Exploiting the Student Model to Emphasize Language Teaching Pedagogy in Natural Language Processing

Trude Heift
Linguistics Department
Simon Fraser University
Burnaby, BC, Canada V5A1S6
heift@sfu.ca

Paul McFetridge
Linguistics Department
Simon Fraser University
Burnaby, BC, Canada V5A1S6
mcfet@sfu.ca

Abstract

One of the typical problems of Natural Language Processing (NLP) is the explosive property of the parser and this is aggravated in an Intelligent Language Tutoring System (ILTS) because the grammar is unconstrained and admits even more analyses. NLP applications frequently incorporate techniques for selecting a preferred parse. Computational criteria, however, are insufficient for a pedagogic system because the parse chosen will possibly result in misleading feedback for the learner. Preferably, the analysis emphasizes language teaching pedagogy by selecting the sentence interpretation a student most likely intended. In the system described in this paper, several modules are responsible for selecting the appropriate analysis and these are informed by the Student Model. Aspects in the Student Model play an important pedagogic role in determining the desired sentence interpretation, handling multiple errors, and deciding on the level of interaction with the student.

Introduction

One of the fundamental problems of any Natural Language Processing (NLP) system is the often overwhelming number of interpretations a phrase or sentence can be assigned. For example, van Noord (1997) states that the Alvey Tools Grammar with 780 rules averages about 100 readings per sentence on sentences ranging in length between 13 and 30 words. The problem is not always improved with deeper analysis, for though a semantic analysis may rule some of the possible syntactic structures, it will introduce lexical and scope ambiguity.

The problem of resolving multiple interpretations is compounded in an Intelligent Language Tutoring System (ILTS) because the grammar must not only admit grammatical structures, but must also be able to navigate over ungrammatical structures and record the errors that the student has made. As a consequence, a grammar for an ILTS will not only assign structures to a grammatical sentence, but may also find analyses which interpret the sentence as ungrammatical, a set of analyses that a traditionally constrained grammar would not find.

The usual method of limiting the number of parses that an ILTS grammar assigns is to examine the effects of relaxing those constraints that represent likely sources of error by students and introduce new constraints into the grammar rules to block unlikely parses (Schneider & McCoy 1998). Such techniques, however, overlook individual learner differences as a key factor in language teaching pedagogy.

The system introduced in this paper differs from the traditional approach by permitting the grammar to freely generate as many parses as it can and using separate pedagogic principles to select the appropriate interpretation and response. The system tightly integrates the Student Model into the process of selecting the appropriate interpretation and generating a response tailored to the student's level of expertise. The Student Model keeps a record of students' performance history which provides information essential to the analysis of multiple parses, multiple errors, and the level of interaction with the student.

In the *German Tutor*, the ILTS described, the process leading to the creation of an instructional message in the event of an error has

three stages:

> (1) Given a forest of parse trees created by the grammar and parser, the parse most likely representative of the intentions of the student must be selected;
>
> (2) In the cases when the parse representing a student's intentions contains several errors, one of the error must be selected as the one that will be addressed. This step is necessary because empirical studies have found that reporting all the errors in a sentence is pedagogically inappropriate. For example, in evaluating her own system Schwind (1990) reports that "[s]ometimes, however, the explanations were too long, especially when students accumulated errors."[1];
>
> (3) Given an error, an instructional message must be constructed that is appropriate to the student's level of expertise and background.

In Section 1, the theory behind the grammar and its formalism is briefly discussed. Section 2 describes the process leading to the selection of a particular parse and how the Student Model participates in this process. We further discuss the pedagogic role of the Student Model in handling multiple errors and deciding on the level of interaction with the student. Section 3 presents conclusions and Section 4 looks at further research.

1 Design of the Grammar

1.1. Grammatical Formalism and Implementation

The grammar for the German Tutor is written in ALE (The Attributed Logic Engine), an integrated phrase structure parsing and definite clause programming system in which grammatical information is expressed as typed feature structures (Carpenter & Penn 1994).
The grammar formalism used is derived from Head-driven Phrase Structure Grammar (Pollard & Sag 1994). This theory is one of a family which share several properties. Linguistic information is presented as feature/value matrices. Theories in this family are to varying degrees lexicalist, that is, a considerable amount of grammatical information is located in the lexicon rather than in the grammar rules. For example, Figure 1 illustrates a minimal lexical entry for *geht*. The subcategorization list of the verb, notated with the feature subj, specifies that *geht* takes a subject which is minimally specified as a singular noun. Rules of grammar specify how words and phrases are combined into larger units according to the subcategorization list. In addition, other principles govern how information such as the head features, which inter alia determine the grammatical category of a constituent, is inherited.[2]

$$\begin{bmatrix} \text{phon} < \text{geht} > \\ \text{cat} \begin{bmatrix} \text{head } v \\ \text{subj} \left\langle \begin{bmatrix} \text{cat} [\text{head } n] \\ \text{content } \boxed{1} \left[\text{index} [\text{num sg}]\right] \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{content} \begin{bmatrix} \text{Reln geht} \\ \text{Geher } \boxed{1} \end{bmatrix} \end{bmatrix}$$

Figure 1 : Partial Lexical Entry for *geht*

Unification-based grammars place an important restriction on unification, namely that two categories A and B fail to unify if they contain mutually inconsistent information (Gazdar & Pullum 1985). However, this inconsistent information constitutes exactly the errors made by second language learners. For example, if the two categories A and B do not agree in number a parse will fail. To overcome this restriction, we relax the constraint on number agreement by changing its structure so that, rather than checking that the noun is singular, the system records whether or not the subject of *geht* is in the singular. To achieve this, the noun is no

[1] Schwind [1990a], p. 577.

[2] Inheritance in feature-value matrices is indicated by multiple occurrences of a coindexing box labeling the single value.

longer marked as [num sg], but instead the path numlsg terminates with the values error or correct. For example, for a singular noun phrase, the value of the path numlsg is correct, while it is error for a plural noun phrase. The two partial lexical entries are given in Figure 2(a) and Figure 2(b), respectively.

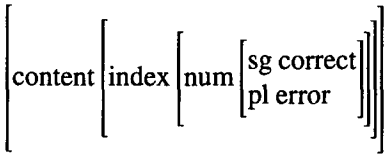

Figure 2a : Marking Number Features for Singular Nouns

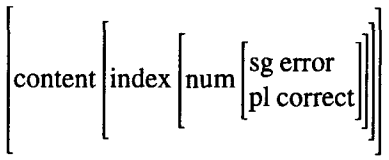

Figure 2b : Marking Number Features for Plural Nouns

The verb *geht* records the value of sg from its subject (Figure 3). If the value of the path numlsg is correct, the subject is in the singular. In case of a plural noun, *geht* records the value error for number agreement.[3]

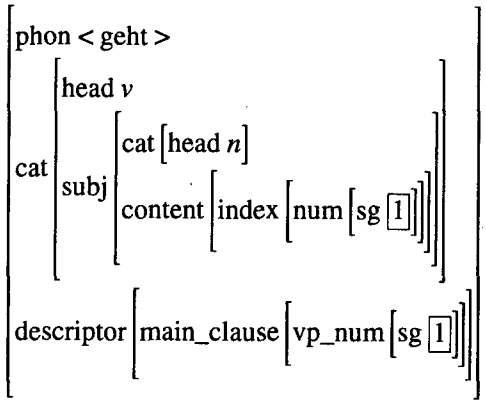

Figure 3 : Recording Number Features for *geht*

1.2 Phrase Descriptors

The goal of the parser and the grammar is the generation of phrase descriptors, each of which describes a particular grammatical constraint, its presence or absence in the input sentence and the student's performance on this constraint. Phrase descriptors correspond to structures in the Student Model and are the interface medium between the Student Model and other modules in the system.

A phrase descriptor is implemented as a frame structure that models a grammatical phenomenon. Each member of the frame consists of a name followed by a value. For example, subject-verb agreement in number is modeled by the frame [number, value] where value represents an as yet uninstantiated value for number. If the grammatical phenomenon is present in the student's input, the value is either correct or error depending on whether the grammatical constraint has been met or not, respectively. If the grammatical constraint is not present in the student's input, the feature value is absent. Consider examples (4a) and (b):

(4a) *Er gehen.

(4b) Er geht.

He is leaving.

The phrase descriptor for subject-verb agreement in number in example (4a) is [number,error], while that for the sentence in (b) is [number,correct]. For either sentence, (4a) or (b), the information will be recorded in the Student Model. A system presented with (4a), however, will also instruct the learner on the nature of subject-verb agreement in number.

In addition to the grammatical features defined in HPSG the grammar uses a type descriptor representing the description of the phrase that the parser builds up. This type is set-valued and is initially underspecified in each lexical entry. During parsing, the values of the features of descriptor are specified. For example, one of the members of descriptor, vp_num in Figure 3, records the number agreement of subject-verb in a main-clause. Its value is inherited from the sg feature specified in the verb *geht*. Ultimately, descriptor records whether the sentence is grammatical and what errors were made.

[3] For an analysis of errors in linear precedence, see [Heift 98].

2 The Role of the Student Model in Analysis and Feedback

The initial goals of the analysis of the results of parsing a student's sentence are selecting the appropriate parse and, from it, selecting the error (if there is one) that the system will focus on for instruction.

A parse of a sentence is a collection of phrase descriptors. For example, the phrase descriptor given in (5) indicates that the learner has violated subject-verb agreement in number in a main clause.

(5) [main_clause [vp_num [sg error]]]

The constraint that generated this descriptor has a correlate in the Student Model, in this case a record labelled vp_nummaincl. For each grammar constraint, the Student Model keeps a counter which, at any given instance in the evaluation process, falls in the range of one of the three learner levels, given in (6a) – (c).

(6a) novice: $20 \leq X \leq 30$
(b) intermediate: $10 \leq X < 20$
(c) expert: $0 \leq X < 10$

Initially, the learner is assessed with the value 15 for each grammar constraint, representing the mean score of the intermediate learner.[4] Once a student uses the system, the Student Model adjusts the counter of each grammar constraint accordingly. If a grammatical constraint has been met, the counter is decremented. If the constraint has not been met, the counter is incremented and, ultimately, a feedback message is displayed to the learner.

The result of the parsing process is a set of collections of phrase descriptors, each collection representing a separate parse. The step of winnowing this set down to a single collection is performed by a set of licensing conditions.

2.1 Selecting the Desired Parse

A sentence which is unambiguous in many other NLP applications can nonetheless result in multiple sentence readings in a system designed to parse ill-formed input. For example, consider a system which relaxes the constraint on grammatical case. Without case marking, the parser has no way of knowing if a constituent is a subject or a verb complement. As a result, more than one sentence analysis will be produced and the errors flagged in each sentence reading can vary. For instance, for the sentence *Sie liebt er* the parser can assign at least two legitimate syntactic structures. The two sentence readings are given in (7a) and (7b).

(7a) *Sie liebt er.
Sie liebt ihn.
She loves him.

(7b) Sie liebt er.
It is her he loves.

For the sentence *Sie liebt er*, *er* could be taken to be either the direct object or the subject of the sentence. Assuming that the choice between the two parses were arbitrary, sentence structure (7a) where *er* is the object of the sentence contains an error and would yield the feedback *This is not the correct case for the direct object.* In contrast, the alternative sentence reading given in (7b) where *er* is the subject of the sentence and the direct object *sie* is topicalized contains no errors.

The example illustrates two important points. First, an algorithm that selects the appropriate parse by counting the numbers of errors flagged in each parse and selecting that which has the least number of errors [as in Weischedel (1978), Covington & Weinrich (1991)] is inadequate. If the student is at an introductory level, the appropriate analysis is the sentence reading given in (7a), the parse that has more errors.[5]

The algorithm promoted here uses instead a set of licensing conditions to mark sentences for the rules that were used to parse them and select the appropriate sentence reading on the basis of the likelihood of a grammatical construction. The task of the licensing conditions in this example is to distinguish multiple sentence readings conditioned by word order. During parsing, three of the syntactic rules, the Subject-Head, the Head-Subject-Complement, and the Head-Subject rule each assign a distinct licensing descriptor. Any of the three licensing conditions

[4] The intermediate learner has been chosen as a reasonable default. While the messages might be initially too overspecified for the expert and too underspecified for the novice, they will quickly adjust to the actual learner level.

[5] Object topicalization is a rare construction and not explicitly taught at the introductory level where the focus is on the grammar of more commonly used rules of German.

can license a sentence. After parsing, the Licensing Module prioritizes multiple sentence readings so that, in the event of a choice, the parses are selected in a particular order. The chosen parse is passed on to the next module of the system for further processing.
The second point important to the current discussion is that it is not sufficient to exclude all other alternatives whenever they appear. An advanced student may be practicing object topicalization and in that case the system should preferably choose the alternative parse given in (7b).
This consideration illustrates the importance of the Student Model to the problem of sorting out a set of parses to find the intended interpretation. To provide input to the licensing conditions, we generate a figure representing the student's overall mastery of the grammar by averaging expertise levels on each grammar constraint in the Student Model. A threshold of expertise is set, below which the analysis in (7a) is preferred and above which that in (7b) is chosen.
After licensing, a single parse has been selected and a learner model update on each of the grammatical constraints present in students' input has been extracted. A single parse, however, can contain a number of errors and the system has to decide on how to communicate these errors to the learner. The following section will discuss the task of filtering multiple errors to select the appropriate one.

2.2 Filtering Multiple Errors

A further challenge in analyzing student input is presented by multiple errors. The sentence given in (8a) illustrates an example.

(8a) *Heute die Kindern haben gespeilt mit das Auto.

(8b) Heute haben die Kinder mit dem Auto gespielt.
Today the children were playing with the car.

In example (8a) the student made the following five errors:

1. word order: the finite verb *haben* needs to be in second position
2. word order: the nonfinite verb *gespielt* needs to be in final position
3. spelling error with the past participle *gespielt*
4. wrong plural inflection for the subject *Kinder*
5. wrong case for the dative determiner *dem*

From a pedagogical and also motivational point of view, a system should not overwhelm a student with instructional feedback referring to more than one error at a time. Little research has been done in Computer-Assisted Language Learning regarding the volume of feedback for different kinds of learners at different stages in their language development. However, van der Linden (1993) found that "feedback, in order to be consulted, has to be concise and precise. Long feedback (exceeding three lines) is not read and for that reason not useful."[6] She further states that displaying more than one feedback response at a time makes the correction process too complex for the student. The task for an Intelligent Language Tutor is to develop an error filtering mechanism that incorporates language teaching pedagogy. The sheer amount of feedback should not overwhelm the student. In addition, if feedback messages are displayed one at a time they need to be ordered in a pedagogically sound way.
To filter the possible errors, an Error Priority Queue is implemented. This queue takes student errors and selects the most important error. Criteria for selection can be set by the language instructor based on her knowledge of the difficulty of a grammatical construction, the likelihood of an error and/or the focus of the exercise.
However, the Student Model can also be invoked to rank errors. One criterion for ranking is the students' performance history as indicated by the Student Model: the grammar constraint most often violated will be reported first. The rationale for this criterion is that this grammatical property has been mastered the least and therefore needs the most attention.
After student errors have been ranked and the most important one has been selected, the system needs to generate instructional feedback messages to be displayed to the learner. This is

[6] Van der Linden [1993], p. 65.

achieved by an Analysis Module which will be discussed in the following section.

2.3 Generating Instructional Feedback

A further difficulty in ILTSs lies in framing instructional feedback to student input. In a typical student-teacher interaction, feedback depends on the students' previous performance history. Inexperienced students require detailed instruction while experienced students benefit best from higher level reminders and explanations (LaReau & Vockell 1989).

For instance, in example (9a) the student made an error with the determiner *einen* of the prepositional phrase. *Von* is a dative preposition and *Urlaub* is a masculine noun. The correct article is *einem*.

(9a) *Sie träumt von einen Urlaub.
(9b) Sie träumt von einem Urlaub.
She is dreaming of a vacation.

In the German Tutor, the Analysis Module generates instructional feedback of different levels of specificity. The pedagogical principle underlying this design is guided discovery learning. According to Elsom-Cook [1988], guided discovery takes the student along a continuum from heavily structured, tutor-directed learning to where the tutor plays less and less of a role. Applied to feedback, the pedagogy scales messages on a continuum from least-to-most specific guiding the student towards the correct answer.

For the error in example (9a), the system generates feedback of increasing abstraction that the instruction system can use when interacting with the student. The level of the learner, either expert, intermediate, or novice according to the current state of the Student Model, determines the particular feedback displayed. The responses, given in (10a) - (c) correspond to the three learner levels for the error in example (9a), respectively:

(10a) There is a mistake with the article *einen* of the prepositional phrase.
(10b) There is a mistake in case with the article *einen* of the prepositional phrase.
(10c) This is not the correct case for the article *einen* of the prepositional phrase. *Von* assigns the dative case.

For the expert, the feedback is most general, providing a hint to where in the sentence the error occurred (prepositional phrase). For the intermediate learner, the feedback is more detailed, providing additional information on the type of error (case). For the beginner, the feedback is the most precise. It not only pinpoints the location and type of the error but also refers to the exact source of the error (dative preposition).

The Analysis Module is implemented in DATR [Evans and Gazdar 1990], a language designed for pattern-matching and representing multiple inheritance. For each grammar constraint, the Analysis Module creates three categories of instructional feedback corresponding to the three learning levels. Provided with three categories of feedback, the system selects an error response suited to students' expertise. The student level is determined by the numerical value for each grammatical constraint maintained in the Student Model. Each value is adjusted each time the learner interacts with the system.

For example, the grammar constraint pp-dat records the student's performance on dative assigning prepositions. A learner who violates the constraint on dative prepositions will, at first, obtain the feedback message for the intermediate. If the student commits the same error in subsequent exercises, s/he will soon be assessed a novice. At this point, the system will display the more detailed feedback message suited to the beginner. However, each time the student applies the grammatical constraint correctly, the Student Model records the success. After demonstrating proficiency, the student will again be assessed as intermediate, or, even expert. Maintaining a large number of grammatical constraints allows for a very detailed portrait of an individual student's language competence over a wide-range of grammatical phenomena.

After instructional feedback for student input has been generated, the feedback message is passed to the Teaching Module. The Teaching Module interacts with the learner. It displays instructional feedback and, at the end of an exercise set, shows the student's performance history on each grammatical constraint. Students performance history informs learners and instructors of the grammatical construction the

student has mastered as well as the ones that require remedial work.

Conclusion

In this paper, we have described a Student Model that implements language teaching pedagogy in guiding the analysis of student input in an ILTS for German. The Student Model keeps a record of students' previous performance history which provides information essential to the analysis of multiple parses, multiple errors, and the level of interaction with the student.

For multiple parses, the system implements licensing conditions which select one of the possible parses by taking into account the likelihood of the error. The likelihood of an error is determined by the performance level of the student as indicated by the Student Model. For multiple errors, the system implements an Error Priority Queue which takes student errors and selects the most important error. Criteria for selection can be set by the instructor or evoked by the Student Model. The Student Model ranks errors with respect to students' performance history. Finally, by consulting the Student Model, the Analysis Module selects instructional feedback of different levels of specificity.

From a language teaching perspective, the system reflects a pedagogically informed, student-centered approach. System decisions are based on a dynamic Student Model rather than static computational factors. As a consequence, the learning process is individualized throughout the analysis of student input.

Further Research

The ILTS described in this paper has been implemented on the World Wide Web. While the system encompasses all the necessary grammar rules, we are currently expanding the lexicon for an introductory course of German.

Our immediate goal is to test the system with learners of German to assess accuracy. Long-term goals include expanding the Student Model. Student performance is only one criterion to individualize the language learning process. The native language of the student as well as different learning styles might also be key factors in the analysis of student input.

References

Carpenter, B., and Penn, Gerald. (1994) *The Attribute Logic Engine: User's Guide, Version 2.0.* Computational Linguistics Program, Carnegie Mellon University, Pittsburgh.

Covington, M.A., and Weinrich, K.B. (1991) "Unification-Based Diagnosis of Language Learners' Syntax Errors." *Literary and Linguistic Computing,* 6 (3): 149-154.

Elsom-Cook, M. (1988) "Guided Discovery Tutoring and Bounded User Modelling." *Artificial Intelligence and Human Learning.* Self, J., ed. Bristol: J. W. Arrowsmith Ltd.: 165-178.

Evans, R., and Gazdar, G. (1990*). The DATR Papers, Volume I.* Brighton, School of Cognitive and Computing Sciences. The University of Sussex.

Gazdar, G., and Pullum, G. (1985) "Computationally Relevant Properties of Natural Languages and their Grammars." *New Generation Computing.* 3: 273-306.

Heift, T. (1998) *Designed Intelligence: A Language Teacher Model.* Doctoral Dissertation, Simon Fraser University.

LaReau, P., and Vockell, E. (1989) *The Computer in the Foreign Language Curriculum.* Santa Cruz, CA: Mitchell Publishing.

Pollard, C., and Sag, I. (1994*) Head-Driven Phrase Structure Grammar.* Chicago University Press.

Schneider, D., and McCoy K. (1998) "Recognizing Syntactic Errors in the Writing of Second Language Learners." *Proceedings of the 17th International Conference on Computational Linguistics.*

Schwind, C. B. (1990) "An Intelligent Language Tutoring System." *International Journal of Man-Machine Studies,* 33: 557-579.

Van der Linden, E. (1993) "Does Feedback Enhance Computer-Assisted Language Learning." *Computers & Education,* 21 (1-2): 61-65.

Van Noord, G. (1997) "An Efficient Implementation of the Head-Corner Parser." *Computational Linguistics,* 23 (3): 425-456.

Weischedel, R.M., Voge, W.M., and James, M. (1978) "An Artificial Intelligence Approach to Language Instruction." *Artificial Intelligence,* 10: 225-240.