Abstract

This paper describes an implemented online English grammar checker for students of English as a second language. This system focuses on a limited category of frequently occurring grammatical mistakes in essays written by students in the English Language Programs at the University of Pennsylvania. The grammar checker exploits the syntactic domain of locality from a Combinatory Categorial Grammar for the purpose of identifying specific types of grammatical mistakes as well as accepting grammatical expressions. It includes grammatical mistakes as ungrammatical variations of the constituents that can be related to given lexical entries in a categorial lexicon. The grammar checker is developed using one set of essays and tested against another set of essays. Unpredicted grammatical mistakes are either incorporated into the next revision or left as they are, depending on the difficulty and desirability of detecting them. The system also provides an interactive Internet interface for convenience of use. Finally, we discuss issues of constructing a full-scale English lexicon for an educational domain.

1 Introduction

In this paper, we describe an implemented English grammar checker intended as an aid for students who are learning English as a second language. While the grammar checker does not attempt to handle all possible grammatical errors, it identifies certain specific types of grammatical mistakes in the proposed domain of application that are more regular than others. The grammar checker itself utilizes a version of Combinatory Categorial Grammar written in SICStus Prolog, and the user interface is given as an Internet page written in HTML/CGI, which interacts with the run-time image of the Prolog code and the categorial lexicon.

Grammar checking is a natural part of natural language processing, whose primary goal lies in dealing with everyday expressions, including not only grammatical sentences but also ill-formed ones. Existing grammar checking systems, such as those described in Thurmai (1990), Bololi et. al. (1992), Genthial & Courtin (1992), Bustamante & León (1996), fall into this discipline, addressing the issue with a collection of heuristic rules that approximate a natural language grammar. This paper addresses an implementation of grammar checking as a serious application of computational linguistics, whose primary focus in syntax is on identifying what constitutes grammatical expressions, using a Combinatory Categorial Grammar as an underlying framework and confining the domain of application to the said setting. The implemented grammar checker accepts English expressions written by students who are learning English as a second language.

This domain of application also shows interesting aspects of the system that have not been previously discussed in the literature. For instance, consider the sentence below.

(1) I want leave.

The English word leave can function as an NP as in “I took leave of her with a heavy heart,” in addition to the more common uses of intransitive and transitive verbs. The sentence is therefore on solid grammatical grounds. In particular, a general purpose grammar checker should not complain. On the other hand, if this sentence appears in an essay where the author shows evidence of restricted knowledge of English grammar, it is very likely that the sentence is simply missing the infinitive marker to from the sentence I want to leave. In this particular essay, it is clear that a grammar checker treating the original sentence as grammatical is not very helpful to the author. It is also clear that decisions of this kind must be made with respect to the actual data, or essays written by students in the proposed application.

The rest of this paper is structured as follows. First, we present the data analysis of grammatical
mistakes that show up in the essays written by students who are learning English as a second language (§2). We then present the grammar formalism, or a version of Combinatory Categorial Grammar, that can identify grammatical mistakes as well as grammatical expressions (§3). We show the organization of the categorial lexicon for the present application in §4 and the architecture of the implemented grammar checker in §5.

2 Grammatical Mistakes

The present project is originally conceived as part of a collaboration with the English Language Programs (ELP) at the University of X in an effort to provide a computational tool for students who are learning English as a second language. Although students are assigned to a certain level of class according to their performance on a battery of placement tests, including a writing test, in writing they may show different profiles due to their previous writing experience, the stage of second language acquisition, their native language, and/or their previous learning experiences (Raines, 1985; Brooks, 1985; Arndt, 1987).

While teachers are perfectly capable of identifying and correcting such grammatical mistakes, current approaches to second language learning hold that learning will be more effective if students have a more active role in the correction process, for example by trying to correct errors that have been identified. The grammar checker that is developed here is expected to help teachers to focus on a higher level of writing problems, such as the proper use of transition words or the semantic properties of words or the organization of the argument, rather than more or less mechanical and purely syntactic mistakes.

Even when we confine our attention to syntactic mistakes, it will be impractical to expect the complete coverage of possible grammatical errors. An analysis of actual essays written by students shows, however, that particular classes of errors dominate, although there are still limited cases where the “sentences” are apparently a word salad. This section illustrates some of the dominant kinds of grammatical mistakes.

The students in a given writing class at ELP are regularly asked to write various essays on a specific topic. The set of 9 essays that we initially addressed in the project were concerned with students’ role-taking reactions to the film, “Children of a Lesser God,” with introductory comments in a test book (Mejia, Xiao, Pasternak, 1992). On average, each essay contains about 19 sentences (and 15 words per sentence), though the presence of many run-on sentences renders the average number of sentences lower than it actually is.

The essays show grammatical mistakes in the uses of verb forms, noun forms, articles/determiners, relative pronouns, agreement between two constituents, coordination, transition words, pronouns, modifiers, and prepositions, along with missing fragments, run-on sentences, wrong choice of words, wrong expressions, and wrong paragraph boundaries. It is beyond the scope of a purely syntax-based grammar checker to identify some of them, such as wrong transition words, wrong choice of words, or run-on sentences.

In the following sentences, the problematic fragments are underlined and missing fragments are shown with matching square brackets.

(2) (a) He just only know how to communicates with them.
(b) I suppose the person who are in the same situation as I am will give more attention to me and do anything well for me.
(c) Generally, everybody have [a] right to speech and write or anything under the law.
(d) ... if someone who are deaf run to be a president, ...
(e) There are a lot of people who are suffered from different kinds of pains ...
(f) I think that the most important thing to cure patients is the belief that can be recovered.
(g) But it’s significant to give them a opportunity.

3 Grammar Formalism

In the present implementation of the grammar checker, we have exploited the syntactic domain of locality implicit in categorial lexical entries under the combinatorial categorial grammar (CCG) framework.

CCGs are a family of grammar formalisms that provides a straightforward account of “difficult” linguistic constructions such as non-standard coordination, as well as well-understood linguistic constructions, using lexical combination of local information. A brief sketch of a version of unification-based CCG will be shown in §3.1. In §3.2, we show how to modify the local information in a CCG to accept ungrammatical inputs with a tagging message.

3.1 Combinatory Categorial Grammar

Categorial Grammars, or CGs, are a class of grammar formalisms, originally proposed by Ajdukiewicz and further developed by Bar-Hillel. The reader is referred to Wood (1993) for a general introduction to CGs. CGs encode syntactic information in a categorial lexicon, where each lexical entry specifies how the corresponding lexeme is interpreted syntactically. In the following sample lexical entries, the operator ‘:=’ connects lexemes and categories.

(3) (a) a := np/n
(b) man := n
(c) stays := s\np

(3) (a) encodes the fact that the article ‘a’ is syntactically a constituent of category np, or a noun
phrase, provided that it is combined with a constituent of category n, or a common noun, on its right. Note that the category np/n applies to all articles and determiners. A constituent of category X will henceforth be referred to as simply X for convenience of presentation. For instance, ‘a man’ is analyzed as np since ‘man’ is n. Similarly, according to (3) (c), ‘stays’ is an s (or a sentence) provided that it is combined with an np on its left. Again, the category s\np applies to all intransitive verbs. As such, the directional symbols, ‘\’ and ‘/’, have the following distinct interpretations.\1

(4) (a) Function Application (forward, >)

\[
\frac{X/Y \ Y}{X}
\]

(b) Function Application (backward, <)

\[
\frac{Y \ X \ Y}{X <}
\]

The following derivation shows how the sentence “A man stays” is understood as such by a CG.

(5) \[
\frac{A \ man \ stays}{\frac{\frac{np/n \ n \ s\np}{n \ np \ s <}}{snp}}
\]

The derivation np/n n => np is achieved by repeatedly replacing the values np and n with the patterns X and Y in the rule >, where the pattern Y is unified with the value n, and the pattern X with the value np.\2 For further details of unification, the reader is referred to Shieber (1986).

There are a fixed number of elementary categories, such as s, np, and n, and categories are defined recursively as the smallest set that contains elementary categories and categories separated by a directional symbol. The following shows another derivation for the sentence “A man will stay.”

(6) \[
\frac{A \ man \ will \ stay}{\frac{\frac{np/n \ n \ (s\np)/(s\np) \ s\np}{n \ np \ (s\np)/(s\np) \ s\np \ s \ np \ s <}}{s \ np}}
\]

The following lexicon shows how features are utilized in a categorial grammar framework to accept the sentence a man will stay, but reject the sentence a man will stays, among others.\3

(7) (a) a := np(sg, Case, third)/n(sg)
(b) man := n(sg)
(c) will := (s(fu)\ np(Num, subj, Deg))/ (s(rt)\ np(Num, subj, Deg))
(d) stay := s(rt)\ np(Num, subj, Deg)
(e) stays := s(pr)\ np(sg, subj, third)

In particular, the categories s(rt) and s(pr) are not unifiable and the fragment ‘will stay’ does not have a grammatical category.

Combinatory CGs, or CCGs, extend the purely applicative CGs described above to include a limited set of combinatory rules corresponding to combinators such as type raising T, function composition B, function substitution S, etc., for the combination of two adjacent categories. The reader is referred to Park (1995) or Steedman (1997) for further discussion. While we will not make use of the distinguishing characteristics of CCGs as opposed to purely applicative CGs in the following discussion, we should point out that they are essential in dealing with sentences that contain non-standard coordination and many other “difficult” constructions such as parasitic gaps (Steedman, 1990).

3.2 Relaxed Local Domain

Each entry in the categorial lexicon defines a particular local domain in which the given lexeme operates. For instance, consider the entry (8) for the article ‘a’ intended for subject NPs.

(8) a := (s(T))/n(np(sg, subj, third))/n(sg)

This entry shows that ‘a’ is a constituent which expects a constituent of category n on its right to further expect a constituent of category s\np, i.e., a verb phrase (VP), to result in a sentence category. In particular, it is not fussy about the internal structure of the VP it ultimately takes: The VP can contain an intransitive verb or a transitive verb with or without modifying prepositional phrase, and so on, as far as the article is concerned. What is expected of the VP by the article, however, is that whatever verb form it contains must be compatible with a subject NP which is a third-person singular form. If the VP violates this expectation, there is no successful derivation, unless of course there is another entry that will tolerate such incompatibility. The entry also requires that the subsequent noun must be singular, ruling out a noun phrase such as ‘a men’.

The following shows our proposal for making use of this “local” information in dealing with ungrammatical sentences. Let us first start with lexical entries for the transitive verb leave. The following lexical entries show some of them, using the user-defined Prolog predicate cat/2 that associates the lexical head with its category.

(9) cat(leaves, (s(pr)\ np(sg, subj, third))/np(sg, obj, _))

For instance, the entry (9) can be used to accept the sentence “A man leaves the office” since ‘a man’ is
type of a third-person singular (subject) NP. The features with the anonymous variable _ are meant to be unrestricted, in the sense that any feature can be unified with it. Since each elementary category such as np or s has a fixed number of predefined features at predetermined argument positions, those features that will be unified with such a variable are strictly bounded.

Consider now the ungrammatical sentence “A man leave the office.” If the goal were to just accept the sentence as well, we could simply add another lexical entry as follows to the existing lexicon.

(10) cat(leave, (s(pr)\np(sg,subj,third)) /np(_,obj,_)).

Since we want not only to accept the sentence but also to distinguish it from the rest of the grammatical sentences, we also need to label the resulting derivation as having an ungrammatical instance of the verb form. For this purpose, we can change the format of the lexicon so that each lexical entry contains two arguments, where the second argument is now a pair, C:M, in which C is a category and M is an associated list of error messages. (11) shows an example.

(11) cat(leave, (s(pr)\np(sg,subj,third)) /np(_,obj,_):
    [leave:’wrong subject-verb agreement’]).

For consistency of the format in the lexicon, we can modify grammatical entries to contain an empty list, or [], as the list of error messages. The parser should also be modified to collect all the error messages, in addition to applying rules of combination to adjacent categories. Since the parser will check the sentence against lexical entries one by one until a successful parse is found, we need to place such entries for ungrammatical inputs after those for grammatical ones.

As explained in the following section, our grammar checker utilizes a large-scale CCG lexicon. It is therefore essential that we organize the lexical assertions in a modular way. For this purpose, we have clustered all the assertions related to a transitive verb, grammatical or not, into one Prolog clause, and those related to an intransitive verb in another, and so on. Thus the actual assertions related to the transitive verb leave are made in terms of a call to the single Prolog clause tv(leave, leaves, left, left, leaving).

Here are the error messages that we can associate with lexical entries for transitive verbs:

(12) (a) wrong subject-verb agreement
    *He leave the office.
(b) missing object (see the discussion below)
    *He left.
(c) missing subject
    *Left the office.
(d) wrong verb form (see the discussion below)
    *He leaving the office.

As for (12) (a), notice that the sentence “I demand that he leave the office” does not receive this error message, since the verb demand induces subjunctive mood as follows and leave is not the matrix verb.

(13) cat(demand, (s(pr)\np(sg,subj,first)) /sb(rt)),
    cat(that, sb(Tense)/s(Tense)).

When the lexical entry in question, such as leave or left, does not have other parts-of-speech, including intransitive use, the error message (12) (b) is what we should generate for sentences with missing object NP, as exemplified below.

(14) cat(left, s(pa)\np(_,subj,_):
     [left:’missing object’]).

Unfortunately, when the verb has many parts-of-speech, the resulting analysis will be sensitive to the relative order of lexical assertion. Since left is also an intransitive verb, we certainly do not want to place the following entry after (14), lest the parser should generate the incorrect error message.

(15) cat(left, s(pa)\np(_,subj,_):[]).

One way of achieving this goal is to make all the assertions regarding grammatical parts-of-speech, before making those regarding ungrammatical ones.

In order to generate the error message “missing subject” for the fragment (12) (c), we need lexical entries such as the one shown below.

(16) cat(left, s(pa)\np(_,obj,_):
     [left:’missing subject’]).

We should not have such lexical entries for base forms though, if we also need to take care of imperative sentences with the missing pronoun you. Notice that the entry (16) does not interfere with the correct interpretation of the following sentence, where the relative pronoun which is heading the subject-less clause ‘left the office’.

(17) I am the person who left the office.

The reason is that the relative pronoun is of category (n\n)/s\np, that is, expecting a constituent of category s\np on its right, whereas the clause will have the wrong category s according to the entry (16). The correct category for the clause, s\np, does not have an associated error message, as expected.

---

4We have thus made a choice to dispense with an explicit module for morphology.

5We put the asterisk mark before an ungrammatical sentence following the tradition in linguistics.

6In the present implementation, we chose to utilize two different predicates for lexical entries, cat/2 and ucat/2, for fast prototyping and in favor of the modularity suggested earlier.
(12) (d) assumes that the author used the progressive form ‘leav[ing]’ incorrectly as a fully tensed verb, such as for present or past progressive tense.

4 Categorial Lexicon

In constructing a categorial lexicon for the project, we have made use of an existing grammar-independent lexicon that had been made available as part of the XTAG project (XTAG, 1995). The original public domain lexicon contains about 37K quintuples, \((\text{INDEX, ENTRY, POS, CAT, FS})\), where \(\text{POS}\) and \(\text{CAT}\) are associated with a part-of-speech (such as \(V\) or \(N\)) and a set of categories, respectively, for the lexical item associated with \(\text{ENTRY}\). The lexical items are supplemented by a morphology table, which has about 317K entries for various grammatical inflections. We have also made use of public domain word lists and consulted an on-line electronic dictionary called noah,\(^7\) for more commonly used lexical items and their parts-of-speech information.\(^8\)

At the moment, the lexicon has about 18K nouns, 6.5K adjectives, 4.5K transitive verbs, 2K intransitive verbs, among other categories. In a typical natural language processing application, we would expect that a larger lexicon almost always serves better than a smaller one. There is however an interesting twist as to the desired coverage of the lexicon in the present application.

We have discussed in §1 that the noun use of \(\text{leave}\) interferes with the ability of the grammar checker to identify potential grammatical problems. Continuing the discussion, notice that the verb \(\text{leave}\) has not only \(\text{left}\) as a past form but also \(\text{leave}d\) as another, when it means, intransitively, to put forth foliage. When the lexicon has the two past forms in its command, as our lexicon did at one point, the following sentence is taken as grammatical.

\((18)\) They leaved again.

Notice that, together with the noun use of \(\text{leave}\) this reveals the limitation of the syntax-alone approach to grammar checker, since properly enhancing the system with a module of semantics and pronoun reference resolution, among others, should be able to see whether the past form \(\text{leave}d\) is acceptable or not in the sentence \((18)\).\(^9\) Compare the sentence with the following, which is perfect.

\((19)\) The plants in our garden leaved very early this year.

Lacking such a module, we believe that it is appropriate to reject the sentence \((18)\), as a safety measure for students who are learning to practice English conjugation.

The present system accepts the sentence \((20)\) (a), but rejects the sentence \((20)\) (b). The first three occurrences of \(\text{can}\) are analyzed to have different parts-of-speech as common noun, auxiliary verb, transitive verb, in that order. The lexical item \(\text{can}\) is a perfect NP, but \(\text{can}\) is not.

\((20)\) (a) One can can can cans.

(b) *One can can can can.

This might be a fancy demonstration of the system’s ability to understand different parts-of-speech of simple words such as \(\text{can}\), but there is a downside to it. Consider the following sentences. For a syntax-alone grammar checker with an adequate lexicon, they are surprisingly all grammatical.

\((21)\) (a) They will to go.

(b) They will likes.

(c) He wills go es.

(d) A must comes.

In the sentences (a) through (c), \(\text{will}\) (or \(\text{wills}\)) could have been used as a transitive verb, and the words \(\text{likes}\) and \(\text{goes}\) as NPs. Unless we augment the purely syntactic grammar checker with a more elaborate feature system, any NPs with a proper object case must be able to function as object NPs. In (d), a \(\text{must}\) is understood as an NP. Depending on the level of English proficiency of the author in question, accepting all of them as grammatical does not appear helpful. (a) may contain the extra infinitive marker ‘to’; (b) may contain a wrong transitive verb form for \(\text{like}\) along with a missing object NP; (c) may have the incorrectly inflected auxiliary verb ‘\(\text{will}\)’ and intransitive verb ‘\(\text{go}\)’; and (d) may have a missing noun after ‘\(\text{a}\)’, along with an incorrectly inflected verb ‘\(\text{come}\)’ after the auxiliary ‘\(\text{must}\)’.

A possible purely syntactic approach to handling such personal differences in commanding lexical knowledge would be to identify those advanced entries as such in the lexicon and to inform the students that they may have made grammatical mistakes whenever such an advanced usage is attempted. In the present implementation, however, we have decided to simply take out those advanced entries from the lexicon, with an understanding that advanced students will not be bothered by consequent error messages but that unwary students will benefit from such warnings. This approach of restricting a big lexicon has an advantage over writing it from scratch, since such advanced uses of basic lexical items are comparatively rare.

\(^7\) Noah is American Heritage Electronic Dictionary, copyrighted in 1991 by Houghton Mifflin Company.

\(^8\) We predict that we could make use of the comlex dictionary (Grishman, Macleod, and Meyers, 1994) to fortify the categorial lexicon.

\(^9\) The on-going discussion should not be interpreted to indicate that CCG is not compatible with such a module. CCG is in fact understood to offer a unifying grammatical framework that encompasses the full range of natural language phenomena, from intonation, syntax, and semantics to discourse analysis. The reader is referred to Steedman (1997) for a related discussion.
5 English Grammar Checker

The English Grammar Checker is implemented in SICStus Prolog\textsuperscript{14} that runs under the Unix system. The first version was implemented to specifically address the grammatical mistakes in the original set of 9 essays. After emacs-based interactions using X windows with volunteer beta testers, the system detects the following kinds of mistakes: Wrong capitalization (sentence initial, wrong lowercase/uppercase initial letter), missing fragments (subjects, objects, some prepositions, complements, articles, clauses, the, than, etc), some extra elements (e.g., the infinitive marker after auxiliary verbs), wrong agreement (number, case, etc), wrong verb form, and various mismatches (verb tense with adverbs, etc).

We then received a new set of 8 essays, written by different students at ELP. This new set, unlike the earlier one, shows relatively shorter essays on various topics, such as comparing two countries, describing daily business, and so on. The essays contain 11 sentences on average. The grammar checker, tested against this new set, made 100 comments. Among them, all grammatical sentences (27 of them) were acknowledged as such. Three ungrammatical sentences were incorrectly accepted as grammatical, due to the system's oversimplified grammar for prepositions. Correctly detected grammatical mistakes included mismatches between adverbs and verb tense (3), wrong subject-verb agreement (6), wrong or missing article/determiner (5), missing fragments (3), extra elements (2), spelling errors (13). Most of the spelling errors included proper nouns, such as city or personal names.\textsuperscript{11} There were 3 misdiagnoses, rejecting “about weekend [on weekends]” as missing an article, “which is relation with my major [which is related to my major]” as missing an article, and “I visited to supermarket [I visited a supermarket]” as missing an article. There were 25 unexpected syntactic forms, including complete word salads, wrong prepositions, multi-word prepositions, compound nouns, enumeration, wrong adverbial phrases, phone numbers and addresses, idioms, wrong verb forms, determiner coordination, and so on.

Interface: The user interface of the system includes a direct interaction with the Prolog interpreter, as well as an Internet interface. The Internet interface is written in HTML and CGI/PERL, which invokes the run-time image of the Prolog code. The architecture of the Internet grammar checker is shown in Figure 1.

\textsuperscript{10}Copyrighted in 1995 by SICS (Swedish Institute of Computer Science), Sweden.

\textsuperscript{11}While all of the sentences containing spelling errors were determined to be ungrammatical, most of them are otherwise syntactically acceptable and would have been determined as such by the grammar checker.

\textsuperscript{12}The window image is drawn only for electronic submission purposes. The real image will be provided.

Figure 2 shows an example window in which a tester has interacted with the grammar checker with four sentences shown in (21). During a light system load, this particular sequence of sentences takes about 2 seconds (wall clock) for the response.

The initial reaction to the grammar checker is encouraging. In particular, the feedback from students is that they appreciate the chance of making grammatical changes in their essays before handing them in for further evaluation. There are still problems of making the parser faster for better performance and extending the coverage of predicted mistakes, however, and it appears that it is more important to address them correctly than to attempt to extend the size of the lexicon further. Since correction is not attempted for pedagogical reasons, no scoring mechanism has been necessary.

Finally, we should note that this is an on-going project, and the cycle of adding a new class of mistakes for coverage and keeping the performance at an acceptable level will repeat until the coverage of predicted syntactic mistakes becomes acceptable.
6 Conclusion

We have described an implemented and tested grammar checker for English as a second language, focusing on a limited category of frequently occurring grammatical mistakes. The grammar checker is based on a linguistic theory, and the presented implementation in CCG should be equally implementable in other theories with a similar definition of local domain, such as TAG. The use of local domain in CCG makes it particularly suitable for such theories.

Acknowledgements

We appreciate the students who agreed to provide their essays for the present project. We also thank our beta testers, especially Laura Siegel and Jinah Park, who have shared their time to test the system.

Appendix

The following segment shows the interaction between an earlier version of the system and a beta tester. We have added line breaks to the original script to make it readable in the two-column format.

SICStus 3 #1: Mon Aug 28 16:01:57 EDT 1995
| ?- [uccg].
[initial messages deleted]

yes
| ?- elp.

1> yesterday I goed to store.
[punctuation or syntax]

2> oh, yesterday I went to store.
[[oh]:cap/init,[store]:missing/art or pl]

3> to the store.
[punctuation or syntax]

4> yesterday I go to the store
[[yesterday]:cap/init,[yesterday]:mismatch/adverb&tense]

5> a store.
 okay

6> there is many thing for sale.
   [[there]:cap/init,[is]:agr/number with verb,[thing]:agr/number with article]

7> There are many things for sale.
 okay

8> There is many beautiful, inexpensive hat, too.
   [[is]:agr/number with verb,[hat]:agr/number with article]

9> There are many beautiful, inexpensive hats, too.
 okay

10> In Panama, we has hat like this.
   [[we]:agr/pronoun,[hat]:missing/art or pl]

11> In Panama we have hats like this.
 okay

12> When I was 16, my family visited one hat factory.
 okay

13> I learned how can make hats.
   [[can]:missing/subject]

14> I learned how people can to make hats.
   [extra/to]

15> I learned how people can make hats.
 okay

16> Is difficult.
   [[is]:missing/subj]

17> It is difficil.
   [punctuation or syntax]

18> My father make hats like these my
grandfather make hats too my uncle didn’t make hats.
[punctuation or syntax]

19> My father made hats like these and my grandfather too.
okay

20> my uncle sold hats for indians.
[[my]:cap/init, [indians]:cap/upper]

21> He gave a hat by me.
[punctuation or syntax]

22> He gave a hat to me but I lost it since two weeks ago.
[extra/since]

23> I lost it two weeks ago.
okay

24> I decision to buy other hat.
[[decision]:verb form, [hat]:agr/number with article]

25> I decided to buy other hat.
[[hat]:agr/number with article]

26> I decided to buy another hat.
okay

27> bye.
Good Bye!

References
Arndt, V., “Six writers in search of texts: A protocol based study of L1 and L2 writing,” ELT Journal, 41, 257 – 267, 1987.

Bolioli, Andrea, Luca Dini, and Giovanni Malnati, “JDIE: Parsing Italian with a Robust Constraint Grammar,” Proceedings of the International Conference on Computational Linguistics (COLING-92), pages 1003 – 1007, 1992.

Brooks, E., “Case studies of the composing processes of five "unskilled" English-as-a-second-language writers,” Unpublished doctoral dissertation, New York University, 1985.

Bustamante, Flora Ramírez and Fernando Sánchez León, “GramCheck: A Grammar and Style Checker,” Proceedings of the International Conference on Computational Linguistics (COLING-96), 1996.

Genthial, Damien and Jacques Courtin, “From Detection/correction to Computer aided Writing,” Proceedings of the International Conference on Computational Linguistics (COLING-92), pages 1013 – 1018, 1992.

Mejia, Elizabeth, Maida Xiao, and Lucyna Pasternak, “American Picture Show: A Cultural Reader,” Prentice Hall Regents, 1992.

Grishman, Ralph, Catherine Macleod, and Adam Meyers, “Complex Syntax: Building a Computational Lexicon,” Proceedings of the International Conference on Computational Linguistics (COLING-94), pages 268 – 272, 1994.

Konrad, Karsten, Holger Maier, Manfred Pinkal, and David Milward, “An Education and Research Tool for Computational Semantics,” cmp-lg/9608005, 1996.

Jong C. Park, “A Unification-Based Semantic Interpretation for Coordinate Constructs,” The Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-92), pages 209 – 215, Newark:DE, 1992.

Jong C. Park, “Quantifier Scope and Constituency,” The Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-95), pages 205 – 212, Cambridge:MA, 1995.

Raimes, Ann, “What unskilled writers do as they write: A classroom study of composing,” TESOL Quarterly, 19, 229 – 258, 1985.

Leonard J. Rosen, “Guide to Grammar and Punctuation: A Writer’s Handbook,” Barnes & Noble, 1985.

Laurie Rozakis, “The Random House Guide to Grammar, Usage, and Punctuation,” The Random House, 1991.

Stuart Shieber, “An Introduction to Unification Based Grammar,” CSLI Lecture Note, No. 4, 1986.

Mark J. Steedman, “Gapping as Constituent Coordination,” Linguistics & Philosophy, volume 13, pages 207 – 263, 1990.

Mark Steedman, “Surface Structure and Interpretation,” Linguistic Inquiry Monograph 30 (to appear), MIT Press, 1997.

Gregor Thurair, “Parsing for Grammar and Style Checking,” Proceedings of the 13th International Conference on Computational Linguistics (COLING-90), pages 365 – 370, 1990.

Mary McGee Wood, “Categorial Grammars,” Routledge, 1993.

The XTAG group, “A Lexicalized Tree Adjoining Grammar for English,” IRCS Technical Report 95-03, University of Pennsylvania, 1995.