Lexical Error Diagnosis for Second Language Learners of Arabic

Marwa Magdy[1] Khaled Shaalan[2] Aly Fahmy[1]

[1]Faculty of Computers & Information, Cairo Univ., 5 Ahmed Zewel St., Giza 12613 Egypt
[2]The British Univ. in Dubai, PO Box 502216 Dubai, UAE

ABSTRACT

This paper addresses the development of an automated lexical error diagnosis system, which helps Arabic second language learners to learn well-formed *weak verbs*. The learners are encouraged to produce input freely in various situations and contexts and guided to recognize by themselves the erroneous or inappropriate functions of their misused expressions. In this system, we successfully used *constraint relaxation* and *edit-distance* techniques to provide error-specific diagnosis and feedback to second language learners of Arabic. We demonstrated the capabilities of these techniques to diagnose errors related to the Arabic weak verb which is formed using complex morphological rules. Furthermore, the developed system allows for individualization of the learning process by providing feedback that conforms to the learner's expertise. Inexperienced learners might require detailed instruction while experienced learners benefit from higher level reminders and explanations.

Keywords: Lexical Errors Diagnosis, Second Language Learners of Arabic

1. Introduction

Language is a way of communicating ideas and feelings among people by the use of conventional symbols. People need to learn second languages to be able to communicate with each other. They face difficulties in word formation, word recognition, sentence construction and disambiguation. Their errors can violate linguistic expectations at all levels: the lexical, syntactic, semantic, and pragmatic/discourse level [6] [13]. Traditional error analysis studies do not address lexical errors well enough although there are large numbers of word related errors that are committed by nonnative language users [17]. Lexical errors can be categorized to:

- *Errors in word formation* which are related to the correct application of morphological rules,
- *Errors in semantic or word choice*, which are to some extent related to ambiguity in word senses and phonetics, and
- *Errors at the interface of lexical and grammar*, which are related to the morpho-syntactic features of words.

Existing studies of lexical error analysis fall into two main closely related systems: *Spelling Checkers* and *Intelligent Language Tutoring Systems (ILTS)*. The purpose of most spelling checkers is neither teaching nor learning languages as they are only designed for detecting spelling errors and suggesting possibly correct spelling [12]. SLLs not only ask for correcting their errors but also they want to improve their language skills in order not to do same errors over and over again. Moreover, most of checkers are inappropriate for nonnative speakers because they are mainly designed for native speakers and as such they are not suitable for detecting and correcting competence errors made by nonnative speakers. For example, recent Microsoft office's Arabic spell checker© detects the word يقالن* as an error but it doesn't suggest the correct form يقلن. On the contrary, intelligent language tutoring systems (ILTS) try to overcome these problems and be more useful to SLLs by making true diagnosis of errors. Consequently, they point the learner to the right direction on how to correct their errors rather than providing the correct version directly [6].

Many researchers developed an Intelligent Language Tutoring System (ILTS) for lexical error analysis in a variety of languages [12] [15] [7] [8]. They incorporate morphological knowledge and non-native intuitions into their algorithms in order to be able handle competence errors made by nonnative writers. However, they still keep behave like a spell checkers by only offering a short list of possible alternative words to replace the unknown word. To the best of our knowledge, there is no research which addresses the problems of diagnosing lexical errors made by Second Language Learners (SLLs) of Arabic. Moreover, our solution is based on natural language processing (NLP) techniques which suit the highly inflected nature of Arabic words.

This paper describes the development of an automated Arabic lexical error diagnosis system which is an integral part in the development of an Intelligent Language Tutoring System for second language learners of Arabic. Since the task of automatic diagnosis of free Arabic text is hard, the *objective test* method is used. The proposed system uses *constraint relaxation* and *edit-distance* techniques to diagnose lexical errors made by learners. In order to demonstrate the benefits of these techniques, we limited our scope to the problem of diagnosing errors related to ill-formed weak verbs made by Arabic SLLs.

Constraint relaxation technique is based on the following principle: partial structures can combine only if some constraints or conditions are met; when the constraints are relaxed, attachment is allowed even if the constraint is not satisfied. The system uses this technique to split erroneous word into three segments *prefix + stem + suffix*. In Arabic, some conditions should be met to form well-formed word such as usage of certain pronouns with respect to Arabic verb tense. The role of the *edit-distance* technique comes into play just after performing constraint relaxation technique. It is based on the number of edits (i.e. insertions, deletions, and substitutions) it takes to convert the wrong learner answer into the well-formed form. The results of analysis will be used to construct an informative feedback that indicates the location of errors and their types. Unlike most systems, the proposed system will improve language learners' skills by helping them not to do the same mistakes over and over again.

The rest of this paper is structured as follows. Section 2 presents a brief overview of Arabic Verbal System. Section 3 introduces a classification of common Arabic lexical errors. Section 4 describes our proposed lexical error diagnosis system. Section 5 focuses on details related to our error diagnosis techniques and show how we and implemented them within our system. Finally, section 6 presents some concluding remarks.

2. Arabic Verbal System

One of the most puzzling problems in the study of Arabic is its verbal system which is very rich in forms and meaning [16]. Arabic verbs can be generated from either trilateral or quadrilateral roots then they can be conjugated according to one of the traditionally recognized patterns (or conjugations). There are 15 trilateral forms and 4 quadrilateral ones [2] [18].

Arabic verbs appear in three tenses (perfect, imperfect and imperative), two voices (active and passive) and four moods (indicative, subjunctive, jussive and energetic).The derivation of verbs in different tenses, voices and mood is achieved using well behaved morphological rules. The irregularities are due to the phonological constraints of certain root consonants [16] [5]. Arabic weak letters can be deleted or replaced by other letters because of Arabic linguistic theory. For example, the conversion of the letter (و) by letter (ت) then combining it to another letter (ت) can be explained by taking the prefect tense of the trilateral root و-ص-ل according to the pattern افتعل. Using regular rules, it would generate اوتصل but as it is an assimilated (weak first) verb it should be generated according to irregular rules and thus it should be generated as ائصل.

Arabic verbs can be categorized into *weak* and *strong* verbs. Weak verbs have a weak letter (‘و’ or ‘ي’) as one or more of their radicals; strong verbs do not have any weak radicals.

Strong verbs can be categorized into three classes: *regular, hamzated* and *doubled.* Regular and doubled verbs can be generated using well behaved morphological regular rules while hamzated verbs are irregular ones [18]. The hamza is changed to other different realizations due to the influence of the vowels before and after the hamza [5]. The different realizations of the hamza are (أ, إ, ؤ, ء, ئ).

Weak verbs, the major concern of this work, can be categorized into three classes: *assimilated, hollow* and *defective*. *Assimilated verbs* are those with a weak initial radical as وعد. They fall into two classes. *Hollow verbs* are those with a weak middle radical as صام. They fall into four classes [3] [16]. *Defective verbs* are those with a weak final radical as نجا. They fall into five classes [18] [19].

3. A Classification of Arabic Lexical Errors

Some linguistic studies examined errors made by SLLs of Arabic [14] [20]. Errors can be classified according to level of analysis to *orthographic*, *phonological*, *morphological*, *syntactic*, *semantic* and *pragmatic errors*. Phonological and morphological errors are closely related to our research of Arabic Lexical errors. Table1 shows error categories which can be handled by the proposed system.

Table1. Error categories treated by lexical diagnosis system

Error Class	Error Type
Word formation errors	*Phonological errors:* Incorrect usage of letters with the closely related pronunciation.
	Phonological errors: Making short vowel long one.
	Phonological errors: Making long vowel short one.
	Morphological errors: Incorrect usage of pronouns with respect to verb tense.
	Morphological errors: Incorrect conjugation of prefect verb.
	Morphological errors: Incorrect conjugation of imprefect verb.
	Morphological errors: Incorrect conjugation of imperative verb.
Semantic errors	*Morphological errors:* Incorrect usage of root pattern.
Error at the interface of lexical and grammar	*Morphological errors:* Switching a conjugated verb with its infinitive.
	Morphological errors: Switching an infinitive with its conjugated verb.
	Morphological errors: Incorrect usage of verb tense.
	Morphological errors: Person disagreement of a connected pronoun with the subject.
	Morphological errors: Gender disagreement of a connected pronoun with the subject.
	Morphological errors: Number disagreement of a connected pronoun with the subject.
	Morphological errors: Incorrect case ending of subjunctive or jussive imprefect verb.

4. An Overview of the Proposed Arabic Lexical Error Diagnosis System

The primary objective of the proposed Arabic Lexical Error Diagnosis System (ALEDS) is to provide error specific diagnosis and feedback in a way that allows for individualization of the learning process. For example, if a learner writes a wrong Arabic weak verb, the cause of this error might be due to verb conjugation, pattern selection, tense selection and/or subject-verb agreement. In such a case, the system should distinguish between these error types. Then, it should provide the feedback that conforms to the learner's expertise. There are three learner levels considered in ALEDS: beginner, intermediate and advanced. The beginner learner will receive the most specific (detailed) feedback. Providing feedback messages according to learner level follows the pedagogical principle of guided discovery learning [10] [11].

For example, given the following learner input

a. *زورت مصر عدة مرات.[1]
b. زرت مصر عدة مرات.

The system detects that the learner made a phonological error *"vowel error"* with the word زورت. S/he made short vowel (الضمة) a long one (و); the concept of *Arabic vowel letters* has been missed. For the advanced learner, the system issues a hint indicating that *"a word formation mistake occurred due to phonology*". For the intermediate learner, the type of the error is provided (*vowel letters*). For the beginner learner, the exact source of the error is also provided (*making short vowels long*). Note, however, that the beginner learner is still required to decide the correct vowel letter in the word.

Figure 1 shows the architecture of ALEDS. It consists of the following components: answer analyzer, feedback system, learner (student) model, item (question) banking, test generator, and graphical user interface.

The *answer analyzer* is an NLP component that analyzes the learner's answer and detects possible source of errors. It includes *morphological analyzer, morphological generator* and *error analyzer*. *Feedback system* is responsible for issuing appropriate error specific feedback message suited to the learner's expertise. It includes *feedback message generator* and *error database*. The proposed system keeps a record of the learner's performance history. This information is held in the *learner model* component. It is used to determine level of specificity of feedback message displayed to learner. The *item* (question) banking and *test generator* component are responsible for generating different questions (tests) to learner. The proposed system is developed using SICStus Prolog.

[1] The asterisk indicates an incorrect word or sentence.

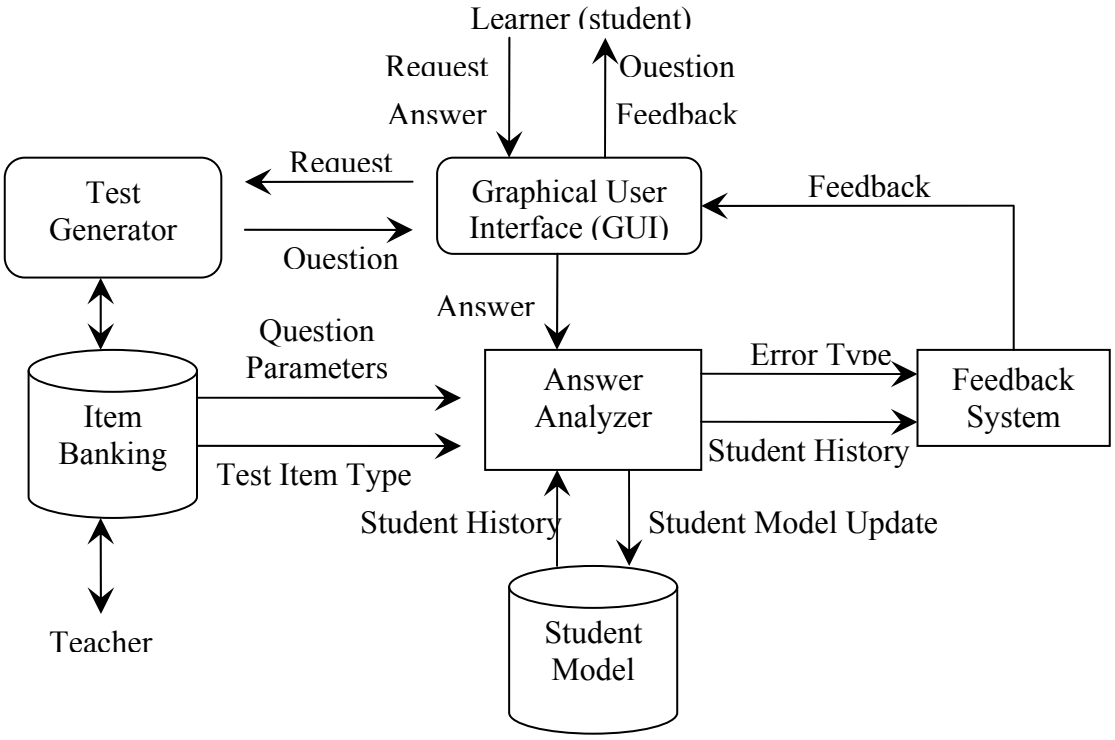

Figure1 The overall architecture of the proposed the Arabic Lexical Error Diagnosis System (ALEDS)

4.1 Learner Model

The *learner model* is a representation of the current state of a specific learner, which includes the student's knowledge and skills on the domain (*domain specific information*) and the student's subject-independent characteristics relevant to the learning processes (*domain independent information).* There are different techniques to represent domain specific information: *scalar, overlay, error (perturbation or differential)* and *genetic models* [9] [1]. The most commonly used technique is the *overlay model* where, the student knowledge is considered as a subset of the domain knowledge and is represented as a set of concept/value pairs. For each domain model concept the student model stores some values that estimate the student knowledge level of this concept [1]. This model cannot represent errors student made.

The ALEDS uses a student model which contains only information about current skill level of the student (*domain specific knowledge*). We used perturbation error model to represent this knowledge. In this model, we assume one or more misconceptions[2] exist for each concept in an introductory course for Arabic weak verbs. By this way, the student model knowledge is represented by a union of a subset of the domain knowledge (Arabic concepts of weak verbs) and another subset of the misconception set. In our implementation, we used the prolog predicate *knowledge/*3 to express student model knowledge. For example, the following predicate defines that the *vowel letters* concept has two associated misconceptions: *make short vowel long one* and the vice versa.

```
knowledge(vowel_letters, short_long, con2_bug1).
knowledge(vowel_letters, long_short, con2_bug2).
```

Notice that arguments represent concept name, bug name, and buggy knowledge name, respectively. The third argument is an index that refers to a combination of the first and second arguments.

The proposed student model tracks 40 Arabic concepts and 116 misconceptions, for each concept and its associated bug; the student model keeps a frequency of this error type, to each student, which falls in the range of one of the three learner levels. The frequency of the bugs is expressed by a pair of numbers

[2] We used terms misconception, bug and error type interchangeably

[S, T]; where the variable *S* represents how many times the student has made this error and the variable *T* represents the total number of times in which the student has met this concept. As the frequency of the bug becomes higher, the student requires detailed instruction. The frequency of all bugs in the student model is initialized with [0, 0]. We used the prolog predicate `student/3` to express current skill of each student. For example, the following predicate records that learner with id 2007123 has made the error "*made short vowel long one*" three times while s/he has met the *vowel letters* concept four times.

```
student( 2007123, con2_bug1, [3, 4]).
```

Notice that the arguments represent a learner id, buggy knowledge name, and frequency of a bug for this buggy knowledge.

The ALEDS uses student model to determine the specificity of the feedback message displayed to the learner. A feedback message is selected according to the current frequency of bug at a particular buggy knowledge. The learner model assumes three types of learners: beginner, intermediate, and expert. Each learner level is represented by a range of values:

Beginner: frequency of bugs > 50%
Intermediate: 20% < frequency of bugs ≤ 50%
Advanced: 0% ≤ frequency of bugs ≤ 20%

Notice that the learner skill level can vary for a specific learner as it is associated with each specific bug. For example, the learner may be beginner with regard to *'make short vowel long one'* error but an advanced with '*make long vowel short one*' error.

4.2 Item Banking

The item banking is a database of test items. This component is used to generate different types of test items each time the learner is allowed to take a test. The test generator selects test items in random order. There are two main types of test items for interaction with the learner in ALEDS: supply-type (short-answer or fill-in-the-blank) or selection-type (multiple-choice or which word is different). From the linguistic point of view, the type of exercises used in our system[3] can be classified as follows:

- *Dictation.* This exercise type focuses on listening skills. Learners should first listen to a sentence then type it. The system accepts the learner answer, if it is correct by a positive feedback message. Otherwise, an error specific feedback message suited to learner's expertise is provided. The learners have two additional options which are consistent for all exercise types. The learner can either correct the error and resubmit the modified answer or peek at the correct answer.
- *Word order.* This exercise displays a number of Arabic roots, the learner task is to rearrange these roots and conjugate them to form a grammatical Arabic sentence.
- *Build a sentence.* In this exercise type, learners are provided with a set of Arabic roots. Their task is to create a grammatical Arabic sentence using all the provided words.
- *Transform the sentence category.* In this exercise type, learners are provided with some sentences. Their task is to transform them to another category. Transform a sentence from nominal to verbal sentence is an example.
- *Word formation practice.* This exercise displays some Arabic roots; the learner task is to conjugate this root as required in the question.
- *Fill in blank.* The learner's task here is to complete sentences by filling in any blanks that appear in the question. The learner is provided with the root of correct answer and is asked to supply the correct verb conjugation.
- *Which word is different?* This exercise displays a list of words of the same category which has only one word which does not belong to this category. The learner task is to identify this word. The divergent word differs morphologically (i.e. root, pattern, verb tense, voice, type of weak verb…etc) from the other words.
- *Multiple choices.* The learner's task here is to choose the correct word from multiple choices. The learner is provided with different forms of the same word.

[3] Most of exercises types used in ALEDS is commonly used is Arabic SLL textbooks.

The structure of the item banking consists of six database relations: question, question type, question content, answer, concepts and parameters relation. Each question is accompanied by an associated list of concepts for each word in the correct answer. There is a list of four concepts associated with writing any Arabic word: *consonant letters*, *vowel letters, usage of suitable lexical category* and *suitable pattern*. As such there is no need to associate this list with any question. Furthermore, each question has some parameters that help the ALEDS to diagnosis errors. These parameters depend on whether the test item type is selection or supply. The parameter list is introduced in details in the next section.

4.3 Answer Analyzer

This component is responsible for analyzing learner's answer and detecting possible error types. It is also responsible for updating the student model by accumulating the frequency of bugs of all associated list of concepts. To perform this functionality, answer analyzer module includes *morphological analyzer*, *morphological generator* and *error analyzer.* It receives both the question's parameters and the test item type from the item banking component then depending on the produced test item type this component will trigger the appropriate procedure for analyzing the learner's answer. Selective test item needs neither morphological analyzer nor generator. However, the supply test item requires that the answer analyzer generates all possible analyses for the learner's answer. Then the error analyzer selects the most appropriate one according to a number of factors: priorities of error categories, instruction learner level and difficulty of Arabic concepts. In general, the answer analyzer performs the following steps:

1. Match the learner's answer to the pre-stored correct answer.
    a. If a complete match is achieved produce a correct feedback and update the learner model by incrementing the variable `T` for all buggy knowledge associated with the corresponding list of concepts associated with the given question. Then exit.
    b. Otherwise, if a partial match is achieved delete the matched words from both answers and update the learner model for all correct words in the learner answer.
2. Apply the morphological analyzer[4] on each word in the remaining learner's answer and send all possible analyses that have the same root as the stored correct answer to the error analyzer module.
3. Apply the error analyzer to get the analysis of each word in the learner answer. It uses constraint relaxation and edit distance techniques to extracts all possible feature structures that describe the main features of the words in the learner answer.
4. Apply the morphological generator component on the corresponding stored roots of the correct answer by applying the extracted feature structures of the learner's answer.
5. Apply the error analyzer again using the complete analyzed and generated information. It uses edit distance technique to detect all possible error types and update the student model accordingly. More illustrations regarding the techniques used during the answer analysis are presented in the next section.

To explain the working of the answer analyzer as a whole, we shall consider the following example. Assume that the learner has made two errors with the word تزرين* in his/her answer: 1) incorrectly used imperfect active verb in indicative mood instead of prefect active verb, and 2) incorrectly made a long vowel (و) a short one (الضمة).

a. أريد أن أشكرك لأنك ***تزرين*** منزلي في العيد.
b. أريد أن أشكرك لأنك زرت منزلي في العيد.

Notice that the system stores for each question the following: 1) the best correct answer, 2) a list of feature structures for Arabic words in the correct answer. Each feature structure[5] includes the following features: *correct word, correct word with diacritics, root*, *pattern*, *type of verb*, *prefix string*, *suffix string*, *lexical category*, *tense*, *voice*, *mood*, *subject* and *object gender*, *number* and *person* and, 3) a list of associated concepts for each word in the correct answer.

The system proceeds as follows in order to detect these two errors:

---

[4] The ALEDS uses one of the commercially widely available morphological analyzers.

[5] The feature structure concerned here is that related to Arabic verbs. For the purpose of handing Arabic verbs, the feature structure of a noun is limited to the following features: root, pattern, prefix string, suffix string and lexical category.

- Match the learner's answer to the pre-stored correct answer and filter out the matched words. This results in having the correct answer contained only the word زرت and the learner's answer contained only the word تزرين.
- The learner model is updated for each correct word in the learner's answer. It does so by recording that the associated list of concepts in the given question for these words is met by the learner and s/he doesn't make any errors related to these concepts. This is done by incrementing their associated variable `T`. For example, the word أريد />u-riyd/ has seven associated concepts. One of them is *vowel letters* which has two possible associated errors: *make short vowel long one* and *make long vowel short one.* Assume the frequency of bugs for these errors before their update are `[0, 2] [1, 2]`. They are updated by incrementing the variable `T` for each of them such that the frequency of bugs became `[0, 3], [1, 3]`.
- Applying the morphological analyzer on the learner's answer تزرين, it did not result in any analysis that matches the root ز-و-ر that was stored with the correct answer. Thus, an empty list is sent to error analyzer module indicating no analysis could be reached.
- The error analyzer applies constraint relaxation and edit distance techniques on the learner answer, which produced three possible feature structures of the word answer تزرين:
    - `[lexical_category: verb, tense: imprefect, voice: active, mood: indicative, subj_person: 2, subj_gender: f, subj_num:sg]`
    - `[lexical category: verb, tense: imperative, subj_person: 2, subj_gender: f, subj_num: sg, obj_person: 1, obj_num: pl, obj_gender: neutral]`
    - `[lexical category: verb, tense: imperative, subj_person: 2, subj_gender: f, subj_num: sg, obj_person: 1, obj_num: sg, obj_gender: neutral]`

- Applying the morphological generator on the root ز-و-ر for all the five feature structures we get their corresponding inflected stems.
- Finally, the error analyzer detects that there are two errors made by the learner in the word زرت. It also updates the learner model for this erroneous word. It does so by recording that the concepts associated with this word are met by the learner but s/he has made errors. This is reflected by incrementing variable `T` for all buggy knowledge associated with the list of concepts and variable `S` for only the buggy knowledge associated with the error the learner has made. For example, the word زرت has seven associated concepts. The learner has made an error in two of them: *vowel letters* and *usage of perfect active verb.* The former concept has two possible associated errors while the latter has four possible associated errors. The frequency of bugs before and after the learner answer the question is shown in Table 2.

Table2. Sample of Learner Model Knowledge

Concept	Error	Buggy Knowledge	Frequency of bugs	
			Before	After
Vowel letters	*Make short vowel long one*	Con2_bug1	`[0,3]`	`[0,4]`
	Make long vowel short one	Con2_bug2	`[1,3]`	`[2,4]`
Use prefect active verb	*switching perfect active verb with prefect passive*	Con24_bug1	`[1,4]`	`[1,5]`
	switching perfect active verb with imprefect active verb in indicative mood	Con24_bug2	`[3,4]`	`[4,5]`
	switching perfect active verb with imprefect passive verb in indicative mood	Con24_bug3	`[2,4]`	`[2,5]`
	switching perfect active verb with imperative verb	Con24_bug4	`[4,4]`	`[4,5]`

Notice that after the learner has written the word incorrectly, the variable `T` is incremented for the relevant concepts. The variable `S` is incremented only for the specific buggy knowledge of these concepts, i.e. `con2_bug2` and `con24_bug2`.

4.4 Feedback System

The feedback system is responsible for generating feedback messages that conform to learner's expertise. It includes *error database* and *feedback message generator*. The error database includes a specification of all different errors categories handled by the system. In our implementation, there are two different relations defined in the error database, one that defines different error types, `error_type/5`, while the other one defines the source of the error `error_source/4`. The following shows an example of a *verb pattern* error:

```
error_type( 1, 'Word Choice', 'اختيار الكلمة', 'Verb Pattern',
'اختيار وزن الفعل').
error_source( 1, 'incorrect use of root pattern', ' استخدام فعل
من أصل واحد', [ X, Y]) :- X \== Y.
```

Note that arguments of `error_type/5` are: error ID, error class, Arabic translation of this error class, error type, and Arabic translation of this error type. Arguments of `error_source/4` are: error ID, source of error, Arabic translation of this source of error, and error parameters- they are in this case variable to be unified with any two different Arabic patterns.

The feedback message generator module receives a number[6] (1 or 2 or 3) that defines learner's level according to this error- beginner or intermediate or advanced. Also, it receives the error type along with its parameters. The feedback message is generated according to learner's level.

5. Error Diagnosis

This section presents the automatic error analyzer system currently implemented in ALEDS. The *objective test* method is used such that learners are provided with some exercises. Their response is analyzed by the system and the appropriate feedback, which conforms to learner's expertise, is provided. Error diagnosis depends on test item type (selective or supply type) and this will be explained in the following subsections.

5.1 Error Diagnosis of Selective Test Item

Selective test type in ALEDS can be categorized into *multiple choices*, *which word is different.* In *multiple choices* question, the associated parameter is a set of feature structures (FSs) that describe the main features of every word, either correct or erroneous, in the answer. For example the FS of the word أقول is given below:

```
[word: 'أقول', root: 'قول', lexical category: verb, pattern:
'فعل', tense: imprefect, voice: active, mood: indicative, subj_
person: 1, subj_num: sg, subj_gender: neutral]
```

Depending on the erroneous selection that the learner has made, a feature-value match is performed between correct and erroneous FSs to identify the corresponding error type[7]. The feedback system receives this error type, its parameters and level of learner related to this error in order to produce an error specific feedback. For example, consider the following question which asks the student to select the correct answer from the list of words in brackets.

اختر الاجابة الصحيحة مما بين القوسين:
..... الحق دائما. (أقول - قال - أقاول)

The correct answer is أقول; If the learner's answer is قال which is a *prefect* verb of the root ق - و - ل and pattern فعل, the system will detect a *verb tense* error with parameters: imprefect active verb in indicative mood and prefect active verb. It will send both the error type and its parameters to the feedback system.

In *which word is different* question: the parameter list associated with this question is the aspect of divergence (i.e. root, pattern, lexical category, tense, voice, type of weak verb…etc) and the value of the divergent feature for each word in the question. For example, consider the following question which asks the student to select the word that differs from the words that followings.

---

[6] This number is associated to frequency of bugs for each error type. If frequency of bugs > 50%, set Number to 1; if frequency of bugs is between 20% and 50%, set Number to 2; set to 3 otherwise

[7] Lexical errors arose in this question are either semantic or at the boarder between lexical and grammar. Word formation errors did not occur as the learner interacts by selecting the answer from a list of answers.

اختر الكلمة المختلفة:

- نجا.
- سعى.
- رمى.
- صام.

The divergent word is صام; all words are defective prefect verbs but the divergent word is hollow prefect verb. In this case, the aspect of divergence, the type of weak verb, is stored with the question. Moreover, we store the type of weak verb for each word in the question.

Error diagnosis in this question is so simple. Depending on the erroneous selection that the learner has made, a corresponding error type "divergence aspect" and its parameters "feature value (either defective or hollow in above example)" are to be sent to the feedback system. Examples of feedback messages that correspond to the three learner levels -beginner, intermediate and advanced- are given in (a) - (c), respectively:

a) Wrong selection: "The divergent word is on the type of weak verb. This word is defective verb while the correct one is hollow".
b) Wrong selection: "the divergent word is on the type of weak verb".
c) Wrong selection.

5.2 Error Diagnosis of Supply Test Item

Error diagnosis of supply test item -where the learner is asked to write a few words- is more sophisticated since in this case the student is allowed to be more creative than in selective type. The system incorporates morphological knowledge and non-native intuitions into its algorithm in order to be able handle competence errors made by nonnative writers. In addition to lexical error checking, the system contains additional error-checking modules -*missing word*, *extra word* and *word order* check- which get evoked when processing a student answer. Lexical error check is the last error-checking module preceded by other checking modules. Figure 2 presents the complete steps of error diagnosis modules. The system is organized in such a way that if any error checking module detects an error, further processing is blocked. As a result, only one error at a time will be displayed to the learner.

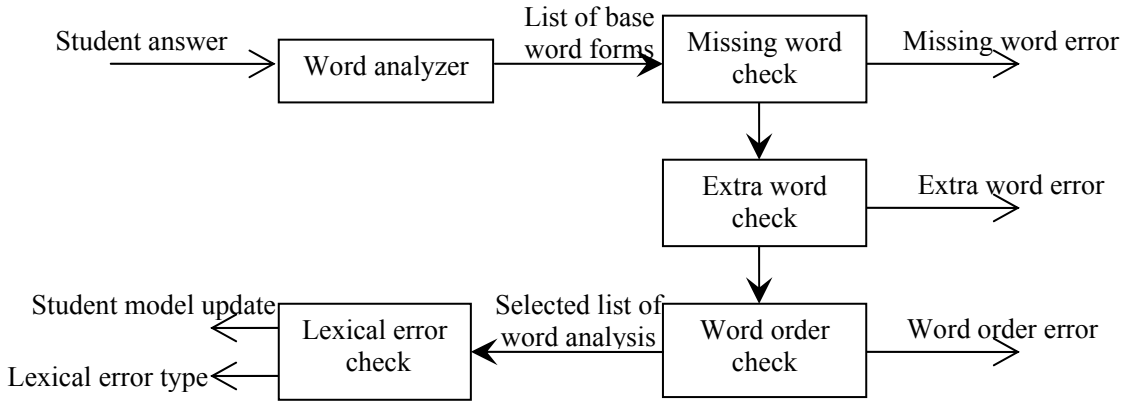

Figure2 The error diagnosis system of supply test item

The parameter list associated with this test item is a list of feature structures for all Arabic words in the correct answer. Each feature structure includes the following features: *correct word, correct word with diacritics, root*, *pattern*, *type of verb*, *prefix string*, *suffix string*, *lexical category*, *tense*, *voice*, *mood*, *subject* and *object gender*, *number* and *person*.

Word analyzer

This module analyzes all words in learner's answer to generate all possible base words that corresponds to each word in learner's answer. The base word form is a normalized form of the root of the word. The system uses this form to ensure that the student doesn't miss or add any words to correct answer. It uses *constraint relaxation* and *edit-distance* techniques to split erroneous word into three segments *prefix+ stem + suffix*. The former technique is based on the following principle: partial structures can combine only if some constraints or conditions are met; when the constraints are relaxed, attachment is allowed even if the constraint is not satisfied. The relaxed constraint must be marked on the structure so that the type and position of the detected error can be indicated later on. In Arabic, some constraints should be met to form well-formed word such as *usage of certain connected pronouns with respect to Arabic verb tense* and *usage of certain affixes* as propositions, conjunction and pronouns. The ALEDS relaxes these two constraints to allow for error diagnosis.

This module makes the following steps to get base word forms:

1. Extract the entire possible suffix list.
2. Filter suffix list according to some heuristic filtering rules.
3. Extract the entire possible prefix list.
4. Filter prefix list according to some heuristic filtering rules.
5. Extract the all possible stem strings.
6. Unify the similar stem strings.
7. Dissolve the stems into base word form.

The system uses regular expression to extract affix list. The list of prefix and suffix is expressed as deterministic finite state automata to implement regular expressions. The suffix list is expressed in reverse order. Figure 3 shows a representation of the suffixes ون, وا, ا.

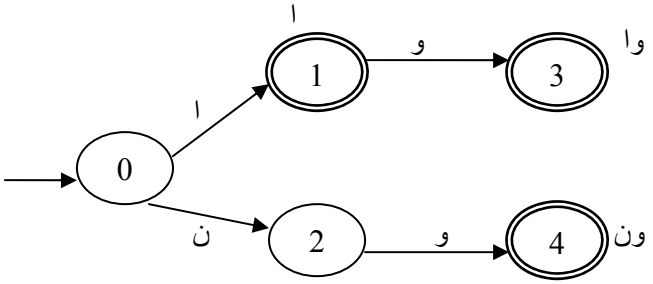

Figure3 A finite state automata for three suffixes

We use prolog terms to present suffixes, prefixes automata:

```
initial(0).
state(0,1,'ا').
state(0,2,'ن').
state(1,3,'و').
state(2,4,'و').
final(1,'ا', 'PVSuff-A', [lexical_category: verb, tense: prefect,
voice: active, subj_person: 3, subj_gender: m, subj_num: dl]).
```

To extract the suffix list, the system matches the input word against the list. The match begins from the end of word and work backwards. The system relaxes the *usage of certain affixes* constraint by using *three-way-match* [4] to compare two words. This method assumes that when a character at location *n* of first word does not match the character at location *m* of second word, we have an error and two other comparisons are made (character at location n with character at location m+1 and vice versa). For example, if the learner writes the word 'قالتو', this module will try to match this word with suffix 'ت' from the end of word. The match fails and it tries with the previous character 'ت' which succeeds. This considers the letter 'ت' a suffix and the letter 'و' an extra letter at the end of the word. It also applies the same method with other suffixes which succeeds with the suffix 'وا' but considers the last suffix letter as missing 'ا'.

Practically, however, the use of constraint relaxation leads to overgeneration of analysis. In order to resolve this problem and to identify the analysis which best reflects the learner's intention, a set of heuristic rules to eliminate highly implausible analysis is applied. For example, we have included

constraints to cope with extra and missing affixes characters attached to weak verbs since SLLs of Arabic find it difficult to differentiate vowel signs from letters.

The suffix list could be minimized by a number of factors: learner answer and error categories handled in the system. For example, if the learner writes the verb 'قالتو', the system will extract the following suffixes: 'ت' which is either (3rdP fem sg) or (2ndP fem sg) or (2ndP masc sg) or (1stPerson sg), 'ات' which is (noun fem pl), 'وا' which is either (prefect or imprefect or imperative suffix), 'و' which is (noun masc pl) and empty suffix. The system deletes first, second, third and fifth solution from suffix list since the end case of these suffixes[8] does not match the added character which is 'و'[9]

Extracting the prefix list is the same as extracting suffix list except that the order of the match process begins from the first letter and goes forward.

To extract the possible stem strings, the system tries to combine every suffix with a prefix that agrees with it from the list of prefixes. It first tries to match the two feature structures, if the match fails the solutions is discarded. Then, the system tries to apply a relax process, e.g. the *usage of certain pronouns with respect to verb tense* constraint could be accepted by the match process even if tense feature is not compatible. After accepting the input word, it tries to remove the prefix and suffix string away from the word to get its stem. The stem should be consisting of at least one character long. Therefore, if the resultant stem is zero character long the solution is discarded.

After applying all previous steps on the word قالتو, the solution list contains the following word analysis in the form *prefix structure* contains *prefix string*, *prefix ID* and *feature structure* describes this prefix, *stem string* represented as prolog list, *suffix structure* contains the same elements as prefix structure, *feature structure* describes this word – it is a combination of both feature structures of affixes and *error representation* in affixes.

```
1-('',Pref-0,[lexical_category: neutral]),['ق', 'ا', 'ل', 'ت',
'و'],('',Suff-0,[lexical_category: neutral]),[lexical_category:
neutral],[]

2- ('',Pref-0,[lexical_category: neutral]),['ق', 'ا', 'ل'],('ت',
PVSuff-t, [lexical_category: verb, tense: prefect, voice: active,
subj_person: 1, subj_num: sg, subj_gender:
neutral]),[lexical_category: verb, tense: prefect, voice: active,
subj_person: 1, subj_num: sg, subj_gender: neutral],[insert('و',5,
affix)]

3- ('', Pref-0,[lexical_category: neutral]),['ق', 'ا', 'ل',
'ت'],('وا', CVSuff-wA,[lexical_category: verb, tense: imperative,
subj_person: 2, subj_gender: m, subj_num: pl]), [lexical_category:
verb, tense: imperative, subj_person: 2, subj_gender: m, subj_num:
pl],[delete('ا',6, affix)]

4- ('',Pref-0,[lexical_category: neutral]),['ق', 'ا', 'ل',
'ت'],('وا',PVSuff-awA,[lexical_category: verb, tense: prefect, voice:
active, subj_person: 3, subj_gender: m, subj_num: pl]),
[lexical_category: verb, tense: prefect, voice: active, subj_person:
3, subj_gender: m, subj_num: pl],[delete('ا',6, affix)]
```

To dissolve each stem string to all possible base forms, the list of Arabic verb patterns[10] is expressed as deterministic finite state automata. The system matches the stem string against pattern list; it uses the three-way-match method but with missing and transformed letters only. If the match succeeds, it deletes from the resultant word any weak or hamza letters to get base word form then puts it in a solution list. This list is sent to next module for further processing. For example, if the learner writes 'قالتو', the resultant stem list would contain {'قالتو', 'قال' and'قالت'}. The first solution is discarded as it

[8] The end case of first is سكون , second is كسرة, third is فتحة and fifth is سكون

[9] The error type here is "short vowel became long one".

[10] The system should keep all Arabic patterns but we apply our technique only on verbs.

does not match any Arabic pattern. The second solution creates a list of base forms {'قال' and 'قل'} but after removing the weak letters it becomes {'قل'}. The third creates a list of base {'قالت' and 'قلت'} but also after removing the weak letters it becomes {'قلت'}. The final solution list of base becomes {'قل' and 'قلت'}.

Missing word check

The ALEDS stores the root of all words that forms a correct answer. This module processes every root in the correct answer by deleting any weak or hamza letters then matches them with the extracted base word forms. If any of the words in the task are not included in the student answer, the system reports a missing word error. If a match is succeeded then discard all other base forms from the solution list and send the modified list to next module. For example, consider that the correct answer contains the word 'قلت' which is derived from the root 'قول' and the learner answer is 'قالتو'. The extracted base word forms list will contain {'قل' and 'قلت'} and correct answer base word form becomes 'قل' after removing weak letter 'و'. So now; the extracted base word forms list contains only the base form 'قل' which corresponds to the following word analysis:

```
('',Pref-0,[lexical_category: neutral]),['ق', 'ا', 'ل'],('ت', PVSuff-
t, [lexical_category: verb, tense: prefect, voice: active,
subj_person: 1, subj_num: sg, subj_gender:
neutral]),[lexical_category: verb, tense: prefect, voice: active,
subj_person: 1, subj_num: sg, subj_gender: neutral],[insert('و',5,
affix)]
```

Extra word check

The system checks for base forms in the learner's answer that are not in the correct answer. The system reports an error if extraneous words are found in the learner input.

Word order check

The system here determines whether the base forms in the learner's answer are in order.

Lexical error check

The lexical error check is the most elaborate of the modules. Figure 4 presents the complete modules of lexical error check.

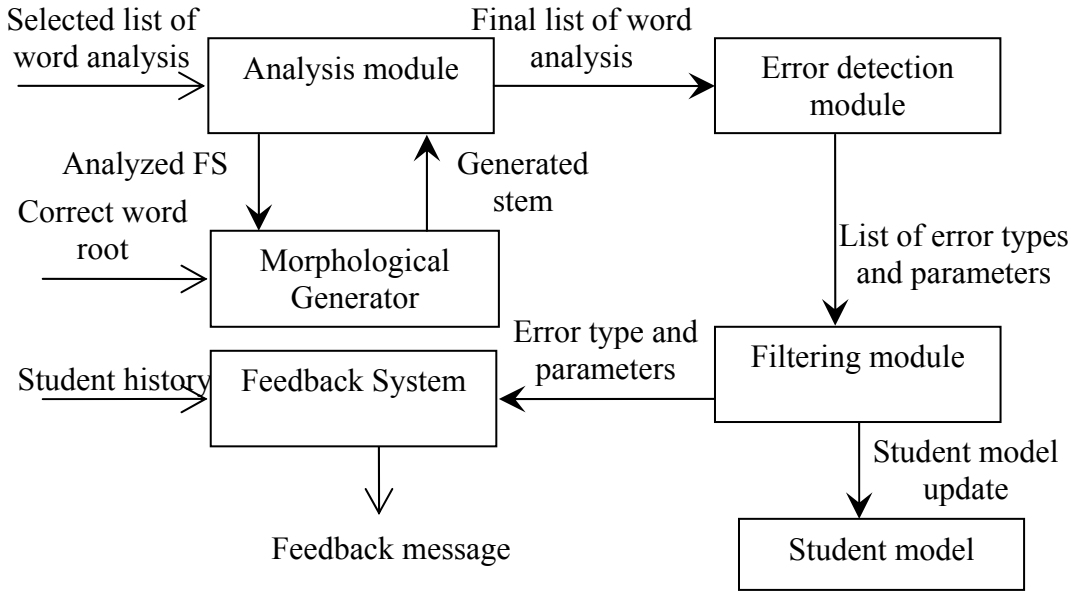

Figure4 the complete modules of lexical error diagnosis

Analysis module

This module proceeds with the analysis of all words in learner's answer to generate all possible complete analysis of input words. It receives a list containing all possible word analysis from the previous module. The word analysis structure contains: prefix structure, stem string, suffix structure, FS tag that describes the main features of word and error representation structure in affixes.

This module makes the following steps to get the final word analysis:

1. Filter the word analysis solution list input according to some heuristic filtering rules.
2. Send the filtered solution list to the answer analyzer module to trigger morphological generator component to get a well-formed correct stem of the analyzed FS.
3. Match the generated stem with the extracted stem using three-way-match method. The deleted and inserted characters should be only weak letters and the converted letters should be pronounced in the same way.
4. Put the final word analysis in the final solution list.

The input solution list could be minimized by a number of factors: learner answer, question parameters and error categories handled in the system. For example, if the learner writes the word 'تزرين' instead of 'زرت', the system will extract fifty different suffixes; some of them begin with 'ا' and 'و' characters[11]. It should delete them from solution list since this verb isn't defective and if it is added to suffixes that begin with a weak letter; last character will have a diacritic sign combatable with weak letter in the suffix (i.e. 'فتحة' with the suffix that begins with 'ا').

For every solution in the list, the system will morphological generate a well-formed stem. We build a shallow morphological generator that is based on the notion of a morphological hierarchy or tree [3] [16].

An example of final output of this module for the word 'قالتو' is:

```
('',Pref-0,[lexical_category: neutral]),['ق','ل'],('ت', PVSuff-t,
[lexical_category: verb, tense: prefect, voice: active, subj_person:
1, subj_num: sg, subj_gender: neutral]),[lexical_category: verb,
tense: prefect, voice: active, subj_person: 1, subj_num: sg,
subj_gender: neutral],[insert('و',5, affix), insert('ا', 2, stem)]
```

Error detection module

This module will recognize different error types from word analysis structure. It contains a set of if-then rules to recognize different error types. For example, to recognize *verb tense* error the system checks the tense feature in either correct and extracted FS. If they are different, the error is reported with two different tense as error parameters.

Filtering module

The final step in analyzing learners' answer is handled by the *filtering module*. The task here is to accommodate multiple errors, instructional feedback messages need to be prioritized by the system and displayed one at a time to the student to avoid multiple error reports. While it is desirable to construct a system capable of detecting and accurately explaining all errors, it does not follow that the system should display each and every error detected.

The ALEDS maintains an *error priority queue* which determines the order in which instructional feedback messages are displayed to the learner. It ranks instructional feedback with respect to the dependency of errors: word formation errors lower and hyper word error "constituent" higher. For example, the learner may write the word 'باعتو' instead of 'البيع'. Here the learner made three errors: *switching an infinitive with its conjugated verb*, *made short vowel long one "this error in suffix"* and *incorrect conjugation of hollow verb in prefect tense active voice*. The error priority queue gives a high priority to first error. It will send this error type to feedback system. However, the system updates student model for all the three errors but only gives feedback message to the first error.

[11] This is because we use three way match method. So it records suffix as 'ان' with deleted letter 'ا'

6. Conclusions

In this paper, we have described the development of a novel automated lexical error diagnosis system which based on its feedback helps Arabic SLLs to learn well-formed weak verbs. It has two main types of test items for interaction with the learner: selection-type and supply-type requiring the learners to write a few words. The objective test method is used such that the expected learner's answer is relatively short and well-focused.

The rule-based approach is used to give some freedom to the language learners in the way they phrase their answers, while enabling the exercise author to enter only one possible correct answer, thus saving much time compared to the traditional pattern matching answer coding approach. The rule-based approach has the advantage of providing detailed analysis of the learner's answer using linguistic knowledge. NLP tools in Arabic Lexical Error Diagnosis System include a morphological analyzer, morphological generator and error analyzer. These tools are used to diagnose and handle ill-formed input.

In this system, we successfully used *constraint relaxation* and *edit-distance* techniques to provide error-specific diagnosis and feedback to second language learners of Arabic. The system allows for individualization of the learning process by providing feedback that conforms to the learner's expertise. Inexperienced learners might require detailed instruction while experienced learners benefit from higher level reminders and explanations.

References

1. Bonfigli, M.E., Casadei, G. and Salomoni, P. (2000). "*Adaptive Intelligent Hypermedia using XML*". In Proceedings of the ACM symposium on Applied computing 2000**,** Como, Italy.
2. Bowden, T. and Kiraz, G.A. (1995). "*A Morphographemic Model for Error Correction in Non-concatenative Strings*". In Proceedings, 33 rd Annual Meeting of the Association for Computational Linguistics (ACL), Boston, MA, 24-30.
3. Cavalli-Sforza, V., Soudi A and Mitamura T. (2000). "*Arabic Morphology Generation Using a Concatenative Strategy*". In Proceedings of NAACL 2000. Seattle, WA.
4. Elmi, M.A. and Evens, M. (1998). "*Spelling Correction Using Context*". In proceedings of 36 rd Annual Meeting of the Association for Computational Linguistics (ACL), Montreal, Canada.
5. El-Sadany, T. A. and Hashish, M. A. (1989) "*An Arabic Morphological System*" In IBM Systems Journal, Vol. 28, No. 4, 600-612, 1989.
6. Anne Vandeventer Faltin. "*Syntactic Error Diagnosis in the context of Computer Assisted Language Learning*". PhD thesis, Geneva University, 2003.
7. Faltin, A.V., L'haire, S. and Ndiaye, M. (2005). "*A spell checker for language learners of French and a learner corpus*". In Proceedings of EUROCALL 2005. Cracow, Poland.
8. Faltin A.V. (2003). "*Natural Language Processing Tools for Computer Assisted Language Learning*". In Linguistik online Journal, May 17, 2003.
9. Binglan Han. "*STUDENT MODELLING AND ADAPTIVITY IN WEB BASED LEARNING SYSTEMS*". M.A. thesis, Massey University, New Zealand, 2001.
10. Trude Heift. "*Designed Intelligence: A Language Teacher Modal*". PhD thesis, Simon Fraser University, Canada, 1998.
11. Heift, T., Toole, J., McFetridge, P., Popowich, F., Tsiplakou, S. (2000). "*Learning Greek with an Intelligent and Adaptive Hypermedia System.*" in IME Journal of Computer-Enhanced Learning, Vol. 2, (2), October, 2000.
12. Hsieh, G., Tsai, T.H., Wible, D. and Hsu, W.L. (2002). "*Exploiting Knowledge Representation in an Intelligent Tutoring System for English Lexical Errors*". In Proceedings of the International Conference on Computers in Education ICCE 2002.
13. Peter Ingels. "*A Robust Text Processing Technique Applied to Lexical Error Recovery"*. PhD thesis, Linkping University, Sweden, 1997.
14. Jassem Ali Jassem. (2000). "Study on Second Language Learners of Arabic: An Error analysis Approach". Kuala Lumpur (Malaysia): A.S. Noordeen. ISBN 983-065-093-6.
15. Rimrott, A. Term Paper Computational Linguistics (2003). *"SANTY: A Spell Checking Algorithm for Treating Predictable Verb Inflection Mistakes Made by Non-Native Writers of German"*. Term Paper for LING 807 – Computational Linguistics at Simon Fraser University (Burnaby, Canada).

16. Soudi, A., Cavalli-Sforza, V. and Jamari, A. "*A Computational Lexeme-based Treatment of Arabic Morphology*". In Proceedings of the Workshop on Arabic Language Processing: Status and Prospects, ACL 2001, Toulouse (2001).
17. Tschichold, C. (2003). "*Lexically Driven Error Detection and Correction*" in CALICO Journal, Vol. 20, No. 3, 549-559.
18. Wright, W. (1967). "A Grammar of the Arabic Language". Cambridge University Press; Third edition.
19. مصطفى عبد العزيز (1975) "الوسيط في علم الصرف- قسم تصريف الافعال"
20. محمد بخيت بن حاج علي. "تحليل الاخطاء اللغوية لدى طلاب جامعة ماليا بماليزيا: دراسة وصفية تحليلة". رسالة ماجستير، جامعة القاهرة 1998