**Linguistic Knowledge and Reasoning for Error Diagnosis and Feedback Generation**

**RODOLFO DELMONTE**  
Department of Language Sciences  
Università Ca’ Foscari - Ca’ Garzoni-Moro  
Venice, Italy

**ABSTRACT**

We present four sets of NLP-based exercises for which error correction and feedback are produced by means of a rich database in which linguistic information is encoded either at the lexical or at the grammatical level. One exercise type “Question-Answering” utilizes linguistic knowledge and inferential processes on the basis of the output generated by GETARUN, a system for text understanding. GETARUN produces a complete parse of a text and a semantic mapping in line with situational semantics in the form of a Discourse Model. Another exercise, Grammcheck, uses a ‘robust’ version of the parser to produce suitable environments for grammatical error spotting and consequent accurate and precise feedback generation for German. The parser of GETARUN is then presented as an analytical tool for students who study Lexical Functional Grammar (LFG). Finally, exercises on “Essay Evaluation,” which are cast into the more general problem of text summarization, are discussed. In this case, the system is used to perform multidocument sentence extraction on the basis of a statistically based Summarizer. This summary is then compared with the student’s summary. All applications can be found at our web site, project.cgm.unive.it.

**KEYWORDS**

NLP Techniques, CALL Tools, Cooperative Question-Answering, Summarization

**1. INTRODUCTION**

The GETARUN program (Delmonte, 1990; Delmonte, Bianchi, & Pianta, 1992; Delmonte & Bianchi, 1998) is a system for text and reference understanding, which is currently being used for summarization and text generation, and has a sophisticated, linguistically based semantic module used to build up a discourse model (DM). Semantic processing is strongly modularized and distributed among a number of different submodules which take care of spatio-temporal reasoning, discourse level anaphora resolution (Delmonte & Bianchi, 1999), and other subsidiary processes like topic hierarchy—which impinges on relevance scoring when creating semantic individuals. The system uses a parser that requires in its deep
version a complete lexicon of the domain in which it will perform its analysis. This deep version is used for students of linguistics as an aid in the assessment and control of grammatical principles. It allows for the parsing of grammatical and ungrammatical sentences. The “shallow” version of the parser allows students of German to get detailed information on their grammatical mistakes. We will show how GETARUN is used for different linguistic exercises for learners of different languages. We will concentrate on how GETARUN facilitates the provision of adequate feedback.

1.1 Self-assessment and Feedback

Generally speaking, assessment in self-instructional courses is problematic but of course very important. Within learner-centered self-instruction, or self-directed learning, self-assessment is a necessary part. Decisions about whether to go on to the next item, exercise, or unit; the allocation of time to various skills; or the need for remedial work are all based on feedback from informal and formal assessment. This concept then is central both to learners and to the kind of courseware we wish to build. We consider self-assessment important as an educational goal in its own right, and training learners to use self-assessment is beneficial to learning. In fact, language learners regularly engage in self-assessment as part of their learning. They complete exercises and check, by whatever means available, whether their responses are correct or not.

In this paper, we present an approach to teaching the comprehension of spoken and written texts by facilitating related text production and by providing explanatory feedback. We consider understanding texts, whether oral or written, an important objective of language learning. A system for CALL that is aimed at tutoring and testing students in text understanding should ideally be equipped with a feedback module to provide explanations for mistakes made by the students. However, many systems today provide very limited feedback: an answer is either right or wrong, and no explanation is made available. An additional limitation is that drills for text understanding on the computer are often of one of two types: multiple choice and/or true/false decisions. Drills that permit students to answer questions by producing free text, even short segments, are rare because automatic analysis and feedback are hard to implement for written language. Production tasks constitute a challenge in that the right feedback may be unavailable if students make an unanticipated mistake, one not included in a list of possible mistakes.

What kind of feedback could be given? In their paper, Lyster and Ranta (1997:45) make the following classification of feedback by human tutors:

1. Explicit correction: “the explicit provision of the correct word or part phrase, usually making clear that this is a correction— e.g. you mean ..., you should say ...”
2. Recast: “the teacher’s reformulation of all or part of the student utterance, minus the error, without making it clear that this is a correction.”

514
3. Clarification request: “What? What do you mean? (only coded in response to language error).”

4. Metalinguistic feedback: “Comments, information or questions regarding the well-formedness of the student’s utterance, but without giving the correct form: that’s not quite right, is that right?”

5. Elicitation: “Getting the student to give the correct form by pausing for her to continue the sentence, or by asking the student to reformulate the utterance.”

6. Repetition: “the repetition, in isolation, of the student’s utterance, usually with error intonationally marked.”

We believe that recast, clarification request, elicitation, and repetition are totally inadequate for feedback generation on a computer. As to explicit correction, perhaps it could be done for grammar drills, but it is certainly much harder in semantically based drills. We assume that only metalinguistic feedback is fully compliant with the current state of human-computer interaction.

In all cases, we want learners to be informed about the error they made, the kind of error they made, and the possible reason why they made it. In addition, they can be directed to carry out some linguistic activity appropriate to help them remedy the problem.

### 1.2 Our Applications

The applications presented in this paper are all concerned with text comprehension. The first one, Grammcheck, presented in section 2 below, is an application for students of German which prompts them to create sentences for which they are given a sequence of base forms or lemmata. These lemmata are taken from a database of correct and incorrect sentences that constitutes the Linguistic Knowledge Database (LKD). We also use a large lexicon of German where lemmata are fully classified with subcategorization frames and morphological features. Knowledge in this case is resident both in the database and in the grammar contained in the analysis program—a robust parser of German.

The second application uses the same system as Grammcheck for German (see section 3 below). It is called GETARUN. Here we use its complete and deep version. The idea for these activities is to help students understand the relevance of linguistic and extralinguistic information in the grammatical analysis and the representation of sentences of a given language—English in this case. The system uses a top-down, depth-first definite clause grammar (DCG) parser with lookahead and a well-formed substring table (WFST) lookup in case of failure to improve efficiency. It implements the core and periphery grammar rule model accompanying the notion of Universal Grammar. This allows it to be multilingual, that is, it parses with the same grammar and set of parameters for German, English, and Italian. The important feature of the parser is the implementation of parsing strategies to allow for multiple analyses of a single input sentence to be appropriately executed.
In section 4, we discuss the generation of “Question-Answering” exercises which utilize linguistic knowledge and inferential processes on the basis of the output generated by GETARUN, our system for text understanding. The GETARUN system produces a complete parse of a text and a semantic mapping in line with situational semantics in the form of a discourse model (DM). The DM is used to generate questions and answers based on the text that the system analyzed and that the students had to read. Students are then given feedback on the question and answers they selected.

Finally, in section 5, exercises on “Essay Evaluation,” which are cast into the more general problem of text summarization, are discussed. In this case, the system is used to perform multidocument sentence extraction on the basis of a statistically-based Summarizer. This summary is then compared with the student’s summary.

2. GRAMMCHECK

The first application is a grammar checker for Italian students of German (and English) (see Delmonte, Chiran, & Bacalu, 2001; Delmonte, 2000a). It is based on the shallow parser of Italian used to produce the syntactic constituency for the National Treebank. The output of the parser is a bracketing of the input tagged word sequence which is then passed to the higher functional processor. This is a Lexical Functional Grammar (LFG)-based c-structure to f-structure mapping algorithm which has three tasks: the first task is to compute features from heads; the second is to compute agreement, and the third is to impose LFG’s grammaticality principles of coherence and consistency to insure that the number and type of arguments are constrained by the lexical form of the governing predicate.

The parser uses a recursive transition network (RTN) which has been endowed with a grammar and a lexicon of German of about 8,000 entries. The grammar is written in the usual arc-transition nodes formalism, well known in augmented transition networks (ATN). However, the aim of the RTN is to produce a structured output both for well formed and ill formed grammatical sentences of German. To this end, we allowed the grammar to keep part of the rules of Italian at the appropriate structural level. Grammar checking is not accomplished at the constituent structure building level, but at the function-structure level.

2.1 The Shallow Cascaded Parser

The function of the shallow cascaded parser is to create syntactic structures eligible for grammatical function assignment. This task is made simpler given the fact that the disambiguator associates a net or constituency label with each disambiguated tag. Parsing can then be defined as a bottom-up collection of constituents which contain either the same label or which are contained in or are a member of the same net or higher constituent. No attachment is performed in
order to avoid being committed to structural decisions which might then reveal themselves to be wrong. We prefer to perform some readjustment operations after structures have been built rather than introducing errors from the start. Readjustment operations are in line with the LFG theoretical framework which assumes that f-structures may be recursively constituted by subsidiary f-structures (i.e., by complements or adjuncts of a governing predicate). So the basic task of the shallow parser is that of building shallow structures for each safely recognizable constituent and then pass this information to the following modules.

The tagset we use for German consists of 85 tags which encore a number of important features for the parser such as transitivity, modality, and auxiliary class for verbs and semantic classes like color, human, and evaluative for nouns. Tags are disambiguated by a statistical and syntactic procedure which is set up for special ambiguity classes. In some cases, we use appropriately organized Finite State Automata. The output of the disambiguator is a partially disambiguated input which is then processed by the shallow cascaded parser (see Figure 1).

Figure 1
*GETARUN* Shallow Parser Architecture

### 2.2 Syntactic Readjustment Rules

Syntactic structure is derived from shallow structures by a restricted and simple set of two categories of rewriting operations: deletions and restructuring. In building syntactic constituents, we obey the general criteria below:
1. We accept syntactic structures which belong to either language—German or Italian.

2. Constituency should allow for the recovery of errors in the higher structural layers where functional mapping takes place.

3. The tensed verb is treated in a special manner. If it is sentence final, it belongs to a separate ibar constituent called IBAR2, and it triggers the building of a specific IP clausal constituent called FYESNO in all “aux-to-comp”-like structures and structures subject to inversion. Otherwise, it is treated as in Italian.

### 2.3 From C-structure To F-structure

Before working at the functional level, we collected 2,500 grammatical mistakes from students’ final tests. We decided to keep track of the following grammatical mistakes which are typical for Italian learners of German: lack of agreement NP internally; wrong position of argument clitic pronouns; lack of subject-verb agreement; wrong position of the finite verb in main clauses, subordinated clauses, or coordinated clauses; and wrong case assignment. Example (1) illustrates this process.

(1) *Heute willst ich mich eine bunte Krawatte umbinden.*

‘today want I me a colorful scarf tie’

(today I want to wear a colorful scarf)

\[
\text{cp-[
  \text{advp-[adv-[heute]],}
  \text{vsec-[vmod-[willst],}
  \text{fvsec-[subj2-[np-[pers-[ich]]]],}
  \text{obj-[np-[clitdat-[mich]]],}
  \text{obj1-[np-[art-[eine],adj-[bunte],}
  \text{n-[krawatte]]],}
  \text{ibar2-[vit-[umbinden]]]
], punct-[.]}\]

The parser issues two error messages. The first one regards case assignment: *mich* is in the accusative whereas dative is required. The second one concerns subject-verb agreement: *willst* is second person singular whereas the subject *ich* is first person singular. In order to recognize errors, full morphological and lexical subcategorization information for all words must be available. For instance, the entries for *ich, wollen,* and *umbinden* are specified in example (1).

### 2.4 Sentence Creation and Automatic Evaluation

In order to build exercises automatically, we duplicated all the sentences with mistakes from our database and created the corresponding correct sentences. This procedure allowed us to generate exercises for students by picking at random a
certain number of sentences, say three or four, from the correct subset and mix them with one or two sentences from the mistakes subset. The task for students could be either to identify the sentences with error(s) or correct the error(s). In either case, their response could be easily checked. Rather than discussing these exercises, we will concentrate on the “Sentence Creation” exercise which requires students to produce a correct sentence from a sequence of input hints consisting of lemmata (uninflected content words). This procedure first selects one of the correct sentences. It then deletes the function words in the sentence and displays the lemma for each content word. The resulting sequence of words is presented to students who are asked to build a correct sentence.

Given the fact that students can produce any sentence using the lemmata provided, we cannot evaluate their responses by a simple pattern-matching operation. The parser has to check for correctness. Figure 2 shows the student window.

Figure 2
Sentence Creation Exercise: Input Window

In this window, and the ones shown below, we use English rather than Italian for student instructions to allow English or French students to use the system. We also prompt students not to type upper case letters because the system only uses lowercase letters. After they type in the sentence, they click on the OK button, and the parser produces a complete parse and an evaluation by the Grammar Checker. The output is shown in Figure 3.
The system addresses issues for students of German who are enrolled in degree programs in Linguistic Sciences where General Linguistics and other similar courses are required. Students are asked to repeat an exercise after they have checked for mistakes in the feedback window. In the case of a sentence being correctly entered, the system simply confirms the correctness and proposes a new sentence. Whenever students decide to interrupt the exercise, an evaluation is issued for the whole interaction, and the result is shown graphically by turning previous successes and failures into scores and then transforming scores into colored bars: red for mistakes and green for correct sentences. A comment is generated based on the severity of the errors and by relying on the overall score.

3. GETARUN: A PARSER FOR LFG STUDENTS

We have seen how the shallow version of the GETARUN parser is used for the analysis of linguistic errors. Here, the detailed description and disambiguation of sentences in linguistic analysis is the task of the ‘deep’ version of the parser. The GETARUN program is a web-based multilingual parser which relies mainly on Lexical Functional Grammar (LFG) theory and partly on Chomskian theories and incorporates a number of parsing strategies which allow students to parse ambiguous sentences using the appropriate strategy in order to obtain an adequate grammatical output.
The underlying idea was that of stimulating students to ascertain and test linguistic hypotheses by themselves by means of a linguistically motivated system architecture. The parser builds c-structure and f-structure and computes anaphoric binding at the sentence level; it also has provisions for quantifier raising and temporal local interpretation. Predicates are provided for all lexical categories, and their description is a lexical form in the sense of LFG (see example in Figure 4).

**Figure 4**
Web Version of the LFG Parser

It is composed both of functional and semantic specifications for each argument of the predicate: semantic selection is operated by means both of thematic role and inherent semantic features or selectional restrictions. Moreover, in order to select adjuncts appropriately at each level of constituency, semantic classes are added to the more traditional syntactic ones. Semantic classes are of two kinds: the first class is related to extensionality versus intensionality and is used mostly to build discourse relations; the second class is meant to capture aspectual restrictions which decide the appropriateness and adequacy of adjuncts so that inappropriate ones are attached at a higher level (see Figure 5).
However, the most interesting part is how the system behaves in the presence of ungrammatical sentences in which students should be told which grammatical principle has been violated. We used test suites of sentences which had been gathered for that task such as the one advertised by the LINGO-Redwoods initiative (see lingo.stanford.edu). Examples of such sentences are:

1. Who does Mary like John?
2. Who did you mention Bill’s belief that you saw?
3. John believes that himself likes Mary.
4. John was believed that is clever.
5. Who did he try to win the race?

The system rejects these sentences as ungrammatical but then, according to whether it has generated a wh-operator or not, activates different feedback strategies. For instance, for examples 1, 4, and 5, the presence of a wh-operator which was unable to bind a variable is interpreted as a syntactic binding violation and in cases 1 and 4 also as a violation of grammatical coherence. In case 2, the NP headed by “belief” does not allow the operator to carry its variable to be bound in the lower sentence. Case 3 is explained because it is impossible to have a reflexive pronoun bound as the subject of a clause. Sentence 4 shows that the sentential complement cannot be interpreted for lack of an expressed lexical subject. These are the linguistic phenomena students can learn more about and understand better when they work with our parser.
3.1 Parsing Strategies

Another phenomenon which receives some attention in the study of linguistics is ambiguity (see Schubert, 1984; Altman, 1989; Frazier, 1987). Ambiguities arise, for example, if a pronoun in the subclause could have one of two antecedents in the main clause:

(2) The authorities refused permission to the demonstrators because they feared violence.
The authorities refused permission to the demonstrators because they supported the revolution.

The underlying mechanism for ambiguity resolution takes one analysis as the default in case it is grammatical. The other plausible interpretations are obtained by activating one of the available parsing strategies which are linguistically and psychologically grounded (see Delmonte, 2000b, 2000c). These strategies allow us to check in the example above whether there is more than one antecedent for a pronoun. Generally, the strategies are used to re-assess syntactic structures which are prone to ambiguity. With this application, we hope to help students understand syntactic analysis better.

4. QUESTION-ANSWER SEQUENCES FOR LISTENING COMPREHENSION TASKS

We are not using the complete version of the GETARUN parser only for syntactic analysis. The parser also provides the basis for the generation of exercises which follow from text understanding. Text understanding (see Iwanska & Shapiro, 2000; Herzog & Rollinger, 1991; Delmonte, 2002b, 2002c) is a task which constitutes a challenge in that the right feedback may not be available if students provide an incorrect answer which is not included among the list of possible mistakes.

We use question-answer dialogs with listening comprehension tasks. Students hear a text read by the internal text-to-speech module or a previously recorded text. No written version is provided to students. At the end of the listening activity, a certain number of questions appear on the screen, and students are prompted to provide answers to each of them.

Each text given to students is represented in the system in the form of a discourse model (DM) and turned into an appropriate database structure. This structure can then be analyzed by our programs. The system takes as the starting point the feature structures represented as direct acyclic graphs (DAGs) of each input sentence analyzed by the parser. Then, in the semantic analysis, the f-structure is turned into a logical form (i.e. a set of well formed formulas). These formulas are mapped onto semantic representations, that is, predicate-argument structures with a polarity and a couple of indices for spatiotemporal locations (based on situation semantics). The final output is a DM. The sequence of internal processes is shown in Figure 6.
We shall discuss the system’s behavior on the basis of the following short text:

At the Restaurant
John went into a restaurant. There was a table in the corner. The waiter took the order. The atmosphere was warm and friendly. He began to read his book.

In the knowledge representation, we establish a semantic relation that holds between a sentence and an interval in the spirit of interval semantics. We specify what property of an interval is entailed by the input sentence and then compositionally construct a representation of the event from the intervals and their associated properties. The DM provides us with the output of the temporal reasoning algorithm which allows us, together with the spatial location inferential module, to determine when and where entities, their relations, and their properties are situated.

4.1 Queries to the System
We present a Question-Answer module which can be used for well defined domains (see Figure 7).
The domain in our case coincides with the text the system has just analyzed and transformed into a DM. Below are some of the queries that can be addressed to the system (see Delmonte, 2002a), here generated by the system. The reason why we let the system generate both questions and answers is that we want to prevent the dangers related to the “open dialogue” mode of questions and answers; we work within the much safer “close dialogue” mode. We also want to prevent having to check for appropriate orthography and grammar and to concentrate on text understanding. The queries generated include questions on spatio-temporal locations (see Bianchi & Delmonte, 1996), identity, and activities.

What has John begun?
Who has begun doing something?
Where was John before going into the restaurant?
How was the atmosphere?
Did the book go into the restaurant?
Did the waiter read the book?

Answers are also automatically generated. However, the interesting part of the program is obviously the possibility of recovery from failure in case of wrong inferences. According to the type of query, failure may be recovered and appropriate feedback generated.

In the exercise, students are presented with four questions generated at random. They choose a question and are then given automatically generated answers to choose from (see Figure 8).
Theoretically, there are three possible errors: wrong answers, inconsistent questions, and inconsistent answers. Wrong answers are produced because there has been some misunderstanding. The problem is how to give cooperative responses in the case of semantic inconsistency. These mistakes may have been made either because students did not fully understand the semantic relations explicitly or implicitly stated in the text. In the case of implicit semantic relations, mistakes can be due to false presuppositions, violations of pragmatic constraints related to the “restaurant” scenery, or simply misconceptions. Feedback is provided after retrieving information related to the wrong answer, and a message consisting of two parts is generated: an explanation of the error in a first sentence and the right answer in the second sentence (see Figure 9).
5. EXTRACTING AND SUMMARIZING WITH GETARUN

In this section we shall present the use of GETARUN for the generation of short summaries (see Boguraev & Kennedy, 1997; Mani & Maybury, 2000). The system builds a semantic database of facts describing the entities of the world contained in the text(s) under analysis along with their properties and relations. This is achieved by tagging and shallow-parsing the tokenized text. The text is then transformed into a functional representation at the sentence level. This representation is passed on to the semantic module which is responsible for the creation of predicate-argument structures from verb subcategorization information (mainly derived from the lexicon made available by the University of Pennsylvania) and semantic features associated with each predicate in a big dictionary (derived from WordNet and Corelex). Main arguments are turned into referential expressions to be filtered by the anaphora resolution module which implements a slightly modified version of the centering algorithm. In this way, pronominal and nominal expressions are co-referred to their antecedents. Systems for information extraction rely crucially on the availability of structural counterparts of semantic entities which constitute the pivoting elements of their recognition task. In particular, recognition tasks may be ranked for difficulty along the following lines:

1. named entity recognition,
2. canned template matching, and
3. generalized relevant information extraction and summarization.

Whereas the first two tasks above may be dealt with by resorting to a certain number of heuristics and a good list of named entities of the relevant domain, the third task is solely dependant on the solution of the basic problems of:

1. recognition of clausal structure,
2. recognition of arguments from adjuncts, and
3. recognition of predicate-argument structures.

It is a fact that these three tasks are ineludible prerequisites for any type of summary generation if one wants to summarize texts with unlimited vocabulary. It is a well established fact that shallow parsing does not ensure carrying out these structural tasks smoothly; it does so only with a certain level of approximation. Therefore, we use GETARUN’s DM to check and compare semantic representations.

5.1 Using Automatically Generated Summaries for Essay Evaluation

One of the possible implementations of automatically generated summaries could be a task very similar to a multidocument summary generation: students are given a newspaper article which deals with a topic related to current local or international events, and they are told to write a summary on that topic by using information made available in the article. Since the summary has a length limitation which can be expressed in number of words, students will be obliged to use
some summarization strategy. The task specification will also enable students to use as many words or sentences taken from the text as they deem sufficient to convey the most relevant facts.

Student summaries have been evaluated by comparing automatically generated summaries with ones produced by students. At first, the comparison procedure tries to gauge the relevance of the text from the percentage of shared concepts and from their order of presentation. The GETARUN program produces a DM of the input text and of the student’s summary. The comparison is concerned with the semantic similarity between the two texts. Whenever a given concept is expressed with the same linguistic description, it is checked for its semantic interpretation. Semantic roles associated with this linguistic description, causal relations, and further inferential links with other concepts are analyzed in order to ascertain whether students have adequately understood the original text. All semantic relations are recovered from WordNet. According to the rate of overlapping information, a score is issued and then weighted. This produces a suitable means of evaluation.

We show here how the DM generated by GETARUN can be made to apply to the task at hand. The list of facts generated sentence by sentence is merged into a single list in which each entity is assigned a score according to whether it participated as main, secondary, or expected topic in the topic hierarchy. Every entity is listed with a semantic type, a semantic index, a score, and a list of facts. The result is the DM, part of which we show below. (T has been substituted for temporal index to improve readability).

(3) entity(ind, id3, 30, facts([fact(infon5, inst_of, [ind:id3, class:man], 1, univ, univ), fact(infon6, name, [john, id3], 1, univ, univ), fact(id5, go, [agent:id3, locat:id4], 1, T, id2), fact(id8, sit, [actor:id3, locat:id7], 1, T, id2), fact(id14, take_order, [agent:id13, goal:id3], 1, T, id2), fact(infon64, poss, [john, id3, id19], 1, id1, id2), fact(id20, read, [agent:id3, actor:id19], 1, T, id2), fact(id22, begin, [actor:id3, prop:id20], 1, T, id2)]).

... fact(infon29, part_of, [restaurant, id10, id4], 1, T, id2])).

The entity with the highest topicality score is John with semantic identifier id3. Students have to produce a short summary which deals with John by mentioning explicit, and also possibly implicit, relations like the fact that John sat at a table and that he ordered something. In case students mistakenly take the waiter to be the most relevant participant in the story and write about the waiter reading the book we can easily gather that they did not understand the text.

Students can make use of synonyms, hypernyms, hyponyms, meronyms, holonyms, and other relevant semantic relations. However, we would like to
stress the fact that here we are dealing with second language learners. Cohesion recovery can be accomplished by using one of the following four procedures:

1. pronominalization,
2. passivization with agent deletion,
3. relative clause formation, and
4. coordination and subject deletion.

All four procedures can be checked appropriately by the system. In particular, the system has been used with Italian students of English for Economics whose lexical knowledge is often very limited. As a matter of fact, students are told to use the original text as much as possible and to concentrate on reporting the most important facts and relations. The length of the summary must be less than 100 words. Our experiments with the system have given good results and we intend to make it fully automatic in the near future. At present, all decisions made by the system go through a screening phase during which a human tutor checks the automatically assigned scores.

6. CONCLUSIONS

The four exercises presented above have all gone through the preliminary experimental phase of software robustness testing in which an extended number of “crash” tests have been carried out in order to prevent the system from freezing or the web server from crashing. The results in terms of student reactions have so far demonstrated the validity of the system. The human tutors are currently working on improving and extending the gamut of feedback messages to be produced and presented to users. As to exercises themselves, we find that those bound to parsing performance and lexical information are well suited to their task, in particular because the architecture required is very simple: both parsing and information refer to a limited domain and written production. However, the exercises based on text understanding and summarization have by far larger importance: the first one has spoken input and the second one uses free text as input, even though semantic updating may take place before the system is actually applied to a text chosen by the human tutor. We are thinking of spoken interaction in the text understanding exercise by introducing a talking head who will be in charge of entertaining students with more extended dialogue exchanges. This will require using the automatic speech recognition module, called SLIM, currently available on the main system for self-instructional second language learning (see Delmonte et al., 1996; Delmonte (2000d).
NOTE

1 Word and constituent tags are to be interpreted as follows: cp = functional constituent where complementizers and interrogative and relative pronouns are taken; advp = adverbial phrase, that is, headed by an adverb—tagged “adv”; vsec = constituent marking the position for verb second or subject inverted structures; vmod = modal verb; fvsec = constituents following verb second; subj2 = constituent for the inverted subject; np= noun phrase; pers = personal pronoun; obj= object; clitdat = dative clitic pronoun; obj1 = another object constituent, that is, “np” or pronominal; art = article; adj = adjective; n = noun; ibar2 = sentence final nonfinite verb; punct = punctuation.

REFERENCES


AUTHOR’S BIODATA

Rodolfo Delmonte was appointed Associate Professor of Applied Linguistics at the University of Venice in 1985. He collaborated with the University of Parma for the implementation of an NLP system. Since 1993 he has been in charge of a course in Computational Linguistics. He was responsible for SLIM, a program for language learning sponsored by his university, and collaborated in a number of European projects promoting the use of speech technology in language learning. He was an invited speaker at EUROCALL2001. Lately, he has developed a number of NLP-driven CALL tools which are strongly linguistically based.
AUTHOR’S ADDRESS

Rodolfo Delmonte
Department of Language Sciences
Università Ca’ Foscari - Ca’ Garzoni-Moro
San Marco
3417 - 30124 Venezia
Italy
Phone:  +041-2578464/52/19
Email:  delmont@unive.it
WWW:  project.cgm.unive.it