Text generators, error analysis and feedback

Juan Rafael Zamorano Mansilla
Department of Anglistik, Universität Bremen, Bremen
juanrafaelz@terra.es

Abstract

The purpose of this paper is to show how the linguistic resources contained in the multilingual text generator KPML can be used for detecting errors in fill-in-the-blank exercises of English and Spanish and providing relevant feedback. This constitutes the first step in the creation of a module that will detect errors in full sentences in future. In this paper I describe the algorithm employed to diagnose the cause of an error and some examples of the results obtained in initial tests. The algorithm has two important advantages: a) it was easy to create because it consists mainly of operations already present in the KPML environment, and b) it does not contain lists of anticipated errors or malrules. The final outcome of the algorithm is a collection of grammatical features that explain the difference between the user's input and the right solution. Although no list of anticipated errors or malrules were included, the features obtained seemed useful material for the production of feedback text and the algorithm turned out to be able to handle a wide range of grammar-based errors.

1. Introduction

The detection of errors and the production of adequate feedback has been regarded as one of the areas of CALL which could benefit most from the advances provided by natural language processing systems [1],[4],[7],[8],[9],[11],[12].

In the wake of these studies and as part of my dissertation thesis I am exploring how the multilingual generator KPML ([2],[3],[5],[10]) can be used to detect errors in language learning exercises, make a diagnosis and produce relevant feedback.

What I describe in this paper is the algorithm designed for the recognition of errors in simple exercises, mainly of the fill-in-the-blank type. These exercises were automatically generated from previously stored sentences by a module, the Exercise Manager, currently being developed also as part of my dissertation thesis.

At present the algorithm is still at a developing stage, and the results presented here were obtained in simple tests in which different inputs were type in to see how they were handled by the algorithm. Besides what the algorithm returns at present is a collection of grammatical features that hint at the cause of the error, not a full message. These features however will be used in future for producing feedback messages.

In spite of these factors, the results of the initial tests have been rather promising. The main advantages we have found over other CALL programs that make use of natural language technology are:

a) the amount of effort and time spent on programming to obtain acceptable results. All the operations that make up the algorithm described here can be implemented with KPML at present. Only the comparison between strings or lists and the decisions derived from the comparisons are completely new.

b) the algorithm defined for the Exercise Manager intends to be "universal". I have tried to avoid the necessity of including lists of malrules or anticipated mistakes by using a single algorithm that copes with as wide a range as possible of errors related to the grammar or the lexicon.

2. An overview of the KPML system

In order to understand how the algorithm for the analysis of errors works it is essential to know something about the architecture of KPML first.

The two components that are relevant for the purposes of this paper are the grammar and the lexicon.

2.1 The grammar

KPML has been developed in the systemic-functional tradition, which means that the grammatical descriptions it contains are based on the theories of systemic-functional linguistics (SFL). This is not the place to enter into the details of this approach to language, but there are two important characteristics that should be mentioned: first, language is regarded as "resource for expressing and making meaning" [2]; second, language is considered to be structured as a semiotic system, where "the process of language use is a process of making meanings by choosing." [6]. As a consequence systemic-functional grammars are essentially an attempt to specify the potential meanings or communicative intentions speakers may wish to express. Those potential meanings or communicative intentions are called features and are grouped in so-called systems. The choice of one feature or another within a system is guided by the communicative intentions, and each choice leads to a new system with further choices between features. These choices become more and more specific as we move deeper into the grammar.

A good example of this is provided by the region of mood in English. Here we find an initial choice between two features: Indicative and Imperative. The feature Indicative leads to a new system containing the features Interrogative and Declarative. And the feature Interrogative leads in turn to a new system with the features Wh- and Yes/no.

Notice that these features represent semantic choices: the selection of the feature Imperative indicates that the speaker wishes to modify the addressee's behaviour rather than provide or ask for information. Of course these choices have a realization in the language, which are indicated in the grammar by means of realization statement associated to the features. The realization statement associated to the feature Declarative, for instance, is Subject^Finite, which indicates that the function Subject precedes the function Finite. There are different types of realization statements to cover all the needs of the grammar; some of them indicate that a function such as Subject or Attribute must be inserted; others regulate the order of the functions within the sentence; some realization statements inform about the specific lexical item that will realize a function, while others inform about the inflection of lexical items.

2.2 The lexicon

Lexical items are stored in KPML as shown in figure 1:

```
(LEXICAL-ITEM
 :NAME FACTORY
 :SPELLING "factory"
 :FEATURES (NOUN ES COMMON-NOUN COUNTABLE)
```

Figure 1. A lexical item in KPML

There are three basic slots where different types of information are stored: under :name we include the label by which the word is known to the grammar. Under :spelling we include the basic spelling of the word. And finally the slot :features contains important information such as the word-class the item belongs to (noun, in this case), the type of plural ending it takes (es) and other strictly lexically conditioned features with syntactic or morphological relevance, such as common-noun or countable.

In the case of words with irregular inflection, there is an additional slot, :properties, where the various inflected forms of the word are kept. Figure 2 shows the specification of the Spanish definite article:

```
(LEXICAL-ITEM
:NAME definite-article
:SPELLING "el/la"
:FEATURES (definite)
:PROPERTIES ((singularmasculine "el") (singularfeminine "la") (pluralmasculine "los") (pluralfeminine "las")))
```

Figure 2. A lexical item with irregular inflection in KPML

3. Using the linguistic resources of KPML for error diagnosis

In this section I briefly describe in human terms the algorithm that makes use of the KPML linguistic resources in order to detect errors and diagnose its cause. But before starting with the description of the algorithm, it is important to realize that every grammar exercise implemented with Exercise Manager automatically generates the right answer. This does not only consist of a string of characters, but also includes a significant amount of information about the parts of the grammar activated in the process of generation. For example, in the sentence He got married ____ 1985, the constituent from which the preposition has been removed contains the following collection of grammatical features: (groups-phrases prepositional-phrase nonwh-phrase prepositional-phrase-simplex minirange-thing spatio-temporal-process location-process temporal-process unordered strong-inclusive).

The algorithm is activated when the program finds a mismatch between the user's input and the solution. Then the program first determines if the mismatch is due to a wrong lexical choice or a wrong inflection of the right lexical item. This is done by checking if the stem of the word that constitutes the right answer is part of the user's input. If the result is negative, the program tries comparing other stems of the word included in the :properties slot of the lexicon. If the result is still negative, this means that the user has chosen a different word. If any of the two comparisons described above is positive, the program concludes that the word is the right one, and the mismatch is due to the inflexion applied by the user.

The next step is roughly the same in both cases: the program calculates the path in the grammar that would generate the user's input and compares it with the path that generates the right answer. The result of the comparison in all the tests carried out was a couple of features that belonged to the same system, one corresponding to the right solution and the other to the user's input. The fact that these features belong to the same system means that there is a meaningful opposition between them; and since these two features mark the precise point of

divergence between the user's input and the right answer, the information they convey is in all cases a very accurate diagnosis of the cause of the mistake. But before considering some examples, I would like to say something about the process of "calculating the path in the grammar that would generate the user's input" mentioned above. Natural language generators have in principle all the resources necessary to recognize any structure by the simple process of generating all the alternatives their linguistic resources can predict and comparing the results with the input. But as has been pointed out [9], the number of combinations an average grammar can predict is so high that the process is in reality not feasible.

In the algorithm presented here this is not a problem, because the calculation of the path that leads to the user's input is guided in a very precise way. The first thing the program does is find out how many features from the grammar can impose the user's word as a realization or the user's morphology, depending on the case. For example, if the program asks what features impose the realization of in as the preposition of a prepositional phrase, the result is: In-extent, Strong-inclusive and Containment-implicit.

This operation is very fast and simple and is already available in the KPML environment as a matter of fact. The tests revealed that in most cases the result is a list of features rather than one feature. So in most cases the program must decide what feature from the list will offer more relevant information. To do this we resort to another function already present in KPML: Show path to. This function shows the collection of features in the grammar that lead to the specified feature. For example, if we apply the function Show path to to the feature In-extent, we obtain the following result: (in-extent temporal-process extent-process prepositional-phrase groups-phrases spatio-temporal-process relative-extent-process).

The tests revealed that the most accurate results were obtained when we chose the feature whose path contained more points in common with the collection of features activated during the generation of the right answer, so this is the strategy followed by the program.

After this process of selection the program ends up with two lists of grammatical features: one for the right solution and another for the hypothetical generation of the user's input.

At this point the program only has to compare both lists to find the features that are not shared. Among these, only two, one on each list, can belong to the same system. As I mentioned above these two features mark the precise point of divergence between the user's input and the right answer and so convey very accurate information about the origin of the error.

The following are some examples of the grammatical features returned by the algorithm with different exercises.

In the sentence Don't put your feet ___ the table, a clear case of exercise purely based on lexical selection, the insertion of the preposition with returned the features accompaniment-process for the user's input and spatio-temporal-process for the right answer. The preposition in returned three-dimensions as opposed to one-two-dimensions corresponding to the right answer. And the preposition onto returned the feature motion-process for the user's input and rest-process for the right answer.

In the Spanish sentence Usted ____ (comer) mucho [You-polite form (eat) a lot] the learner must provide the right inflexion of the verb. The form como returns the feature firstperson-form for the user's input and secondperson-form for the right answer. The form comes returns the feature informal-form for the user's input and formal-form for the right answer. The form comen returns the features thirdperson-form and plural-form for the user's input and secondperson-form and singular-form for the right answer.

Finally, there is also the possibility that both mistakes, i.e. lexical and morphological, can be present in the same exercise. In this case the algorithm gives priority to the lexical choice. In the sentence Who lives in ____ house?, intended for the demonstrative that, the insertion of the demonstrative these returns the feature near for the user's input and far for the right answer, ignoring the fact that the choice of the plural form is also incorrect. The article the returns the feature nonselective for the user's input and demonstrative-selection for the right answer. Finally, if the form inserted is those, the algorithm returns the feature plural-form for the user's input and singular-form for the right answer.

In the Spanish sentence Juan _____ cansado [Juan _____ tired], where the student must choose between the verbs ser and estar, the form es returns the feature permanent-property for the user's input and the feature temporary-property for the right answer. The form estoy returns the feature firstperson-form for the user's input and thirdperson-form for the right answer. The flexibility and potential of the algorithm is revealed when we introduce verbs that do not even belong to set of possible answers in the exercise. The verb tener [have], for instance, returned the feature possessive for the user's input and intensive for the

right answer. And the verb ver [see] returned the feature mental for the user's input and relational for the right answer.

As I said above, I believe the features returned by the algorithm are excellent material for the automatic production of feedback text. They are very accurate comments about the origin of the learner's error and they also offer the possibility of producing different types of messages. If we take the feature corresponding to the user's input we can produce negative messages of the type "The preposition you need here does not express motion". If we take the feature corresponding to the right answer the message generated would be something like "The preposition you need here is used with three dimensional objects". Notice that even in the case of messages relative to the right answer the information offered is highly personlized and not fixed at all. This is due to the fact that the feature returned corresponding to the right answer varies according to the user's input, as can be observed from the examples above.

As a third option one can take both features returned by the algorithm and produce a complex message of the type "You used the third person of the verb, but the context requires the second person". Nevertheless the exact form and content of the feedback messages is a matter that will be resolved in the future, being the main problem that these features are often highly specialized terms.

4. Conclusions and future work

In this paper I have shown how the linguistic resources contained in the multilingual generator KPML can be used to recognize errors in simple grammar exercises and return information that can be employed to generate a feedback message.

The results obtained in the initial tests have been satisfactory, although the real usefulness of the algorithm will be assessed in experiments with real students to be carried out during the winter semester of 2004 at the universities of Bremen and Complutense of Madrid.

The main advantages of the algorithm described here are:

a) it is universal. This means that the same algorithm is used to handle all kinds of grammatical and lexical errors. No list of anticipated errors is necessary;

b) the diagnosis of the cause of the error is highly accurate and personalized, increasing the quality of the feedback text that can be generated.

Future lines of investigation include three chief areas:

a) the possibilities offered by KPML in the recognition of errors related to linguistic levels of analysis different from the grammar and the lexicon, such as punctuation, alternations in the spelling of inflected forms (e.g. English sto p sto pping) and phonological variations of inflected forms (e.g. Spanish p oder p uedo).

b) the possibilities offered by KPML in the recognition of errors in full sentences, not just words.

c) the possibilities offered by KPML in the recognition of errors caused by linguistic transfer.

References

[1] Allen J R (1996). The Ghost in the Machine: Generating Error Messages in Computer Assisted Language Learning Programs, CALICO Journal, 13/2-3: 87-103

[2] Bateman J A (1997). Enabling technology for multilingual natural language generation: the KPML development environment, Journal of Natural Language Engineering, 3: 15—55

[3] Bateman J A (1998). Automated Discourse Generation. In Kent A (ed.) Encycolpedia of Library and Information Science, vol 62, supplement 25, Marcel Dekker, Inc., New York: 1—54

[4] Chen J F (1997). Computer Generated Error Feedback and Writing Process: A Link, TESL-EJ, 2/3.

[5] Dale R and Reiter E (2000). Building natural language generation systems, Cambridge University Press, Cambridge, 2000.

[6] Eggins S (1993). An Introduction to Systemic Functional Linguistics, Pinter Publishers, London, 1993.

[7] Holland V M and Kaplan J D (1995). Natural Language Processing Techniques in Computer Assisted Language Learning: Status and Instructional Issues, Instructional Science, 23: 351-380.

[8] Koller T (2003). Knowledge-based intelligent error feedback in a Spanish, Proceedings of The 14th Irish Conference on Artificial Intelligence & Cognitive Science, 2003, Dublin, Trinity College, 117-121.

[9] Levison M, Lessard G, Walker D (1998). The Intelligent Detection of Second Language Learner Errors, ALLC/ACH Joint Annual Conference, 1998, Lajos Kossuth University, Debrecen, Hungary.

[10] Matthiessen C M I M and Bateman J A (1991). Text Generation and Systemic-Functional Linguistics. Experiences from English and Japanese, Pinter Publishers, London, 1991.

[11] Reuer V (2003). Error Recognition and Feedback with Lexical Functional Grammar; CALICO Journal, 20(3):497-512

[12] Yang J C and Akahori K (1997). Development of computer assisted language learning system for Japanese writing using natural language processing techniques: A study on passive voice Proceedings of the workshop "Intelligent Educational Systems on the World Wide Web", 8th World Conference of the AIED Society, August 1997, Kobe, Japan, 18-22.