The experimental EPISTLE system is intended to provide "intelligent" functions for processing business correspondence and other texts in an office environment. This paper focuses on the initial objectives of the system: critiquing written material on points of grammar and style. The overall system is described, with some details of the implementation, the user interface, and the three levels of processing, especially the syntactic parsing of sentences with a computerized English grammar.

The EPISTLE text-critiquing system

by G. E. Heidorn, K. Jensen, L. A. Miller, R. J. Byrd, and M. S. Chodorow

The long-term objectives of the EPISTLE project are to provide office workers, particularly middle-level managers, with a variety of application packages to help them interact with natural language texts. Initially we are focusing on business letters and on the first of two classes of applications. This first class will provide services for the author, initially furnishing critiques of a draft of a letter or other text, and eventually helping him write an initial draft based on a terse statement of what he wants to say. The second class of applications will deal with incoming texts, synopsizing letter contents, highlighting portions known to be of interest, and automatically generating index terms based on *conceptual* or *thematic* characteristics rather than key words.

In its current experimental form, the EPISTLE system addresses only the tasks of grammar and style checking of texts written in English. Grammar checking deals with such errors as lack of number agreement between subject and verb; style checking points out such

problems as overly complex sentences. Even in this limited form, the system is still under development and is not available for general use at this time.

The processing in EPISTLE is done at three levels. Word level processing is mostly a matter of efficient dictionary lookup, but also includes handling of suffixes and prefixes. The information retrieved from the dictionary provides parts of speech and other attributes of words needed for later processing. Grammar checking is done by a general language processing system that attempts to parse each sentence according to the rules of English grammar. If any rules have to be relaxed to bring about a parse, a grammar error is noted. Style processing uses the parse trees developed during grammar checking to detect potential problems in exposition.

This paper begins with a description of the natural language processing system that underlies EPISTLE, along with a discussion of the user interface developed for this application. Then the processing is described for each of the three levels. Most emphasis is placed on the grammar processing because it is considered to be the central element. Finally, the current status of the system and both our immediate and longer-range plans are discussed. Earlier descriptions of EPISTLE can be found in Miller[1] and in Miller, Heidorn, and Jensen.[2]

Overview of the EPISTLE system

EPISTLE is built upon a general language processing system called NLP, which is used here primarily to parse English sentences, i.e., to determine their syntactic structures. In this section, NLP is discussed first and then the EPISTLE user interface is described.

The NLP natural language processing system

NLP is based on the concept of *augmented phrase structure grammar* (APSG).[3,4] The current implementation is embedded in the Yorktown LISP system, a revision of LISP/370,[5] and, for EPISTLE, requires a four-megabyte virtual machine under VM/370. (However, a new version of NLP is being implemented in PL/I and should be able to run EPISTLE in about 500K bytes.)

The basic units of data in NLP are attribute-value pairs, which are grouped into *records*. *Named records* initially hold static information; *segment records* hold dynamic information and are created and destroyed during processing. Most of the attribute values in a record, except those low-level records associated with individual words, are pointers to other records, thus forming a network of information called a "record structure."

Processing in NLP is specified by augmented phrase structure rules. There are *decoding* rules that specify how input text is to be

Figure 1A Initial set of grammatical errors

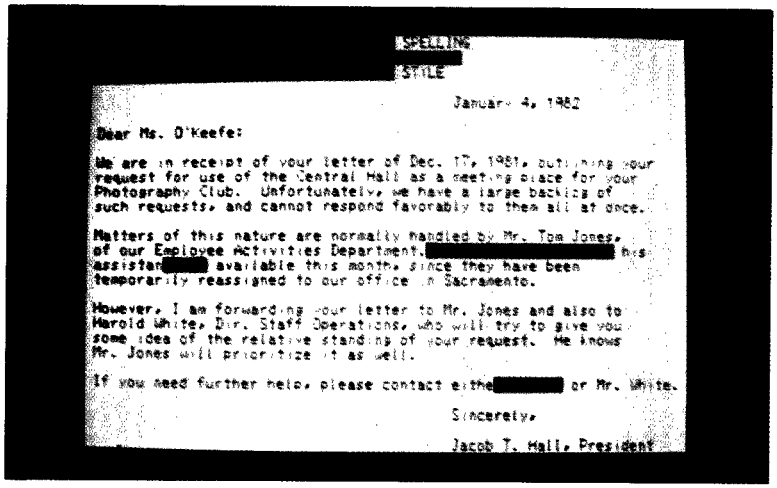
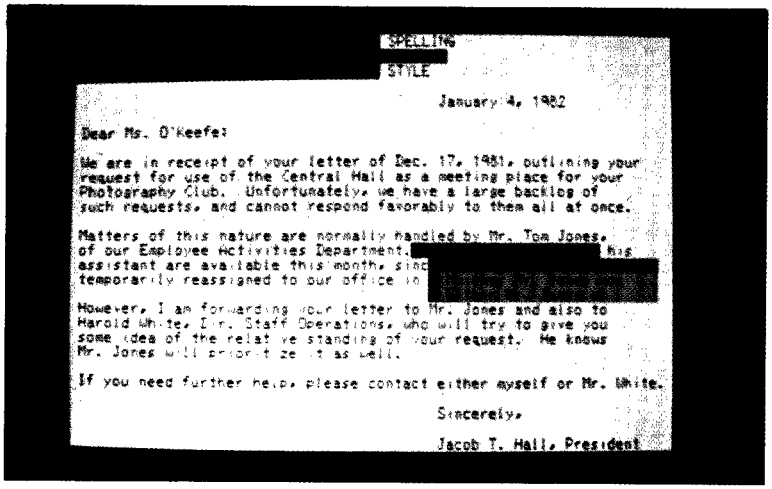


Figure 1B Error with fix window



converted into a record structure, and there are *encoding* rules that specify how output text is to be generated from a record structure. Encoding rules also manipulate internal record structures when necessary. NLP rules look like context-free phrase structure rules, similar to BNF (Backus Naur Form), but augmented with arbitrary tests and structure-building actions stated in parentheses in a special notation on both sides of the rules. For EPISTLE, decoding rules produce parse trees and diagnose grammar errors, and encoding rules diagnose style errors. Examples of NLP rules are discussed later.

Figure 1C Error with fix window and help window

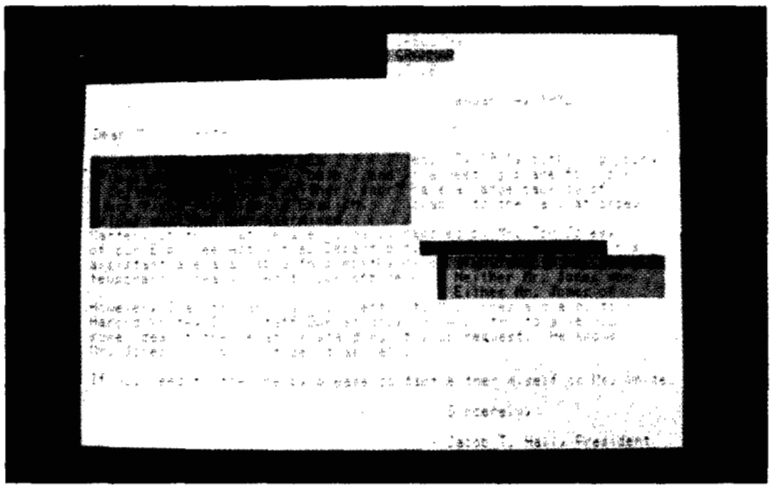
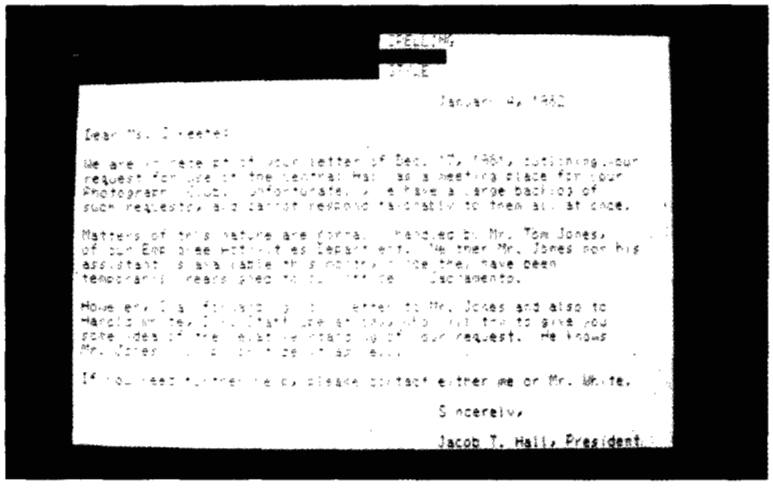


Figure 1D No remaining grammatical errors



The NLP system consists of a rule translator and a run-time environment with two processing algorithms. The translator converts NLP rules into LISP functions, which are then compiled into System/370 machine code by the LISP compiler. The two run-time algorithms apply the decoding and encoding rules. The decoding algorithm operates in a left-to-right, bottom-up, parallel-processing fashion, similar to a syntax-directed compiler, but with the addition of nondeterminism. The encoding algorithm operates in a top-down, serial-processing fashion, producing output from left to right.

Until last year, dictionary processing in NLP used a named record for each word stem, with attributes for parts of speech and valid suffixes, along with morphological decoding rules that specified how the individual characters in an input text were assembled into words. Now, in order to make the system's vocabulary essentially unlimited, it uses a standard on-line dictionary of over 100 000 entries, in which it finds the parts of speech. A separate routine does some morphological processing for prefixes and suffixes. Currently, NLP gets most of its word information from this on-line dictionary. A small file of named records is still maintained for some function words and syntactic data (such as verb complement types), but this older method of dictionary processing will soon be phased out completely. The latest version of the newer method is described in some detail in the next section, dictionary processing.

The EPISTLE user interface

The interface processor controls the user's session. It reads input text, passes it to NLP for analysis, and then displays the results. The user sees the system as a text editor with sophisticated enhancements.

Much recent work on programming and office automation systems suggests that multiple overlapping windows provide a natural model for the interaction between the user and the computer (e.g., see Ingalls[6]). Window location, size, texture, and color all play a role in associating and dissociating portions of the information. When combined with a flexible pointing mechanism, windows provide an easy-to-use interface. We have adopted that model in the interface to EPISTLE and have implemented the scenario described below on an IBM 3279 color terminal with a light pen.

The main window in the interface is white and displays the current state of the text. If the text is too large for the screen, only a portion of it is shown at one time. Appended to this window are mode indicators, essentially a window containing a menu, which allows the user to select the service to be invoked—spelling, grammar, or style checking. (The system does not currently include a spelling checker, but an available one will eventually be incorporated into it to form a complete text-critiquing package.)

Figure 1A shows the screen that results from selecting grammar checking to be applied to a sample document. Notice that the GRAMMAR item in the mode window has turned yellow, suggesting caution because of potential grammatical errors. The erroneous words or phrases are highlighted in red within their original context.

The user focuses on one of the errors by selecting it with a light pen. Immediately, the other errors resume their background white color, and a new window appears in the vicinity of the selected error, overlapping the original document. This *fix* window contains a brief

description of the error found and a list of words or phrases that the system proposes as likely fixes for the red-highlighted error (Figure 1B).

At this point the user may take one of four actions:

1. Ignore the system's suggestion by selecting the red-highlighted text a second time. This action causes the system to treat the word or phrase as correct. (For personalized style checking, a future version of the system could store such a decision in its catalog of user preferences, to avoid signaling similar errors to this user in subsequent documents.)
2. Request additional information about the error by selecting the title of the *fix* window. The system responds with a *help* window which, depending upon the error type, contains (1) an expanded description of the error, (2) the system's strategy for producing the recommended fixes, or (3) a strategy for the user to apply in generating his own fix. If this action were taken for Figure 1B, the result would appear as Figure 1C.
3. Accept the system's determination of an error, and select one of the menu items in the fix window. The selected material replaces the red-highlighted text, and the system treats the result as correct.
4. Agree that there is an error but not wish to use any of the suggested fixes. In this case, the user may enter his/her own correction in place of the red-highlighted text, and the system will assume that the new text is correct.

When the error has been corrected, the red highlighting disappears from that portion of the text and reappears for any remaining grammar errors. Figure 1D shows that when all grammar errors have been corrected, the GRAMMAR indicator in the mode window is displayed in green.

When the user selects SPELLING or STYLE, he/she goes through the same sequence of steps.

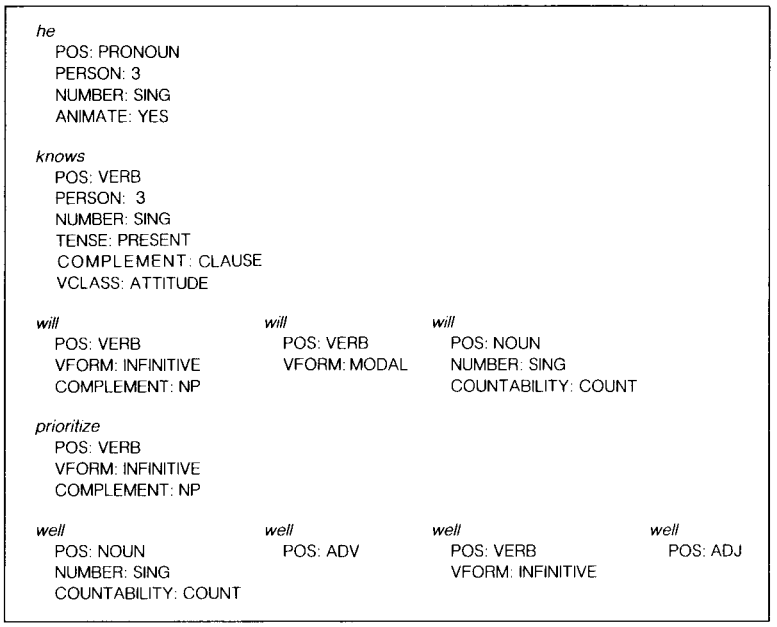
Dictionary processing

The dictionary contains information about words. The current dictionary processor is written in the EXEC-2 programming language, but an improved version is being done in PL/I. We now briefly discuss morphological processing, the form of the dictionary output, improved spelling checking, and fact identification.

Morphological processing

The morphological component takes advantage of derivational and inflectional regularities in the structure of English words to reduce the number of words that must be explicitly stored and to generate syntactic and semantic information that is predictable from word

Figure 2 Word segment records for the example sentence



structure. The mechanism used is an extension of the one described by Amsler[7] for dealing with plural inflections on nouns. The extensions exploit some of the theoretical results of Aronoff's study[8] of word formation rules.

A simple example of the first benefit of morphological processing is that plurals for most nouns need not be stored explicitly, since they may easily be formed from the singular forms. The second type of benefit can be illustrated by the word *complexity*. If the stored data contains *complex* as an adjective and if there is a morphological rule that a word is an abstract noun when it has the form Adjective + *ity*, then we can generate the syntactic information that *complexity* is a noun and the semantic information that it is abstract.

Dictionary output for grammar processing

During the parsing of a sentence, the dictionary processor looks up each word and produces NLP records needed by the parser. For example, Figure 2 shows a portion of the records produced for some of the words in the sentence

He knows Mr. Jones will prioritize it as well.

taken from the letter in Figures 1A-1D. Only those attributes of the records that are relevant to our discussion have been included in the figure. In actual use, the dictionary provides much more information about these words. There are several points to notice about this.

Many words in English are ambiguous. The word "well" has *multiple parts of speech*; it could be a noun ("John fell into the well"), an

adverb ("John sang well"), a verb ("Tears did well up in John's eyes"), or an adjective ("John got well in three days"). Another ambiguity can be seen for the word "will," which has *multiple verb senses,* one for the transitive verb ("John did will his fortune") and the other for the modal ("John will go"). Since the dictionary processor has no way of knowing which form of the word is appropriate for the input being parsed, it returns records for all of them. In most cases, the ambiguity will be resolved by the grammar, as described in the next section, grammar parsing.

The word "prioritize" is not stored in the dictionary. Rather, a morphological rule specifies that a transitive verb can be formed from a base noun ending in *-y* by removing the *-y* and adding *-ize*. Since the necessary base ("priority") is in the dictionary, the system determines that "prioritize" is a verb. The fact that it is transitive is signaled by an attribute that says that it takes an NP (*Noun Phrase*) complement.

In contrast to the basic *infinitive* form marked for "prioritize," the dictionary can determine that "knows" is a third person singular, present tense verb because of the *-s* inflection found on a base consisting of the verb "know." Note that, as far as the dictionary processor is concerned, the *third person singular* designation for this verb is independent of the same designation for the pronoun "he." However, as discussed later, the co-occurrence of words with such designations allows the grammar to check for subject-verb agreement in a clause.

Possibilities for improved spelling checking

Current spelling checkers produce a list of words that are close in spelling to some misspelled word and let the user make a choice. A shortcoming of this approach is that the lists frequently contain words that could not possibly be intended. Many words in such lists either have the wrong part of speech or are inappropriately inflected for the context in which they occur. For the misspelling "receieve," one such system produced the list: "receiver," "received," "reprieve," "retrieve," "reactive." In EPISTLE, the combination of a dictionary component that can analyze inflections and generate correctly inflected words and a parser that can analyze syntactic context and determine the permissible attributes of a misspelled word will allow correction lists that more closely match the user's intention. The capability for creating such lists, however, has not yet been developed.

Fact identification

Although the dictionary currently provides almost exclusively syntactic information, work is being done to augment it in the near future with some semantic information about the words. For example, nouns can have attributes related to animacy, time, place, or measure. This

type of information would be helpful both in writing grammar rules and for identifying facts: either the *name* of something or the *quantification* of something in terms of time, location, or some other measurement attribute.

Identifying facts in a document has at least two important applications. First, the job of parsing text can be simplified somewhat by replacing a string of words with a token indicating the fact category. Thus, in the example letter of Figures 1A–1D the string "Harold White, Dir. Staff Operations," could be replaced with a person-name-title token, and "Dec. 17, 1981," could be replaced with a date token. Such replacements would occur between dictionary processing and parsing.

The second application of fact identification is document-indexing—the characterization of a document in terms of a set of features or indices to be used to retrieve the document. (This is one of the longer-range objectives of the EPISTLE project.) Facts, being specific or unique references, provide an excellent base for supporting retrieval, since most queries have specific components—like places, names, and dates. However, fact-indexing provides a potentially much more powerful retrieval technology than present-day key-word-in-context methods since the fact identification algorithms would map potentially widely differing token strings into the same fact category.

Although fact identification has not yet been incorporated into the EPISTLE system, algorithms have been developed and tested for a comprehensive and detailed taxonomy of facts.[9]

Grammar processing

Grammar processing is the central component of the EPISTLE system. In this section, parsing is treated in some detail, followed by a description of the grammatical error diagnosis done by the system.

Parsing in EPISTLE

We now discuss what parsing of English sentences is, and then describe the way it is done in the EPISTLE system. Thereafter, the current system's coverage of English grammar is discussed, including the results of benchmark testing.

parsing English sentences

To parse means to break down a sentence into its component parts of speech with an explanation of the form, function, and syntactic relationship of each part. The component parts of a sentence include, first of all, its subject and predicate. The subject is the thing that the sentence is talking about, and the predicate is what is being said about the subject.

Figure 3 The structure of a sentence

```
SENT    NP      PRON*   "HE"
        VERB*   "KNOWS"
        VP      ""
                NP      NP      NOUN*   "MR."
                        NOUN*   "JONES"
                VERB    "WILL"
                VERB*   "PRIORITIZE"
                NP      PRON*   "IT"
                AVP     ADV*    "AS WELL"
```

A common way of illustrating the parsing of a sentence is to draw brackets around its components and label them:

[[He][knows [Mr. Jones] [will prioritize it as well]]]

SUBJECT SENTENCE — PREDICATE — SUBJECT — PREDICATE

This sentence consists of two clauses (subject-predicate pairs), one clause inside the other. Hence, this sentence is really two sentences in one.

We could draw more brackets around the subparts of each subject and predicate until we had broken the sentence down into its parts of speech—nouns, verbs, adjectives, adverbs, and so on. But these brackets, when nested inside one another, can be confusing to look at. Another way to display the components of a sentence is to pull them apart and align the smaller parts to the right of the larger, in something like an outline form, as in Figure 3.

In this display, which is produced by the EPISTLE parser, NP stands for Noun Phrase, VP for Verb Phrase, and AVP for Adverb Phrase. These terms are the subparts of the subjects and predicates of this sentence. Their form and linear relationships are graphically presented here; their functions (subject, predicate, direct object, etc.) and syntactic relationships are not displayed but are carried in the NLP records created during the processing.

In the figure, the parts of speech marked with asterisks identify the *heads* of their respective phrases, where "head" means the central element around which the other elements of the phrase are grouped. Elements of the phrase that precede its head are *premodifiers*; elements of the phrase that follow its head are *postmodifiers*. The concepts of "head" and "modifier" are important in understanding how the grammatical description is constructed.

For example, the head of the entire sentence in Figure 3 is the main verb "knows." "Knows" has as its only premodifier the noun phrase

Figure 4 Sample NLP decoding rules with explanations

RULE	INTERPRETATION
(1) NOUN → NP (HEAD=NOUN)	(1) A noun can be called a noun phrase.
(2) PRON → NP (HEAD=PRON)	(2) A pronoun can also be called a noun phrase.
(3) NP (¬PRON) NP (¬PRON) → NP	(3) Consecutive NPs (provided they are not pronouns) can be put together to form another NP.
(4) VERB → VP (HEAD=VERB)	(4) A verb can be called a verb phrase.
(5) VERB ('WILL') VP → VP (FUTURE='YES')	(5) The modal verb "will" followed by a VP can be combined into a new VP. The new VP will indicate future time.
(6) VP NP → VP (OBJECT=NP)	(6) A VP followed by an NP can become a new VP, with the NP becoming its direct object.
(7) VP AVP → VP	(7) A VP followed by an adverb phrase also combines into a new VP.
(8) PREP ('AS') ADV ('WELL') → AVP ('ALSO')	(8) The preposition "as" followed by the adverb "well" functions like the one-word adverb "also."
(9) PREP NP → PP	(9) A preposition (such as "to," "by," "as," etc.) followed by an NP forms a prepositional phrase.
(10) NP VP (NUMB.AGREE.NUMB(NP)) → VP(SUBJECT=NP)	(10) An NP followed by a VP, where the number of the VP agrees with the number of the NP (singular with singular, or plural with plural), can be called a VP, with the NP functioning as subject of the new VP. This rule forms a complete clause.

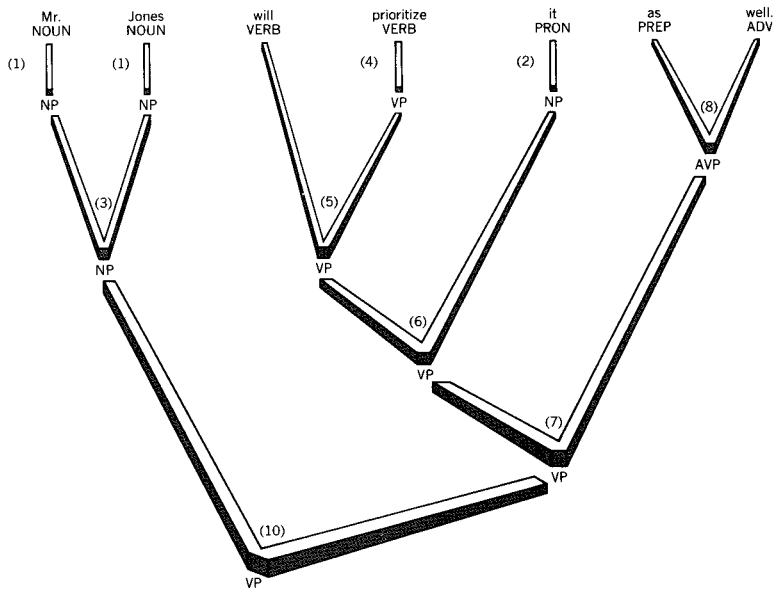
(consisting of a pronoun) "he." "He" has neither premodifiers nor postmodifiers. The postmodifier of "knows" is the VP containing the entire second clause of the sentence. That clause, in turn, has a head, two premodifiers, and two postmodifiers; the subject NP of this clause has one premodifier.

The EPISTLE parser can produce sentence outlines (called "parses" or "parse trees") like the one above for the majority of English sentences encountered in a large data base of business letters. In addition to the outlines, records for each sentence are constructed, containing information about the heads and modifiers of its component phrases. The information contained in these records forms the basis for the grammatical diagnostic work of EPISTLE.

parsing in EPISTLE

Figure 4 gives a set of simplified NLP decoding rules. The actual rules used in the grammar contain much more detail. This list includes the rules that would be necessary to produce the parse in Figure 3. Terms that are on the left of the arrow, outside the parentheses, are the names of the sentence parts that are being put together. Terms on the left of the arrow inside the parentheses are conditions that must be met before the rule can be applied. The term on the right of the arrow outside the parentheses is the name of the sentence part being formed. Terms inside the parentheses on the right of the arrow specify how to create a record structure to describe the new segment being formed. The equal sign is the attribute assignment operator.

Figure 5 A parse tree, showing rule numbers



These rules use information that comes from the dictionary component described earlier. For example, only a word whose part of speech is either noun or pronoun can become the head of a noun phrase, according to Rules (1) and (2). And the condition on Rule (10) checks to see if the number of the verb in the VP matches the number of the subject noun phrase. Since, in the sentence parsed in Figure 3, the dictionary marks the pronoun "he" as singular and marks any verb form ending in *-s* as singular, "he knows . . . " will get put together as a sentence, but "he know . . . " would not.

Figure 5 shows a detailed tracing of the parse for the second clause of the sentence in Figure 3. The numbers in parentheses indicate the rules in Figure 4 that are applied at each step.

This parse reflects the way in which English speakers would understand the sentence "Mr. Jones will prioritize it as well." The subject is the noun phrase "Mr. Jones"; the main verb is "prioritize," with the modal "will" serving as a verbal auxiliary to show future tense; the direct object of the verb is the noun phrase (from the pronoun) "it"; and "as well" is a final adverb phrase, meaning, roughly, "also."

Sometimes a set of grammar rules allows more than one parse for a particular sentence. Notice that the rules as written in Figure 4 allow a second interpretation of the phrase "as well." Since "as" is listed in the dictionary as a preposition (among other things), and "well" can

Figure 6 A second parse for the example sentence

```
SENT    NP      NP      NOUN*   ""MR."
                NOUN*   "JONES"
        VERB    "WILL"
        VERB*   "PRIORITIZE"
        NP      PRON*   "IT"
        PP      PREP    "AS"
                NOUN*   "WELL"
```

be a noun ("John fell into the well"), the phrase in question could be parsed by Rule (9) to yield a prepositional phrase. The output from this second parse is shown in Figure 6.

It is not clear how this parse might be interpreted by an English speaker. "Into the well" would be comprehensible, but "as well" (where "well" means the same thing as it does in "into the well")—? Clearly the grammar is in trouble here if its purpose is to produce a parse that will explain the functions and relationships of the sentence and its parts.

One way around this problem would be to use more of the information supplied by the dictionary and carried in the records. The noun "well" is a *count* noun. Therefore, it should not stand alone as it does in Figure 6. If we are referring to one well, we must say "the well," or "a well," or "my grandfather's well," but not just "well." Therefore, the prepositional phrase in Figure 6 is not a possible construction in English.

A simple addition to Rule (9) in the grammar of Figure 4 can block the unwanted parse

(9) PREP NP(MASS(HEAD)|PRMODS) → PP

The condition added here says that if the NP following a preposition has a head which is a *mass* noun, then it is acceptable, but if it does not, the noun phrase must contain premodifiers (like "the," or "a," or "my grandfather's"). Since "well" is not a mass noun and does not have any premodifiers, it will fail to satisfy the revised Rule (9), and the incorrect parse in Figure 6 will never be produced. This example illustrates the kind of modifications that are regularly made to the EPISTLE grammar during its development.

English grammar coverage

The EPISTLE grammar must cover all the topics traditionally treated in a grammar book. There are rules that describe noun phrases, verb phrases, adjective, adverb, and prepositional phrases, subordinate clauses, participial phrases, and other typical topics of grammar. The current rules can identify the main sentence types: declarative, imperative, question, and exclamatory.

There are groups of rules that treat variations on basic sentence structures. For example, the sentence

He wrote that report last September.

might be varied by moving the final time phrase to the front:

Last September he wrote that report.

or even by moving the direct object to the front:

That report he wrote last September.

All of these variations have to be accepted by the grammar to result in structures that are different in their shape and in the ordering of their parts, but that reveal that those parts have identical functions and relationships.

The EPISTLE grammar currently contains about 250 NLP decoding rules for English. These rules are intended to produce what we call "a unique approximate parse" for each sentence. Such a parse may not always be semantically correct, e.g., some prepositional phrases may be attached incorrectly, but it is adequate for the critiquing tasks of the system.

A benchmark test of the parser was run in December 1981 on 2254 sentences from 411 business letters. Many of these sentences are long and extremely complex (the longest sentence contains 63 words). The average number of words per sentence was 19. The average processing time per sentence was 10 CPU seconds on an IBM 3033 computer, and the average working storage used per sentence was 150K bytes. The grammar produced parses for 64 percent of these sentences (which was up from 43 percent six months earlier). Of the total, 41 percent had single parses, 11 percent had double parses, 11 percent had numbers of parses ranging from three to nine, and 1 percent had ten or more parses.

Although these percentages fall short of the near-perfect score that will be needed for the final implementation, the rate of improvement is strong and continues to be so. Furthermore, since the tested sentences had not been edited before processing, many of them contained grammatical, punctuation, or other errors, which prevented some parses that would otherwise have been obtained. Since EPISTLE is currently implemented in LISP, it is expensive both in time and in space. If these facts are taken into consideration, the performance of the EPISTLE parser seems quite satisfactory at this stage.

Grammar checking

The types of grammatical errors diagnosed by the current system are now presented; then the algorithm used to diagnose them is briefly described. We conclude the discussion with some types of errors that the current system does not handle.

errors currently diagnosed

A catalog of grammar critiques can be compiled in two ways: (1) by consulting textbooks and authorities on English usage (e.g., Warriner and Griffith,[10] Strunk and White[11]), and (2) by reading real business correspondence and watching for stigmatized constructions. Both methods have been used to collect the grammar problems that EPISTLE addresses. The EPISTLE critiques do not cover all possible grammar errors in English, but they do address those that are most often mentioned in the literature and most frequently found in the observed correspondence.

Most grammar errors violate important conditions in the decoding rules and prevent a sentence from being parsed. For example, consider how Rule (10) in the grammar of Figure 4 would interact with the sentence

Your file and your note of correction does not contain proper information.

The condition on Rule (10) says that the number of the subject noun phrase must agree with the number of the verb phrase in order to form a sentence. But the subject of the sentence is the conjoined noun phrase "Your file and your note of correction," which is plural, and the verb phrase "does contain" is singular. Therefore, this sentence could not be accepted by Rule (10), and it could be said that the sentence contains a grammatical error.

EPISTLE currently diagnoses five classes of grammatical errors:

Class 1: Subject-verb disagreement

(a) Your statement of deficiencies *have* not been completed (should be *has*).
(b) Mr. Jones, as well as his assistants, *are* entitled to the commission (should be *is*).
(c) Neither Mr. Smith nor his associates *wishes* to participate (should be *wish*).
(d) Either of the models *are* acceptable (should be *is*).

Class 2: Wrong pronoun case

(a) The Harrison contract was written by Bob Lee and *I* (should be *me*).
(b) The company will sell this product to *whomever* asks for it (should be *whoever*).
(c) I would not advise that course of action, if I were *him* (should be *he*).
(d) If you have any further questions, please call either *myself* or Arthur Hill (should be *me*).

Class 3: Noun-modifier disagreement

(a) These *report* must be checked by our trained personnel (should be *reports*).
(b) Several of the misplaced *memo* were found in the files (should be memos).

(c) *Such large group* cannot be served in the allotted time (should be *such a large group* or *such large groups).*

Class 4: Nonstandard verb forms

(a) The judge cannot forget his *preconceive* notions (should be *preconceived*).

(b) The completed manuscript was *wrote* by Tom Brown and Jeffrey White (should be *written*).

Class 5: Nonparallel structures

(a) We will accept the funds, send receipts to the payers, and *crediting* their accounts at the same time (should be *credit*).

(b) Sorting equipment would *save time, money, and provide greater control* (should be *save time and money, and provide. . .*).

algorithm for grammatical error detection

The EPISTLE system uses the following three steps to detect grammatical errors:

1. Attempt to parse the sentence, using fully grammatical rules (where "fully grammatical" includes conditions on, for example, number agreement between subject and verb). Only sentences that fit the constituent class patterns and obey all conditions on the patterns will be parsed successfully.
2. If the sentence was not parsed in the first step, then try again, but this time with some of the conditions relaxed and with some additional rules.
3. If the sentence is parsed in the second step, then make note of what condition had to be relaxed and where in the sentence the problem occurred, and pass appropriate information back to the interface processor for display to the user.

The general approach stated here is similar to techniques described in Weischedel and Black[12] and in Kwasny and Sondheimer.[13]

errors not currently handled

Although the current system can handle the five classes of errors previously listed, there are other types of errors whose detection requires information that the system does not yet have available. For instance, the phrase "*the the* standard operating procedure" is incorrect, because English does not allow repeated sequences of determiners (words like "the," "a," "these," etc.). But how about "the *standard standard* operating procedure"? It is difficult to be completely sure that "standard standard" is not a legitimate adjective phrase. Certainly "very very good" is acceptable English, although it may be questionable style in the business environment. So it is impossible to state generally that all sequences of identical words are grammatically unacceptable.

In addition to the problem of repeated words, there are three more categories of grammatical errors that the system does not currently handle well. These problems do not necessarily cause the parse to fail, but they do make it difficult to guarantee that the parse is correct.

Class 6: Repeated words (just discussed)

Class 7: Apostrophes

(a) This critique is complimentary to the *writers* effort (should be *writer's*).
(b) Their cooperation is worthy of *thanks'* (should be *thanks*).
(c) It is unfortunate when management does not seem to care about *other's* needs (should be *others'*).

Class 8: Faulty comparisons

(a) His price is much lower than our competitor (how low is our competitor?).
(b) The special features of our operating system justify a higher price than that charged for other inferior systems (is our system also inferior?).

Class 9: Position of modifiers

(a) We received a letter from your secretary, Ms. Hinchley, dated August 16 (which one was dated, the letter or Ms. Hinchley?).
(b) Walking across the aisle, a mail cart hit Mr. Phelps (it's unusual to see a mail cart walking).
(c) You have been most helpful to us in advising our customers (who actually did the advising, you or we?).

Style processing

We define style as the author's strategy for organizing information. It is not, in our terms, simply a lexical phenomenon (that is, it does not have to do only with choice of words.) This section lists several types of style errors and tells how the style critiques were developed, followed by an explanation of a style rule in NLP form. Style errors in the example sentence are then discussed.

Types of style errors

Style can be critiqued on several different structural levels: word, phrase, sentence, and paragraph. Categorizing the errors by level helps to maintain a perspective on the complicated array of advice that is given in textbooks. Some errors that EPISTLE detects are

Word-level critiques

1. "Business-ese" (e.g., "prioritize," "dollarization")
2. Spelling nonpreferred (e.g., "labelled" versus "labeled")
3. Bad connotations (e.g., "hate")

Phrase-level critiques

1. Awkward, redundant or jargonistic phrases (e.g., "effect an alternative procedure," "merge together," "surface the recommendation")
2. Too much qualification of a noun (e.g., "The disk pack holder mount flange tip")

3. Too many intensifiers (e.g., "This seems like a *very very* good idea")

Sentence-level critiques

1. Sentence too long (or too short)
2. Too many negatives (e.g., "They don't know nothing," "That is a not unwise decision")
3. Too many "attitude" verbs (e.g., "I *know* that you *think* that I *believe* that you *feel* confused")

Paragraph-level critiques

1. Too many passive sentences (e.g., "All requirements for that program have not *been fulfilled*. . . . Eligibility may *be attained* by you if the deficiencies can *be overcome*")
2. Too many compound or complex sentences (i.e., sentences containing more than one clause)
3. Poor readability score (as measured by some standard readability index)

Another system that does style critiquing is the "Writer's Workbench."[14-16] It does many of the same style critiques as EPISTLE, but because it does not have a parser, it cannot do critiquing that requires having a parse tree for the sentence, such as "subject-verb distance too great."

The development of style critiques

The EPISTLE style critiques were developed in five stages. First, principles of good style (especially those for the business environment) had to be identified. This stage was done largely by reviewing several text and "how-to" books on writing (e.g., Bates,[17] Cloke and Wallace,[18] Dyer,[19] Wilkinson *et al.*[20]). For example, three commonly agreed-upon principles of good business style are

1. Prefer the active voice
2. Use strong verbs
3. Make sentences readable

The second stage in developing style critiques was to determine the ways of violating each principle. For example, the readability principle could be violated in several ways:

a. Too many words in the sentence
b. Too many dependent clauses
c. Too great a distance between the subject and the verb

Given this expansion of a principle into classes of violation, the third step was to identify for each the specific grammatical cues that could be used to detect an instance of violation and its severity. For example, violation class *c* above has two obvious grammatical cues: the locations of the subject and of the verb; these can be identified as *Loc(Subj)* and *Loc(Verb)*. Less obvious, perhaps, is the consideration

of the syntactic nature of the text intervening between the subject and verb. If one had psychological evidence that people's comprehension of sentences becomes poorer as the number of nouns intervening between the subject and verb increases, this could then be a third syntactic cue, identified as *Num(Noun)*. All of this can be summarized by saying that the subject-verb distance critique, *CRIT(SVD)*, is some function F of these cues:

$$\text{CRIT(SVD)} = \text{F(Loc(Verb), Loc(Subj), Num(Noun))}.$$

The fourth step was to specify a function that combines these syntactic cues into a single number. A primary criterion for selecting the function is that the value of the function should increase more or less linearly with perceived "badness" of style. Furthermore, the function should not produce anomalous results for small, zero, or other critical values of its variables (e.g., as when the sentence being evaluated is imperative and has no subject). The following function has these desired properties. It contains a weighting factor w for Num(Noun); the absolute value handles the cases where subject and verb are inverted (as in questions):

$$\text{CRIT(SVD)} = |\,\text{Loc(Verb)} - \text{Loc(Subj)}\,| \times (1 + w \times \text{Num(Noun)})$$

A value of 0.2 for w gave good results on our test sentences.

The fifth and final step in the development of specific style critiques was to establish one or more thresholds against which to compare the values of the critique functions. When the value of a function exceeds the threshold, a violation is reported; the lower the threshold, the more sensitive is the function in reporting style violations. For example, with w set to 0.2, if the threshold for the above function were set to 10.1 and a sentence occurred in which the subject was in the second word location (e.g., "The man. . ."), with no nouns intervening between subject and verb, the verb would have to be in word location 13, or further, before the value of the CRIT(SVD) function would signal a *hit*. If there were two intervening nouns in such a sentence, a verb in location 10 or further would result in a hit.

Thresholding, together with adjustable weights, allows tailoring of style critiques to individual environments without rewriting the functions. Users can vary the thresholds to obtain very critical or gross-level critiques.

NLP encoding rules for style error diagnosis

The diagnosis of style errors in EPISTLE is done by a set of NLP encoding rules which are consulted during a top-down traversal of the parse tree of a sentence. One such rule is explained here after the error it detects is described.

On the phrase level, a construction that is often blacklisted by style books is the "split infinitive," where words are inserted between "to" and a verb, as in "to *finally* finish the book." According to some

books, it is acceptable to split an infinitive with one short adverb, but not with more. "To *quickly* read the book" is permitted, but "to *as fast as possible* read the book" is not.

The following is a slightly simplified NLP encoding rule that identifies split infinitives and issues an advisory message:

```
ERRTST(SEGTYPE.EQ.'INFPH',COUNT.GE.THRESHOLD) →
    ERRTST(ERRORS=ERRORS. . .<'ERROR',STYLERR='PHRASELEVEL',
            TYPE='SPLIT-INFINITIVE',
            MESSAGE="IT MIGHT BE BETTER TO MOVE THE INTERVEN-
            ING MATERIAL">)
```

When an infinitive phrase (type INFPH) is encountered, and the COUNT attribute of the phrase is greater than or equal to the threshold for split infinitives, this rule is invoked. COUNT, the number of words between "to" and the verb, is calculated during the parsing of the sentence. For example, if the threshold were two, and the phrase "to *very quickly* read" had been parsed, then the count would equal two and both conditions on the application of the rule would be satisfied.

If the conditions are satisfied, the rule is applied and a new record is created with an ERRORS attribute added to it, as can be seen from the specifications on the right side of the rule. The ERRORS attribute points to a record that contains information about the level of the construction, the error type, and the diagnostic message to be displayed.

Style errors in the example sentence

Having described the mechanism for detecting stylistic weaknesses, let us return to the sentence, "He knows Mr. Jones will prioritize it as well." There are two aspects of this sentence that would require some processing by the style rules.

First, according to Figure 2, the verb "knows" belongs to the class of *attitude* verbs. Too many attitude verbs in one sentence will trigger a style critique. What constitutes "too many" is determined by the threshold. It would be unwise to be critical of the use of just one attitude verb such as "knows," so the threshold would probably never be set to 1, and no diagnostic message would be produced for this sentence. The system would, however, have noted the presence of one such verb and would have identified more if they had occurred.

Second, the verb "prioritize" is one of the jargon words that most style manuals warn against. Since EPISTLE's style component checks for such words, a critique would be produced, suggesting that the author substitute some other verb such as "rank" or "order."

Status and plans

In its current form, the EPISTLE system is not yet ready for general use. In the two years that it has been under development, the most

concentrated effort has been on developing and testing the grammar. We are satisfied with our progress toward a grammar that will cover almost all sentences that appear in business letters. During this same time we have improved the underlying natural language processing system and have demonstrated a capability for doing grammar checking and style checking. Also, a considerable amount of work has been done with an on-line version of a standard dictionary to produce the part-of-speech information that is used by the parser.

At present, we are working on several items toward the goal of making an experimental version of EPISTLE available. We are implementing techniques for dealing with sentences that are only partially parsed by the grammar, and we are refining techniques for dealing with sentences that result in multiple parses.[21] Then we have to expand the grammar-checking and style-checking functions from a demonstration level to a level of real usefulness. Along with this, work on the dictionary is continuing, both to improve the access times and to provide more syntactic and semantic information. As stated earlier, the underlying natural language processing system is being reimplemented in a more efficient language to reduce the time and space requirements.

In the longer term, in addition to handling whatever deficiencies are found by experimental use, we will increase the amount of semantic information that is available to the processor so that the system can diagnose the more difficult kinds of errors not currently handled. Critiquing at the paragraph level would also become possible if the system could represent the "meaning" of each sentence as a series of related propositions and then assess their continuity by comparing them to known patterns of good exposition. We would also like to do the semi-automatic creation of first drafts, using some techniques discussed in Jensen.[22] Eventually, there is the second class of applications mentioned in the introduction to be worked on, dealing with the synopsizing, indexing, and retrieving of incoming documents.

ACKNOWLEDGMENTS

We would like to thank Evon Greanias, Tony Hwang, and Ken Niebuhr for their various contributions to the EPISTLE system, and John Sowa and the referees for their careful reading of the manuscript.

CITED REFERENCES

1. L. A. Miller, "A system for the automatic analysis of business correspondence," *Proceedings of the First Annual National Conference on Artificial Intelligence,* Stanford University (1980), pp. 280–282.
2. L. A. Miller, G. E. Heidorn, and K. Jensen, "Text-critiquing with the EPISTLE system: An author's aid to better syntax," *AFIPS Conference Proceedings* **50,** 649–655 (May 1981).
3. G. E. Heidorn, *Natural Language Inputs to a Simulation Programming System,* Naval Postgraduate School Technical Report No. NPS-55HD72101A (1972). (Copies are available from the author at the IBM Thomas J. Watson Research Center in Yorktown Heights, NY.)

4. G. E. Heidorn, "Augmented phrase structure grammars," in *Theoretical Issues in Natural Language Processing,* Editors: B. L. Nash-Webber and R. C. Schank, Association for Computational Linguistics, Menlo Park, CA (1975).
5. *LISP/370 Program Description/Operations Manual,* SH20-2076-0, IBM Corporation (1978); available through IBM branch offices.
6. D. H. H. Ingalls, "The Smalltalk-76 Programming System: Design and implementation," *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages* (1978), pp. 9–16.
7. R. A. Amsler, *The Structure of the Merriam-Webster Pocket Dictionary,* Doctoral Dissertation TR-164, University of Texas, Austin, TX (1980).
8. M. Aronoff, *Word Formation in Generative Grammar,* M.I.T. Press, Cambridge, MA (1976).
9. L. A. Miller and G. Carriero, *The Automatic Identification of Facts in Natural Language Text,* Research Report, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 (to be published in 1982).
10. J. E. Warriner and F. Griffith, *English Grammar and Composition,* Harcourt, Brace, and World, Inc., New York (1963).
11. W. Strunk, Jr. and E. B. White, *The Elements of Style,* Third Edition, Macmillan Publishing Company, Inc., New York (1979).
12. R. M. Weischedel and J. E. Black, "Responding intelligently to unparsable inputs," *American Journal of Computational Linguistics* **6,** No. 2, 97–109 (April–May 1980).
13. S. C. Kwasny and N. K. Sondheimer, "Relaxation techniques for parsing grammatically ill-formed input in natural language understanding systems," *American Journal of Computational Linguistics* **7,** No. 2, 99–108 (April–June 1981).
14. L. L. Cherry, *PARTS—A System for Assigning Word Classes to English Text,* Bell Laboratories Computing Science Technical Report No. 81, Bell Laboratories, Murray Hill, NJ (1978).
15. L. L. Cherry, *Writing Tools—The STYLE and DICTION Programs,* Bell Laboratories Computing Science Technical Report No. 9, Bell Laboratories, Murray Hill, NJ (1980).
16. N. Macdonald, *Pattern Matching and Language Analysis as Editing Support,* paper presented at the American Educational Research Association Meeting, Boston (April 1979).
17. J. D. Bates, *Writing with Precision,* Acropolis Books, Ltd., Washington (1978).
18. M. Cloke and R. Wallace, *The Modern Business Letter Writer's Manual,* Doubleday and Co., New York (1969).
19. F. C. Dyer, *Executive's Guide to Effective Speaking and Writing,* Prentice-Hall, Inc., Englewood Cliffs, NJ (1962).
20. C. W. Wilkinson, P. B. Clarke, and D. C. M. Wilkinson, *Communicating through Letters and Reports,* Seventh Edition, Richard D. Irwin, Inc., Homewood, IL (1980).
21. G. E. Heidorn, "Experience with an easily computed metric for ranking alternative parses," presented at the *Twentieth Annual Meeting of the Association for Computational Linguistics,* Toronto (June 1982); also to be published in 1982 as a Research Report, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.
22. K. Jensen, "Computer generation of topic paragraphs: structure and style," presented at the ACL sessions of the *56th Annual Meeting of the Linguistics Society of America,* New York (December 1981); also to be published in 1982 as a Research Report, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

G. E. Heidorn, K. Jensen, L. A. Miller, and R. J. Byrd are located at the IBM Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598; M. S. Chodorow is with the Psychology Department of Hunter College, City University of New York, 695 Park Avenue, New York, NY 10021.

Reprint Order No. G321-5171.