CN YUR CMPUTR RAED THS?

Linda G. Means
Computer Science Department
General Motors Research Laboratories
Warren, Michigan 48090

ABSTRACT

This paper describes strategies for automatic recognition of unknown variants of known words in a natural language processing system. The types of lexical variants which are detectable include inflexional aberrations, ad hoc abbreviations and spelling/typographical errors. This technique is independent of any particular grammar or parsing formalism, and can be implemented as a lexical lookup routine which heuristically prunes and orders the list of possible fixes found in the lexicon, then allowing the parser to treat the list of candidates as a set of multiple meanings for a polysemous word.

1 INTRODUCTION

This paper describes a technique for automatic recognition of unknown variants of known words in a natural language processing system. "Known word" refers here to a word which is in the lexicon. The types of lexical variants which are detectable include inflexional aberrations, ad hoc abbreviations and spelling/typographical errors. The strategies presented here have been implemented fully in an English database query system and play a crucial role in a text-understanding system which is in the early stages of design. This technique, however, is independent of any particular grammar or parsing formalism, and can be implemented as a lexical lookup routine which heuristically prunes and orders the list of possible fixes found in the lexicon. First, a context-free plausibility assessment is based on a comparison of the structure of each candidate fix with that of the unknown word, and determines the order in which fixes will be considered by the parser. Then, the parsing process can choose among the candidate fixes in the same way that it tests multiple meanings of polysemous words for a good syntactic and semantic fit. The use of heuristics to identify the most plausible fixes for a hypothesized ad hoc abbreviation or spelling error will be the focus of this paper.

Unknown words have traditionally been handled by natural language processing systems in the following ways:

1. Query the user for a replacement, possibly offering a menu of spelling corrections. This strategy will allow correction of misspelled words as well as correctly spelled words which are not in the lexicon, and generally ensures an accurate interpretation by the computer. However, continued interaction of this sort may prove frustrating to a poor typist, and is, of course, unsuitable for a non-interactive natural language processor.

2. Enter into a dialogue with the user to provide a definition for a new word. This strategy requires a lexicon interface based on a metalanguage which would specify grammatical properties for a word without necessitating an inordinate degree of linguistic sophistication or knowledge of the database on the part of the end user. Although various attempts have been made to design such interfaces[1], many outstanding research issues remain, and this approach too requires an interactive environment.

3. Try to infer syntactic and/or semantic features of the unknown word from the linguistic context, with no user interaction. This strategy can be used to choose a plausible correction for a misspelled word as well as to parse an expression containing an unknown word. Early research in this area attempted to model human reasoning about unknown words in a script-based parser [5], and has since come to encompass a variety of multistrategy, expectation-based techniques as exemplified in the DYPAR [2] and NOMAD [4] systems. This technique shifts the burden of linguistic expertise from the end user to the computer system, but has met so far with only limited success, and accuracy can only be assured by interaction with the user to

[1]Two outstanding examples are the TELI [1] and TEAM [6] systems.

confirm the hypothesized interpretation. An additional limitation to this approach is that many existing natural language parsers cannot accomodate this sort of analysis.

At General Motors Research Laboratories we have encountered two situations in which the traditional interactive strategies are inadequate, and where it is preferable to try to fix an unknown word automatically.

The first involves Datalog, our research prototype of an English database query system [7]. This system was designed with the philosophy that a more humanlike sort of interaction is obtained by letting the system reason about ambiguities of all sorts as well as it can, always informing the user of its interpretation before displaying the response. Automatic lexical correction satisifes the objectives of this design principle.

A more compelling need for this capability has arisen in a text-understanding project that we have undertaken, which aims to read and summarize the content of free-form text records in a diagnostic database. The cases are entered by automotive technicians in the process of solving a variety of vehicle failure problems referred to them by service personnel in dealerships. The technicians are generally people who are not expert typists, may not have excellent language skills, and have a time constraint imposed by a heavy load of calls. Also, two of the three free-form text fields in the database are abstract lines which are limited to a few words, which imposes a severe space constraint. As a result, the language used in the free-form text tends to be highly ill-formed, with an abundance of ad hoc abbreviations and typographical and spelling errors. And although the technicians' least compelling concern at data-entry time is linguistic in nature, their errors come back to haunt them, as the lexical errors impede the success of subsequent keyword searches done to find analogous cases in the database. So automatic lexical correction, in addition to being necessary to our text-understanding task in the longer text field, could also be of service in making corrections to the abstract lines in the database. The size of the database precludes the feasibility of an interactive system which would read text from existing cases and query a program operator for fixes.

Incidentally, all examples used here of spelling errors and ad hoc abbreviations are taken from our free-form text data.

2 FLEXIBLE MORPHOLOGY

One common type of spelling error involves spelling changes in base forms when adding an inflexional suffix. Two sorts of errors must be addressed in this area: failure to make a necessary spelling change (e.g. *plug/pluged*), and the occurrence of inappropriate spelling changes (*come/comming*). Inflected forms containing either of these errors can be detected by a forgiving morphology algorithm.

Our algorithm currently recognizes only one prefix and/or suffix per word. Flexibility regarding inflexional spelling changes pertains only to suffixes; although additional suffixes can be specified easily, those currently recognized are:

-s	(noun, verb)	*-er*	(adj)
-ed	(verb)	*-est*	(adj)
-ing	(verb)	*-ly*	(adj, adv)
-ment	(verb)		

The flexible morphology algorithm looks for a known suffix at the end of an unknown word. If found, the suffix is stripped off, and the remainder, a postulated base form, is looked up in the lexicon. If not found, spelling change transformations are performed on the base form, and lexical lookup is performed after each transformation. Morphological analysis succeeds when a postulated base form is found in the lexicon with a syntactic category which is compatible with the suffix.

Consider the two ill-formed inflexions mentioned above. For an unknown word *pluged*, the *-ed* suffix is stripped off. The remainder, *plug*, is found in the lexicon as a verb, so morphology succeeds. For the unknown word *comming*, the postulated base form after stripping off the *-ing* is *comm*. Transformations specified for an *-ing* suffix are:

1. If base ends in a double consonant, reduce it.
2. If base ends in a single consonant, append an *-e*.
3. If base ends in *-i*, change to *-y*.

In the case of *comm*, morphology succeeds after performing the first two transformations and finding the verb *come* in the lexicon.

This algorithm has proven quite effective in recognizing inflected forms which contain common errors such as erroneously doubling a word-final consonant or failing to double a consonant before an inflected suffix; failure to drop word-final *-e* before *-ing*; and failure to change *-y* to *-ie* before adding *-s* or *-ed*.

Morphology is the first fix tried for an unknown word, as it is less computationally intensive than detection of spelling errors and ad hoc abbreviations, and occurs more frequently.

3 AD HOC ABBREVIATION RECOGNITION

When morphology fails to detect a variant of a known word, an ad hoc abbreviation or spelling error is hypothesized. Each possible abbreviation expansion found in the lexicon is assigned a plausibility score on the basis of a comparison of the structure of the unknown word and of the potential fix.

There are definite identifiable tendencies used by humans to abbreviate words. Although the tendencies which will be discussed here have not been tested empirically, our abbreviation-plausibility heuristic which incorporates them performs well in ordering lists of candidate abbreviation expansions. Further experimentation with the algorithm will undoubtedly produce even better results, but the heuristics described here have performed satisfactorily.

Abbreviation occurs as a result of truncation or contraction. To identify candidate abbreviation expansions for a postulated abbreviation, the lexicon is searched for words whose initial substring coincides with the unknown word (truncation-type fixes) and words which contain all letters of the unknown word in the same relative order (contraction- type fixes). Bear in mind that another property of abbreviations is that they generally begin in the same letter as the word from which they are derived[2], so it is only necessary to search the portion of the lexicon beginning in the same letter as the unknown word, and it is only necessary to consider lexical words whose length exceeds that of the unknown word.

Finding possible expansions of an abbreviation is a simple task.[3] Rating their plausibility in the absence of contextual evidence is more difficult. For each possible abbreviation expansion found in the lexicon for an unknown word, a comparison is made between the structure of the unknown word and that of the expansion, and the candidate fix is assigned to one of five categories, which serves as a measure of its plausibility. Distinct heuristics are used to rate contraction-type and truncation-type fixes. The criteria used to classify candidate fixes and the ordering of the five plausibility categories are described below.

3.1 Truncation-type Abbreviations

Truncation-type fixes are classified as either plausible or implausible, based on the extension string, i.e. the string which is chopped off to abbreviate a word. I will refer to the two truncation classifications as trunc-good (plausible) and trunc-bad (implausible).

If the extension consists entirely of vowels or entirely of consonants, it is classified as trunc-bad, as people generally tend to truncate words by deleting at least an entire syllable. If the unknown word ends in a vowel and the extension begins in a consonant, again it gets a trunc-bad rating, as people tend to truncate words by using the entire initial syllable(s), plus the initial consonant(s) of the following syllable if the preceding syllable ends in a vowel. If the entire initial consonant cluster of the expansion does not occur at the start of the unknown word, assign trunc-bad.

Any truncation-type fix not classified as trunc-bad is rated as trunc-good (plausible). So the lexical word *hesitation* is classified as trunc-bad for an unknown word *hesi*, and trunc-good for an unknown word *hes*.

3.2 Contraction-type Abbreviations

Three degrees of plausibility are distinguished for contraction-type fixes. Because the elimination of vowels from a word is a common contraction strategy, and the absence of vowels in an unknown word is a strong indicator of an ad hoc abbreviation as opposed to a spelling error, the highest degree of plausibility is assigned to those lexical words from which only vowels have been excised to derive the unknown word. The classification designated for such fixes is called missing-vowels.[4]

[2]The rare exceptions such as *xmit* for *transmit* or *xtra* for *extra* will not be treated here.

[3]The only exceptions are those infrequent abbreviations which contain letters not occurring in the expansion, like *no.* as an abbreviation for *number*. These tend to be common abbreviations which should be entered into the lexicon; ad hoc abbreviations generally do not have this characteristic.

[4]The missing-vowels criterion requires two qualification. If a doubled consonant in the abbreviation is simplified, the abbreviation can still be classified as missing-vowels.

For an unknown word *assm*, for instance, *assume* will be classified as a missing-vowels fix.

For contraction-type fixes in which consonants as well as vowels have been excised to derive the unknown word, one of two classifications is assigned: contract-good (plausible) and contract-bad (implausible). The favorable rating is assigned if none of the following criteria for an implausible rating apply:

1. If the abbreviation cannot be derived from the lexical word by removing a single substring, assign contract-bad. This criterion reflects the common tendency to contract a word by using some initial portion plus some final portion. For an unknown word *ht*, some fixes classified as contract-bad for this reason are *hatch*, *hertz* and *heater*.

2. If the abbreviation differs from the lexical word by a single substring, but the substring contains only consonants, assign contract-bad. This is motivated by the fact that at least one entire syllable is generally removed to contract a word. For an unknown word *satch*, the fix *scratch* is classified as contract-bad.

3. If the abbreviation differs from the lexical word by a single substring, but a vowel immediately precedes or follows the substring in the expansion, assign contract-bad. The implausibility here arises from the fact that a vowel in the abbreviation is adjacent to a different consonant than in the lexical word, which is unlikely, or that two adjacent vowels in the abbreviation are not adjacent in the lexical word, which is even less likely. Consider as an example the fix *regulate* for an unknown word *reate*.

4. If the entire initial consonant cluster of the lexical word does not occur at the start of the abbreviation, assign contract-bad, e.g. *strain* as a fix for *stn*.

Consider the classification of fixes from our lexicon for an unknown word *compt* as an example of the ordering capability of the contraction heuristics:

missing-vowels	contract-good	contract-bad
compute	*compartment*	*consumption*
	compact	*compensate*
	component	*computer*
	complaint	*composite*

3.3 Ordering of Plausibility Categories for Abbreviations

The five abbreviation plausibility classifications themselves are ordered in terms of plausibility. From high acceptability to low, the ordering is:

1. missing-vowels
2. trunc-good
3. contract-good
4. trunc-bad
5. contract-bad

As a result, missing-vowels fixes are always preferred over truncation or other contraction fixes. Plausible truncation fixes are preferred over plausible contraction fixes, but plausible contraction fixes are preferred over implausible truncation fixes. For an unknown word *spr*, fixes are found in our lexicon for four of the five classifications:

missing-vowels	trunc-good	contract-good	contract-bad
spare	*sprocket*	*speaker*	*separate*
super	*spring*	*spacer*	*spark*
	spray	*speedometer*	*sport*
	sprint	*sputter*	*sulphur*
		spicer	*suppressor*

3.4 Interaction of Morphology and Abbreviation Detection

Abbreviations are easily recognizable when inflected with suffixes known to the morphological analyzer. If an unknown word ends in a known suffix, the suffix is stripped off, and if the remainder does not end in a vowel[5], two comparisons are made with each lexical entry: one with the entire unknown word, and one with the suffix stripped off, checking for syntactic category compatibility between the suffix and the lexical definition. So for an unknown word such as *outs*, expansions will be found for the entire word (e.g. *outside*), and also for *out*, with a syntactic category restriction of noun or verb (e.g. *outlet*, *outline*).

It is not unusual to encounter an abbreviation with an inflected suffix. It is unusual, however, for

For instance, the abbreviations *probbl*, *probl*, *prbbl* and *prbl* all have a missing-vowels relationship with the expansion *probable*. Also, for each vowel removed to derive the abbreviation, all adjacent vowels in the expansion must also have been removed. Thus, *count* is not classified as a missing-vowels fix for an unknown word *cont*.

[5]Abbreviations rarely end in a vowel (see Section 5), so if the string left after stripping off a suffix ends in a vowel, the unknown word is not likely to be an abbreviation.

an ad hoc abbreviation to incorporate the spelling changes required to inflect the expansion. Given the contraction *assy* for *assembly*, for instance, the plural form of the abbreviation is likely to be *assys*, not *assies*. As a result, I don't see a need to perform the usual spelling change transformations on an inflected abbreviation.

4 SPELLING CORRECTION

The currently implemented spelling corrector is capable of identifying five types of spelling/typographical errors: wrong letter, missing letter, extra letter, transposed letters, and missing blank (incorrect segmentation in which two words are run together). The first four of the error categories listed above have been found to account for over 80% of the spelling errors found in studies performed with end users [3], [9]. Like the abbreviation detector, the spelling corrector only treats words which consist entirely of alphabetic characters.

4.1 Spelling Error Plausibility Assessment

Unlike the abbreviation plausibility strategies, the spelling correction plausibility heuristics cannot identify implausible fixes; they can only distinguish the most highly plausible fixes from the other fixes, which are considered plausibility-neutral.

No discriminators have been established for missing-blank fixes. In the wrong-letter category, the plausibility rating is boosted for fixes in which the correct letter phonologically resembles its erroneous replacement (e.g. substitutions within sets of sibilants, vowels, or nasals), or when the two letters are adjacent on the keyboard.

Missing-letter fixes are rated by comparing the missing letter with the letters that precede and follow it in the fix. A fix is assigned a higher score if a missing vowel is preceded or followed by another vowel (reflecting the common spelling error of reducing a diphthong, triphthong, vowel digraph or trigraph), or if a missing consonant is preceded or followed by an identical consonant (capturing the propensity to reduce doubled consonants).

Extra-letter fixes are also evaluated on the basis of the letters preceding and following the extra letter. A higher rating is assigned to those fixes in which an extra consonant is preceded or followed by an identical consonant, or an extra vowel occurs adjacent to another vowel, capturing the proclivity to unnecessarily double consonants or create diphthongs or triphthongs. Transposed-letters fixes are considered plausible when the two transposed letters are adjacent (even more so when adjacent transposed letters are both vowels), and when two consonants have been transposed around vowels.

Other researchers have expressed doubt about the feasibility of searching a large lexicon for all possible spelling corrections [2], preferring an expectation-based search in which the syntactic and/or semantic features expected by the parser upon encountering an unknown word are used to prune the search space for corrections. For a parser which can accommodate this approach, perhaps a combination of expectation-based pruning and plausibility heuristics would yield the best results. Even in the absence of parser expectations, a couple of strategies can be employed to reduce the search space considerably.

First, for the five categories of errors which we can detect, a lexical word need not be considered as a fix unless the length differential with the unknown word is less than 2 (accounting for all error types except missing blank), or unless the unknown word begins in the lexical word (in the case of a missing blank).

Secondly, another pruning strategy which merits mention is that of searching only the subset of the lexicon beginning in the same letter as the unknown word. An examination of one abstract field in 11,000 cases from our diagnostic database revealed only two misspelled words in which the initial letter differs from that of the correction (*oight* for *light* and *irratic* for *erratic*). There occurred in the same data set 1207 misspelled words beginning in the same letter as the correction. Perhaps typists tend to notice and correct a misspelling more readily when the initial letter is in error. At any rate, our data seem to indicate that this pruning strategy is not unreasonable, particularly in an interactive application like database query where the fail-safe method of querying the user exists in the event of spelling correction failure.

When spelling correction fails to find an acceptable fix beginning in the same letter, other lexical subsets may be searched as a last resort, chosen perhaps on the basis of phonetic similarity or keyboard adjacency to the first letter of the unknown word (which would prove successful in our exceptional cases of *oight* and *irratic*).

Using the pruning strategies of a <2 length differential in a same-first-letter subset of the lexicon, we have experienced very good response time with a lexicon of over 2600 words, which would be a good indicator of success for applications like database query where lexicons tend to be rather limited.

4.2 Interaction of Morphology and Spelling Correction

The interaction of the spelling corrector with the morphological analyzer is more problematic than abbreviation-morphology interaction. The difficulty lies in the occurrence of morphological spelling changes in inflected forms (recall that inflected abbreviations generally do not incorporate the usually requisite spelling changes). For spelling correction as well as abbreviation recognition, two comparisons are made with each entry in the appropriate lexical subset when the unknown word ends in an inflected suffix: one comparison with the entire unknown word, and a second with the postulated base form, checking for syntactic category compatibility with the possible fix.

One obvious problem is that while our spelling correction algorithm can identify one and only one spelling error per word, an inflexional spelling change in an already misspelled word results in two deviations from the lexical word. Consider, for instance, *seperating* as a misspelled inflexion of *separate*. After stripping off the *-ing* from the unknown word, a comparison of the postulated misspelled base form *seperat* with the lexical word *separate* will fail.

One possible solution to this problem would involve the use of an inflect-word function which composes its base and suffix arguments into an inflected form with requisite spelling changes. The lexical search for fixes then would compare the inflected unknown word with the inflected form of the lexical word (e.g. compare *seperating* with *separating*, computing the latter form from the lexical entry *separate* inflected with suffix *-ing*).

5 ADJUDICATION BETWEEN SPELLING AND ABBREVIATION FIXES

Once the lists of abbreviation fixes and spelling fixes have been ordered internally, priority must be given to one of these categories. Although a numerical score is assigned to each fix, the relativity of the scores holds only within the broader categories of spelling and abbreviation, and not across categories. Various criteria may be applied to determine whether the unknown word is more likely to be an abbreviation or a misspelling.

While there occasionally occur such fortuitous signals of abbreviation as a word-final period or an apostrophe before an inflected suffix (e.g. *repl'ed* for *replaced*), other subtle discriminators can also be identified. With the exception of standard abbreviations for U.S. states and a few other common abbreviations (e.g. *limo*, *demo*, *info*), abbreviations of single words (as opposed to acronymic abbreviations) rarely tend to end in vowels. An unknown word ending in a vowel, then, is much more likely to be a misspelling than an ad hoc abbreviation.

Consider too the fact that while the spelling heuristics fail to identify implausible fixes, the abbreviation heuristics do identify classes of implausibility for truncations as well as contractions. Generally speaking, abbreviation fixes classified as trunc-bad or contract-bad are implausible to such a high degree that they should not be considered at all when more plausible abbreviation fixes or any spelling fixes have been identified. So the final ordering of the entire list of fixes may be established by listing spelling fixes first (or maybe only) when the unknown word ends in a vowel or when no abbreviation fix is classified higher than trunc-bad, and placing abbreviation fixes before spelling fixes otherwise.

6 ROLE OF THE PARSER

The Datalog parser has a Cascaded ATN architecture, in which semantic feedback is provided to the working parser during the parse. A successful parse yields a semantic representation of the input expression as well as a syntactic parse tree. Lexical ambiguity is resolved by the parser through the rejection of unacceptable interpretations by either syntax or semantics. Multiple abbreviation and/or spelling fixes for an input word are treated by the parser as cases of lexical ambiguity, differing only in how the user is informed of the fix when the response is displayed. In the original implementation of the lexical correction algorithm in Datalog, even though the list of candidate fixes found during lexical lookup was not ordered or pruned before parsing, good results were obtained in the recognition of spelling errors and abbreviations. The incorporation of plausibility heuristics, however, reduces parsing time considerably, as they eliminate unlikely candidates from consideration before parsing while also ordering candidates in terms of plausibility, thus obtaining a successful parse earlier.

7 FURTHER RESEARCH ISSUES

Although initial experimentation with the lexical correction algorithm has yielded fairly good results in our database query system and text-understanding project, many research issues remain unresolved. These pertain primarily to lexical variants which the algorithm cannot identify as possible fixes.

7.1 Robust Spelling Correction

Spelling errors which are not handled include extra blank (incorrect segmentation which splits one word into two), more than one letter in error (e.g. *droan* for *drone*), and misspellings which coincide with a lexical word (this would be a rare occurrence for an ad hoc abbreviation). Further work is needed to design a more robust spelling correction algorithm which can account for a greater variety of spelling errors as well as discriminate among good and bad correction candidates[6]. The five types of spelling errors found by our program are often the result of keyboard slips instead of a misconstrual of the correct spelling. An ideal spelling correction algorithm would assess the differences between an unknown word and a postulated fix in terms of keyboard layout and typing habits, as well as phonetic similarity and interference from other words with a similar pronunciation (e.g. *fluxuate* for *fluctuate*).

7.2 Unknown Words

Identification of truly unknown words (those which should be added to lexicon) is difficult. We get around this problem with our free-form text by preprocessing all new text to find new words to add to the lexicon. Morphology weeds out inflected forms of known words. A list of unknown words is then sent to a lexicon building program, which allows a lexicon builder to make decisions about each word and automatically creates a lexical definition with features selected by the user. Misspellings, abbreviations and inflexions can be skipped over easily, or renamed as new lexical entries. In the lexical lookup stage, an acceptability threshold can be established below which a fix will be rejected from consideration, so if no candidate fixes exceed the threshold, none will be considered by the parser.

[6]An excellent source of information on spelling correction algorithms is Peterson's annotated bibliography in [8].

7.3 Misspelled Inflexional Suffixes

Spelling correction of inflexional suffixes is lacking. Morphology cannot recognize a word with a misspelled suffix such as *engagemant*, and as only the base form *engage* occurs in the lexicon, spelling correction cannot find the correct fix either. Perhaps spelling correction of inflexional suffixes can be implemented as a last resort measure. I haven't seen much of a need for it, possibly because the suffixes we identify are quite short, although it could be a problem if multiple suffixes were recognized by morphology (e.g. *standardization* = *standard* + *ize* + *ation*).

7.4 Abbreviation Irregularities

Misspelled abbreviations are another difficulty; the algorithm will not recognize *accell* as a truncation of *acceleration*.

Abbreviated inflected suffixes on abbreviations are also beyond our current capabilities; whereas *repled* can be recognized as an inflected contraction of *replaced*, *repld* cannot be recognized (and the latter is probably the more likely contraction).

7.5 Short Words

One- and two-letter words are difficult to fix because of the high number of candidate fixes that are found. We do not attempt to fix one-letter words, and generally find an unwieldy number of spelling fixes as well as abbreviation fixes for two-letter words.

7.6 Syllabic Analysis of Abbreviations

We have not yet experimented with a syllable-based comparison of the unknown word with candidate fixes as a method of context-free plausibility assessment. Although this approach may prove to be more effective, it would also be more computationally intensive and may be unnecessary. The strategies described here have been designed to capture many of the same generalities regarding abbreviation plausibility which would be inherent to a syllable-based approach.

8 CONCLUSION

Several decades of research in natural language processing have resulted in significant advances in our ability to parse well-formed input within a well-specified domain. One challenge which we now face is the ability to forgive linguistic deviations which do not obscure meaning.

The lexical correction techniques described herein appear to be promising for natural language applications in which it is necessary to curtail user interaction in resolving ambiguities. Even in a more interactive environment, the usefulness of this capability should not be ruled out. It accommodates users who lack good spelling and/or typing skills by forgiving spelling errors and by allowing considerable conservation of keystrokes through ad hoc abbreviation recognition.

9 ACKNOWLEDGEMENTS

Many thanks to my GMR colleagues Kurt Godden and Sam Uthurusamy for their continued interest and feedback on this topic.

References

[1] Bruce W. Ballard and Douglas E. Stumberger. Semantic Acquisition in TELI: A Transportable, User-Customized Natural Language Processor. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 20–29, 1986.

[2] Jaime G. Carbonell and Philip J. Hayes. Recovery Strategies for Parsing Extragrammatical Language. *American Journal of Computational Linguistics*, 9(3-4):123–146, 1983.

[3] Fred J. Damerau. A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the ACM*, 7(3):171–176, 1964.

[4] Richard H. Granger. The NOMAD System: Expectation-Based Detection and Correction of Syntactically and Semantically Ill-Formed Text. *American Journal of Computational Linguistics*, 9(3-4):188–196, 1983.

[5] Richard H. Granger, Jr. FOUL-UP: A Program that Figures Out Meanings of Words from Context. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 172–178, 1977.

[6] Barbara J. Grosz, Douglas E. Appelt, Paul A. Martin, and Fernando C.N. Pereira. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. *Artificial Intelligence*, 32:173–243, 1987.

[7] Carole D. Hafner and Kurt S. Godden. *Design of Natural Language Interfaces: A Case Study.* Research Publication GMR-4567, General Motors Research Laboratories, Warren, MI, 1984.

[8] James L. Peterson. Computer Programs for Detecting and Correcting Spelling Errors. *Communications of the ACM*, 23(12):676–687, 1980.

[9] James L. Peterson. A Note on Undetected Typing Errors. *Communications of the ACM*, 29(7):633–637, 1986.