Arboretum: Using a precision grammar for grammar checking in CALL

Emily M. Bender[†], Dan Flickinger[*], Stephan Oepen[‡*], Annemarie Walsh[†], and Timothy Baldwin[*]

[*]Stanford University, Stanford, CA 94305 (USA)
[†]University of Washington, Seattle WA 98195 (USA)
[‡]Universitetet i Oslo, 0317 Oslo (Norway)

{danf|oe|tbaldwin}@csli.stanford.edu {ebender|awalsh}@u.washington.edu

Abstract

We present a tutorial system for language learners, using a computational grammar augmented with mal-rules for analysis, error diagnosis, and semantics-centered generation of corrected forms. Corrections are produced using a novel strategy of *aligned generation*.

1 Introduction

Developing fluency in a second language requires extensive practice, using the language in conversation, making mistakes and getting feedback on how to do better. In a classroom setting, this feedback can be quite direct, but necessarily limited in quantity, so most practice takes place outside the classroom where feedback is often less clear. Students of the language could thus benefit from a conversational partner with infinite patience, a precise mastery of the rules of the language, and the ability to gather the intent of students' error-prone utterances.

In this project, we have adapted the English Resource Grammar (ERG: Flickinger (2000)) and the LKB parser and generator (Copestake, 2002) into a prototype of an NLP engine for interactive grammar checking, robust for interpreting input, but precise when doing analysis and responding to the student. The ERG is a broad-coverage precision grammar for English. It compositionally relates semantic representations to surface strings, and as such, is suitable for both parsing and generation. The LKB includes an efficient Lisp-based parser and generator which interpret typed-feature structure grammars such as the ERG. Based on this technology, we have built a prototype system called Arboretum which is capable of producing well-formed semantic representations for ill-formed (erroneous) input, diagnosing the nature of the errors, and providing corrected forms. Crucially, in producing the corrected forms, the generator will not make spurious "corrections" of other parts of the input which may confuse the learner.

In the Arboretum system, we have augmented the existing ERG with mal-rules (Schneider and McCoy, 1998) to handle specific kinds of mal-formed input. These mal-rules serve to relate the erroneous input to well-formed semantic representations. We can then both generate from the well-formed semantic representations (with the mal-rules turned off) to produce corrected forms and diagnose the nature of the error based on the mal-rules which were used in the parse. To minimize spurious "corrections", we have modified the LKB chart generator so its search is guided by the input parse where possible.

In this paper we describe the mal-rules strategy and the types of mal-rules we encountered (§2) and the implementation of aligned generation as a kind of best-first generation (§3). We then evaluate the usefulness of the mal-rules strategy against the range and frequency of error types in an error-tagged learner corpus (SST: Izumi et al. (2003)) (§4) and evaluate the effectiveness of the aligned-generation strategy based on a sample of the corpus. Finally, in §6, we consider how the error detection problem can be addressed using existing stochastic parse selection techniques without requiring training treebanks of learner English.

2 Mal-rules: Types and Examples

Our goal in this project is to map ill-formed surface strings to well-formed semantic representations. Doing this leaves the grammar checking system ready to be embedded in an interactive dialogue system which can engage the language learner's interest and provide motivation and opportunities for the learner to practice.

In developing the prototype, we found that the mal-rules required fell into three classes:

- Syntactic constructions which parallel existing syntactic constructions but are subject to a different set of constraints (e.g., a rule which allows for determinerless NPs based on singular count nouns, rather than plural or mass nouns).
- Lexical rules which parallel existing lexical rules but match the wrong orthographic change to a given morphosyntactic effect (e.g., a rule which produces verb forms like *chases* morphosyntactically marked for non third singular agreement)
- Lexical items inheriting from a lexical type already in the grammar, but incorrect (in native speaker English) for the word in question (e.g., the use of *allow* in **we allow to travel*, c.f., *we allow traveling*).

For the evaluation described in §4 below, we merely added 4 syntactic construction mal-rules, 6 lexical mal-rules and 21 mal-lexical entries to the grammar, as well as 25 supporting mal-types (see examples in §3). In doing so, we strove to ensure that the additions to the grammar inherited any information they share with the standard rules they shadow from a common supertype, so that as the coverage of the core grammar is extended

In Proceedings of the InSTIL/ICAL Symposium: NLP and Speech Technologies in Advance Language Learning Systems, Venice, Italy.

and improved the mal-components remain functional and benefit from the extensions.

All of the mal-rules and mal-entries are flagged with a feature so that they can be selectively applied (for example, in parsing but not in generation) and so that the grammar-checking algorithm can discover where in the parse tree a mal-rule has been used. The latter is the key to error diagnosis.

3 Aligned Generation

The fact that our mal-rules map from ill-formed strings to well-formed semantic representations forms the basis of our solution to two of the problems outlined in §1: the understanding problem and the correction problem. On the first front, it brings our treatment of ill-formed input in line with our treatment of well-formed input, giving both consistent semantic representations which are suitable for further processing by a natural language understanding/dialogue system. On the second front, having normalized semantic representations allows us to simply generate from these and produce well-formed strings, essentially treating this as a machine translation problem.

However, the process of generating the corrected forms is complicated by the fact that there is ambiguity in generation as well as in parsing, i.e., multiple ways of expressing the same semantic concept. While one could conceivably pick the most probable of the generated forms based on some corpus of native speaker input, that strategy is not appropriate for this task: the corrected form should match the input in all ways except those affected by the correction, lest the learner be misled into thinking that they had also made an error with respect to one of the points of linguistic variation which leads to generator ambiguity (e.g., adverb placement, *that*-deletion, topicalization, alternative spellings of lexical items).

In order to address this concern, we developed a strategy for best-first generation which we call 'aligned generation'. In aligned generation, the goal is to generate the sentence which most closely matches some reference sentence in structure and lexical yield. If all of the reference sentences were well-formed, this would be trivial (one could in fact return the input string). In our case, however, the sentences which are well-formed according to the grammar used to parse (with mal-rules) will not be well-formed according to the grammar used to generate (without mal-rules), and so the problem becomes to find the closest match given that not all of the components of the input sentence are available.[1]

We implemented aligned generation in terms of a scoring strategy which assigns priorities to generation tasks in the LKB's agenda-driven chart generator (Carroll et al., 1999). The modified generator assigns one of four possible priorities to a given task, arranged highest to lowest as follows:

- Highest: For edges licensed by lexical entries, the same lexical item was used in the reference parse. Else, the edge (active or passive) to be built by the task corresponds to a configuration found in the reference parse. In this context, a configuration is a subtree (incomplete in the case of active edges) with rule identifiers as labels on the nodes.
- The edge (active or passive) to be built by the task is licensed by the same rule and will have the same lexical yield as some edge in the reference parse.
- The rule to be used in the task is used somewhere in the reference parse.
- Lowest: All other tasks.

The generator is working bottom-up, instantiating lexical entries based on the semantics given as input, and then building larger phrases over those lexical entries using lexical rules and phrase structure rules. At any given point, there are several tasks which can be undertaken, and the point of the strategies listed above is to guide the choices of the generator among the tasks so that the first complete tree it finds (covering the semantics of the input) is the closest one to the reference parse.

The first strategy considers local subtrees of depth one: either terminal nodes (lexical entries) or mother-daughter(s) configurations. If a lexical entry was used in the reference parse and is available to the generator it should be considered. In the case of non-terminal nodes, a task which corresponds to a subtree (partial or complete) is to be preferred if it matches a subtree found in the reference parse. This 'match' is calculated in terms of the rule and lexical entry identifiers (rather than particular features of the signs involved such as are represented in the syntactic category abbreviations N, NP, etc).[2]

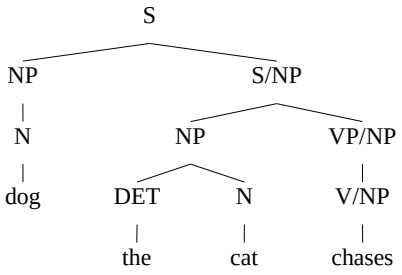
However, in many cases the presence of mal-rules in the parse will lead to local subtrees which cannot be matched by the generator. For example, the analysis of *Dogs barks* involves a lexical mal-rule for verbs which gives the form *barks* non-third-singular agreement. Thus the configuration corresponding to the topmost local subtree (S over NP and VP) has the daughters BARE_NP[3] and MAL_NON_THIRD_SG_FIN_VERB, while the corresponding subtree that the generator will try to produce will have the daughters BARE_NP and NON_THIRD_SG_FIN_VERB. In order to assign some priority to this configuration, we apply the second strategy, which values it because it has the same mother (SUBJH) and the same lexical yield (DOG_N1 and BARK_V1) as the configuration in the reference parse.

[1] We believe there are other possible applications of this strategy, as in dialogue systems where the computer's turns can be made to sound more natural by matching as closely as possible the user's turns.

[2] Neither strategy is absolute: It is possible for the generator to propose a local subtree at one point which happens to match a local subtree from elsewhere in the reference parse.

[3] Daughters are identified by the rule that licensed them. ERG rule names are cited in small caps.

The final strategy applies when neither the proposed daughters nor the lexical yield are found with the proposed mother in the tree. For example, the ill-formed sentence *dog the cat chases* is assigned the structure shown below. We correct this sentence to *A/the dog*



the cat chases, crucially leaving the topicalization even though it is in general dispreferred. When the generator considers the task of building the topmost subtree licensed by the rule FILLHEAD_NON_WH with daughters licensed by the rules HSPEC and SUBJH, this configuration is not found in the reference parse, as the left-hand daughter was instead built with BARE_NP_SG. Furthermore, FILLHEAD_NON_WH is not found with the same lexical yield in the reference parse as the corrected string has an additional determiner ('*a/the*'). Thus the third strategy assigns some priority (lower than the first two but still higher than other tasks) to this task since the rule used in the task is indeed found in the reference parse.

As discussed further in §5, there is still room for further refinements to this list of strategies. In particular, we do not yet use head-lexicalization in matching configurations, nor account for the ordering of modifiers (e.g., from *The big red barking dog finally left* we can generate *The red big barking dog finally left*). Here, we believe that heuristics based on the order of lexical items rather than on tree configurations will be more effective.

4 Evaluation of Mal-rules Strategy

Matthews (1993) and Menzel and Schroeder (1999) argue that adding mal-rules to a precision grammar is an ineffective way to approach the problem of grammar checking for CALL, because the range of learner errors is too broad, making it infeasible to anticipate every possible error combination. They advocate instead the more widely-used technique of developing algorithms for relaxing constraints in the grammar until a parse is found (see also Granger et al. (2001), Heinicke et al. (1998), Khader (2003) and Vandeventer (2001)). While a mal-rules approach will never be fully robust, we believe that it is nonetheless interesting, for two reasons: (i) The increased precision afforded by adding particular mal-rules rather than removing constraints on existing rules increases the utility of the grammar as a component of a dialogue-based language tutoring system. (ii) Constructing a system which can accurately and helpfully diagnose some common errors already benefits language learners. Indeed, there are error types such as word choice in both closed class cases such as *a* v. *the* and open class cases such as (1) which we believe are in principle beyond the scope of either a mal-rules or a constraint-relaxation approach. Characteristically, these errors involve sentences which are grammatical but not 'idiomatic'. From the language learner's point of view, however, the error types are not necessarily distinct.

(1) Because, you know, yesterday was Valentine's Day, and I had a *time* (corr: *date*) with my girlfriend.

Therefore, the question is not whether the mal-rules strategy scales up to complete coverage, but whether it scales up to covering a high enough proportion of error tokens to be helpful to the language learner.

The most common error type in the SST corpus is determiner errors (27%), and two of the three subtypes of this error (extraneous determiner and missing determiner) are plausibly handled by a the mal-rules approach. Similarly, the noun number errors and many of the verb tense errors (8.1% and 8.8%) are within reach. 11 of the 15 next most common types involve lexical choice, but the other four (verb agreement 3.7%, verbal complements 3%, missing auxiliaries 1.5% and verb form errors 1.2%) are manageable.

5 Evaluation of Aligned Generation

We evaluated the effectiveness of the aligned generation strategy by randomly selecting a sample of 221 items from the SST corpus, representing the error types we attempt to handle in our prototype and balanced to reflect the proportion of those errors over the whole corpus, including both single-error and multiple-error sentences.

Of the 221 items, we were able to parse 195. Hand-inspection of the parses indicated that 178 items were assigned a correct parse. Thus for 80.5% of the sample, the parses assigned by the grammar (including mal-components) included an appropriate one. We treebanked those successful parses to serve as input for generation. The generator was able to produce a string for 167 of its input items, or 93.8%. On the remaining items, either the input semantics was incompatible with the grammar without mal-components[4] or the generator ran out of edges before finding a string.

The 167 items we were able to generate for can be broken down according to whether or not they required any change (no change was required if the string was in fact grammatical, if perhaps infelicitous in context, or the error was one we cannot catch, despite being classified in the corpus as the same type as others which we can) and whether or not the current system generated multiple strings for the item. These results are shown in Table 1.

[4] E.g., extraneous determiners, the mal-rule for which creates a semantic representation which needs further correction before it can be well-formed. We anticipate handling such cases by proposing a 'transfer' component which operates on the semantic representations.

		Strings generated		
		1	> 1	total
correction needed	yes	74	70	144
	no	13	10	23
	total	87	80	167

Table 1: Generation by ambiguity & need for correction

Where there was no correction needed (i.e., the mal-rules didn't fire), we returned the input string in all but four cases. All four cases (two ambiguous, two not) involved the generator systematically not producing the correct output because it currently won't generate contracted forms or discourse particles.

Of the 74 cases where some correction was needed (i.e., a mal-rule fired) and the generator found only one string, 10 had a spurious change in addition to the required change. Seven of these are related again to contraction and discourse particles, one to a misapplied mal-rule, and three others to an apparent treebanking error.

Finally, of the 70 cases where mal-rules fired and there was a choice of output for the generator, 22 cases showed spurious changes. 12 of these involved failure of the aligned generation strategies (sometimes in combination with grammar or generator failures). In the other 10 cases, the generator was unable to supply a string without spurious changes. One of these had to do with the generator not producing certain forms (e.g., contractions, discourse particles). The remainder were effectively treebanking errors: either the wrong parse was chosen, or all of the parses should have been rejected.

The aligned generation failures fell into two main classes: failure to distinguish between lexical entries and failure to distinguish between phrasal tasks (largely involving adjunct placement). Both of these failures can be potentially addressed by more sophisticated propagation of edge priorities. In the current system, the priority of an edge is independent of the priorities of its daughters. Since all lexical entries are retrieved and all lexical rules are applied before any phrase structure rules are attempted, lexical priorities are effectively lost, and phrasal priorities can get masked in sufficiently large structures.

Despite these drawbacks, aligned generation is performing well above baseline. Our sample included 60 examples requiring corrections and involving ambiguity where the generator was able to return a correct string. A baseline strategy of choosing randomly among these alternatives would be expected to produce the right string (i.e., a string without spurious corrections, assuming that there is only one such per generated set) 34% of the time. Aligned generation did so 80% of the time.

6 Conclusion and Future Work

Our semantics-centered approach to the analysis and correction of errors produced by language learners enables our prototype implementation of a tutorial system to present corrected utterances with high precision, and pointed us to next steps in this research, especially the task of choosing correctly among several alternatives.

The problem of error detection in this framework is closely related to a problem of parse selection, since parses for a string with and without mal-rules will be competing. Current state-of-the-art techniques for parse selection with unification-based grammars involve training statistical models over annotated corpora (see e.g., Oepen et al. (2002)). We expect to take advantage of this technology without needing a treebank of learner English, because our system produces normalized semantic representations. We will train a semantics-based parse selection model on an existing native-speaker corpus, and use it for parse selection on learner English.

Acknowledgments

This work was supported by a generous research grant from MediaX at Stanford University.

References

John Carroll, Ann Copestake, Daniel Flickinger, and Victor Poznanski. 1999. An efficient chart generator for (semi-)-lexicalist grammars. In *Proc. of the 7th European Workshop on Natural Language Generation*, pages 86 – 95, Toulouse.

Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford, CA.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):15–28.

Sylviane Granger, Anne Vandeventer, and Marie Jose Hamel. 2001. Analyse des corpus dapprenants pour lelao bas sur le tal, linguistique de corpus. *TAL*, 42 (2):609–621.

Johannes Heinicke, Jurgen Kunze, Wolfgang Menzel, and I. Schroder. 1998. Eliminative parsing with graded constraints. In *Proc. Coling-ACL'98*, pages 526 – 30, Montreal.

Emi Izumi, Toyomi Saiga, Thepchai Supnithi, Kiyotaka Uchimoto, and Hitoshi Isahara. 2003. The development of the spoken corpus of japanese learner english and the applications in collaboration with nlp techniques. In *Proceedings of Corpus Linguistics 2003 Conference*, pages 359 – 366.

Isa Khader. 2003. Evaluation of an English LFG-based grammar as error checker. Master's thesis, University of Manchester Institute of Science and Technology.

Clive Matthews. 1993. Grammar frameworks in intelligent CALL. *CALICO*, 11 (1):5–27.

Wolfgang Menzel and Inge Schroeder. 1999. Error diagnosis for language learning systems. *Special edition of the ReCALL journal*, pages 20–30.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Chris Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods treebank. Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan.

David Schneider and Kathleen McCoy. 1998. Recognizing syntactic errors in the writing of second language learners. In *Proc. of Coling-ACL'98*, pages 1198 – 1204, Montreal.

Anne Vandeventer. 2001. Creating a grammar checker for call by constraint relaxation: a feasibility study. *ReCALL*, 13 (1):110–120.