overcome a frequent difficulty encountered by students. In addition, taking a corrective action on this error type allows teachers to improve learning of a significant number of students.

“I prefer taking a corrective action on an error type that has a high number of reported errors because this error type enables me to overcome a frequent problem”.

Theme 2: Number of difficulties.
The number of difficulties to overcome by taking one corrective action is a parameter that 16 teachers took into account (10 experienced teachers and 6 novice teachers). Indeed, these teachers preferred taking a corrective action on a general error type when errors were reported equally on its subtypes because this corrective action helps them overcome several specific students’ difficulties (subtypes of the general error type).

“I prefer taking a corrective action on a general error type if errors are reported equally on its subtypes in order to deal with several students’ problems”.

Theme 3: Corrective action precision.
The precision of corrective actions was discussed by 15 teachers (10 experienced teachers and 5 novice teachers). These teachers preferred taking a corrective action on specific error types because they enable them to deal with precise students’ difficulties.

“I prefer taking corrective action on a specific error type because it enables me to take a precise corrective action”.

Theme 4: Time required.
The time required to take corrective actions is a factor that 11 teachers (8 experienced teachers and 3 novice teachers) considered. Thus, these teachers preferred taking a corrective action on an error type that has a small number of subtypes. This strategy enables teachers to overcome few students’ problems and therefore take corrective actions more quickly.

“I prefer taking a corrective action on a type that has few subtypes in order to deal with students’ difficulties more quickly”.

Theme 5: Error type importance.
Three experienced teachers and one novice teacher mentioned that an error type for a lab work mission can be more important than others. In this case, these teachers preferred taking corrective actions on these important error types.

“I consider the importance of an error type before its number of reported errors”.

As is seen in Figures 11 and 12, number of students, number of difficulties and corrective action precision are the most frequent themes discussed by teachers, however, error type importance is the less frequent theme cited by teachers. In addition, Time required is also cited frequently by experienced teachers in contrast to novice teachers.

To compare the evaluation findings with the algorithm that computes the most common error types, we identify relationships between the themes and the algorithm rules. As a result, four of the themes explained above represent one or more algorithm rules. First, taking a corrective action on an error made by a significant number of students (theme 1) consists to select an error type that was reported frequently on a task type (rule 1).
Second, taking a precise corrective action (theme 3) corresponds to the generality level rule (rule 2) that consists of selecting specific error types. Third, taking a corrective action quickly (theme 4) means selecting an error type that has fewer subtypes (rule 3). Finally, the theme number of students’ difficulties to overcome is the combination of the frequency rule and the generality level rule.

**Figure 11** Experienced teachers (see online version for colours)

**Figure 12** Novice teachers (see online version for colours)
To conclude, the algorithm developed uses and combines the strategies that experienced teachers discussed frequently to identify the most common error types. Thus, the frequency rule, the generality level rule, their combination and the number of subtypes rule are the most frequent strategies cited by teachers. However, novice teachers did not take frequently the number of subtype rule to select the most common error types. In addition, we identified a new strategy that the algorithm does not take into account, the importance of error types for lab works even if it was not cited frequently by teachers.

9 Conclusion

This paper studied how to compute the most common error types reported by teachers on students’ lab reports in LabBook using MemoNote annotation tool. Computing these common error types is not obvious because of their taxonomic structure. Thus, occurrences of an error type belong to its supertypes and considering only the type occurrences implies selecting the most general type as the most common one.

We defined formally the most common error types using three rules: the frequency, generality level and number of subtype rules. For a given error type, these rules use three parameters: its frequency, generality level and the number of subtypes.

We developed an algorithm to compute common error types that respect all at once these three rules. It is based on a ranking function that combines the three parameters to assign a score to an error type. From the various possible ranking functions evaluated against these rules, only one corresponds. It multiplies the error type frequency with an adapted version of information content function (combining type depth and the number of type subtypes).

Thanks to formal semantics of error types, the most common error type feature has been implemented using semantic web technologies such as reasoning services provided by pellet reasoner, SPARQL and Jena API.

A qualitative study conducted with teachers showed that most experienced teachers used and combined the algorithm rules to select the most common error types on which they should take corrective actions, whereas novice teachers took into account more the frequency and the generality level rules. Both categories of teachers preferred taking corrective actions on error types made by a lot of students, on error types that help teachers overcome several students’ difficulties and on specific error types. In addition to these strategies, experienced teachers took into account the time required to take a corrective action, which corresponds to the number of subtype rule.

For some lab work mission, error types and task types are more important than others. For this purpose, we will investigate how to measure importance of error types and task types in order to incorporate them in the algorithm proposed.

References


