DEVELOPMENT OF AN ENGLISH GRAMMAR CHECKER
A PROGRESS REPORT

Hsien-Chin Liou

Abstract: In order to leave more time for EFL teachers to work on higher-level re-writing tasks, we decided to develop a computer grammar checker. The first stage of development was devoted to error analysis of 125 writing samples collected from our students. We found 1659 errors and classified them into 14 main types and 93 subtypes. The analysis served as the basis for constructing a taxonomy of mistakes and ranking the categories according to frequency of occurrence and comprehensibility. To implement the grammar checker, we first built a small electronic dictionary with 1402 word stems and necessary features, and designed a suffix processor to accommodate morpho-syntactic variants of each word stem. We then constructed an ATN parser, equipped with phrase structure rules and error patterns. In addition, a set of disambiguating rules for multiple word categories was designed to eliminate unlikely categories and thus increase the parser's efficiency. The current implementation detects seven types of errors and provides corresponding feedback messages. Future research will be focused on detecting more types of mistakes with greater precision and on providing appropriate editing strategies.

Keywords: grammar checker, error analysis, error patterns, electronic dictionary, word features, suffix processor, phrase structure rules, parser, feedback.

I. Introduction

One of the reasons language teachers in Taiwan, R.O. C., find EFL (English-as-a-Foreign-Language) writing classes formidable is the seemingly endless task of correcting grammatical mistakes in student compositions. Our own experiences in this area led us to investigate computer-assisted language learning (CALL). If a computer program could help detect or even correct grammatical mistakes in students' papers, it would reduce, the tiring part of the revision process and leave more time for human teachers to work on higher-level re-writing tasks.

We[2] began testing to what extent a commercial software package, *Grammatik IV* (1989), could help our EFL students.[3] We asked 28 college students to use *Grammatik IV* individually, observed how they responded to feedback messages the package generated, and requested them to fill out a questionnaire which elicited their affective reaction toward the process. Mistakes detected and marked by *Grammatik IV* were recorded on a hard copy of student essays. It was found that though most of the students were using CALL (computer-assisted language learning) for the first time, they did not find the experience discomfiting. Seventy percent found the process interesting and the package easy to use, as well as helpful for one aspect or another of the revision process. Nevertheless, comparison of the marked essays with the originals revealed that only fourteen percent (10 out of 70) of the mistakes *Grammatik IV* detected were substantive grammatical errors; the rest were stylistic. Worse, the package missed significant errors frequently made by students, and generated false positives and misleading messages such as those in italics below:

(1) *Having listening_the teachers' word, I was not surprised at the poor score I got as I didn't do the question with caution.* [Passive voice: 'was surprised' Consider revising using active]
(2) *There <u>were</u> great man in the world whom I respected forever.* [The context of 'whom' indicates you may need to use 'who']
(3) *These occupy successively lower ranges on the scale of computer translation ambition.* [Usually 'these' should be followed by a plural noun.]

Grammatik IV's deficienciess led us to try to develop an automatic English grammar checker which could detect the kinds of major errors our students frequently make.[4]

II. Error Analysis and Categorization

We collected over 1000 two-hundred-word compositions from students with mainly engineering backgrounds. In analyzing 125 of these, we found 1659 errors which were classified into 14 major types (see Table 1).

Each of the major types was then divided into several subtypes, a process which yielded 93 subtypes in total. To measure the gravity of the error types, we adopted two criteria: frequency of occurrence and level of comprehensibility. Frequency of occurrence was

measured by dividing the number of occurrences of an error type by the total number of errors, 1659 (see Appendix A). To obtain a measure for the second criterion, level of comprehensibility, we asked two native English speakers (associate professors in linguistics) to grade examples taken from each subtype on a four-point scale.

I.	Verbs
II.	Nouns
III.	Adjectives
IV.	Adverbs
V.	Auxiliaries
VI.	Pronouns
VII.	Determiners
VIII.	Conjunctions
IX.	Prepositions
X.	Subject-verb/predicate concord
XI.	Lexicon
XII.	Form-Classes (part of speech)
XIII.	Sentence-level
XIV.	Mechanics

We then selected those categories which occurred frequently, hindered comprehension significantly, and could be processed by a grammar checker with relative ease. For these, we either formulated error patterns or represented the errors as explicitly as possible so that computer programs could recognize/detect them. For example, the V1 error patterns (a subtype under the major type verbs; see sentences (4), (5), and (6) for examples) can be described as a be verb (optionally plus one or more words), plus either an intransitive verb, or a transitive verb followed by a noun phrase at the verb phrase level.

(4) *It is seem great for the results coming out from science.*
(5) *...although they are not necessary improve our material life directly.*

To accommodate exceptions such as sentence (6), we needed another pattern: causative verb, make (optionally plus one or more words), plus finite verb be.

(6) Scientists have done a lot of works which made our living pattern is different from the days.

The pattern can be written formally in the following manner:

a'V[b]X -{V[vi] }
-{V[vt] NP }
b'V[c]X- V[b]

(V[b]: be verb; X: wildcard symbol; V[vi]: intransitive verbs; V[vt]: transitive verbs; NP: noun phrase; V[c]: causative verbs; {}: selectional symbol)
(Note: Tentatively X has been defined as an arbitrary number of words.)

Patterns such as these provided the basis for the error identification component described in section IV.

III. The Electronic Dictionary

A survey of relevant literature indicated that there are several comprehensive machine readable dictionaries available such as *Longman Dictionary of Contemporary English*, *Webster's Seventh Collegiate Dictionary*, *Collins Bilingual Dictionary*, and *Collins Thesaurus* (see Boguraev & Briscoe, 1987; Boguraev & Briscoe, 1989; Byrd, Calzolari, Chodorow, Klavans, Neff & Rizk, 1987 for examples). However, because our students have limited English vocabulary and the project is exploratory in nature, we decided to make a small dictionary on our own to meet immediate needs. Our experiences with this small dictionary will help us to select crucial information and to determine efficient access methods when we adopt a comprehensive electronic dictionary in the future.

For our own dictionary[5], a program was written to extract word types from a sample of the analyzed student compositions and to form the core of our dictionary entries. There are currently 1402 entries in our dictionary, including proper nouns. Attached to each word is part-of-speech (or word category) information and necessary features. Note that we have selected only the more likely part-of-speech information which our learners use in their English writing; we have not encoded rare usage in our dictionary. The selection and ordering of word categories are intended to reflect frequency of occurrence for the usage of each word, although further lexicographic research is required to ensure that our representations are correct. The selective approach means that more unknown words could be encountered in higher quality essays. However, the simplification strategy saves space in memory and increases the parser's efficiency. A sample of word categories and associated features in the electronic dictionary is shown in Table 2.

The entries in our dictionary are mainly stems of words or headwords. To accommodate suffix changes of word stems, we have designed a suffix processor (as suggested in Heidom, Jensen, Miller, Byrd & Chodorow, 1982) by adopting the concept of a distributional lexicon (Beale, 1987). The processor is equipped with information about (a) rules of change concerning word categories (e.g., from verb to noun) and inflectional features (e.g. from plural noun to singular noun), and (b) associated actions (e.g., omitting *–s* in a plural noun can result in a singular noun form). By means of a search procedure to correlate rules and suffix changes between the variants and headwords, the suffix processor ensures that the dictionary can identify the following three types of morpho-syntactic variants of headwords in the dictionary: (a) inflectional suffixes such as *–ing*, *-ed*, *-s* (for both verbs and nouns), (b) derivational suffixes such as *–ly* in *happily* (from *happy*), and *–ful* in *cheerful* (from *cheer*), and (c) markers of comparative and superlative degrees, *-er*, *-est* (such as *hotter*, or *fastest*). In this way, our dictionary can cope with natural English texts without listing all the derivations as separate entries. To increase processing efficiency, we grouped the rules above so that when a word like *getting* is encountered, it is assigned to the *–ing* group. This means we

can avoid searching all the suffix rules. To cope with irregular forms of verbs, we have designed a table which lists the root form and irregular changes of verbs. In this way, an irregular verb (for example, *began*) can be associated with its root (*begin*).

IV. Parsing and Error Detection

Noun:	count/noncount; vowel/consonant in the initial phoneme (V/C)
Adjective:	single/multiple syllable (S/M); V/C
Adverb:	subcaegories (8 classes); S/M; V/C
Verb:	subcategories (33 classes)
Pronoun:	singular/plural/both (S/P/B); person (1^{st}, 2^{nd}, 3^{rd}); case (subject/object/possessive)
Determiner:	S/P/B

Table 2. A Sample of Word Entries and Associated Features in the Dictionary

To analyze the input text structurally, a top-down parser was constructed. It was formulated as an augmented transition network (ATN) grammar (Woods, 1970). To increase its precision, a set of word category disambiguation (WCD) rules was devised to pre-process multiple word categories of some input words. For example, if a word has two categories, verb and adjective, and it is preceded by a determiner and followed by a noun, then the category, adjective is chosen (such as *falling* in *the falling rock*). The rules cut down on multiple word categories, and thus reduce both processing time and the number of ambiguous sentence structures.

So that the parser could debug grammatical errors (besides judging whether the sentence is grammatical or not), two types of information were included in the program: an expert model and a bug model. The expert model represents all the structural possibilities of correct sentences, while the bug model represents the error patterns we formulated. The following is a small segment of the phrase structure rules we use in the expert model to represent the structure of correct sentences:

S-> NPVP
NP-> (Det) AN ({PP,S'})
AP-> (Det) ("more") A {PP,S'}
VP-> V (NP) ({NP, PP})
PP-> PNP
S'-> CompS

(S: sentence; NP: noun phrase; VP: verb phrase; Det: determiner; AP: adjective phrase; N: noun; S': embedded sentence; A: adjective; V: verb; PP: prepositional phrase; P: preposition; Comp: complementizer; (): optional symbol; {}: selectional symbol)

The bug model currently has three groups of error patterns: those manifested at noun phrase, verb phrase, and clause levels. A group is activated while the parser is analyzing/reconstructing its corresponding constituent. In addition, there are errors which occur infrequently and/or are idiosyncratic. For these cases, we plan to map the expert model onto the input sentence and to diagnose the problem by some devised heuristic. The dual-model mechanism we applied is similar to that described in Weischedel and Sondheimer (1983).

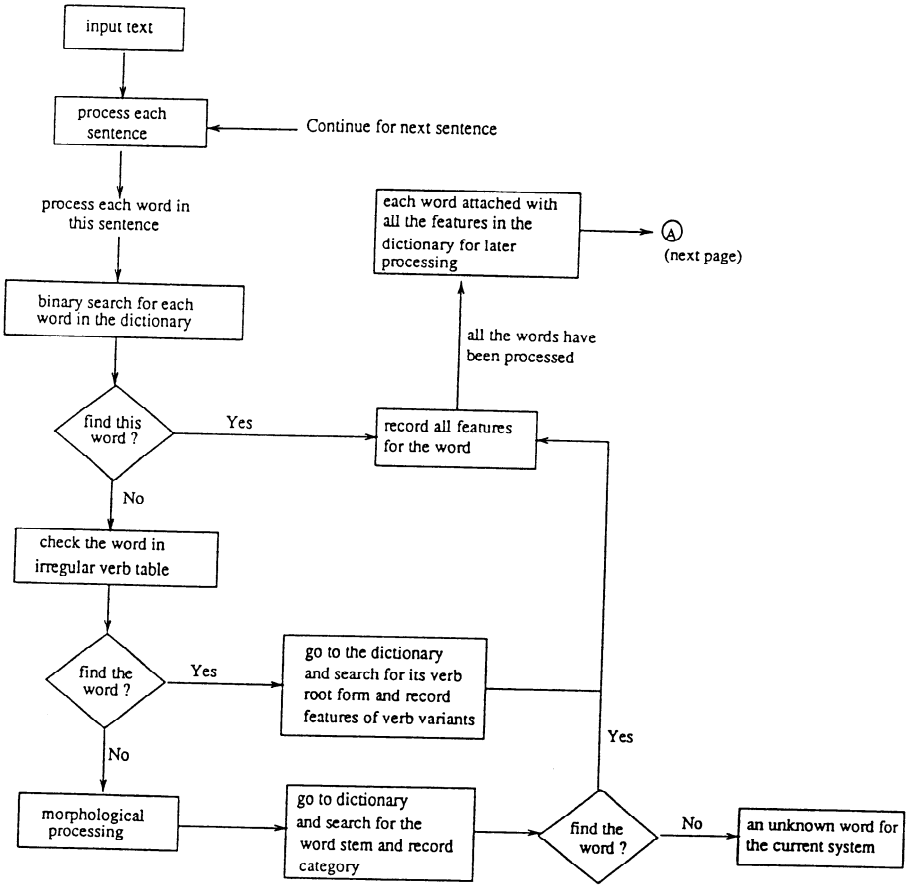

Figure 1. The flowchart of processing a sentence in the program

How does our grammar checker operate to detect a mistake? The flow chart in Figure 1 demonstrates how the grammar checker processes each sentence and detects errors. First, our program allows regular English text as its input and processes sentence by sentence. For each sentence, the program uses the binary search algorithm to locate each word in the dictionary. If the program finds the word, it then records all associated features of this word. If the program fails, it proceeds to search for the word in the irregular verb table. If it finds the irregular verb form and thus the root form, it returns to the dictionary and obtains features of the root form as well. If the program still cannot find the word at this stage, it activates the suffix processor to do morphological processing. Note that the category of a word before morphological processing is unknown if the form of the word in the text is not the same as the form in the dictionary. After the word is processed by the suffix processor, it may be reformed and obtains its category information from this process. If the program still fails at this stage,

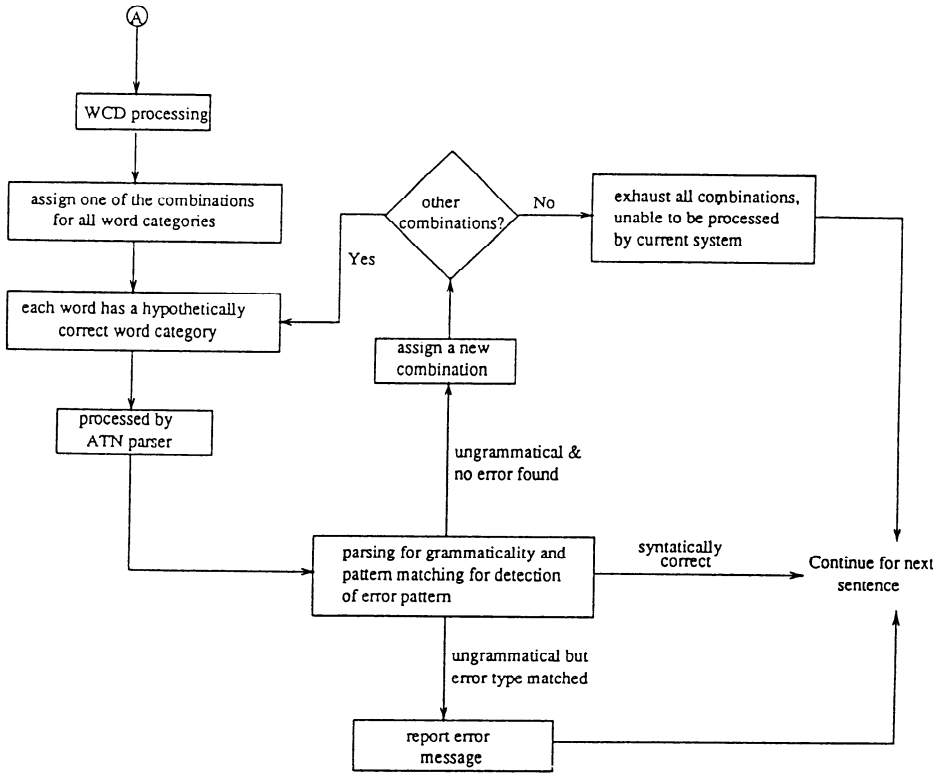

Figure 1 (continued)

the word is tagged as unknown for our current system.

After each word has been associated with category information, the program activates word category disambiguation (WCD) rules to cut down unlikely categories if a word has more than one category. After WCD processing, each sentence obtains a hypothetically correct combination of word categories to be processed by the parser.

If the parser determines that the sentence is grammatical, the program proceeds to the next sentence. If the sentence is determined as ungrammatical and detected by any of the error patterns, the program reports the error/feedback message and continues to the next sentence. If neither the parser nor pattern matching can determine the status of the input sentence, the sentence is assigned another combination of word categories (if there are any), and the program repeats the parsing/pattern-matching processing. If the program exhausts all the possible combinations of word categories but still cannot determine the status of the sentence (grammatical or ungrammatical), then the sentence cannot be analyzed by the current system.

```
Parse sentence:  No matter he say, he like these job.

Searching in the dictionary...

Using WCD-rules...

Assigning category...
no<av> matter<n> he<ppn> say<v> he<ppn> like<v> these<d> job<n>

Syntax error!! -> No matter...
no matter (?) he say, he like these job.

Syntax Error!!  -->  Number disagreement determiner-noun
no matter he say, he like (these) (job).

Syntax Error!!  -->  Subject-verb disagreement
no matter (he) (say), he like these job.
no matter he say, (he) (like) these job.

This is not a correct sentence.  There are four errors.
```

Table 3: An Output Trace.

The operation of the grammar checker itself is an interaction between the parsing and the error pattern matching processes. By way of illustration, let us look at sentence (7). The trace in Table 3 illustrates how the checker diagnoses this sentence.

(7) *No matter_he say_, he like_ these job_.*

Because sentences are presumed to be ungrammatical or erroneous, the program activates the error pattern matching process before the parser. First, pattern matching of clause level errors is activated. Error types such as *although*...*but* and *no matter* are classified as clause level errors. Sentence (7) matches the error pattern of *no matter*, a fact which the program notes. Second, because there is a noun phrase (NP), *these job*, error types at the noun phrase level are also examined; the phrase in question is found to match the type *determiner-noun disagreement*. Lastly, subject-verb (S-V) agreement is checked for each NP and VP (verb phrase) in each clause. The program first locates the head of each NP and VP and returns the number values (singular or plural) of both. Then, a comparison is made to see whether they agree. In sentence (7), two incidents of S-V disagreement are found. If none of the error patterns are matched in any of the constituents, the parser is activated and determines whether the sentence is grammatical under our current phrase structure representation.

Currently, our checker can locate the following seven types of errors:

(I) although...but combination

(8) Although he is poor, but he is happy.

(II) erroneous usage of no matter

(9) People can produce many things, no matter bad or good.

(III) determiner-noun disagreement

(10)We can know many informations.
(11)This is a books.
(12) I like an book[6]

(IV) unbalanced coordinated phrases

(13) He likes a dog but hate_ a cat.

(V) capitalization misuse

(14) There are not the exist of Television, computer, airplane, and so on.

(VI) erroneous morphological changes in verb phrase, and

(15) I should went with you

(VII) subject-verb disagreement

(16) Human create_ the science.
(17) Human already have the ability to research the phenomena of space.
(18) But the development in science have bring great change.
(19) A man who like_ art like_ books.

V. Feedback

When the program detects a grammatical error, appropriate feedback messages are essential for the grammar checker to achieve its educational goal. For this, we designed a message generating routine which basically matches a flag that is attached to each processing rule with a message file, and outputs the message to the users, possibly with some examples. We use a template to output a complete feedback message: the message consists of some variables (as those italicized in (20)) and literal texts (those in plain type in (20)). For example, a feedback message for sentence (20) is illustrated in the square brackets.

(20) The *development* in scientific technologies *have* bring great change. [*development* is the subject of theverb *have*. The subject is in 3rd person singular form. The following are 2 correct examples:
The *clerk* beside the bookshelves *is* watching television.
The *lady* who the workers *love* teaches English.]

For technical terms, we are considering the use of Chinese for the messages. In addition, the correction and feedback given should be set up with a user-friendly interface environment so that language teachers and learners will not be confused.

VI. Future Research

In the short term, we will complete the analysis of the remaining compositions and continue to develop the program to include more error patterns. In addition, the grammar checker's performance must be tested against a corpus. Finally, we will consider at which point in the program it is appropriate to give feedback messages and student editing strategies to improve writing revision.. Clearly, there is still a long way to go. Nevertheless, despite the problems and difficulties, we believe we have made some important first steps.

NOTES

[1] Versions of this paper were presented at TESOL '91 (New York, 26) and CALICO '91 (Atlanta, April 4). The research project is funded by a grant(#NSC80-0301-HO07-15) from the National Science Council of Taiwan, Republic of China.
[2] A research group, including a professor in foreign languages, a professor in computer science, two part-time graduate students, and a full-time research assistant.
[3] We also examined *Right Writer* (Rightsoft, 1988), but found it to be an inferior product for our purpose.
[4] For a similar critique of *Grammatik IV*, see Brock 1990a and 1990b. Concurrent efforts such as Chen and Xu (1990) have been initiated, as complementary to the present research.
[5] The current project does not deal with misspellings which spelling checkers in commercial word processing packages identify to a very satisfying extent.
[6] The initial phoneme of *book* is encoded in the dictionary.

REFERENCES

Beale, A. 1987. "Towards a distributional lexicon." In R. Garside, G. Leech, and G. Sampson (Eds.), *The Computational Analysis of English*. London: Longman, 149-162.

Boguraev, B. and T. Briscoe. 1987. "Large lexicons for natural language processing: Utilising the grammar coding system of LDOCE," *Computational Linguistics*, 13:3-4, 203-218.

Boguraev, B. and T. Briscoe, (Eds.). 1989. *Computational Lexicography for Natural Language Processing*. London: Longman.

Brock, M.N. (1990a). "Customizing a computerized text analyzer for ESL writers: Cost versus gain," *CALICO Journal*, 8:2, 51-60.

Brock, M.N. (1990b). "Can the computer tutor? An analysis of a disk-based text analyzer," *System*, 18:3, 351-359.

Byrd, R.J., N. Calzolari, M.S. Chodorow, J.L. Klavans, M.S. Neff and O.A. Rizk. 1987. "Tools and methods for computational lexicography," *Computational Linguistics*, 3:3-4, 219-240.

Chen, S. and L. Xu. 1990. "Grammar-debugger: A parser for Chinese EFL learners," *CALICO Journal*, 8:2, 63-75.

Grammatik IV. 1989. Computer Software. San Francisco: Reference Software International.

Heidorn, G.E., K. Jensen, L.A. Miller, R.J. Byrd and M.S. Chodorow. 1982. "The EPISTLE text-critiquing system," *IBM Systems Journal*, 21:3, 305-326.

Rightsoft, Inc. 1988. *Right Writer: User's Guide*. Sarasota, FL: Right Soft Incorporated.
Weischedel, R.M. and N.K. Sondheimer. 1983. "Meta-rules as a basis for processing ill-formed input," *American Journal of Computational Linguistics*, 9:3-4, 161-177.
Woods, W.A. 1970. "Transition network grammars for natural language analysis," *Communications of the ACM*, 13:10, 591-601.

AUTHOR'S BIODATA

Hsien-Chin Liou took her Ph.D. at the University of Illinois. She is currently Associate Professor in Foreign Languages at National Tsing Hua University in Hsinchu, Taiwan, R.O.C. She teaches CALL, EFL writing, and introductory computational linguistics. Her research interests include CALL for writing instruction, multimedia courseware development, and CALL research. She is a contributor to *Computer Assisted Language Learning and Testing* (ed. by Patricia Dunkel, 1991).

AUTHOR'S ADDRESS

Hsien-Chin Liou

Foreign Languages, National Tsing Hua University
101, Sec. 2 Kuang-Fu Road
Hsinchu, Taiwan, R.O.C. 30043
E-mail: hcliu@fl.nthu.edu.tw
nthu048@twnmoe10.bitnet

APPENDIX

Distribution of errors in main types and subtypes

Main Type	N	Per Cent
Det	326	19.65%
Verb	231	13.92%
Noun	178	10.73%
PS	174	10.49%
Concord	168	10.13%
Sent	158	9.52%
Prep	123	7.41%
Lex	115	6.93%
Conj	67	4.04%
Mech	54	3.25%
Adv	27	1.63%
Adj	23	1.39%
Pron	9	0.54%
Aux	6	0.36%
Total	**1659**	**100.00%**

Type	Subtype	N	Per Cent
Det	A-3	154	9.28%
Noun	CN	129	7.78%
Det	A-1	105	6.33%
Lex	Dict	94	5.67%
Prep	Prep-1	81	4.88%
Concord	3S-1	75	4.52%
Sent	Run-on	65	3.92%
PS	PS-nadj	65	3.92%
Verb	V-sub	59	3.56%
Sent	Frag	57	3.44%
Verb	VT-1	55	3.32%
Conj	Conj-1	55	3.32%
Verb	VT-3	51	3.07%
Mech	Cap	49	2.95%
Det	Det-a	49	2.95%
PS	PS-adjn	41	2.47%
Verb	VF	39	2.35%
Concord	SV	39	2.35%
Noun	UN	27	1.63%
Adj	Comp-1	23	1.39%
Prep	Prep-2	23	1.39%
Concord	3S-4	21	1.27%
Noun	NN	20	1.21%
Prep	Prep-3	19	1.15%
Concord	3S-5	15	0.90%
PS	PS-nv	14	0.84%
PS	PS-adjadv	12	0.72%
Verb	V1	12	0.72%
PS	PS-advadj	12	0.72%
Det	A-2	9	0.54%
Sent	E	9	0.54%
PS	PS-vn	8	0.48%
Concord	3S/paral	8	0.48%
Adv	ED	8	0.48%
Sent	2S	8	0.48%
Sent	Paral	8	0.48%
Conj	NM	7	0.42%
Verb	VT-2	7	0.42%
Pron	Pron-1	7	0.42%
Adv	Adv-2	6	0.36%
Concord	SP	5	0.30%
Conj	AB	5	0.30%
PS	PS-vadj	5	0.30%
Adv	OS	5	0.30%
Sent	Rel-1	5	0.30%
Verb	V2	4	0.24%
Aux	Aux-to	4	0.24%
PS	PS-prepv	4	0.24%

APPENDIX
(continued)

Type	Subtype	N	Per Cent
Det	Det-O	4	0.24%
Lex	Dict-v	4	0.24%
Lex	2V-1	3	0.18%
Det	A-4	3	0.18%
Adv	ASP	3	0.18%
Mech	Ap	2	0.12%
Lex	Dict-p	2	0.12%
Verb	VT-4	2	0.12%
Lex	Red	2	0.12%
Sent	WH	2	0.12%
PS	PS-adjv	2	0.12%
Concord	3S-2	2	0.12%
PS	PS-advcon	2	0.12%
Adv	very/much	2	0.12%
Verb	VT	2	0.12%
Sent	Rel-3	2	0.12%
Noun	One-N	2	0.12%
Pron	anaf	2	0.12%
Lex	SM	2	0.12%
Concord	3S-3	2	0.12%
Mech	Punct	2	0.12%
PS	PS-conjprep	2	0.12%
PS	PS-nadv	2	0.12%
Lex	Sem-1	1	0.06%
PS	PS-prepconj	1	0.06%
PS	PS-N.PP	1	0.06%
Lex	to/too	1	0.06%
Lex	Dict-Es	1	0.06%
Lex	A/E	1	0.06%
Det	some/any	1	0.06%
Det	Num-a	1	0.06%
Concord	WS	1	0.06%
Sent	Rel-2	1	0.06%
PS	PS-infprep	1	0.06%
PS	Red-Comp	1	0.06%
Lex	PH	1	0.06%
Sent	WHi	1	0.06%
PS	N-adj	1	0.06%
Aux	Aux-2	1	0.06%
Aux	Aux-1	1	0.06%
Lex	Dict-mb	1	0.06%
Lex	Dict-e	1	0.06%
Adv	TA	1	0.06%
Adv	SA	1	0.06%
Adv	Adv-1	1	0.06%
Total		**1659**	**100.00%**