Techniques for Automatically Correcting Words in Text

KAREN KUKICH

Bellcore, 445 South Street, Morristown, NJ 07962-1910

Research aimed at correcting words in text has focused on three progressively more difficult problems: (1) nonword error detection; (2) isolated-word error correction; and (3) context-dependent word correction. In response to the first problem, efficient pattern-matching and *n*-gram analysis techniques have been developed for detecting strings that do not appear in a given word list. In response to the second problem, a variety of general and application-specific spelling correction techniques have been developed. Some of them were based on detailed studies of spelling error patterns. In response to the third problem, a few experiments using natural-language-processing tools or statistical-language models have been carried out. This article surveys documented findings on spelling error patterns, provides descriptions of various nonword detection and isolated-word error correction techniques, reviews the state of the art of context-dependent word correction techniques, and discusses research issues related to all three areas of automatic error correction in text.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning—*connectionism and neural nets*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*language models*; *language parsing and understanding*; *text analysis*; I.5.1 [**Pattern Recognition**]: Models—*neural nets*; *statistical*; I.5.4 [**Pattern Recognition**]: Applications—*text processing*; I.7.1 [**Text Processing**]: Text Editing—*spelling*

General Terms: Experimentation, Human Factors, Performance

Additional Key Words and Phrases: Context-dependent spelling correction, grammar checking, natural-language-processing models, neural net classifiers, *n*-gram analysis, Optical Character Recognition (OCR), spell checking, spelling error detection, spelling error patterns, statistical-language models, word recognition and correction

INTRODUCTION

The problem of devising algorithms and techniques for automatically correcting words in text has become a perennial research challenge. Work began as early as the 1960s on computer techniques for automatic spelling correction and automatic text recognition, and it has continued up to the present. There are good reasons for the continuing efforts in this area. Although some excellent academic and commercial spelling checkers have been around for some time, existing spelling correction techniques are limited in their scope and accuracy. As a consequence, many current computer applications are still vulnerable to costly text, code, and data entry mistakes. For example, the disruptive Bell Atlantic and Pacific Bell telephone network outages that occurred during the summer of 1991 were due in part to a typographical error in a software patch. Similarly, although some good commercial text recognition devices are available today, they perform optimally only under ideal conditions in which input consists of clean text set in a

CONTENTS

standard type font. Furthermore, even a character recognition accuracy rate as high as 99% yields only a 95% word recognition accuracy rate, because one error per 100 characters equates to roughly one error per 20 words, assuming five-character words. One study [Cushman 1990] found that in order for optically scanned documents to contain no more residual errors than typed documents, at least 98% character recognition accuracy coupled with computer-assisted proofreading would be required.

Evolving human-computer and computer-communications technologies have opened the door for a host of new applications that will require better word recognition and error correction capabilities. Pen-based interfaces will enable users to provide handwritten input to computers; text recognition devices will make scanning of printed material feasible under everyday conditions; voice synthesis (text-to-speech) devices will make textual material audible; and voice recognition (speech-to-text) technology will eventually allow voice input to computer systems. But none of these applications will become practical until significant improvements are made in the area of word recognition and correction. Some of the other applications that will benefit from such improvements include more sophisticated software tools for text and code editing, computer-aided authoring, machine translation, language learning, computer-aided tutoring, and database interaction, as well as various voice input and voice output business applications and aids for the disabled such as fax-to-voice devices and phonetic-transcription services.

Early on, researchers working within the paradigms of automatic spelling correction and automatic text recognition proceeded somewhat independently using different techniques. Over time the various techniques began to migrate between the fields so that today numerous hybrid approaches and many reasonably successful systems exist. But those who would consider spelling correction a solved problem may be missing some subtle and some not so subtle points about the nature of the problem and the scope of existing techniques.

A distinction must be made between the tasks of error **detection** and error **correction**. Efficient techniques have been devised for detecting strings that do not appear in a given word list, dictionary, or lexicon.[1] But correcting a misspelled string is a much harder problem. Not only is the task of locating and ranking candidate words a challenge, but as Bentley [1985] points out: given the morphological productivity of the English language (e.g., almost any noun can be verbified) and the rate at which words enter and leave the lexicon (e.g., *catwom-*

[1] The terms "word list," "dictionary," and "lexicon" are used interchangeably in the literature. We prefer the use of the term *lexicon* because its connotation of "a list of words relevant to a particular subject, field, or class" seems best suited to spelling correction applications, but we adopt the terms "dictionary" and "word list" to describe research in which other authors have used them exclusively.

anhood, *balkanization*), some even question the wisdom of attempts at automatic correction.

Many existing spelling correctors exploit task-specific constraints. For example, interactive command line spelling correctors exploit the small size of a command language lexicon to achieve quick response times. Alternatively, longer response times are tolerated for noninteractive-mode manuscript preparation applications. Both of the foregoing spelling correction applications tolerate lower first-guess accuracy by returning multiple guesses and allowing the user to make the final choice of intended word. In contrast, some future applications, such as text-to-speech synthesis, will require a system to perform fully automatic, real-time word recognition and error correction for vocabularies of many thousands of words and names. The contrast between the first two examples and this last one highlights the distinction between **interactive spelling checkers** and **automatic correction**. The latter task is much more demanding, and it is not clear how far existing spelling correction techniques can go toward fully automatic word correction.

Most existing spelling correction techniques focus on **isolated** words, without taking into account any information that might be gleaned from the linguistic or textual **context** in which the string appears. Such isolated-word correction techniques are unable to detect the significant portion of errors, including typographic, phonetic, cognitive, and grammatical errors, that result in other valid words. Some examples are the use of *form* where *from* was intended, the misuse of the words *there*, *their*, and *they're*, and the occurrence of the word *minuets* in the phrase *see you in five minuets*. It is clear that contextual information is necessary for the detection and correction of such errors. A context-based correction technique would not only address the problem of real-word errors, i.e., errors that result in another valid word, but it would also be helpful in correcting those nonword errors that have more than one potential correction. An example might be the string *ater*. Without context there is little reason, apart from a priori frequencies of words, to prefer *after*, *later*, *alter*, *water*, or *ate*, among others, as the intended correction. Developing context-based correction techniques has become the foremost challenge for automatic word recognition and error correction in text.

For descriptive purposes then, automatic word correction research may be viewed as focusing on three increasingly broader problems: (1) **nonword error detection**; (2) **isolated-word error correction**; and (3) **context-dependent word correction**. Work on the first problem spanned a period from the early 1970s into the early 1980s. During that time, effort was directed mainly toward exploring efficient pattern-matching and string comparison techniques for deciding whether an input string appears in a predefined word list or dictionary. Work on the second problem spanned a broader time frame, from as early as the 1960s into the present. During that time various general and special-purpose correction techniques were devised, some in conjunction with studies of spelling error patterns. Experimentation on the third problem began in the early 1980s with the development of automatic natural-language-processing models, and interest has recently been rekindled with the development of statistical language models.

The Unix™ *spell* program [McIlroy 1982] is an example of an effective and efficient program for spelling error detection that typifies work on the first problem. *Spell* takes a whole document as input, looks up each string in a 25,000-word dictionary tailored to include terms found mainly in the domain of technical writing, and returns a list of all strings that were not found in the dictionary. It does not make any attempt to correct the strings that it believes are misspelled. That task is left to the user, who might

™ *Unix* is a registered trademark of UNIX System Laboratories.

make use of an isolated-word spelling correction tool, such as *grope* [Taylor 1981]. *Grope*, another Unix tool, is one of many existing isolated-word spelling correctors that were developed to address the second problem. *Grope* acts as an interactive decision aid by accepting a string as input and providing the user with an ordered list of the most similar strings found in its dictionary as output, allowing the user to choose among them. The Unix *Writer's Workbench* package [Cherry 1983] represents one of the first efforts to address the third problem, that of detecting and correcting real-word errors in text. Its *style* and *diction* tools flag common grammatical and stylistic errors and suggest possible corrections. Although some commercial grammar-checking programs are available, no general-purpose context-dependent word correction tool yet exists.

This article presents a review of the techniques and issues related to automatic error correction in text. It is divided into three sections corresponding to the three problems identified above. Section 1 reviews nonword error detection work and includes a discussion of issues related to dictionary size and coverage and word boundary infractions. Section 2 reviews findings on spelling error patterns and provides descriptions of many of the techniques that have been used for isolated-word error correction. It notes the application domains for which the techniques were devised and highlights application-specific constraints that were exploited by some techniques. Section 3 reviews preliminary work in the area of context-dependent error correction in text. It includes a brief history of early work based on natural language processing as well as recent experiments based on statistical language modeling. Possible future directions for the field are hypothesized in an epilogue.

1. NONWORD ERROR DETECTION RESEARCH

The two main techniques that have been explored for **nonword error detection** are n-gram analysis and dictionary lookup. N-grams are n-letter subsequences of words or strings, where n is usually one, two, or three. One-letter n-grams are referred to as unigrams or monograms; two-letter n-grams are referred to as digrams or bigrams; and three-letter n-grams as trigrams. In general, n-gram error detection techniques work by examining each n-gram in an input string and looking it up in a precompiled table of n-gram statistics to ascertain either its existence or its frequency. Strings that are found to contain nonexistent or highly infrequent n-grams (such as the trigram *shj* or *IQN*) are identified as probable misspellings. N-gram techniques usually require either a dictionary or a large corpus of text in order to precompile an n-gram table. Dictionary lookup techniques work by simply checking to see if an input string appears in a dictionary, i.e., a list of acceptable words. If not, the string is flagged as a misspelled word. There are subtle problems involved in the compilation of a useful dictionary for a spelling correction application.

Historically, text recognition systems have tended to rely on n-gram techniques for error detection while spelling checkers have tended to rely on dictionary lookup techniques. In both cases, problems arise when errors cross word boundaries resulting in run-on or split words. Issues related to each of these techniques and their accompanying problems are discussed next.

1.1 *N*-gram Analysis Techniques

Text recognition systems usually focus on one of three modes of text: hand-printed text, handwritten text (sometimes referred to as cursive script), or machine-printed text. All three modes may be processed by optical character recognition (OCR) devices. OCR devices typically rely on feature analysis to recognize individual characters within words. Examples of features might include counts of the number of vertical, horizontal, curved, and crossing lines in

a character. Errors made by OCR devices tend to be those that confuse characters with similar features, such as *O* and *D*, *S* and 5, or *t* and *f*. *N*-gram analysis has proven useful for detecting such errors because they tend to result in improbable *n*-grams. Following up on early work by Sitar [1961], Harmon [1972] reported that "1) in large samples of common English publication text, 42% of all possible digram combinations never occur and 2) random substitution of one letter in a word by another letter which is selected with equal likelihood from the other 25 of the alphabet will produce at least one new digram which has zero probability 70% of the time" (p. 1172). These facts have been exploited for detecting errors in the output of optical-character recognizers.[2]

N-gram tables can take on a variety of forms. The simplest is a *binary bigram* array, which is a two-dimensional array of size 26 × 26 whose elements represent all possible two-letter combinations of the alphabet. The value of each element in the array is set to either 0 or 1 depending on whether that bigram occurs in at least one word in a predefined lexicon or dictionary. A *binary trigram* array would have three dimensions. Both of the above arrays are referred to as **nonpositional** *binary n-gram* arrays because they do not indicate the position of the *n*-gram within a word.

More of the structure of the lexicon can be captured by a set of **positional** *binary n-gram* arrays. For example, in a positional binary trigram array, the i, j, kth element would have the value 1 if and only if there exists at least one word in the lexicon with the letters l, m, and n in positions i, j, and k. The tradeoff for representing more of the structure of the lexicon is the increase in storage space required for the complete set of positional arrays. Any word can be checked for errors by simply looking up its corresponding entries in binary *n*-gram arrays to make sure they are all 1s.

Two studies give some indication of how effective nonpositional and positional binary *n*-gram arrays are at detecting errors in OCR output. Hanson et al. [1976] studied the output of an optical scanner that had been fed a corpus of hand-printed six-letter words. The output contained 7,662 words each having a single substitution error. The best detection performance was achieved using positional binary trigram arrays. These detected 7,561, or 98%, of the errors. Nonpositional *n*-gram arrays scored substantially lower. Hull and Srihari [1982] found that positional binary trigrams representing a subdictionary of 5,000 seven-letter words detected over 98% of substitution errors. It is not clear how well these results generalize to shorter-length words and other types of errors such as insertions, deletions, and framing errors that alter the lengths of words. An example of a framing error is the substitution of the single letter *m* for the two-letter sequence *ni* or vice versa.

Another study was designed to evaluate the effectiveness of trigram frequency statistics for spell-checking applications. Zamora et al. [1981] compiled a trigram table containing frequency counts or probabilities instead of binary values. Such statistics must be compiled from a sufficiently large corpus of text (e.g., at least a million words) covering the domain of discourse. Then they analyzed a collection of 50,000 word/misspelling pairs from seven machine-readable databases to determine "whether there is sufficient difference between the trigram compositions of correct and misspelled words for the latter to be reliably detected" (p. 306). They found that although trigram analysis was able to determine the error site within a misspelled word accurately, it did not distinguish effectively between valid words and misspellings.

To address this problem, Morris and

[2] Although text recognition researchers often refer to techniques that use *n*-gram statistics as context-dependent techniques, this use of the term "context" refers only to within-word context. In contrast, the techniques that are discussed in Section 3 of this article exploit extraword context for correcting real-word errors.

Cherry [1975] devised an alternative technique for using trigram frequency statistics to detect errors. Rather than generating a trigram frequency table from a general corpus of text, they generated the table directly from the document to be checked. Their rationale for this approach was based on their finding that "for a given author, the number of distinct words [used in a document] increases approximately as the square root of the total number of words" (p. 54). Thus, misspellings might show up as words containing trigrams that were peculiar to the document itself.

To check a document for spelling errors, Morris and Cherry [1975] generate a trigram frequency table based on the document itself. Then, for each unique word in the document they compute an *index of peculiarity* as a function of the trigram frequencies of the word. Finally, they rank the words in decreasing order of peculiarity. They hypothesize that misspelled words will tend to appear near the top of the list. As an example of the technique's success, they point out (1) that it took only ten minutes for an author of a 108-page document to scan the output list and identify misspelled words and (2) that 23 of the 30 misspelled words in the document occurred in the top 100 words of the list.

1.2 Dictionary Lookup Techniques

Dictionary lookup is a straightforward task. However, response time becomes a problem when dictionary size exceeds a few hundred words. In document processing and information retrieval, the number of dictionary entries can range from 25,000 to more than 250,000 words. This problem has been addressed in three ways, via efficient dictionary lookup and/or pattern-matching algorithms, via dictionary-partitioning schemes, and via morphological-processing techniques.

The most common technique for gaining fast access to a dictionary is the use of a hash table [Knuth 1973]. To look up an input string, one simply computes its hash address and retrieves the word stored at that address in the preconstructed hash table. Sometimes a link or two must be traversed if collisions occurred during construction of the hash table. If the word stored at the hash address is different from the input string or is null, a misspelling is indicated.

Turba [1981] provides a quick review of the pros and cons of using hash tables for dictionary lookup. The main advantage is that the random-access nature of a hash code eliminates the large number of comparisons needed for sequential or even tree-based searches of the dictionary. The main disadvantage is the need to devise a clever hash function that avoids collisions without requiring a huge hash table. Fox et al. [1992] recently described a technique for devising "minimal perfect hash functions whose specification space is very close to the theoretical lower bound" (p. 266). Some earlier systems circumvented the storage problem by not storing the character representation of the word itself but instead using a single bit to indicate that a given hash code maps into a valid word. This trick explains the occasional odd behavior of some spelling checkers in which an invalid string goes undetected because it happened to map into a hash address for a valid word. The Unix *spell* program is one example of a program that employs a hash table for fast dictionary lookup.

Other standard search techniques, such as tries [Knuth 1973], frequency-ordered binary search trees [Knuth 1973], and finite-state automata [Aho and Corasick 1975], have been used to reduce dictionary search time. Knuth [1973] defines a trie as "an M-ary tree, whose nodes are M-place vectors with components corresponding to digits or characters. Each node on level l represents the set of all keys that begin with a certain sequence of l characters; the node specifies an M-way branch, depending on the $(l + 1)$st character" (p. 481). Knuth notes that the name *trie* was coined by Fredkin [1960] because of its role in information re*trie*val applications. Sheil [1978] introduced a fast lookup technique called median split trees for searching lexicons with highly skewed distributions such as English text. A technique for fast regu-

lar-expression pattern matching invented by Aho and Corasick [1975] prescribes a method for creating a finite-state automaton to represent a small set of terms (keywords) that must be matched against a corpus of running text. Other algorithms for fast pattern matching, some of which have been adopted for spelling correction applications, are described in a review article by Aho [1990].

Peterson [1980] suggests partitioning a dictionary into three levels for spelling error detection in text-processing applications. The first level, to be stored in cache memory, would consist of the few hundred most frequently used words that account for 50% of dictionary accesses; the second level, to be stored in regular memory, would consist of a few thousand domain-specific words which account for another 45% of dictionary accesses; a third level, to be stored in secondary memory, would consist of tens of thousands of less frequently used words which account for the remaining 5% of dictionary accesses.

Memory constraints also motivated many early spelling checkers to avoid storing all possible morphological variants of individual words (e.g., plurals, past tenses, adverbials, nominals, etc.) as separate dictionary entries. Instead, only root forms of words were stored. If a lookup procedure failed to find an input string in the dictionary, it would issue a call to a morphological-processing routine which would iteratively check for known suffixes and prefixes to strip (and possibly replace with alternative strings, as in *dictionaries* → *dictionary*) before re-searching the dictionary. However, as Elliott [1988] describes in a memo on an improved morphological-processing component for the Unix *spell* program, simple affix-stripping methods often lead to false acceptances, such as *adviseed* or *disclam*, when affixes are stripped with no regard to grammar. Elliott provides an efficient solution based on creating affix equivalence classes that are motivated by spelling similarities.

Increases in computational speed and memory have made dictionary storage and processing-time constraints less critical, so the current trend is toward storing all inflectional variants of words as separate dictionary entries. However, the need for morphological processing will never be completely obviated because it is virtually impossible to anticipate all of the morphologically creative terms that people tend to generate in both text and conversation (e.g., "large software projects must avoid creeping *featurism*," "these actors are highly *watchable*").

1.3 Dictionary Construction Issues

A lexicon for a spelling correction or text recognition application must be carefully tuned to its intended domain of discourse. Too small a lexicon can burden the user with too many false rejections of valid terms; too large a lexicon can result in an unacceptably high number of false acceptances, i.e., genuine mistakes that went undetected because they happened to form valid low-frequency or extra-domain words (e.g., *lave*, *fen*, *veery*, etc). But the relationship between misspellings and word frequencies is not straightforward.

Peterson [1986] calculated that approximately half a percent of all single-error transformations of each of the words on a 350,000-item word list result in other valid words on the list. He noted that the number of actual undetected errors may be much higher in practice due to the fact that shorter words, which tend to occur more frequently, also have a higher tendency to result in other valid words when misspelled. So he went on to compute frequency-weighted estimates of the percentage of errors that would go undetected as a function of dictionary size. His estimates ranged from 2% for a small dictionary to 10% for a 50,000-word dictionary to almost 16% for a 350,000-word dictionary. This led Peterson to recommend that word lists for spelling correction be kept relatively small.

However, Damerau and Mays [1989] challenge this recommendation. Using a corpus of over 22 million words of text from various genres, they found that by increasing the size of their frequency rank-ordered word list from 50,000 to

60,000 words, they were able to eliminate 1,348 false rejections while incurring only 23 additional false acceptances. Since this 50-to-1 differential error rate represents a significant improvement in correction accuracy, they recommend the use of larger lexicons.

Dictionaries themselves are often insufficient sources for lexicon construction. Walker and Amsler [1986] observed that nearly two-thirds (61%) of the words in the *Merriam-Webster Seventh Collegiate Dictionary* did not appear in an eight million word corpus of *New York Times* news wire text, and, conversely, almost two-thirds (64%) of the words in the text were not in the dictionary. In analyzing a sample of the novel terms in the *New York Times* text, they found that "one fourth were inflected forms; one fourth were proper nouns; one sixth were hyphenated forms; one-twelfth were misspellings; and one fourth were not yet resolved, but some are likely to be new words occurring since the dictionary was published" (p. 79). Many such novel terms could be desirable, frequently used terms in a given application. Some applications require lexicons specialized for command and file names or specific database entries. On the topic of construction of the spelling list for the *spell* program, McIlroy [1982] provides some helpful insights into appropriate sources that may be drawn upon for general word-list construction. A more recent article by Damerau [1990] provides some insights into and guidelines for the automatic construction and customization of domain-oriented vocabularies for specialized natural-language-processing applications.

Mitton [1986] has made available a computer-usable dictionary derived from the *Oxford Advanced Learner's Dictionary of Current English*. Its nearly 38,000 main entries yield over 68,000 inflected forms. Sampson [1989] has evaluated the coverage of this dictionary against a 50,000-word cross section of a corpus of written English. Both the dictionary and the corpus "are British-based, though both include some material pertaining to other varieties of English" (p. 29). Sampson reports that he found the dictionary's coverage of the corpus to be surprisingly good—only 1,477, or 3.24%, of the word tokens in the corpus were not in the dictionary.[3] In analyzing the 1,176 distinct word types represented by the 1,477 word tokens, Sampson found that over half, about 600, were proper names or adjectives derived from proper names (e.g., *Carolean*). Of the remainder, he felt that some, including abbreviations and numbers written digitally with alphabetic affixes, could be added with relative ease. But he found foreign words, hyphenated variants, and derived morphological variants (e.g., *inevitably*, *glassily*, *fairness*), especially missing negative forms (e.g., *uncongenial*, *irreversible*), to be problems for spelling dictionary construction.

1.4 The Word Boundary Problem

For virtually all spelling error detection and correction techniques, word boundaries are defined by white space characters (e.g., blanks, tabs, carriage returns, etc.). This assumption turns out to be problematic since a significant portion of text errors involve running together two or more words, sometimes with intrinsic errors (e.g., *ofthe*, *understandhme*), or splitting a single word (e.g., *sp ent*, *th ebook*). Kukich [1992] found that a full 15% of all nonword spelling errors in a 40,000-word corpus of typed textual conversations involved this type of error (i.e., 13% were run-on words, and 2% were split words). Mitton [1987] found that run-ons and splits frequently result in at least one valid word (e.g., *forgot* → *for got*, *in form* → *inform*), so one or both such errors may go undetected. In contrast to human-generated errors, Jones et al. [1991] found that OCR devices are more likely to split words than to join

[3] A *type vs token* convention is used to distinguish actual occurrences of an individual word, i.e., *tokens*, from the name of the word itself, i.e., its *type*. For example, a document may contain many *tokens* of the word **the**, but the word **the** itself represents only one *type*

them. The difficulty in dealing with run-ons and splits lies in the fact that weakening the word boundary constraint results in a combinatorial explosion of the number of possible word combinations or subdivisions that must be considered.

No spelling error detection systems treat word boundary infractions any differently than other types of errors, but a few spelling error correction systems attempt to handle some types of run-on and split-word errors explicitly. One such corrector was built by Lee et al. [1990] for a natural language interface to an intelligent tutoring system in the domain of cardiovascular physiology. This corrector exploits the limited vocabulary of its domain of discourse to check for run-ons or splits when no candidate corrections exceed a minimum threshold of similarity. Another corrector implemented by Means [1988] for a limited-domain natural language interface handles split-word errors in the same way. A corrector designed by Kernighan [1991] for a text-to-speech application handles run-on words by checking for space bar deletions at the same time it checks for other letter deletions. It sets a "likelihood score for a two-word correction equal to a constant (determined by trial and error) times the frequencies of the two words" (p. 4). CLARE, a front end to a natural language parsing and generation system implemented by Carter [1992], was explicitly designed to handle word boundary infractions. It does so by maintaining "a lattice of overlapping word hypotheses from which one or more complete paths are subsequently selected" (p. 159). This system was able to find a single, accurate correction for 59 of 108 errors in a test set of 102 sentences. Twenty-four of the 59 corrections involved word boundary infractions. An error-correcting postprocessor for OCR devices designed by Jones et al. [1991] includes a processing phase that explicitly checks for split-word errors.

There is some indication that a large portion of run-on and split-word errors involves a relatively small set of high-frequency function words (i.e., prepositions, articles, quantifiers, pronouns, etc.), thus giving rise to the possibility of limiting the search time required to check for this type of error. The SPEEDCOP corrector, implemented by Pollock and Zamora [1984] for a general text-editing application, includes a final subroutine that checks for run-on errors involving function words after other routines have failed. However, the general problem of handling errors due to word boundary infractions remains one of the significant unsolved problems in spelling correction research.

1.5 Summary of Nonword Error Detection Work

In general, although n-gram analysis may be useful for detecting machine-generated errors such as those produced by optical-character recognizers, it has proven to be less accurate for detecting human-generated errors. Hence, most current spelling correction techniques rely on dictionary lookup for error detection. Since access speed may be a factor when dictionary size is moderate to large (e.g., > 20,000 entries), efficient algorithms for exact pattern matching that exploit hash tables, tries, and other techniques have been used. Dictionaries must be carefully tailored to the domain of discourse of the application in order to avoid frustrating the user with too many false acceptances and rejections. More research in the area of generative morphology is warranted to address the problem of creative morphology if not dictionary size reduction. Finally, errors due to word boundary infractions remain a significant unsolved problem for spelling error detection and correction.

2. ISOLATED-WORD ERROR CORRECTION RESEARCH

For some applications, simply detecting errors in text may be sufficient, but for most applications detection alone is not enough. For example, since the goal of text recognition devices is to accurately reproduce input text, output errors must be both detected and corrected. Simi-

larly, users have come to expect spelling checkers to suggest corrections for the nonwords they detect. Indeed, some spelling correction applications, such as text-to-speech synthesis, require that errors be both detected and corrected without user intervention. To address the problem of correcting words in text, a variety of **isolated-word error correction** techniques have been developed.

The characteristics of different applications impose different constraints on the design of isolated-word error correctors, and many successful correction techniques have been devised by exploiting application-specific characteristics and constraints. It is worth reviewing some isolated-word error correction applications and their characteristics before delving into the details of individual techniques.

Text recognition and **text editing** are the most studied applications. A number of papers covering the first two decades of text recognition research is available in an anthology [Srihari 1984]. Some more recent work in this area includes the following papers: [Burr 1987; Goshtasby and Ehrich 1988; Ho et al. 1991; Jones et al. 1991]. A variety of spelling correction techniques used in text editing is represented by the following papers: [Damerau 1964; Alberga 1967; Gorin 1971; Wagner 1974; Hall 1980; Peterson 1980; Yannakoudakis and Fawthrop 1983a; Pollock and Zamora 1984; Kernighan et al. 1990].

Other applications for which spelling correction techniques have been devised include **systems programming applications** [Morgan 1970; Aho and Peterson 1972; Sidorov 1979; Spenke et al. 1984], **command language interfaces** [Hawley 1982; Durham et al. 1983], **database retrieval** and **information retrieval interfaces** [Blair 1960; Davidson 1962; Boivie 1981; Mor and Fraenkel 1982a; Bickel 1987; Kukich 1988a; Salton 1989; Gersho and Reiter 1990; Cherkassky et al. 1990; Parsaye et al. 1990], **natural language interfaces** [Veronis 1988a; Means 1988; Berkel 1988; Lee et al. 1990; Deffner et al. 1990a], **computer-aided tutoring** [Tenczar and Golden 1972], **computer-aided language learning** [Contant and Brunelle 1992], **text-to-speech applications** [Kukich 1990; Tsao 1990; Kernighan 1991], **augmentative communication systems for the disabled** [Alm et al. 1992; Wright and Newell 1991; Demasco and McCoy 1992], **pen-based interfaces** [Rhyne and Wolf 1991], and even searching for historical word forms in databases of 17th century English [Robertson and Willet 1992].

Most application-specific design considerations are related to three main issues: (1) **lexicon** issues, (2) **computer-human interface** issues, and (3) **spelling error pattern** issues. **Lexicon** issues include such things as lexicon size and coverage, rates of entry of new terms into the lexicon, and whether morphological processing, such as affix handling, is required. These were discussed in the preceding section on dictionary construction issues.

Computer-human interface issues include such considerations as whether a real-time response is needed, whether the computer can solicit feedback from the user, how much accuracy is required on the first guess, etc. These issues are especially important in command language interfaces where trade-offs must be made to balance the need for accuracy against the need for quick response time and where care must be taken to provide the user with helpful corrections without badgering her with unwanted suggestions. An empirical study by Durham et al. [1983] demonstrated, among other things, that considerable benefits accrued from a simple, speedy algorithm for correcting single-error typos that only corrected about one quarter of the spelling mistakes made by users but did so in an unobtrusive manner. Other work, by Hawley [1982], demonstrated the ease with which spelling correction could be incorporated into the Unix command language interface. In studying alternative methods for correcting recognition errors in pen-based interfaces, Rhyne and Wolf [1993] found that the cost to repair er-

rors is a critical factor in the design of these interfaces and in their ultimate acceptance by users. Some of the correction methods they are studying include allowing the user to write over an error, allowing for keyboard correction, and providing n best matches for the user to select from.

Issues related to **spelling error patterns** include such things as what the most common errors are, how many errors tend to occur within a word, whether errors tend to change word length, whether misspellings are typographically, cognitively, or phonetically based, and, in general, whether errors can be characterized by rules or by probabilistic tendencies. Spelling error pattern issues have had perhaps the greatest impact design of correction techniques.

2.1 Spelling Error Patterns

Spelling error patterns vary greatly depending on the application task. For example, transcription typing errors, which are for the most part due to motor coordination slips, tend to reflect typewriter keyboard adjacencies, e.g., the substitution of *b* for *n*. In contrast, errors introduced by optical-character recognizers are more likely to be based on confusions due to featural similarities between letters, e.g., the substitution of *D* for *O*. More subtly, even two similar text entry modes, such as transcription typing and conversational text typing, e.g., electronic mail, may exhibit significantly different error frequency and distribution statistics due to the greater cognitive overhead involved in the latter task. So care must be taken not to overgeneralize findings when discussing spelling error patterns.

Distinctions are sometimes made among three different types of nonword misspellings: (1) typographic errors, (2) cognitive errors, and (3) phonetic errors. In the case of typographic errors (e.g., *the* → *teh*, *spell* → *speel*), it is assumed that the writer or typist knows the correct spelling but simply makes a motor coordination slip. The source of cognitive errors (e.g., *receive* → *recieve*, *conspiracy* → *conspiricy*) is presumed to be a misconception or a lack of knowledge on the part of the writer or typist. Phonetic errors (e.g., *abyss* → *abiss*, *naturally* → *nacherly*) are a special class of cognitive errors in which the writer substitutes a phonetically correct but orthographically incorrect sequence of letters for the intended word. It is frequently impossible to ascribe a single category to a given error. Is *recieve* necessarily a cognitive error, for example, or might it simply be a typographic transposition error? Similarly, is *abiss* a phonetic or typographic error? Fortunately, it is often unnecessary to categorize errors in order to develop a useful spelling correction technique because many correction techniques handle typographic and cognitive misspellings equally well. In fact, only a few researchers bother to distinguish among categories of misspellings. Phonetic errors, however, tend to distort spellings more than typographic errors and other cognitive errors, so the limited findings related to these errors are reported below.

Most studies of spelling error patterns were done for the purpose of designing correction techniques for nonword errors (i.e., those for which there is no exact match in the dictionary), so most findings relate only to nonword as opposed to real-word errors (i.e., those that result in another valid word). However, at least two studies that did address real-world errors are included in this discussion. It should be noted that the increasing use of automatic spelling checkers has probably reduced the number of nonword errors found in some genres of text. Consequently, the relative ratio of real-word errors to nonword errors is probably higher today than it was in early studies, at least for some applications.

2.1.1 Basic Error Types

One of the first general findings on human-generated spelling errors was observed by Damerau in 1964 and has been substantiated for many applications since

then. Damerau [1964] found that approximately 80% of all misspelled words contained a single instance of one of the following four error types: **insertion, deletion, substitution,** and **transposition**. Misspellings that fall into this large class are often referred to as **single-error misspellings**; misspellings that contain more than one such error have been dubbed **multi-error misspellings**.

The 80% single-error rule cannot be taken for granted for all applications, however. Pollock and Zamora [1984] found that only 6% of 50,000 nonword spelling errors in the machine-readable databases they studied were multi-error misspellings. Conversely, Mitton [1987] found that 31% of the misspellings in his 170,016-word corpus of handwritten essays contained multiple errors.

OCR-generated misspellings do not follow the pattern of human-generated misspellings. Most error correction techniques for OCR output assume that the bulk of the errors will be substitution errors. However, Jones et al. [1991] report that "the types of [OCR] errors that occur vary widely, not only from one recognizer to another but also based on font, input quality and other factors" (p. 929). They also report that "a significant fraction of OCR errors are not one-to-one errors (e.g., $ri \rightarrow n$ or $m \rightarrow iii$)" (p. 927). Rhyne and Wolf [1993] classify recognition errors into four categories: (1) substitutions; (2) failures (no character exceeds a minimal recognition threshold); (3) insertions and deletions; and (4) framing errors (one-to-one mapping failures).

2.1.2 Word Length Effects

Another general finding, a corollary to the first, is the observation that most misspellings tend to be within two characters in length of the correct spelling. This has led many researchers, especially within the OCR paradigm, to partition their dictionaries into subdictionaries by word length in order to reduce search time.

Unfortunately, little concrete data exists on the frequency of occurrence of errors by word length. This characteristic will clearly affect the practical performance of a correction technique since errors in short words are harder to correct, in part because less intraword contextual information is available to the corrector. Furthermore, according to Zipf's law [Zipf 1935], short words occur more frequently than long words, and, according to an empirical study by Landauer and Streeter [1973], high-frequency (i.e., short) words tend to have more single-error neighbors than low-frequency words, thus making it difficult to select the intended correction from its set of neighbors. On the other hand, if spelling errors occur less frequently in short words, then the problem may be less pressing.

Pollock and Zamora's study [Pollock and Zamora 1983] of 50,000 nonword errors indicated that errors in short words are indeed problematic for spelling correction even though their frequency of occurrence may be low. They state: "although 3–4 character misspellings constitute only 9.2% of total misspellings, they generate 42% of the miscorrections" (p. 367). There appears to be wide variance in the proportion of errors that occur in short words across different applications. For example, Yannakoudakis and Fawthrop [1983b] found an even lower frequency of occurrence of errors in short words, about 1.5%, in their analysis of 1,377 errors found in the literature. In contrast, Kukich [1990] analyzed over 2,000 error types in a corpus of TDD conversations and found that over 63% of the errors occurred in words of length 2, 3, and 4 characters. These differences emphasize the need for determining the actual characteristics of the spelling errors of the intended application before designing or implementing a correction system.

2.1.3 First-Position Errors

It is generally believed that few errors tend to occur in the first letter of a word. Only a few studies actually document first-position error statistics. Pollock and Zamora [1983] found that 3.3% of their 50,000 misspellings involved first letters, and Yannakoudakis and Fawthrop [1983b] observed a first-position error

rate of 1.4% in 568 typing errors. Mitton [1987] found that 7% of all the misspellings he studied involved first-position errors. In contrast, Kukich [1992] observed a 15% first-position error rate in a 40,000-word corpus of typed textual conversations.

Discounting first-position errors allows a lexicon to be partitioned into 26 subsets, each containing all words beginning with a single letter, thus greatly reducing search time. Many spelling correction techniques have exploited this characteristic. However, a caveat is in order any time dictionary-partitioning schemes are used: a trade-off must be made between quicker response time and reduced accuracy due to the possibility of missing the correction entirely because it is not in the partition searched.

2.1.4 Keyboard Effects

Some extensive studies of typing behavior were carried out by the LNR Typing Research Group [Gentner et al. 1983]. Rather than developing a spelling correction technique, their goal was to develop a working computer simulation model of typing. As part of this work, Grudin [1983] performed an extensive analysis of the typing errors made by six expert typists and eight novice typists while transcribing magazine articles totaling about 60,000 characters of text. He found large individual differences in both typing speed and types of errors made. For example, error rates ranged from 0.4% to 1.9% for experts and averaged 3.2% for novices; the majority of expert errors were insertions that resulted from hitting two adjacent keys simultaneously while the majority of novice errors were substitutions. But Grudin did uncover some general patterns. After compiling a confusion matrix[4] from the 3,800 substitution errors in his corpus, he observed that 58% of all substitution errors involved adjacent typewriter keys. He also found strong letter frequency effects in the adjacent substitution errors, i.e., even after normalizing for frequency in language, a typist is more likely to substitute a higher-frequency letter for a lower-frequency neighbor than vice-versa. Grudin went on to develop a computer model of typing capable of generating the same sorts of errors that human typists make [Grudin 1981]. Until recently, few spelling correction techniques have attempted to directly exploit the probabilistic tendencies that arise from keyboard adjacencies and letter frequencies. Those that have are discussed in Section 2.2.5.

2.1.5 Error Rates

Data concerning the frequency of occurrence of spelling errors is not abundant. The few data points that do exist must be qualified by (1) the size of the corpus from which they were drawn; (2) the text entry mode of the corpus, e.g., handwriting, transcription typing, conversational typing, edited machine-readable text, etc.; and (3) the date of the study, since newer studies for some genres, such as edited machine-readable text, probably reflect lower error rates due to the availability of automatic spelling checkers. Furthermore, whether both nonword and real-word errors were included in the study should also be noted.

Grudin's transcription-typing study [Grudin 1983] found average error rates of 1% for expert typists and 3.2% for novices. His corpus consisted of approximately 60,000 characters of typed text. He did not distinguish between errors that resulted in nonwords and errors that resulted in real words.

Two studies of errors in typed textual conversations by deaf TDD (Telecommunications Device for the Deaf) users by Kukich [1992] and Tsao [1990] found nonword spelling error rates of 6% and 5% respectively. The corpus studied by Kukich contained 40,000 words, or, more accurately, strings, and that studied by Tsao contained 130,000 strings.

Pollock and Zamora [1984] examined a

[4] A confusion matrix is a square matrix whose rows and columns are labeled with the characters of the keyboard and whose ijth element contains the frequency count of the number of times letter i was mistakenly keyed when letter j was intended in a given corpus.

set of machine-readable databases containing 25 million words of text. They found over 50,000 nonword errors, for a spelling error rate of 0.2%. An even lower error rate holds for a corpus of AP (Associated Press) news wire text studied by Church and Gale [1991]. For the purposes of their study, a typo was defined as any lower-case word rejected by the Unix *spell* program. By this measure, they found that "the AP generates about 1 million words and 500 typos per week" (p. 93), which equates to an error rate of 0.05%.

Spelling error rates of 1.5% and 2.5% for handwritten text have been reported by Wing and Baddeley [1980] and Mitton [1987] respectively. Wing and Baddeley's corpus consisted of 40 essays written by Cambridge college applicants totaling over 80,000 words, of which 1,185 were in error; Mitton's corpus contained 925 high school student essays totaling 170,016 words, of which 4,128 were in error. Both of these studies included both nonword and real-word errors. A quick scan of Wing and Baddeley's data suggest that at least 30% were real-word errors. Mitton found that fully 40% of the misspellings in his corpus were real-word errors. These involved mainly substitution of a wrong function word (e.g., *he* for *her*), substitution of a wrong inflected form (e.g., *arrive* for *arrived*), and incorrect division of a word (e.g., *my self* for *myself*). None of these types of errors are detectable by isolated-word error correctors. As noted in Section 1.3, Peterson's [1986] study of undetected spelling errors indicates that single-error typos alone are likely to result in valid words at least 16% of the time. These findings suggest a genuine need for context-dependent spelling correction methods, at least for some applications. This topic will be taken up in Section 3.

While many manufacturers of OCR equipment advertise character recognition error rates as low as 1–2%, or equivalent word error rates of 5–10%, recent research [Santos et al. 1992; Jones et al. 1991] indicates that these devices tend to exhibit marginal performance levels, e.g., word error rates in the range of 7–16%, in actual field applications (as opposed to laboratory testing under ideal conditions). Nevertheless, one recent study [Nielsen et al. 1992] found that despite word error rates of 8.8% for one pen-based system, information retrieval rates were nearly as good for pen-based-entered documents as they were for the same error-free text. The authors caution readers that this result holds only for documents whose average length was 185 words, and that "very short notes would be difficult to find without the use of further attributes such as time or context of writing" (p. 7).

2.1.6 Phonetic Errors

Examples of applications in which phonetic errors abound include (1) the automatic yellow-pages-like information retrieval service available in the French Minitel system [Veronis 1998b], (2) a directory assistance program for looking up names in a corporate directory [Boivie 1981], a database interface for locating individuals by surname in large credit, insurance, motor vehicle bureau, and law enforcement databases [Oshika et al. 1988], and (3) a natural language interface to an electronic encyclopedia [Berkel and DeSmedt 1988]. The predominant phonetic nature of errors in these applications is intuitively explained by the fact that people resort to phonetic spellings for unfamiliar names and words.

Two data points on the frequency of occurrence of phonetic errors have been documented. One is provided by Van Berkel and DeSmedt [1988]. They had 10 Dutch subjects transcribe a tape recording of 123 Dutch surnames randomly chosen from a telephone directory. They found that 38% of the spellings generated by the subjects were incorrect despite being phonetically plausible. The other data point comes from Mitton's study [Mitton 1987]. He found that 44% of all the spelling errors in his corpus of 925 student essays involved homophones.

2.1.7 Heuristic Rules and Probabilistic Tendencies

Three comprehensive studies of spelling error patterns have been carried out with the goal of devising spelling correction techniques. One study, by Yannakoudakis and Fawthrop [1983b], aimed at discovering specific rules that spelling errors tend to follow, with the intent of designing a rule-based spelling correction algorithm. Another study, by Pollock and Zamora [1983], aimed at discovering probabilistic tendencies, such as which letters and which position within a word are most frequently involved in errors, with the intent of devising a similarity key-based technique. A third study, by Kernighan et al. [1990] aimed at compiling error probability tables for each of the four classes of errors, with the intent of exploiting those probabilities directly. There is obviously some overlap among goals of these three studies. Nevertheless, the information they derived was used to devise and implement three very different spelling correction techniques, each of which is described in the next section. This section contains a brief review of their findings.

Yannakoudakis and Fawthrop [1983a, 1983b] sought a general characterization of misspelling behavior. They compiled a database of 1,377 spelling errors culled from a variety of sources, including mistakes found in the Brown corpus [Kucera and Francis 1967] as well as those found in a 60,000-word corpus of text typed by three University of Bradford adults who "believed themselves to be very bad spellers." They found that a large portion of the errors could be accounted for by a set of 17 heuristic rules, 12 of which related to the misuse of consonants or vowels in graphemes,[5] and five of which related to sequence production. For example, heuristics related to consonants included (1) the letter *h* is frequently omitted in words containing the graphemes *ch*, *gh*, *ph*, and *rh*, as in the misspellings *agast* and *tecniques*; and (2) doubling and singling of consonants which frequently occur doubled is a common error. Heuristics related to sequence production included: (3) the most frequent length of a misspelling is one letter short of the correct spelling; (4) typing errors are caused by hitting an adjacent key on the keyboard or by hitting two keys together; (5) short misspellings do not contain more than one error, and so on.

The application motivating Pollock and Zamora's study was that of editing machine-readable text for bibliographic information retrieval services. They extracted over 50,000 errors from 25 million words of scientific and scholarly text taken from seven Chemical Abstracts Service databases. Among other things, they found that (1) 0.2% of the words in their corpus contained spelling errors; (2) 94% of the spelling errors were single errors; (3) 34% of all errors were omissions; (4) 23% of all errors occurred in the third position of a word; and (5) except for a few frequently misspelled words (e.g., *the* → *teh*), most misspellings are rarely repeated.

Kernighan et al. [1990] scanned 44 million words of AP news wire text for spelling errors. Using *spell* to locate nonword strings in the text and a candidate generation technique to locate every possible dictionary entry formed by a single insertion, deletion, substitution, or transposition in the nonword strings, they found over 25,000 misspellings for which there existed exactly one possible correct spelling. (Some nonword strings had no possible corrections because they contained more than one error; others had more than one possible correction, e.g., *acress* → *actress*, *acres*, *access*.) Their list of 25,000 misspelling/correct-spelling pairs was used to compile error frequency tables (a.k.a., confusion matrices) for each of the four classes of spelling errors: insertions, deletions, substitutions, and transpositions. For example, they determined that *a* was incorrectly

[5] A *grapheme* is a letter sequence corresponding to a phoneme. For example, *that* includes the graphemes *th*, *a*, and *t*.

substituted for *e* 238 times; *s* was incorrectly inserted after *e* 436 times; *t* was incorrectly deleted after *i* 231 times; and *it* was incorrectly typed for *ti* 48 times. These frequencies were used to estimate the probability of occurrence of each potential error.

2.1.8 Common Misspellings

Still other sources of data on the nature of spelling errors are published lists of common misspellings and their corrections. *Webster's New World Misspeller's Dictionary* [Webster 1983] is one such list. Its foreword identifies 12 classes of common errors, such as uncertainty about whether a consonant is doubled, errors resulting from mispronunciations, errors resulting from homonyms, etc., some of which overlap with the Yannakoudakis and Fawthrop [1983b] findings. Very few academic spelling correctors report the use of lists of common misspellings. One exception is the technique devised by Pollock and Zamora which did incorporate a routine for checking a short list of frequently misspelled words.

2.1.9 Summary of Spelling Error Pattern Findings

In summary, the frequency of occurrence of nonword spelling errors varies greatly depending on application, data entry mode, and date of study. Estimates range from as low as 0.05% in edited news wire text to as high as 38% in phonetically oriented database retrieval applications. Practical OCR word error rates currently range between 7% and 16%. OCR error patterns tend to vary for different OCR devices and type fonts.

Some general findings on the nature of human-generated spelling errors include the facts that: (1) most errors (i.e., roughly 80%) tend to be single instances of insertions, deletions, substitutions, or transpositions; (2) as a corollary, most errors tend to be within one letter in length of the intended word; and (3) few misspellings occur in the first letter of a word. These findings have been exploited by implementing fast algorithms for correcting single-error misspellings and/or by partitioning dictionaries according to first letter and/or word length to reduce search time. Other general findings of (4) strong keyboard adjacency effects and (5) strong letter frequency effects hold potential for improving correction accuracy. There is general agreement that phonetic errors are harder to correct because they result in greater distortion of the misspelled string from the intended word than single-error typos.

At least one study has examined individual errors to try to identify common mistakes that could be characterized by general rules, such as the tendency to double or undouble consonants. Another study examined errors to identify probabilistic tendencies, such as the fact that over one-third of all errors were omissions. And another study compiled general error probability tables for insertions, deletions, substitutions, and transpositions.

It is worth emphasizing that specific findings do not hold for all applications. It is imperative to collect a sufficiently large sample of errors characteristic of the target application for analysis and testing in order to select or devise a spelling correction technique that is optimally suited to that application. Finally, it is worth noting that one study found that fully 40% of all errors in a corpus were real-word errors, i.e., errors that are not detectable by isolated-word error correction techniques, thus emphasizing the need for research into context-dependent word correction techniques.

2.2 Techniques for Isolated-Word Error Correction

The problem of isolated-word error correction entails three subproblems: (1) **detection of an error**; (2) **generation of candidate corrections;** and (3) **ranking of candidate corrections.** Most techniques treat each subproblem as a separate process and execute them in sequence. The error detection process usually consists of checking to see if an input

string is a valid dictionary word or if its n-grams are all legal. The candidate generation process usually employs a dictionary or a database of legal n-grams to locate one or more potential correction terms. The ranking process usually invokes some lexical-similarity measure between the misspelled string and the candidates or a probabilistic estimate of the likelihood of the correction to rank order the candidates. Some techniques omit the third process, leaving the ranking and final selection to the user. Other techniques, including many probabilistic and neural net techniques, combine all three subprocesses into one step by computing a similarity or probability measure between an input string and each word, or some subset of words, in the dictionary. If the resulting top-ranked dictionary word is not identical to the input string, an error is indicated and a ranked list of potential corrections is computed.

As in detection research, n-gram statistics initially played a central role in text recognition techniques while dictionary-based methods dominated spelling correction techniques. But text recognition researchers quickly discovered that n-gram analysis alone was inadequate to the task of correction, so they began to develop "top-down/bottom-up" approaches that combined the use of dictionaries (the "top-down" aspect) with n-gram statistics (the "bottom-up" aspect). At the same time, n-gram statistics found their way into dictionary-based spelling correction research. In addition, many other clever techniques were invented based on minimum edit distance algorithms, similarity keys, rule-based procedures, probability estimates, and neural nets. A convenient way to organize the approaches is to group them into six main classes:

(1) minimum edit distance techniques;
(2) similarity key techniques;
(3) rule-based techniques;
(4) n-gram-based techniques;
(5) probabilistic techniques;
(6) neural nets.

This organization is somewhat arbitrary as there is some overlap among classes. In particular, the last three classes are closely related in that most implementations rely on some lexical feature-based representation of words and misspellings such as n-grams. Furthermore, many hybrid techniques combining the approaches of more than one class have been developed.

In the following discussion, each of the techniques is briefly described in terms of the candidate generation and ranking subprocesses. For the sake of providing some insight into how techniques differ in their scope and accuracy, some additional relevant characteristics are reported whenever that information is available. These include: (1) **lexicon size**; (2) **test set size**; (3) **correction accuracy for single-error misspellings**; (4) **correction accuracy for multi-error misspellings**; (5) **word length limitations**; and (6) **type of errors handled** (e.g., OCR-generated vs. human-generated, typographic vs. phonetic, etc.). These characteristics are important because techniques differ greatly in the types of errors they handle forming nonoverlapping or partially overlapping error coverage sets. Since few techniques have been tested on the same test set and lexicon, direct comparisons are rarely possible. The following sections are meant to convey a general idea of the scope and accuracy of existing isolated-word error correction techniques.

2.2.1 Minimum Edit Distance Techniques

By far the most studied spelling correction algorithms are those that compute a minimum edit distance between a misspelled string and a dictionary entry. The term *minimum edit distance* was defined by Wagner [1974] as the minimum number of editing operations (i.e., insertions, deletions, and substitutions) required to transform one string into another. The first minimum edit distance spelling correction algorithm was implemented by Damerau [1964]. About the same time, Levenshtein [1966] developed a similar

algorithm for correcting deletions, insertions, and reversals (transpositions). Wagner [1974] first introduced the notion of applying dynamic-programming techniques [Nemhauser 1966] to the spelling correction problem to increase computational efficiency. Wagner and Fischer [1974] also generalized Levenshtein's algorithm to cover multi-error misspellings. Another algorithmic variant, due to Lowrance and Wagner [1975], accounted for additional transformations (such as exchange of nonadjacent letters). Still, other variants were developed by Wong and Chandra [1976], Okuda et al. [1976], and Kashyap and Oommen [1981]. The metric that all of these techniques compute is sometimes referred to as a *Damerau-Levenshtein metric* after the two pioneers. Hawley [1982] tested some of these metrics in his spelling corrector for the Unix command language interface.

Aho's [1990] survey article of pattern-matching algorithms points out that the Unix file comparison tool, *diff*, is based on a dynamic-programming minimum edit distance algorithm. It goes on to describe other dynamic-programming algorithms in terms of their time complexities. Dynamic-programming algorithms have also been studied extensively for other approximate pattern-matching applications, such as comparing macromolecular sequences (e.g., RNA and DNA), comparing time-warped speech signals for speech recognition, comparing gas chromatograms for spectral analysis, and others. These are described in a collection of articles edited by Sankoff and Kruskal [1983].

Although many minimum edit distance algorithms compute integer distance scores, some algorithms obtain finer-grained scores by assigning noninteger values to specific transformations based on estimates of phonetic similarities or keyboard adjacencies. Veronis [1988a] devised a modified dynamic-programming algorithm that differentially weights graphemic edit distances based on phonemic similarity. This modification is necessary because phonetic misspellings frequently result in greater deviation from the correct orthographic spelling.

In general, minimum edit distance algorithms require m comparisons between the misspelled string and the dictionary, where m is the number of dictionary entries. However, some shortcuts have been devised to minimize the search time required. Going on the assumption that deletions are the most common type of error, Mor and Fraenkel [1982b] achieved an efficient response time for a full-text information retrieval application by storing every word in the dictionary $|x| + 1$ times, where $|x|$ is the length of the word, each time omitting one letter. They then invoked a hash function to look up misspellings.

A "reverse" minimum edit distance technique was used by Gorin [1971] in the DEC-10 spelling corrector and by Durham et al. [1983] in their command language corrector. Kernighan et al. [1990] and Church and Gale [1991b] also used a reverse technique to generate candidates for their probabilistic spelling corrector. In reverse techniques, a candidate set is produced by first generating every possible single-error permutation of the misspelled string and then checking the dictionary to see if any make up valid words. This means that given a misspelled string of length n and an alphabet of size 26, the number of strings that must be checked against the dictionary is $26(n + 1)$ insertions plus n deletions plus $25n$ substitutions plus $n - 1$ transpositions, or $53n + 25$ strings, assuming there is only one error in the misspelled string.

Tries have been explored as an alternative to dynamic-programming techniques for improving search time in minimum edit distance algorithms. Muth and Tharp [1977] devised a heuristic search technique in which a dictionary of "correct spellings are stored character-by-character in a pseudo-binary tree. The search examines a small subset of the database (selected branches of the tree) while checking for insertion, deletion, substitution, and transposition errors"

(p. 285). Dunlavey [1981] described an algorithm he called SPROOF, which stored a dictionary as a trie, i.e., "a huge finite-state machine recognizing all and only the strings in the dictionary" (p. 608). The search proceeded by visiting the nodes of the trie in increasing edit distance until the candidate corrections at the leaves were reached. A similar technique was used by Boivie [1981] in his *da* directory assistance program, which in turn became the underlying algorithm for Taylor's [1981] *grope* spelling corrector.

Minimum edit distance techniques have been applied to virtually all spelling correction tasks, including text editing, command language interfaces, natural language interfaces, etc. Their spelling correction accuracy varies with applications and algorithms. Damerau reports a 95% correction rate for single-error misspellings for a test set of 964 misspellings of medium and long words (length 5 or more characters), using a lexicon of 1,593 words. His overall correction rate was 84% when multi-error misspellings were counted. Durham et al. [1983] report an overall 27% correction rate for a very simple, fast, and unobtrusive single-error correction algorithm accessing a keyword lexicon of about 100 entries. For as low as this seems, the authors report a high degree of user satisfaction for this command language interface application, due largely to their algorithm's unobtrusiveness. Kukich [1990] evaluated the *grope* minimum edit distance algorithm, among others, for use in a demanding text-to-speech synthesis application in which 25% of 170 misspellings involved multiple errors and fully 63% of the misspellings involved short words (length 2, 3, or 4 characters). She observed a 62% overall correction rate, using a lexicon of 1,142 words. Other techniques evaluated by Kukich, including the vector distance and neural net techniques described below, outperformed the *grope* minimum edit distance technique for the same test set and lexicon by more than 10%. Muth and Tharp [1977] report a performance level as high as 97% for their trie-based algorithm on a 1,487-word test set, though they do not specify the size of the lexicon they used or the median length of the words in their test set.

2.2.2 Similarity Key Techniques

The notion behind similarity key techniques is to map every string into a key such that similarly spelled strings will have identical or similar keys. Thus, when a key is computed for a misspelled string it will provide a pointer to all similarly spelled words (candidates) in the lexicon. Similarity key techniques have a speed advantage because it is not necessary to directly compare the misspelled string to every word in the dictionary.

A very early (1918), often cited similarity key technique, the SOUNDEX system, was patented by Odell and Russell [1918] for use in phonetic spelling correction applications. It has been used in an airline reservation system [Davidson 1962] among other applications. It maps a word or misspelling into a key consisting of its first letter followed by a sequence of digits. Digits are assigned according to the following rules:

$$A, E, I, O, U, H, W, Y \rightarrow 0$$
$$B, F, P, V \rightarrow 1$$
$$C, G, J, K, Q, S, X, Z \rightarrow 2$$
$$D, T \rightarrow 3$$
$$L \rightarrow 4$$
$$M, N \rightarrow 5$$
$$R \rightarrow 6$$

Zeros are then eliminated and repeated characters are collapsed. So, for example, we can have: Bush → B020 → B2 and Busch → B0220 → B2. However, we could also have: QUAYLE → Q00040 → Q4 while KWAIL → K0004 → K4. It is clear that the SOUNDEX similarity key is a very coarse-grained one. Furthermore, the SOUNDEX system provides no function for ranking candidates. Instead, all candidates are simply presented to the user.

Pollock and Zamora [1984] used the findings of their study of 50,000 spelling errors from seven Chemical Abstracts

Service databases to devise a similarity key technique, called SPEEDCOP, for correcting single-error misspellings. After constructing a key for each word in the lexicon, the lexicon was sorted by key order. A misspelling was corrected by constructing its key and locating it in the sorted key list. Then all nearby words (candidates) within a certain distance in both directions were checked in sequence to find the first word that could have been generated from the misspelling by either an insertion, omission, substitution, or transposition (i.e., the ranking process).

The SPEEDCOP system actually made use of two similarity keys, a skeleton key and an omission key, both of which were carefully tailored by statistical findings to achieve maximum correction power for their application domain, which included few, if any, phonetic misspellings. Both the skeleton key and the omission key consisted of single occurrences of all the letters that appear in a string arranged in a specific order. In the skeleton key, the order is the first letter of the string followed by the remaining unique consonants in order of occurrence and the unique vowels in order of occurrence. In the omission key, consonants come first in reverse order of the frequency of occurrence of omitted letters, which, according to Pollock and Zamora's findings, is

RSTNLCHDPGMFBYWVZXQK.

Vowels follow, in order of their occurrence in the string. They also found that spelling errors tend to preserve the order of occurrence of vowels. The omission key for the word *chemical* is MHCLEIA. The keys for two actual misspellings *chemicial* and *chemcial* are identical, and the key for another misspelling *chemcal* is very close.

The SPEEDCOP algorithm consisted of four steps, each of which was applied only if the previous one failed to yield a correction: (1) consult a common misspelling dictionary; (2) apply the skeleton key; (3) apply the omission key; (4) apply a function word routine to check for concatenations of function words. Pollock and Zamora [1984] mention that the last step increased corrections by only 1%–2% but was entirely accurate. They report that the correction accuracy of their technique ranged from 77% to 96% in correcting the single-error misspellings in the seven databases they studied. Single-error misspellings accounted for an average of 94% of all the errors they observed. They report an overall correction rate ranging from 74% to 88%. These tests were based on a lexicon of 40,000 words.

A creative similarity key spelling correction technique was devised by Tenczar and Golden [1972] for the PLATO computer tutoring system. As in the SPEEDCOP technique, a similarity key was constructed for each word in the application lexicon and the lexicon was sorted by key order. But the PLATO similarity key was based on a set of human cognitive features intuited by the authors, including (1) word length, (2) first character, (3) letter content, (4) letter order, and (5) syllabic pronunciation. Each feature was represented by a bit field in the similarity key, and the authors point out that additional features could easily be included by adding another bit field. Locating the word typed by the student, or its closest approximation, consisted of computing the similarity key for the input word and executing a "binary chop" search through the encoded application lexicon to find an exact or closest match.

Tenczar and Golden [1972, p. 13] tested their technique on a dictionary of common misspellings and found that it corrected 95% of a sample of items taken from the dictionary. And they tested it by checking each of the 500 most commonly used English words against each other and found that it "performed satisfactorily, (i.e., pairs which it calls misspellings usually differ by only one letter)."

An algorithm called *token reconstruction* that computes a fuzzy similarity measure has been patented by Bocast [1991]. Given a misspelled string and a dictionary word, the algorithm returns a

similarity measure that is computed by taking an average of the sum of four empirically weighted indices that measure the longest matching substrings from both ends of the two strings. The technique partitions the dictionary into buckets containing words that have the same starting character c and the same length n. The search space expands from bucket c_i, n_j to buckets $c_i^*, n_j \pm 1$ so as to check first for words with small differences in length and then check for words beginning with other characters in the misspelled string. Bocast reports that the algorithm is simple enough to be blindingly fast, and that its 78% first-guess accuracy on a 123-word test set that includes 15 multi-error misspellings exceeds the accuracy of four popular commercial spelling correctors on the same test set.

2.2.3 Rule-Based Techniques

Rule-based techniques are algorithms or heuristic programs that attempt to represent knowledge of common spelling error patterns in the form of rules for transforming misspellings into valid words. The candidate generation process consists of applying all applicable rules to a misspelled string and retaining every valid dictionary word that results. Ranking is frequently done by assigning a numerical score to each candidate based on a predefined estimate of the probability of having made the particular error that the invoked rule corrected.

Yannakoudakis and Fawthrop [1983] devised a knowledge-based spelling correction program based on the set of rules they inferred from their study of 1,377 misspellings described above. Their goal was a general spelling correction system, so they tested their program on a set of 1,534 misspellings using a dictionary of 93,769 words. Because some of their rules incorporated knowledge of the probable length of the correct word based on the misspelling, their technique employed a dictionary that was partitioned into many subsets according to word length as well as first letter. The candidate generation process consisted of searching specific dictionary partitions for words that differ from the misspelling by only one or two errors and that follow any of the rules. When multiple candidates were found they were ranked by predefined estimates of the probabilities of occurrence of the rules. They found that the correct word was in the partition searched for 1,153 cases, or 75% of the time. For those 1,153 cases, the correct word was returned as the first choice 90% of the time, yielding an overall correction accuracy of 68% (90% of 75%).

In another project, a specialized rule-based spelling correction system was designed by Means [1988] for a natural language data entry system with a high incidence of abbreviation, acronym, and jargon usage. This system first checks a set of morphology rules for common inflectional violations (such as failure to double a final consonant before adding -ing). It next consults a set of abbreviation expansion rules to determine if the misspelled string might expand to a lexicon term. Failing that, it tries all single-error transformations of the misspelled string, including the possibility of an inserted blank (i.e., a split-word error).

2.2.4 N-gram-Based Techniques

Letter n-grams, including trigrams, bigrams, and/or unigrams, have been used in a variety of ways in text recognition and spelling correction techniques. They have been used in OCR correctors to capture the lexical syntax of a dictionary and to suggest legal corrections. They have been used in spelling correctors as access keys into a dictionary for locating candidate corrections and as lexical features for computing similarity measures. They have also been used to represent words and misspellings as vectors of lexical features to which traditional and novel vector distance measures can be applied to locate and rank candidate corrections. Some n-gram-based correction techniques execute the processes of error detection, candidate retrieval, and similarity ranking in three separate steps.

Other techniques collapse the three steps into one operation.

Riseman and Hanson [1974] provide a clear explanation of the traditional use of n-grams in OCR correction. After partitioning a dictionary into subdictionaries by word length, they construct positional binary n-gram matrices for each subdictionary. They note that because these matrices provide answers to the question "is there some word in the dictionary that has letters α and β in positions i and j respectively," the matrices capture the syntax of the dictionary. An OCR output string can be checked for errors by simply checking that all its n-grams have value 1. If a string has a single error, it will have a 0 value in at least one binary n-gram; if more than one 0 value is found, the position of the error is indicated by a matrix index that is common to the 0 value n-grams. Correction candidates can be found by logically intersecting the rows or columns specified by the shared index of the incorrect n-grams. If the intersection results in only one n-gram with value 1, the error can be corrected; if more than one potential n-gram correction is found the word is rejected as ambiguous.

Riseman and Hanson [1974] tested the technique on a test set of 2,755 6-letter words. They found that positional trigram matrices, which outperformed other positional and nonpositional n-gram matrices, were able to detect 98.6% of the errors in their test set and correct 62.4% of them. This technique has at least one advantage over dictionary lookup correction techniques in that it obviates the need for an exhaustive dictionary search. A minor drawback it that it is possible for a correction to result in a nonword on rare occasions. A more serious drawback is that the technique is designed to handle only substitution errors, and it is not clear how well it would handle other errors, such as insertion, deletion, transposition, and framing errors.

Two hardware-based techniques have been proposed that would exploit n-gram databases in parallel. Ullmann [1977] proposed a technique for finding all valid words differing from an input string by up to two insertion, deletion, substitution, and reversal errors by processing binary n-grams in parallel. Henseler et al. [1987] implemented a parallel OCR recognition and correction algorithm on a 16-processor hypercube machine. Their technique used a database of trigram frequencies as opposed to binary trigrams. A test of their technique on a small sample of text yielded 95% initial character recognition accuracy and 100% accuracy after correction.

Angell et al. [1983] have written a seminal paper on the use of trigrams for spelling correction applications. Their technique computes a similarity measure based on the number of (nonpositional binary) trigrams common to a misspelled string and a dictionary word. The similarity measure is the simple function $2(c/(n + n'))$, where c is the number of trigrams common to both the dictionary word and the misspelling and n and n' are the lengths of the two strings. They refer to this function as "the well-known Dice coefficient." They also suggest creating an inverted index into the dictionary using trigrams as access keys. The trigrams of the misspelled string can be used to retrieve only those dictionary words having at least one trigram in common with the misspelling, and the similarity function needs to be computed only for that subset.

This technique achieved an overall accuracy score of 76% on a test set of 1,544 misspellings using a dictionary of 64,636 words. The authors analyzed its performance by error type and found that it worked "very well indeed for omission and insertion errors, adequately for substitution errors, but very poorly for transposition errors" (p. 259). Although they did not break down the technique's performance by word length, they did note that it cannot be expected to do well on short words because a single error can leave no valid trigrams intact. The mean length of the misspelled strings in their test set was 8.4 characters.

A problem with the Dice coefficient was that it tended to make errors when the

misspelled string was wholly contained in a valid word, or vice versa. For example, given the misspelled string *concider*, it assigned values of 0.71 and 0.70 respectively to the words *cider* and *consider*. Angell et al. noted that the tendency for misspellings to differ in length from their intended correction by at most one character could be exploited by changing the similarity function to $(c/\max(n, n'))$. They found that this modified similarity function corrected the containment error problem without affecting the correction rate for multi-error misspellings.

Similar trigram correction techniques have been suggested or devised by Kohonen [1980] and DeHeer [1982] for text editing and information retrieval applications. A related technique, triphone analysis, was devised by Van Berkel and DeSmedt [1988] specifically for a natural language interface application involving the use of proper nouns that tend to be phonetically misspelled. It first applies a set of grapheme-to-phoneme conversion rules to a string and then makes use of an inverted-triphone index to retrieve other candidate corrections. It was highly successful in a test experiment, correcting 94% of a test set of deviant spellings of 123 Dutch surnames. Its lexicon was a database of 254 Dutch surnames randomly culled from a telephone directory. Van Berkel and DeSmedt compared their phonetic-based technique to some well-known nonphonetic techniques and found that the phonetic technique outperformed the others by anywhere from 4 to 40 percentage points on this test set.

In addition to the trigram-based similarity functions suggested by Angell et al. [1983], a variety of traditional and novel vector distance measures based on representations of words as n-gram vectors has been tested. Most of these techniques collapse all three subprocesses (detection, candidate generation, and ranking) into one process. The basic ideas are: (1) to position each lexicon entry at some point in an n-dimensional lexical-feature space and then (2) to project a misspelled string into that same space and measure its proximity to its nearest neighbors (candidates). Unigrams, bigrams, and trigrams are three possible candidates for the features of the lexical space; words and misspellings can be represented as sparse vectors in such spaces. Hamming distances, dot products, and cosine distances are three possible measures of the proximity of any two vectors in lexical space.

Kukich [1992] evaluated the correction accuracy of these three vector distance metrics using the same test set and lexicon as those used to test the *grope* minimum edit distance technique (which scored 62% accuracy); the test set contained 170 single and multi-error misspellings of which nearly two-thirds occurred in short words (less than 5 characters), and the lexicon contained 1,142 words. Using 790-element vectors of unigrams and bigrams to represent words and misspellings, she recorded correction rates of 54% for a dot product metric, 68% for a Hamming distance metric, and 75% for a cosine distance metric.

Correlation matrix memory (CMM) techniques are closely related to vector space techniques. They represent a lexicon of m words as an $n \times m$ matrix of n-dimensional lexical-feature vectors. The correction process consists of multiplying the n-dimensional feature vector representing the misspelled string by the $n \times m$-dimensional matrix representing the entire lexicon, yielding an m-dimensional vector in which the ith element represents the ith lexicon entry. The element with the highest value is taken to be the most strongly correlated entry, thus the correct word.

Cherkassky et al. [1992] have done detailed studies comparing the use of bigram and trigram feature vectors in CMM models. They created two test sets of randomly generated single-error misspellings, one for medium-length words (5–7 characters) and one for long words (10–12 characters), and tested them against lexicons ranging in size from 500 to 11,000 words. In particular, they observed excellent correction rates (above 90%) for long words using trigram fea-

ture vectors and fair-to-good correction rates (averaging from 60% to 85% depending on lexicon size) for medium-length words using bigram feature vectors. An even more detailed study by Dahl and Cherkassky [1990] compared the use of unigrams, bigrams, and trigrams on words of varying lengths for each of the four classes of single-error typos. They found that "[b]igram and trigram encodings give better recall rates for all kinds of errors except transpositions for all word lengths" and that a unigram encoding performs much better for transposition errors. Indeed, a lexical study by Deloche and Debili [1980] found that in both French and English, 96% of the anagrams created from lexicons of over 20,000 uninflected word forms had unique solutions.

Some attempts have been made to transform lexical-feature spaces by known mathematical techniques in an effort to better reflect the similarity relations among lexical entries, and thus improve correction accuracy. One such technique, the computation of the Generalized Inverse (GI) matrix, was explored by Cherkassky et al. [1990]. The goal of the GI technique, which is based on finding a minimum least squares error inverse of the lexicon matrix, is to minimize the "crosstalk" or interference that occurs among similar lexicon entries. Later however, Cherkassky et al. [1991] derived a proof, based on a theorem of linear algebra, that a GI matrix saturates when the number of lexicon entries approaches the dimensionality of the feature space, and a significant decline in correction accuracy follows. This led them to conclude that simple CMM models are more effective and efficient for spelling correction.

Another technique for transforming a feature space, Singular Value Decomposition (SVD), was explored by Kukich [1990]. SVD can be used to decompose a lexicon matrix into a product of three matrices, one representing each of the individual-letter n-grams as a vector of factors, a second, diagonal matrix consisting of a set of singular values that are used to weight each of the factors, and a third matrix representing each of the lexicon entries as a vector of factors. The goal of the decomposition process is to identify and rank the most important factors that determine the relevant similarity relations in the lexical space. The least important factors, which may in fact represent noise in the data, can be discarded, yielding three matrices of reduced rank which better capture the essential similarity relations among lexical entries. SVD has been used successfully to this end in information retrieval experiments [Deerwester et al. 1990]. In those experiments, bibliographic databases consisting of thousands of documents and lexicon entries are represented by sparse term-by-document matrices. The technique has been dubbed Latent Semantic Indexing because the SVD process appears to elicit the inherent semantic relations among terms and documents.

In contrast to the information retrieval application, a spelling correction matrix in which words are represented by bigram and unigram vectors is far less sparse. Once such a matrix has been decomposed into three factor matrices and reduced, a misspelling is corrected by first summing the vectors for each of the individual-letter n-grams in the misspelled string (as represented in the first matrix) and then multiplying the sum vector by the singular-value matrix of weights (represented in the second matrix). The resultant vector determines the location of the misspelled word in the n-dimensional lexical-feature space. Any standard distance measure (such as a dot product or a cosine distance) can be used then to measure the distances between the vector for the misspelled word and the vectors for each of the correctly spelled words (represented by the third matrix) in order to locate and rank the nearest correctly spelled words. In Kukich's studies, SVD matrices yielded improved spelling correction accuracy over undecomposed matrices for a small lexicon (from 76% to 81% for a lexicon of 521 words), but no significant improvement

for larger lexicons. These empirical findings are consistent with the theoretical findings of Cherkassky et al. [1991] regarding the saturation point of Generalized Inverse matrices.

Bickel [1987] devised a hybrid unigram and heuristically based vector distance correction technique for a database retrieval application that used employee names as access keys. The lexicon for this application consisted of nearly one thousand personal names. Bickel's technique represented each valid name as a unigram vector. The value of each unigram vector element was set to either 0 if that letter did not occur in the name or to an integer between 3 and 9 if the letter did not occur in the name. Integer values for each letter of the alphabet were predetermined according to their frequency of occurrence in the lexicon of names, with least frequent letters receiving the highest weights. The assumption was that less frequently occurring letters were more valuable as search information. Potentially misspelled input names were represented by binary unigram vectors. The correction technique assumed the first letter of the name was correct and confined its search to that portion of the lexicon. It derived a similarity measure for the input name and each valid name in the sublexicon by computing the inner product of the two vectors. Bickel found this similarity measure to be more than 95% successful in locating the intended name given another spelling.

2.2.5 Probabilistic Techniques

N-gram-based techniques led naturally into probabilistic techniques in both the text recognition and spelling correction paradigms. Two types of probabilities have been exploited, **transition probabilities** and **confusion probabilities**. Transition probabilities represent probabilities that a given letter (or letter sequence) will be followed by another given letter. Transition probabilities are language dependent. They are sometimes referred to as Markov probabilities based on the assumption that language is a Markov source. They can be estimated by collecting *n*-gram frequency statistics on a large corpus of text from the domain of discourse.

Confusion probabilities are estimates of how often a given letter is mistaken or substituted for another given letter. Confusion probabilities are source dependent. Because different OCR devices use different techniques and features to recognize characters, each device will have a unique confusion probability distribution. For that reason, confusion probabilities are sometimes referred to as channel characteristics. The same OCR device may generate different confusion probability distributions for different font types and points sizes.

A confusion probability model for a specific device can be estimated by feeding the device a sample of text and tabulating error statistics. This process is sometimes referred to as a "training" phase when the results are used in the development of an error-correcting postprocessor. Alternatively, a device can generate a 26-element vector containing a likelihood estimate for each letter of the alphabet at the time a character is recognized. In that case the likelihood estimates may be interpreted as confusion probabilities.

Confusion probabilities based on human errors are simply called error probabilities. They, too, have been estimated by extracting error statistics from large samples of human-generated text containing typos or other spelling errors. One other class of probabilistic information that has been exploited for isolated-word correction techniques is word frequency data, or word unigram probabilities.

Text recognition researchers have intensively explored the use of probabilistic information; spelling correction researchers have only recently joined the fray. Text recognition research has shown that these probabilistic sources alone are insufficient to achieve acceptable error correction rates; however, combining probabilistic information with dictionary techniques results in significantly better error correction ability. A brief history of

probabilistic error correction research follows.

Bledsoe and Browning [1959] pioneered the use of likelihood estimates in text recognition techniques. The Bledsoe-Browning technique entails two phases: (1) an individual-character recognition phase in which a 26-element vector of likelihood estimates is generated for each character in an input word, followed by (2) a whole-word recognition phase in which a dictionary is used to help choose the individual letters whose likelihoods jointly maximize the probability of producing a valid word from the dictionary. In effect, whole-word recognition enhances individual-character recognition by imposing dictionary constraints that help resolve uncertainty and correct errors made at the individual-character recognition level.

The Bledsoe-Browning technique uses Bayes' rule to compute an *a posteriori* probability for each word in the dictionary from the a priori likelihood probabilities for the individual characters. Letting X represent a dictionary word and Y represent an OCR output string, Bayes' rule states that

$$P(X|Y) = \frac{P(Y|X)^* P(X)}{P(Y)} \quad (1)$$

where $P(X|Y)$ is the conditional probability that X is the correct word, $P(Y|X)$ is the conditional probability of observing Y when X is the correct word, and $P(X)$ and $P(Y)$ are the independent probabilities of words X and Y. Finding the most probable dictionary word X given the OCR output string Y amounts to maximizing the function

$$G(Y|X) = \log P(Y|X) + \log P(X) \quad (2)$$

where $P(X)$ is often taken to be the unigram probability of the word X. The a posteriori probability for each dictionary word can be readily computed from the a priori likelihood estimates for individual characters by the equation

$$\log P(Y|X) = \sum_{i=1}^{i=n} \log P(Y_i|X_i) \quad (3)$$

where n is the length of the word, and i indexes the individual characters in the word. For example, given an OCR output string *DOQ* and a dictionary word *DOG*,

$$\begin{aligned} G(DOQ|DOG) = &\log P(D|D) \\ &+ \log P(O|O) \\ &+ \log P(Q|G) \\ &+ \log P(DOG). \end{aligned} \quad (4)$$

In the Bledsoe-Browning technique, the likelihood estimates for each character in the output string are supplied by the OCR device; the unigram word frequency is ignored; and the word with the maximum probability is chosen.

The computational demands of the Bledsoe-Browning technique are linear with the size of the dictionary. Hence, the processing of large dictionaries is infeasible, even if they are partitioned by word length. In an application for a handwriting recognizer, Burr [1983] extended the Bledsoe-Browning technique to incorporate morphological processing. This allowed him to use a smaller dictionary of root forms of words along with prefixes and suffixes.

Kahan et al. [1987] have also combined confusion probabilities with dictionary lookup in a postprocessor for OCR output. When an output word is rejected by spelling checker, they generate successive candidates, based on confusion probabilities obtained through training, until an acceptable dictionary word is found or some threshold is exceeded. They report a 97% character recognition rate ignoring indistinguishable pairs such as 0/O (zero/oh) and 1/l (one/ell).

Techniques have been tried that make use of confusion and transition probabilities without the benefit of a dictionary. Hanson et al. [1976] report on experiments using likelihood probabilities in combination with either trinary transition probabilities or positional binary trigrams. They found that using trinary transition probabilities to directly weight the likelihoods of individual characters

in OCR output words reduced the word error rate on a test set of 2,755 6-letter words, but only from 49% to 29%. They concluded that efficient techniques based on transition probabilities would not be effective. They went on to find that positional binary trigrams alone corrected over 50% of the word errors in their test set, and that additional processing using likelihood probabilities further reduced the error rate by an order of magnitude.

Both Toussaint [1978] and Hull and Srihari [1982] provide general overviews of error correction techniques based exclusively on transition and confusion probabilities. Bayes' formula can be used to combine these two kinds of probabilities. Instead of using a word unigram probability for the final term in the formula, the sum of the transition probabilities in the dictionary word can be substituted. But a more efficient and widely used method for combining transition and confusion probabilities is a dynamic-programming technique called the Viterbi algorithm [Forney 1973]. In the Viterbi algorithm, a directed graph, sometimes referred to as a trellis, is used to capture both the structure of a lexicon (transition probabilities) and the channel characteristics of a device (confusion probabilities). A starting node and an ending node represent word boundary markers; intermediate nodes represent likelihood estimates for individual letters; and edges represent transition probabilities between letters. The graph is efficiently traversed via a dynamic-programming algorithm to find the sequence of letters having the highest probability given the likelihood estimates output by the OCR device and the transition probabilities of the language.

Various modifications to the original Viterbi algorithm, e.g., Shinghal and Toussaint [1979a], that limit the depth of a search based on likelihoods or other factors have been developed. A disadvantage of all these techniques is that the highest probability string is not always a valid word. More importantly, Toussaint, Hull, Srihari, and others have concluded that the correction accuracy of techniques based solely on these two probabilistic sources is only mediocre.

One other technique worth noting for exploiting these two probabilities is a relaxation technique described by Goshtasby and Ehrich [1988]. It calls for setting the initial character probabilities for an OCR output word according to the likelihood estimates for each character. Rosenfeld's relaxation formula [Rosenfeld et al. 1976] is applied to iteratively adjust the likelihood probabilities for each character as a function of the transition probabilities of adjacent characters. The number of confusion candidates for each character can be limited to a small set that exceeds some threshold, thus making the relaxation process itself computationally quite affordable, on the order of $10n^2$ for a word of length n. As with the Viterbi algorithm, the relaxation process can converge on a nonword. For example, during one test it converted the correct word *biomass* to *biomess*. The authors point out that the process can be improved by incorporating additional knowledge. They suggest using syllable digram information to increase the probabilities of the candidates that form compatible syllables at each iteration. They also note that the relaxation process could be integrated with a dictionary-based technique.

Probabilistic information has been used effectively as a preprocessor for a spelling correction application. Oshika et al. [1988] implemented a Hidden Markov Model (HMM) procedure for preclassification of surnames according to ethnic background. HMM's were automatically constructed for each of five different languages based on transition probabilities of letter sequences in 2,000 sample surnames from each language. The HMMs were then used to automatically classify input names before generating language-specific spelling variants of the names. Oshika et al. report that their preprocessing technique improved retrieval accuracy from 69% to 88% on a test of 160 query names.

Better error correction performance has been achieved by techniques that

combine the use of probabilistic information (bottom-up information) with dictionaries (top-down information). As Shinghal and Toussaint [1979b] note, "Dictionary methods have low error-rates but suffer from large storage demands and high computational complexity. Markov methods have the inverse characteristics" (p. 201). So they invented a technique they call the predictor-corrector method that "achieves the error-correcting capability of dictionary look-up methods at half the cost." The technique calls for sorting the dictionary according to a precomputed value V for each word that is based on the word's transition probabilities. The predictor-corrector algorithm first uses a modified Viterbi algorithm to recognize an input word. Then it computes V for the recognized word and does a binary search of the dictionary for the word with a matching value. If that word is not an exact match, it calculates V for words in the neighborhood and chooses the one with the highest score. In an experiment with a dictionary of 11,603 words and two test sets of 2,521 and 2,256 words, the technique achieved character recognition rates of 96.6% and 96.4%. Unfortunately, Shinghal and Toussaint [1979] note that the technique worked best when the neighborhood included the whole dictionary, so they suggest the need for a better method for organizing the dictionary. Even so, they note that the technique still cut dictionary-processing time in half because the modified Viterbi algorithm produced the correct word more than half the time.

Srihari et al. [1983] implemented a technique in which the dictionary is represented as a trie and integrated into the Viterbi algorithm search "so as to retain VA [Viterbi algorithm] efficiency while guaranteeing the most likely dictionary word as output" (p. 72). They tested their technique on a sample text of 6,372 words that yielded a dictionary of 1,724 words. They found that their Viterbi algorithm alone was able to correct 35% of the errors made by the OCR device; their dictionary lookup algorithm alone corrected 82%; and their combination Viterbi-dictionary algorithm corrected 87%.

Sinha and Prasada [1988] developed a technique that not only integrated probabilistic and dictionary information but also incorporated a variety of heuristics. Their objective was to tackle the problem that in practice one cannot assume that a dictionary will contain all the words found in a document. So they compiled a "partial dictionary" by starting with a list of the 10,000 most frequent words from the Brown Corpus [Kucera and Francis 1967] and augmenting it "with all the valid words obtained by single character substitutions" (p. 465). Their algorithm takes two passes. It first corrects only those errors for which confusion probabilities produce a word found in the partial dictionary. Then it uses a modified Viterbi algorithm to estimate the most likely correction for words not in the dictionary. The dictionary is searched as a trie, and the system uses various heuristics to rank confusion candidates, limit the search through the trie, and limit the branches of Viterbi net searched. They tested their technique on a test set of 5,000 words (25,000 characters) of text from typewritten and typeset documents in seven different font types and point sizes. Its overall performance exceeded 98% character recognition accuracy, and its word recognition accuracy was around 97% not counting punctuation errors. They note that the "augmented dictionary approach yields better performance as compared to the unaugmented version in the case of texts that are not very conventional such as technical and literary documents . . . . However, performance [on conventional texts] deteriorates for low performance [$< 90\%$] IOCR [devices] with the augmented dictionary . . . [because] a comparatively larger number of [correction candidates] get generated and many of the high frequency words . . . get mapped to less frequent words in the augmented dictionary" (p. 474).

Jones et al. [1991] have implemented an OCR postprocessor that "integrates a Bayesian treatment of predictions made

by a variety of knowledge sources" (p. 926). In a training phase they build statistical models for both the OCR device and the documents to be processed. Their models include letter unigram and digram frequencies, binary letter trigrams and initial-letter *n*-grams, word unigram frequencies, and some word digram frequencies. They include letter and word digrams involving punctuation and capitalization. They adjust their statistics to reflect undertraining so that, for example, undertrained word digram frequencies do not unduly influence results. Their confusion probability models include non-one-to-one mappings (e.g., *and* → *aml*).

The corrector operates in three phases: (1) it generates a ranked list of candidates for each word of input based on confusion probabilities and a trie-based dictionary; (2) it merges words to undo apparent word-splitting errors; (3) it reranks candidates based on word digram probabilities. Phases 2 and 3 feed back proposed analyses into previous phases.

Jones et al. [1991] tested their system in two experiments. In the first experiment, they trained the system on six software-testing documents totaling 8,170 words and tested on a seventh document containing 2,229 words. The raw OCR word error rate was 16.2%. Their system corrected 89.4% of those errors. In the second experiment, they trained the system on 33 pages of AP news wire text and tested it on 12 additional pages. The raw OCR word error rate on the 12 pages varied from 6.8% to 10.3%, and their system corrected 72.5% of those errors. This system is significant in that it is one of the first to incorporate extra-word contextual knowledge in the form of word and punctuation digram probabilities into the correction process.

Recently, techniques that make explicit use of error probabilities have begun to be explored for spelling corrections applications. Kashyap and Oommen [1984] were motivated to explore using error probabilities in response to the poor performance of traditional spelling correction techniques on short words (less than six characters). Observing that misspellings frequently involve the mistyping of an adjacent key on the typewriter keyboard, they compiled a table of subjective estimates of the probabilities of such substitution errors. They also ascribed subjective estimates to the probabilities of a single deletion or insertion of any character at various positions within a word. Assuming that every spelling error can be reduced to a combination of such deletions, insertions, and substitutions, they devised a recursive algorithm that used their estimates for calculating the probability that a given misspelled string is a garbled version of a dictionary word. When applied to each word in a dictionary, this algorithm computes a probability ranking over the dictionary of correct words for a misspelled input string.

Kashyap and Oommen [1981] tested their technique on artificially generated misspellings in words of length 3, 4, and 5 characters. Misspellings were generated according to their error probability tables with an average of between 1.65 and 1.90 errors per word. They used a dictionary of the 1,025 most frequent words. They report correction accuracies ranging from 30% to 92% depending on word length and number of errors per word. They report also that these figures compare favorably with those reported by Okuda et al. [1976] which ranged from 28% to 64%, for words of similar length and error characteristics. Okuda et al. used a Damerau-Levenshtein minimum edit distance correction technique.

Kernighan et al. [1990] and Church and Gale [1991a] devised an algorithm for correcting single-error misspellings. They used a database of genuine errors extracted from a 44 million-word corpus of AP news wire text to compile four confusion matrices, one for each of the four standard classes of errors (insertions, deletions, substitutions, and transpositions). They estimated confusion probabilities then from these statistics. Their program, called *correct*, used a reverse minimum edit distance technique

to generate a set of single-error candidates and identify a specific error within each candidate, followed by a Bayesian computation to rank the candidates.

Correct starts with a misspelled string detected by *spell*. First, it generates a set of candidate correct spellings by systematically testing all possible single-error typographic permutations of the misspelled string, checking to see if the result is a valid word in the *spell* dictionary. A precomputed deletion table and hashing are used to minimize dictionary accesses during candidate generation. Candidates are then ranked by multiplying the a priori probability of occurrence of the candidate word (i.e., its normalized frequency) by the probability of occurrence of the specific error by which the candidate word differs from the misspelled string.

Correct was evaluated by testing it on a set of 332 misspellings from a subsequent month of AP news wire text. Each of these misspellings had exactly two candidate correct spellings, and both *correct* and three human judges were asked to choose the best candidate. The human judges were allowed to pick "neither," and they were given a surrounding sentence of context. *Correct* agreed with at least two of the three judges 87% of the time.

Kernighan went on to build a version of *correct* into a text-to-speech synthesizer for TDD conversations [Kernighan 1991]. Before pronouncing a word, this system's pronunciation component [Coker et al. 1990] computed a pronunciation difficulty index to avoid unnecessarily correcting misspelled words whose pronunciation would be acceptable, such as *operater* and *wud*. Note that the latter might be incorrectly changed to *wed* when *would* was intended. A misspelling was sent to *correct* only when the pronunciation difficulty index exceeded a threshold. In an experiment with human listeners, Kernighan's system significantly improved the comprehensibility of synthesized text. Kernighan and Gale [1991] also tested some variations of the technique on a spelling corrector for Spanish.

In a related study, Troy [1990] investigated the effects of combining probabilistic information with a vector distance technique. She used a cosine distance metric, word unigram probabilities, and Kernighan-Church-Gale (KCG) [1990] error probabilities on Kukich's [1990] 170-word test set and 1,142-word lexicon. The cosine distance metric was used to generate a candidate correction set; a dynamic-programming technique was used to identify specific errors within the candidates; and either word unigram probabilities or the KCG error probabilities were used to rerank candidates. Since the cosine distance metric could handle single- and multi-error misspellings, candidates could have multiple errors. Troy observed that the word frequency information did not improve the correction accuracy of the cosine distance metric, but error probability information did improve the cosine distance metric accuracy by three percentage points (from 75% to 78%).

2.2.6 Neural Net Techniques

Neural nets are likely candidates for spelling correctors because of their inherent ability to do associative recall based on incomplete or noisy input. Furthermore, because they can be trained on actual spelling errors, they have the potential to adapt to the specific error patterns of their user community, thus maximizing their correction accuracy for that population. One can imagine a neural net learning chip in a personal workstation that continuously adapts its weights to conform to its owner's idiosyncratic spelling error behavior.

The back-propagation algorithm [Rumelhart et al. 1986] is the most widely used algorithm for training a neural net. A typical back-propagation net consists of three layers of nodes: an input layer, an intermediate layer, usually referred to as a hidden layer, and an output layer. Each node in the input layer is connected by a weighted link to every node in the hidden layer. Similarly, each node in the hidden layer is connected by a weighted

link to every node in the output layer. Input and output information is represented by on-off patterns of activity on the input and output nodes of the net. A 1 indicates that a node is turned on, and a 0 indicates that a node is turned off. Patterns of continuous real numbers can also be used to indicate that nodes are more or less active.

Processing in a back-propagation net consists of placing a pattern of activity on the input nodes, sending the activity forward through the weighted links to the hidden nodes, where a hidden pattern is computed, and then on to the output nodes, where an output pattern is computed. Weights represent connection strengths between nodes. They act as resistors to modify the amount of activity reaching an upper-level node from a lower-level node. The total activity reaching a node is the sum of the activities of each of the lower-level nodes leading up to it, multiplied by their connection strengths. This sum is usually thresholded to produce a value of 0 or 1 or put through a squashing function to produce a real number between 0.1 and 0.9.

The back-propagation algorithm provides a means of finding a set of weights for the net that allows the net to produce the correct, or nearly correct, output pattern for every input pattern. It is a convergence technique that relies on training from examples. Weights are initialized to small random values. Then, for each input-output pair in a training set, the input pattern is placed on the input nodes of the net, and activation is sent forward through the weights to the hidden and output nodes to produce an output pattern on the output nodes. The actual output pattern is then compared to the desired output pattern, and a difference is computed for each node in the output layer. Working backwards through the net, the difference is used to adjust each weight by a small amount inversely proportional to the error. This procedure is used to cycle repeatedly through all the examples in the training set until the weights converge. The result is a set of weights that produces a nearly correct output pattern for every input pattern in the training set, as well as for similar input patterns that were not in the training set. This last factor is referred to as the net's ability to generalize to noisy and novel inputs.

For example, in a spelling correction application, a misspelling represented as a binary n-gram vector might serve as an input pattern to the net. Its corresponding output pattern might be a vector of m elements, where m is the number of words in the lexicon, and only the node corresponding to the correct word is turned on. This type of net is sometimes referred to as a 1-of-m classifier because the goal of the net is to maximize activation on the output node representing the correct word and minimize activation on all other nodes. In fact, the values of the nodes in the resultant output vector are sometimes interpreted as likelihood values. The 1-of-m encoding scheme used for the output pattern is referred to as a local encoding scheme because each word in the lexicon is represented by a single node. In contrast, the binary n-gram encoding scheme used for the input pattern is referred to as a distributed encoding scheme because no one node contains all the information needed to represent a word. Rather, the information is distributed across the whole vector of input nodes.

Neural nets are being intensively explored for OCR applications, not so much for word recognition as for applications of handwritten-number recognition such as postal zipcode reading and bank check and credit card receipt reading [Matan et al. 1992; Keeler and Rumelhart 1992]. These OCR applications are inherently difficult due to the lack of contextual information available. Some neural net word recognition work has been done, however. Burr [1987] implemented a two-phase hand-printed-word recognizer that combines a neural net recognition technique with a probabilistic correction technique. In phase one, his system uses a neural net with 26 output nodes, each representing one letter, and 13 input nodes, each representing one segment of a bar mask, for capturing the features of a handprinted letter. The output of the

neural net is a useful approximation to a maximum-likelihood probability distribution over the 26 letters of the alphabet. In phase two, the system uses Bayes' formula to compute a probability estimate for each word in the dictionary based on the likelihood probabilities of the individual characters and the unigram probabilities of the words. In a test experiment, Burr created a set of 208 hand-printed characters, eight instances of each letter in the alphabet. He trained a neural net using half the set and tested it on the other half. The net attained 94% character recognition accuracy on the test set. Then, he tested his word recognition technique by simulating a hand-printed word for each word in a 20,000-word dictionary using instances of letters in the test set. He obtained a word recognition rate of 99.7% for words at least six characters long.

Neural nets have been investigated by both Kukich [1988a, 1988b] and Cherkassky and Vassilas [1989a, 1989b] for name correction applications, by Kukich [1990] for a text-to-speech synthesis application, by Gersho and Reiter [1990] for an address correction application, and by Deffner et al. [1990a, 1990b] for a natural language interface application. The studies by Kukich and Cherkassky and Vassilas explored the use of the back-propagation algorithm for training nets; Gersho and Reiter's experiments made use of both self-organizing and back-propagation nets; and the system devised by Deffner et al. employed a weighted correlation matrix model to combine a variety of sources of lexical knowledge.

For her name correction experiments Kukich [1988a] used a lexicon of 183 surnames to train a standard back-propagation net. The output layer consisted of 183 nodes, one for each name. The input layer contained 450 nodes in 15 sequential blocks of 30 nodes each, so names of up to 15 characters in length could be represented. Each 30-node block contained one node for each character in a 30-character alphabet. So if the letter *C*, for example, appeared as the first character in a string, the node representing *C* would be turned on (set to 1) in the first block, and the remaining 29 nodes in that block would be turned off (set to 0). Thus, it was a positionally encoded shift-sensitive input vector.

The net was trained on hundreds of artificially generated single-error misspellings of each of the 183 names. Tens of hours of training time on the equivalent of a Sun SPARCstation 2™ processor were required for the net to converge. The net was tested on additional randomly generated single-error misspellings. Testing required only a few seconds of CPU time. The net achieved near-perfect (approaching 100%) correction accuracy for all four classes of typos, including the shift-variant insertion and deletion typos. In fact, these latter error types did take slightly longer to learn.

Some of the experimental findings were that: (1) nets trained on names with misspellings learned better than nets trained on perfectly spelled names; (2) as the number of hidden nodes was increased up to 183 nodes, performance increased; above 183 nodes performance leveled off; and (3) nets using local encoding schemes learned more quickly than variants using distributed encoding schemes. Subsequent experiments revealed that nets using the positionally encoded shift-sensitive input scheme and **no** hidden layer actually learned more quickly and performed at least as well as the optimally configured hidden-layered nets. However, nets that used a binary *n*-gram (unigram-plus-bigram), hence shift-invariant, input scheme, while exhibiting the same performance increase as the number of hidden nodes was increased, performed miserably with no hidden layer.

The experiments performed by Cherkassky and Vassilas [1989a, 1989b] corroborated many of Kukich's findings. They separately tested both unigram and

™ Sun SPARCstation 2 is a registered trademark of Sun Microsystems, Inc.

bigram vector encoding schemes for input to the nets, and they used the same local encoding scheme (one node per lexicon entry) for output; thus their nets were also 1-of-*m* classifiers. They trained nets on correctly spelled names and tested them on names with artificially generated deletion and substitution errors. Lexicon sizes ranged from 24 to 100 names. They found that nets attained almost 100% correction accuracy for the smallest lexicons, but the choice of learning rate and number of hidden units made significant differences in the performance of the nets. They found also that the optimum number of hidden units was close to the number of names in the lexicon. One might be tempted to conclude from this fact that nets were simply performing a rote lookup operation, and no generalization was taking place. However, this is clearly not the case, as evidenced by the fact that nets were tested on novel misspellings which they had never seen during training. Cherkassky and Vassilas concluded that because the nets were so sensitive to various training parameters, and because they required extensive computational training time, they are less desirable candidates for spelling correctors than correlation matrix models. An overview paper by Cherkassky et al. [1992] summarizes these experiments and discusses the relative merits and shortcomings of correlation matrix models and neural nets.

Kukich [1990] went on to investigate how such nets might scale up for a text-to-speech spelling correction application requiring a larger lexicon and having a significant portion (25%) of multi-error misspellings. A standard three-layer, back-propagation architecture was used, with a 420-element unigram-plus-bigram vector input layer, a 500-node hidden layer, and a locally encoded 1,142-node output layer. Again, the net had to be trained on artificially generated errors because not enough genuine errors were available, but it was tested on the same set of 170 genuine misspellings previously used to test vector space and other techniques. Like the best vector space techniques, the net achieved an accuracy score of 75%. Computational training time requirements were high, however—on the order of hundreds of hours of Sun SPARCstation 2 CPU time. Thus, it appears that the heavy computational training demands of standard three-layer, 1-of-*m* back-propagation nets impose strict practical limits on scalability with respect to lexicon size, at least with currently available computational power.

However, techniques for reducing training time by partitioning the lexicon or by exploiting alternative network architectures are being explored. One such variation was investigated by Gersho and Reiter [1990] for a database retrieval application involving up to 1,000 street names. They devised a hierarchical two-layer network strategy consisting of a self-organizing Kohonen-type network [Kohonen 1988] and many small back-propagation networks. Input to their system goes first to the Kohonen network to be categorized into one of a few coarse-grained categories and then to the appropriate back-propagation network for more fine-grained categorization. They report accuracy rates ranging from 74% to 97% for a database of 800 street names, and they are continuing to explore larger databases.

The neural net spelling corrector implemented by Deffner et al. [1990a], which is currently being tested on a 5,000-word lexicon, is part of a natural language interface to a database system. During retrieval, certain features (such as syntactic and semantic features) may be fixed by the context of the database query. Lexicon entries and misspellings are represented by feature vectors containing letter *n*-grams, phonetic features, syntactic features (e.g., **is_adjective**), and semantic features (e.g., **is_color**). A Hamming distance metric is used to compute a measure of similarity between a potentially misspelled or incomplete input string and each member of a restricted subset of the lexicon. This technique appears to be a promising step toward incorporating some context

Table 1.

Accuracy of Some Isolated-Word Spelling Correction Techniques on Kukich's 170-Word Test Set

Technique	521-Word Lexicon	1142-Word Lexicon	1872-Word Lexicon
Minimum Edit Distance			
- *grope*	64%	62%	60%
Similarity Key			
- Bocast *Token Reconstruction*	80%	78%	75%
Simple *N*-gram Vector Distance			
- Dot Product	58%	54%	52%
- Hamming Distance	69%	68%	67%
- Cosine Distance	76%	75%	74%
SVD *N*-gram Vector Distance			
- Cosine Distance	81%	76%	74%
Probabilistic			
- Kernighan-Church-Gale Error Probs		78%	
Neural Net			
- Back-propagation Classifier	75%	75%	

dependency into a corrector for domain-specific applications.

2.2.7 *Summary of Isolated-Word Error Correction Research*

It should be clear from the preceding sections that the ideal isolated-word error corrector does not yet exist. Such a system would have broad lexical coverage (> 100,000 words) and be capable of correcting both single-error and multi-error misspellings in short, medium, and long words in real time with near-perfect first-choice accuracy. Real-time text-to-speech synthesis is one example of such a demanding application. It appears that performance levels approaching 90% correction accuracy for isolated-word text recognition and 80% correction accuracy for isolated-word spelling correction represent the state of the art for practical applications.

In most cases it is impossible to make direct comparisons of techniques. In only a few cases were techniques tested on the same test set and lexicon. Furthermore, techniques vary greatly in the kinds of errors they were designed to correct (e.g., OCR-generated vs. human-generated, typos vs. phonetic errors, single vs. multiple errors, errors in long vs. short words, etc.). Hence, the characteristics of the errors in a test set as well as the size and content of a test dictionary all have great impact on the correction accuracy of a technique.

One small set of direct comparisons is available based on Kukich's test set of 170 human-generated misspellings. This test set may be considered difficult but realistic in that it contains relatively large portions of multi-error misspellings (25%) and short words (63%). The results, shown in Table 1, provide at least a partial ordering of the accuracy of some isolated-word spelling correction techniques.

It may be surprising to some that most of the other techniques outperform the minimum edit distance technique. The increased accuracy of the *n*-gram vector distance techniques, especially the cosine distance metric, may be due to the fact that the bigram representation it uses captures more useful information than the unigram representation used in the minimum edit distance technique. Bocast's "token reconstruction," which makes use of bidirectional letter sequence information, also apparently captures some useful information. It is interesting that the neural net technique attained nearly the same performance level as the best *n*-gram vector distance technique despite being trained on artificially generated errors. At the same time, it is notable that computational demands make standard neural net techniques impractical given existing hardware. It is even more interesting that Kernighan et al.'s [1990] error probabilities enhanced the performance of the cosine distance metric despite the fact that those probabilities were estimated from a general source (AP news wire data) while the test set and lexicon came from a specialized domain.

It is not clear whether these techniques approach some theoretical upper bound on isolated-word correction accuracy. It may not be coincidental that human performance on the same test set was nearly identical to the best automatic techniques. Seven human subjects turned in scores ranging from 65% to 83% and averaging 74%. To see why even humans have trouble achieving 100% accuracy levels, consider the problem of guessing the intended word from among the closest candidates for just a few of the misspellings in that test set:

vver → over, ever, very?
ater → after, later, ate, alter?
wekk → week, well, weak?
gharge → charge, garage, garbage?
throught → through, thought, thoughout?
thre → there, three, threw, the?
oer → per, or, over, her, ore?

Given isolated misspelled strings such as these, it is difficult to rank candidate corrections based on orthographic similarity alone.

Nevertheless, it does seem likely that isolated-word spelling correction accuracy levels could be pushed a little higher with good sources of probabilistic information. OCR error correction research has shown that although transition and confusion probabilities provide only mediocre correction capability when used alone, they achieve excellent performance levels when coupled with dictionary lookup techniques. OCR researchers have been able to estimate reliable confusion probabilities for OCR devices by feeding text to those devices and tabulating error frequencies, but confusion probabilities for human-generated errors are harder to come by. Indeed, those compiled by Kernighan et al. [1990] remain the only such source. Given that the KCG error probabilities were able to improve the performance of other techniques despite their generality, it seems reasonable to infer that error probabilities that are specifically tuned to their human sources and lexical domains would achieve even better accuracy. Annotated machine-readable corpora of human-generated errors from which error probabilities may be estimated are sorely needed. A starting collection containing mainly British-English misspellings is available from Mitton [1985].

Error probabilities might be integrated into *n*-gram vector distance techniques by storing their values directly into word vectors, thus creating detailed word models that explicitly represent the specific probabilities of typing any *n*-gram in any position of a correct word. Alternatively, once neural net learning chips become practical, the same probabilistic knowledge could not only be learned for each individual, but it could also be continuously adapted.

Further research is still needed to address the problem of word boundary infractions as well as the problem of neologisms, i.e., novel terms that arise from creative morphology and from the influx of new proper nouns and other words reflecting a changing environment. As

Sinha and Prasada [1988] have aptly pointed out, no dictionary will ever be complete for this reason. Their technique, which relies on the use of transition probabilities when it encounters such terms, seems apropos and warrants further investigation.

Other techniques worth investigating are those that, like Tenczar and Golden's [1972], are modeled after human psychological word recognition processes. Such techniques would attempt to capture gestalt features of words, such as number of characters extending above and below the horizontal base line, etc. Hull [1987] and Ho et al. [1991] have already begun some work along these lines. There is a wealth of experimental-psychology research to be exploited. Rumelhart and McClelland [1982], McClelland and Rumelhart [1981], and Johnston and McClelland [1980] might serve as starting points for exploring this area.

It is interesting to note that while all techniques fall short of perfect correction accuracy when only the first guess is reported, most researchers report accuracy levels above 90% when the first three guesses are considered. Given a small candidate set that contains the correct word over 90% of the time, it is possible to consider exploiting some contextual information to choose from among the candidates. Thus, the potential exists for current isolated-word error correction techniques to be extended by context-dependent knowledge. Some preliminary experiments along this line, along with other context-dependent spelling correction efforts, are discussed next.

3. CONTEXT-DEPENDENT WORD CORRECTION RESEARCH

Regardless of the progress that is made on isolated-word error correction, there will always remain a residual class of errors that is beyond the capacity of those techniques to handle. This is the class of real-word errors in which one correctly spelled word is substituted for another. Some of these errors result from simple typos (e.g., *from* → *form*, *form* → *farm*) or cognitive or phonetic lapses (e.g., *there* → *their*, *ingenious* → *ingenuous*); some are syntactic or grammatical mistakes, including the use of the wrong inflected form (e.g., *arrives* → *arrive*, *was* → *were*) or the wrong function word (e.g., *for* → *of*, *his* → *her*); others are semantic anomalies (e.g., *in five minuets*, *lave a message*); and still others are due to insertions or deletions of whole words (e.g., *the system has been operating system for almost three years*, *at absolutely extra cost*) or improper spacing, including both splits and run-ons (e.g., *myself* → *my self*, *ad here* → *adhere*). These errors all seem to require information from the surrounding context for both detection and correction. Contextual information would be helpful also for improving correction accuracy for detectable nonword errors.

Devising context-dependent word correction tools has remained an elusive goal. This is due mainly to the seeming intractability of the problem, which appears to call for full-blown natural-language-processing (NLP) capabilities, including robust natural language parsing, semantic understanding, pragmatic modeling, and discourse structure modeling. Up to now, successful NLP systems have been confined to highly restricted domains of discourse, and although a few of these systems have addressed the need to handle ill-formed input, none were designed for general use. Recently, however, progress toward robust syntactic parsing has led to the development of at least two general writing aid tools capable of detecting and correcting errors due to some syntactic-constraint violations. At the same time, advances in statistical language modeling, together with a steady increase in computational power, have led to some promising experiments in the area of statistically-based error detection and correction.

The history of research related to context-dependent word correction is reviewed in this section. It is divided into four subsections. The first reviews findings on the frequency and classification of real-word errors. The second reviews prototype NLP systems that address the problem of handling ill-formed input, including two syntactic rule-based writing

aid tools that incorporate spelling and grammar checking. The third reviews recent experiments that attempt to exploit statistical language models for context-dependent spelling error detection and correction. The last provides a summary of context-dependent spelling correction work.

3.1 Real-Word Errors: Frequency and Classification

Only a few studies have attempted to estimate the frequency of occurrence of real-word errors. (One obvious reason for the shortage of data on the frequency of occurrence of real-word errors is the lack of automatic tools for detecting such errors.) The significant findings are: Peterson's [1986] estimates suggesting that anywhere from 2% to 16% of all single-error typos might result in real-word errors, Mitton's [1987] analysis of 925 handwritten student essays in which he observed that a full 40% of misspellings were real-word errors, and Wing and Baddeley's [1980] listing of 1,185 handwritten errors in which roughly 30% involved real words.

An extrapolation from Mitton's [1987] statistics to the 40,000-word corpus of typed textual conversations studied by Kukich [1992] dramatizes the need for context-dependent spelling correction. Only 85% of the 2,000+ detectable nonword error types in that corpus were correctable by isolated-word correction techniques (the remaining 15% being word boundary infractions that are not correctable by current techniques). The best isolated-word correction techniques corrected approximately 78% of that 85%, for an overall nonword correction rate of 66%. But if the nonword errors studied represent only 60% of all errors in the corpus (the remaining 40% being real-word errors), then only 66% of 60%, or 40%, of all the errors in the corpus are ultimately being corrected. Clearly, the greatest gain is to be found in the detection and correction of errors in the 40% block of real-word errors.

Because real-word spelling and grammatical errors pose practical problems for natural language interfaces, a few studies have been undertaken to determine the expected error rate in such environments. Thompson [1980] analyzed 1,615 database input queries. She found that 446 queries, or almost 28%, contained errors. Of those 446, 61 were nonword spelling errors; 161 were errors due to incomplete lexicon coverage; and 72 were punctuation errors. The remaining real-word errors included 62 grammatical errors and comprised 34% of all errors.

Eastman and McLean [1981] studied 693 natural language queries to a database system. They found that over 12% of them contained co-occurrence violations such as subject-verb disagreement, pronoun-noun disagreement, incorrect verb form, uninflected plural, etc. Another 12% exhibited missing words, and 2% contained extraneous words. Overall, more than 25% of the queries contained real-word grammatical errors. In a more recent study, Young et al. [1991] again found real-word spelling and grammar error rates in excess of 25% in a sample of 426 natural language queries to a document retrieval system. Thus, real-word error rates ranging from 25% to 40% of all errors have been empirically documented.

One approach to handling real-word errors is to view them as violations of natural language processing constraints and to apply NLP tools to the task of detecting and correcting them. NLP researchers traditionally identify at least five levels of processing constraints: (1) a lexical level, (2) a syntactic level, (3) a semantic level, (4) a discourse structure level, and (5) a pragmatic level. Nonword errors would be classified as lexical errors since they violate constraints on the formation of valid words. Errors due to lack of subject-verb number agreement would be classified as syntactic errors because they violate syntactic constraints, as would other errors that result in the substitution of a word whose syntactic category does not fit its context, e.g., *The study was conducted* **be** *XYZ Marketing Research*. Errors that do not necessarily violate syntactic constraints but do result in semantic anomalies, e.g.,

see you in five **minuets**, would be classified as semantic errors. Errors that violate the inherent coherence relations in a text, such as an enumeration violation, would be classified as discourse structure errors, e.g., *Iris flowers have* **four** *parts*: *standards*, *petals*, and *falls*. And errors that reflect some anomaly related to the goals and plans of the discourse participants would be classified as pragmatic errors, e.g., *Has Mary registered for Intro to* **Communication** *Science yet?* (where **Computer** was intended).

It would be helpful to know the relative proportions of real-word errors that fall into each of the five categories because the five levels of natural-language-processing constraints represent increasingly more difficult problems for NLP research. Lexical errors are amenable to isolated-word spelling correction techniques; tools are just beginning to emerge for handling syntactic errors in unrestricted texts; but only restricted-domain prototype systems exist for handling semantic, discourse structure, and pragmatic errors. Ramshaw [1989] astutely points out that the same error may be classified differently with respect to detectability and correctability. He cites the following example from a student advisor expert system, *Is there room in* **CIM** 360?, as a case in which lexical constraints may be sufficient to detect the error (since **CIM** is not a valid course code), but pragmatic knowledge about the student's goals and past course history may be needed to determine whether **CIS** (Computer and Information Science) or **COM** (Communication Sciences) was intended.

It would also be helpful to know the relative proportions of real-word errors that depend on local vs. long-distance relations between words. This is because an alternative approach to handling real-word errors is to treat them as irregularities or improbabilities within the framework to statistical language modes. Statistical language models based on word bigram probabilities [Gale and Church 1990; Church and Gale 1991b], word trigram probabilities [Bahl et al. 1983; Jelinek et al. 1991], word n-gram probabilities [Bahl et al. 1989; Brown et al. 1990a], syntactic part-of-speech bigram and trigram probabilities [Derouault and Merialdo 1984a; Garside et al. 1987; Church 1988], and collocation[6] probabilities [Choueka 1988; Smadja and McKeown 1990; Smadja 1991a] have been developed already for other speech- and text-processing applications. Statistical techniques based on such models could be used to detect improbable word sequences and to suggest probable corrections. Most of these models are based on local lexical or syntactic relations, i.e., relations among words that occur within a few positions of each other, so error detection and correction techniques based on these models would be limited to handling local errors. Since it is still not clear what proportion of real-word errors fall within this realm, the error coverage potential of statistically based models is yet to be determined.

One small but insightful study was done by Atwell and Elliott [1987] in conjunction with the University of Lancaster Unit for Computer Research on the English Language (UCREL) probabilistic parsing project [Garside et al. 1987]. These researchers analyzed a small sample of errors to determine the relative proportion of errors in each of the following four categories: (1) nonword errors; (2) real-word errors in locally invalid syntactic contexts; (3) real-word errors in globally invalid syntactic contexts; and (4) syntactically valid real-word errors in invalid semantic contexts. Their sample consisted of 50 errors each from three different sources: (a) published, manually proofread texts, (b) essays by 11- and 12-year-old students, and (c) English written by non-native English speakers. The results of their analysis, shown in

[6] Collocations are word pairs (or triples or quadruples, etc.) that exhibit statistically significant affinities for occurring somewhere in each other's nearby, though not necessarily contiguous, lexical environment, e.g., *salt* and *pepper*, *apple*, and *pie*, *apple* and *big*, *state* and *union*, etc

Table 2.

Distribution of Spelling Errors (*from Atwell & Elliott*)

	Total # of Errors	% Non-word Errors	% Local Syntactic Errors	% Global Syntactic Errors	% Semantic Errors
Text source					
Published Texts	50	52%	28%	8%	12%
Student Essays	50	36%	38%	16%	10%
Non-Native Text	50	4%	48%	12%	36%

Table 2, indicate widely different distributions of errors across text genres, but they do indicate that a significant portion of errors may be detectable as local syntactic violations. The prototype system Atwell and Elliott implemented for detecting such errors is described in Section 3.3.1

The following few sentences, which were randomly culled from existing texts, contain genuine real-word errors that illustrate some of the problems and potential for various NLP and statistically based detection and correction techniques. Most of the errors in these sentences are readily detectable by humans, though errors of this type are frequently overlooked because inherent, human, error correction tendencies are so strong.[7]

Local Syntactic Errors:

(1) The study was conducted mainly *be* John Black.

(2) The design *an* construction of the system will take more than a year.

(3) Hopefully, all *with* continue smoothly in my absence.

(4) Then they *an* going to go to his house.

(5) I need to *notified* the bank of [this problem].

(6) He *need* to go there right *no w*.

Global Syntactic Errors:

(7) Not only we in academia but the scientific community as a whole *needs* to guard against this.

(8) Won't they *heave if* next Monday at that time?

(9) I can't pick him *cup* cuz he might be working late.

(10) This thesis is supported by the fact that since 1989 the system has been operating *system* with all four units on-line, but...

Semantic Errors:

(11) He is trying to *fine* out.

(12) They are leaving in about fifteen *minuets* to go to her house.

(13) Can they *lave* him my message?

(14) Well, I'll call again when they come *hoe*.

Examples (1) through (10) all contain syntactic violations. In theory, they should be detectable by a sufficiently robust automatic natural language parser. The syntactic violations in (1) through (6) involve local (within 1 or 2 words) incon-

[7] See Cohen [1980] for a review of some experiments on human ability to detect nonword, real-word, homophonic, and nonhomophonic spelling errors in context. Experimental results indicate that, not surprisingly, nonword errors are easier to detect than real-word errors, and that, perhaps surprisingly, real-word homophones are easier to detect than real-word nonhomophones. The author suggests three possible explanations for the latter result, a priming explanation, a frequency explanation, and a semantic explanation.

gruities which should make them detectable not only by a parser but also via a statistical language model.

The mistakes in examples (7) through (10) are more problematic. The subject-verb number disagreement in example (7) turns on a long-term dependency between the complex subject phrase and the verb. This mistake requires a parser capable of building the full syntactic structure of the sentence. Even a parser with that capability might have trouble locating the double error in (8) which precipitates a garden path effect.[8] Although the problem in (9) might be detectable by local syntactic-constraint violations, it would be difficult to select between two potential corrections, the phrases *pick him up* and *pick his cup*, on syntactic grounds alone. Furthermore, the same mistake would not have been detectable even via syntactic analysis if the sentence had read *pick her cup*. Deletions and insertions, such as the extra word, *system*, in example (10) might be problematic for both locally based and globally based syntactic analyzers.

The mistakes in examples (11) through (14) would all be undetectable via syntactic analysis alone since they all have valid parses. However, they are readily detectable by humans for their semantic incongruities and/or lexical improbabilities. Unfortunately, there are no general-purpose NLP systems capable of detecting semantic incongruities that might be used for this purpose. There are statistical language models, however, that might hold some promise for detecting not only semantic errors but also local syntactic errors. Intuition suggests that many of the above errors result in low-frequency bigrams, i.e., *an construction*, *with continue*, and *they an* in (2), (3), and (4). However, some seem to require at least trigrams to be detected, e.g., *conducted mainly be*, *been operating system*, and *to fine out*, in (1), (10), and (11). At least one, (7), is resilient to both bigram and trigram improbability detection.

No examples of discourse structure errors or pragmatic errors are included above because none were readily found in a few days' recording of actual errors. Whether this reflects a genuine sparseness of such errors is unknown.

One other characteristic of real-word errors for which little hard data exists is their apparent tendency to involve orthographic intrusion errors, i.e., substitutions of graphemes from nearby words, as in the following examples:

(15) Is th*ere* a w*ere* to display your plots on a DEC workstation running X? (*way* → *were*)

(16) ... and they would *b*ust *b*ill us 1/12 of that each month. (*just* → *bust*)

(17) *As* noted above, such *as* system would give them a monopoly. (*a* → *as*)

(18) I have access to those people through *b*y *b*oss. (*my* → *by*)

(19) We will not meet again until... we get *out* first *out*side speaker. (*our* → *out*)

Although phonemic (aural) lexical intrusions, or slips of the tongue, are well documented in cognitive studies of language production [Fromkin 1980; Garrett 1982], orthographic (visual) lexical intrusions, or slips of the pen, are less well documented [Ellis 1979; 1982; Hotopf 1980]. Studies of these phenomena might certainly be exploited in the context of word correction.

In summary, there is some evidence that real-word errors are ubiquitous, accounting for as much as 40% or more of all textual errors, but more research is needed to determine their true frequency. More research is also needed to characterize or classify such errors. Knowledge of the relative proportions of such errors that violate each of the five natural language processing level constraints, as well as the proportions that entail local vs. global lexical or syntactic violations and the proportions that involve orthographic intrusions, would be

[8] A *garden path effect* occurs when a processor is led down an incorrect search or processing path from which it has difficulty recovering.

helpful to determine which techniques might be most useful in detecting and correcting them. Emerging NLP tools for syntactic processing have potential for detecting and correcting some grammatical errors, while statistical language-modeling tools have potential for handling other certain errors that result in local lexical irregularities. These techniques are described in more detail in the following sections.

3.2 NLP Prototypes for Handling Ill-Formed Input

Natural language researchers were quick to recognize the need for handling ill-formed input as they observed their earliest NLP systems floundering with the naturally erroneous input proffered by their first real users. They soon began incorporating error-handling techniques into prototype NLP systems. Most of the prototypes were designed for specific tasks within restricted domains of discourse, so their error-handling techniques do not scale up well for use on unrestricted text. Nevertheless, this work helps to elucidate some to the issues surrounding the problem of context-dependent spelling and grammar correction, so an overview is included here. Progress has been made toward processing unrestricted text on the syntactic level, so two syntactic rule-based NLP systems for context-dependent spelling and grammar correction are more fully described at the end of this section.

Most NLP systems are parser driven. That is, they view their central task as one of deriving the syntactic structure of the input text (or speech). Sometimes they go on to do further semantic discourse structure and pragmatic processing to interpret and act on the input. A typical NLP system consists of a lexicon, a grammar, and a parsing procedure. The lexicon is the set of terms that are relevant to the application domain. The terms are usually annotated with their potential parts of speech as well as morphological information. The grammar is a set of rules that specifies how words, parts of speech, and higher-order syntactic structures (e.g., noun phrases, verb phrases, etc.) may be validly organized into well-formed sentence fragments and sentences. The parsing procedure often consists of looking up each word in the dictionary to determine its potential parts of speech and applying grammar rules to build up higher-order syntactic structures. Since many words have more than one possible part of speech, multiple partial parses often must be built in parallel until later constraints resolve the syntactic ambiguity of the terms.

Not all NLP systems retain clear-cut divisions of lexical, syntactic, and semantic knowledge. Quite a few systems employ some form of a "semantic grammar" in which semantic restrictions on how words can be meaningfully juxtaposed are incorporated directly into the grammar. A few systems incorporate knowledge of syntactic structures and semantic restrictions into their lexicons. In either case, the availability of knowledge of semantic constraints at an earlier stage of the parsing process helps narrow the space of potential parses that must be explored. Those NLP systems that make use of discourse structure and/or pragmatic knowledge usually keep that knowledge in separate data structures that represent models of the organization of the text, the user's plans and goals, and/or the system's knowledge of the domain. The models may be consulted and modified during the parsing process, and the constraints they impose help to reduce further the search space of potential parses.

Under this paradigm, the detection and correction of errors is nontrivial for at least two reasons: (1) lexical and grammatical coverage is necessarily incomplete; and (2) locating the source of a parsing failure is often difficult. For example, a failure to find a string in the system's lexicon could indicate that a spelling error has occurred, but it could also indicate that a novel term has been encountered. Because the occurrence of novel terms (e.g., proper nouns) is so high, most parsers assume that unknown

terms are not mistakes and assign a default syntactic category (usually noun) to them and then attempt to proceed with the parse. If a parsing failure occurs further along in processing, often it is hard to tell whether it is due to the fact that the input violates some rule(s) of the grammar, or the fact that grammar itself is incomplete, or the fact that an input string has been miscategorized. Furthermore, it is frequently hard to pinpoint the precise rule or word that precipitated the failure let alone suggest a correction. Consider the previous example: *Won't they* **heave if** *next Monday at that time*? Even a robust parser might not recognize that a problem exists until the ending punctuation is encountered, at which time it would be hard to guess that the terms **heave if** are the culprits and that they should be replaced with the terms **have it**.

Various approaches to dealing with ill-formed natural language input have been tried. They can be loosely classified into three general categories: (1) **acceptance-based approaches**; (2) **relaxation-based approaches;** and (3) **expectation-based approaches.**

3.2.1 Acceptance-Based Techniques

Acceptance-based approaches are grounded in the philosophical observation that despite the fact that ordinary language is replete with errors, people nevertheless have little trouble interpreting it. So acceptance-based techniques assume that errors can simply be ignored as long as some interpretation can be found that is meaningful to the given application. Acceptance-based approaches tend to make heavy use of semantic information and little use of any other level of linguistic information.

Many of the earliest NLP systems, for example, those devised by Waltz [1978] and Schank [1980] and their students, focused on semantic understanding, and so were able to adopt acceptance-based approaches for handling ill-formed input. Under these approaches ill-formed input, such as lack of number agreement between a subject and a verb, was treated on an equal basis with well-formed input, either by ignoring or eliminating from the grammar the syntactic or other grammatical constraints that the input violated or by incorporating rules for processing certain common ill-formed constructions directly into the grammar. These approaches proved successful for their particular applications, in part because erroneous input is frequently semantically unambiguous within a restricted domain. But they do not generalize readily to the task of detecting and correcting real-word errors in unrestricted text for at least two reasons. One is that they often accommodate ill-formed input without even detecting a problem. The other is that they require a semantic model of the domain of discourse. No full-blown semantic model is available for unrestricted text.

A variant of an acceptance-based approach that has some intuitive appeal is the preference semantics approach advocated by Fass and Wilks [1983]. It is intended to address the fact that a large portion of all natural language utterances are either ill-formed or metaphorical to some degree, for example, the statement *My car drinks gasoline*. They assume that a formal semantic-representation scheme and an encoding procedure are available that enable every input string to be encoded in such a way that highlights any semantic conflicts or semantic-preference violations. Further processing would then determine the degrees to which either certain terms in the input would have to be changed or the system's semantic model would have to be changed in order to minimize the conflicts. The system would settle on the interpretation resulting in the least change. While this approach has much psychological plausibility, at present it is impractical for processing unrestricted text, again because the rich formal semantic knowledge base it requires is unavailable.

3.2.2 Relaxation-Based Techniques

Relaxation-based approaches are at the opposite end of the error spectrum—they

assume that no errors can be ignored. This philosophical bias arose from the fact that many early NLP systems relied almost exclusively on syntactic rules, and as such they simply broke down when such rules were violated. In relaxation-based techniques, when a parsing failure occurs, the system attempts to locate an error by identifying a rule that might have been violated and by determining whether its relaxation might lead to a successful parse. Locating and relaxing rules allow for both detection and correction of errors.

Relaxation-based techniques, which focus mainly on syntactic processing, are the least knowledge intensive of the three approaches and for that reason the most well suited for use on unrestricted text. In fact, two relaxation-based text-editing systems, the EPISTLE/CRITIQUE system originally designed for editing business correspondence [Heidorn et al. 1982; Richardson and Braden-Harder 1988] and a similar text editor for the Dutch language [Kempen and Vosse 1990] have been tested on unrestricted text. Those systems and their test results are described in more detail later. For now we will simply note that they both employ rule-based parsers to detect and correct certain errors that result in syntactic-rule violations. Neither system makes use of semantic, pragmatic, or discourse structure knowledge, so neither system attempts to detect or correct errors in those categories. The EPISTLE/CRITIQUE system preprocesses input to detect and correct nonword spelling errors before beginning its parsing procedure; certain real-word errors, including commonly confused terms such as *who's* and *whose*, as well as a variety of grammatical errors, such as subject-verb number agreement violations and punctuation problems, are detected during the parsing process, and corrections are suggested. The Dutch system preprocesses each input string to generate a rank-ordered cohort set, i.e., a relatively small set of orthographically and phonetically similar words. Presumably, the cohort sets for both nonword and real-word errors will contain the correct word. The parsing procedure is then applied to the original input string, and, when a failure occurs, a rule is relaxed and retried using a candidate term from the cohort set.

Similar relaxation-based techniques have been implemented or proposed by Trawick [1983], Weischedel and Sondheimer [1983], Suri [1991], and Suri and McCoy [1991]. In his REL system, Trawick used syntactic-rule relaxation, plus ordering strategies for trying easy corrections first, plus cohort set generation as a last resort. Weischedel and Sondheimer advocate adopting a principled approach to rule relaxation. They have defined a set of diagnostic metarules to guide the search for potential rules to relax when parsing failures occur. Their proposed metarules detect problems in the following general categories: omitted articles, homonyms, spelling/typographical errors, violated selection restrictions, personification, metonymy, and other violated grammatical tests such as agreement failures. Suri and McCoy have recently begun work on a system for correcting English written by American Sign Language (ASL) users. Since the grammar and syntax of ASL differ significantly from standard English, text written by ASL users often contains errors with respect to standard English that fall into certain patterns. Suri has developed a taxonomy of such errors by analyzing a corpus of writing samples from ASL users. The error taxonomy will guide the development of an error-correcting language-learning tool.

3.2.3 Expectation-Based Techniques

Expectation-based approaches fall somewhere in the middle of the error spectrum—they acknowledge both that errors occur and that people draw on contextual knowledge to correct them. In expectation-based techniques, as the parsing procedure progresses the system builds a list of words it expects to see in the next position based on its syntactic, semantic, and sometimes pragmatic and discourse structure knowledge. If the next term in the input string is not in the list of expected terms, the system as-

sumes an error has been detected and attempts to correct it by choosing one of the words on its expectation list.

Expectation-based techniques have been devised to exploit various levels of linguistic knowledge. One system that exploits both syntactic and semantic expectations for error correction is the case-based parser referred to as CASPAR, DYPAR, and MULTIPAR [Carbonell and Hayes 1983], listed here in successive generations. This parser derives much of its expectation-forming ability from the fact that it operates under the case frame paradigm. Case frames are slot-filler data structures for representing predicates (usually realized as verbs) and their arguments. For example, a *GIVE* predicate would have a case frame with slots for three arguments, a *FROM*-slot, a *TO*-slot, and an *OBJECT*-slot.

Under the case frame paradigm, both the syntactic and semantic structures of a given domain of discourse are modeled in recursive case frames which are incorporated into a grammar and lexicon. As each word in an input string is parsed, it either invokes a new case frame or fills a slot of an existing one. Existing case frames and their unfilled slots set up expectations for upcoming terms. Thus, an expectation list of potential next terms can be generated for each word in an input string as the parse progresses. The size of the expectation list will vary from the whole dictionary, when no case frame has yet been selected, to small sets of potential terms when a partially filled case frame is being processed, to a unique word when a function word is expected. These "reduced-dictionary" expectation lists can be exploited when both nonword and real-word spelling errors are encountered in the input. Case frames also provide significant help in recognizing missing word errors and in parsing valid instances of ellipsis and anaphoric references.

The most recent version of this parser, MULTIPAR [Minton et al. 1985], pursues multiple parses in parallel. It exploits both expectation-based and relaxation-based error recovery techniques. When the parsing procedure reaches a roadblock, it invokes as many recovery strategies as it needs to achieve at least one valid parse, including nonword and real-word spelling correction, missing-word insertion, and others. The order in which recovery strategies are invoked is governed by a control strategy to ensure that strategies leading to less-deviant errors are tried first.

Another expectation-based parser, the NOMAD system [Granger 1983], makes similar use of syntactic and semantic expectations to correct nonword and real-word errors, to constrain possible word senses, and even to guess the meanings of unknown words. For example, NOMAD is able to guess that the unknown term *scudded* in the input *Enemy scudded bombs at us* is probably a verb whose action is in the *PROPEL* class.

Expectation-based techniques have also been devised to exploit pragmatic and discourse structure knowledge. A system devised by Ramshaw [1989] exploits pragmatic knowledge to detect and correct real-word errors in user input to an expert system. The expert system has a rich pragmatic model "based in part on a library of domain plans for how agents can deal with situations of the given type" (p. 23). Input errors are detected when the system is unable to construct a complete interpretation of an input that is compatible with one of the domain plans. Ramshaw's system uses a wildcarding technique to successively replace each word in the input with an open slot and then combine the syntactic and semantic constraints from the partially interpreted input with the predictions from the pragmatic model to yield a ranked set of corrections. Two other systems, one by McCoy [1989] and one by Carberry [1984] prescribe methods for generating responses to pragmatically ill-formed input. Both rely on heuristics for building a model of the user's plans and goals based on information that can be inferred from the preceding dialogue.

Expectations that can be derived from the structure of the dialogue have also been used to correct errors and resolve ambiguities introduced by speech recog-

nizers. Speech-to-text systems are faced with a compound problem: not only are current voice recognition systems subject to a high rate of error in classifying speech signals into phonemes, but many unique phoneme sequences yield lexically ambiguous homophones (e.g., *hear*, *here*). Fink and Biermann [1986] describe a connected-speech-understanding system capable of executing commands spoken by a user. Their system learns dialogue scripts as it is being used and then exploits the dialogue expectations it has learned to resolve subsequent ill-formed and ambiguous input.

In summary, expectation-based techniques appeal to the intuition that contextual linguistic knowledge at all levels, lexical, syntactic, semantic, discourse structure, and pragmatic, help to constrain the list of possible word choices when natural ambiguities and errors occur in everyday language. Unfortunately, general computational processing techniques exist for only two of those levels, the lexical and syntactic levels. Techniques for representing and processing semantic, pragmatic, and discourse structure knowledge at sufficiently rich levels of detail are currently impractical beyond limited domains of discourse. So, like acceptance-based techniques, expectation-based techniques are not practical yet for attacking the problems of context-dependent spelling and grammatical-error correction for unrestricted text. However, the less knowledge-intensive relaxation-based techniques, which concentrate almost exclusively on syntax, have made some progress in this area, and the extent of their success is described next.

3.2.4 Parser-Based Writing Aid Tools

One obvious application for context-dependent spelling correction is in writing aid and text critiquing systems. The Unix-based *Writer's Workbench* system [Cherry and Macdonald 1983] is perhaps the earliest such system. It consists of a set of automatic tools that include, among others, the *spell* isolated-word spelling corrector, a *grep*-based pattern-matching tool for detecting some 800 commonly misspelled words (including some real-word misspellings), a word usage tool that describes the proper usage of some 300 words and phrases, a tool for detecting common instances of poor writing style, such as the use of sexist language and stilted phrases (e.g., *more preferable*, *collaborate together*, *with the exception of*), and the *style* program, which computes a number of statistics about a document including a readability score, sentence lengths, percentage of words in each part-of-speech category, etc. *Style* makes use of a program called *parts* that guesses the parts of speech of the words in a document. Its authors claim that it achieves 95% accuracy despite the fact that it uses only a small dictionary of function words, irregular verbs, and suffixes to classify some words and then looks for relations among those words to classify the rest.

Perhaps the first system to attempt to bring robust natural language parsing to bear on the problems of detecting and correcting real-word spelling and grammar errors was the IBM EPISTLE system [Heidorn et al. 1982], later known as CRITIQUE [Richardson and Braden-Harder 1988]. EPISTLE was originally designed to diagnose five classes of grammatical errors:

(1) subject-verb disagreement: (e.g., *Mr. Jones, as well as his assistance,* **are** *entitled . . .*);
(2) wrong pronoun case: (e.g., *If I were* **him** . . .);
(3) noun-modifier disagreement: (e.g., *These* **report** *must be checked . . .*);
(4) nonstandard verb forms: (e.g., *The completed manuscript was* **wrote** *by . . .*); and
(5) nonparallel structures: (e.g., *We will accept the funds, send receipts to the payers, and* **crediting** *their accounts . . .*).

Note that EPISTLE's robust parsing approach has the advantage of being able to detect even problems that turn on long-term syntactic dependencies.

The EPISTLE parser and grammar were designed and tested reiteratively around a corpus of 411 business letters consisting of 2,254 sentences whose average length was about 19 words. EPISTLE made use of an on-line dictionary of over 100,000 words. By 1982, EPISTLE's core grammar of augmented phrase structure rules consisted of about 300 rules. Its rule-based parser attempted to arrive at a unique parse for each sentence by invoking first its core grammar rules. When multiple parses were found a general metric for ranking parses [Heidorn 1982] was invoked to select one. When the core grammar failed to produce a complete parse, another technique, called fitted parsing [Jensen et al. 1983], was invoked. The fitted-parsing technique consisted of a set of heuristics for choosing a head constituent[9] and attaching remaining constituents to form a complete sentence. In 1981, 64% of the sentences in the test corpus could be parsed using EPISTLE's core grammar alone. By mid-1982, when the fitted-parsing technique was introduced, the whole corpus could be successfully parsed, 27% via fitted parsing.

Processing of text in EPISTLE began with a dictionary lookup phase that assigned parts-of-speech to input words, and in the process detected nonword spelling errors and certain awkward phrases. Unrecognized words were assigned a default part of speech, usually noun. Next came the parsing phase, during which the detection and correction of real-word grammatical errors occurred. The parsing process might require multiple passes; if a successful parse was not obtained on the first pass, it was retried with selected grammatical constraints being relaxed. The result of the parsing phase was a parse tree that highlighted any relaxed constraints and suggested corrections. A final EPISTLE processing phase diagnosed stylistic problems such as sentences that were too long.

EPISTLE's authors acknowledged that its five original error classes "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" (p. 319) [Heidorn et al. 1982]. Over the years, EPISTLE's grammar and error coverage were extended so that by 1985, its revised version, named CRITIQUE [Richardson and Braden-Harder 1988], covered over 100 grammar and style errors and had been tested on four text genres: business correspondence, freshman student essays, ESL (English as a Second Language), and professional writing. CRITIQUE's output on 10 randomly selected samples from each of these genres was analyzed, and its advice was classified into three categories: **correct, useful**, or **wrong**. The **useful** category designated cases in which CRITIQUE detected a problem correctly but was not exactly correct in its diagnosis. Results of the analysis are shown in Table 3.

Perhaps the lower performance level for professional text might be due in part to a lack of coverage of the lexicon and grammatical structure of the professional domain. Unfortunately, we are not told how many errors CRITIQUE overlooked. The authors indicate they believe CRITIQUE to be "most helpful on straightforward texts before they are significantly revised" and that "In general, CRITIQUE also appears to be more accurate on texts with a shorter average sentence length" [Richardson and Braden-Harder 1988, pp. 201–202].

A parser-based text-editing system for the Dutch language has produced some encouraging results [Kempen and Vosse 1990]. Although its authors are careful to point out that the system was not designed to be a full-scale grammar and style checker, it does appear to function well as a general-purpose tool for detecting and correcting both nonword and real-word syntactic errors. It incorporates the triphone-based isolated-word

[9] Roughly speaking, a constituent is a portion of the syntactic tree structure of a sentence that comprises a complete phrasal unit, such as a noun phrase, verb phrase, or prepositional phrase.

Table 3.

CRITIQUE's Error Correction Ability

Genre	Average Sentence Length	Correct Advice	Correct + Useful Advice
Business Corres	18 words	73%	82%
Student Essays	16 words	72%	86%
ESL Text	21 words	54%	87%
Professional Text	22 words	39%	41%

spelling corrector of Van Berkel and DeSmedt described in Section 2.2.4 and a unification-based parser consisting of some 500 augmented phrase structure rules. It also has access to a 100,000-word dictionary that is cross-indexed to inflected forms, giving it a total lexicon of about 250,000 words.

It processes a document in four stages: (1) preprocessing, (2) word-level analysis, (3) sentence-level analysis, and (4) text resynthesis. The preprocessing stage removes typesetting markup symbols and corrects punctuation. The word-level analysis stage calls upon the Van Berkel and DeSmedt spelling corrector to generate a small set (~ 5) of candidate corrections, including homophones, for each word type in the document. In the sentence-level analysis stage, a unification-based parser is called to produce a high-level parse. Unification rules are relaxed to allow for syntactic violations which are then flagged. When violations are detected, the parser attempts to construct a valid parse using candidates from the correction set. Finally, in the text resynthesis stage the system regenerates the original document with suggested corrections and error diagnostic messages.

Kempen and Vosse [1990] evaluated the system on a standard 150-sentence spelling correction exam for typists that contained 75 correct sentences and 75 incorrect sentences; 59 of the incorrect sentences had nonword errors; 14 had real-word syntactic errors; and 2 had real-word idiomatic errors. The system gave no false alarms on the 75 correct sentences; it detected 56 of the 59 nonword errors[10] and corrected 55 of them; it detected 9 of the 14 syntactic errors and corrected 7 of them; it overlooked the 2 idiomatic errors. All this required under 10 seconds of CPU time on a DECstation 3100. A more recent version of the system, which will soon be available commercially [Vosse 1992], was tested on the same test set. It left only 3 errors undetected, correcting the other 72 and causing no false alarms. This system was also tested on a random sample of 1,000 lines from two texts on employment legislation submitted for publication. The sample contained 6,000 words with 30 spelling errors. The system detected 28 of the errors and accurately corrected 14 of them while producing 18 false alarms.

Recent research has led to significant advances toward robust natural language parsing of unrestricted English text.[11] Given this progress, some useful commercial systems for real-word grammatical-error correction based on robust parsing and syntactic-constraint relaxation techniques should be appearing in the not-too-distant future.

3.3 Statistically Based Error Detection and Correction Experiments

An alternative approach to context-dependent spelling correction is the statistical language-modeling approach. Statistical language models (SLMs) are essentially tables of conditional probability estimates for some or all words in a language that specify a word's likelihood to occur within the context of other words. For example, a word trigram SLM speci-

[10] The nonwords were overlooked because the system's morphological analyzer construed them as legal compounds.

[11] See, for example, Hindle's [1983] Fidditch parser, Abney's [1990] CASS parser, and Sleator's [1992] Link Grammar.

fies the probability distribution for the next word conditioned on the previous two words; a part-of-speech-bigram SLM specifies the probability distribution for the part of speech of the next word conditioned on the part of speech of the previous word; and a collocation SLM specifies the probability distributions for certain words to occur within the vicinity (e.g., within five places in either direction) of another linguistically related word. Some of the various speech and text-processing applications for which SLMs have been compiled include speech recognition [Bahl et al. 1983; Jelinek et al. 1991], information retrieval [Choueka 1988], text generation [Smadja 1990a, 1991b], syntactic tagging and parsing [Garside et al. 1987, Church 1988], machine translation [Brown et al. 1990b], semantic disambiguation [Brown et al. 1991], and a stenotype transcription application [Derouault and Merialdo 1984b].

The probability estimates comprising SLMs are derived from large textual corpora, where "large" is currently defined as tens of millions of words. Only recently have the computational resources become available and the sizes of machine-readable corpora begun to approach the dimensions required to make the compilation and use of such models feasible. Note, for example, that the size of a word trigram probability matrix grows cubically with the size of the lexicon, so that a full-scale word trigram model for a lexicon of only 5,000 words would have 25 trillion entries. Clearly, the majority of those entries should be assigned zero probabilities. But because of Zipf's law [Zipf 1935], which states that the frequency of occurrence of a term is roughly inversely proportional to its rank, there will always be a large tail of terms that occur only once. So even with the availability of corpora as large as 100 million words, hence 100 million trigrams, computing nonzero probabilities is a nontrivial statistical estimation problem. To alleviate this problem, some clever estimation techniques have been devised [Bahl et al. 1983, 1989; Brown et al. 1990a], and some comparative studies of estimation procedures have been carried out [Gale and Church 1990; Church and Gale 1991b]. The latter studies were done in the context of word bigram models, which are somewhat more manageable in size than word trigram models.

Other more manageably sized SLMs have been explored. For example, part-of-speech (POS) bigram and trigram models have been compiled [Garside et al. 1987; Church 1988; Derouault and Merialdo 1984a] based on expanded part-of-speech category sets of approximately 100 categories (for example, **plural-possessive-personal-pronoun**, e.g., *our*, **reflexive-indefinite-pronoun**, e.g., *oneself*, **superlative-degree-adverb**, e.g., *most*, **superlative-general-adverb**, e.g., *best*, etc). Note that a POS bigram model based on 100 categories will have at most 10,000 POS bigram entries for which probabilities must be compiled. Manageably sized collocation-based SLMs have been compiled also [Choueka 1988; Smadja 1991a]. Since the goal of collocation-based models is to locate and estimate probabilities of co-occurrence for only those terms that enter into statistically significant collocation relations, the resulting models contain fewer table entries. The price that is paid for the computational manageability of these smaller models is reduced predictive power: POS n-gram models are able to predict only the part-of-speech of the next word as opposed to the word itself, and collocation models provide predictions for only some terms. Nevertheless, by combining POS n-gram probabilities or collocation probabilities with other lexical statistics, such as unigram word frequency or relative probability that a word will have a particular part-of-speech, these models have been put to productive use in certain applications, including the syntactic tagging and parsing, stenotype transcription, information retrieval, and text generation applications cited above.

The statistical language-modeling approach to context-dependent word correction shares a basic premise with expectation-based NLP approaches: namely, that contextual information can be used to

help set a priori expectations for possible word choices. Thus, low-probability word sequences might be used to detect real-word errors, and high-probability word sequences might be used to rank correction candidates. Word-based SLMs (as opposed to SLMs based on syntactic POS n-grams) might even capture some inherent semantic relations, e.g., the semantic relation between the words *doctor* and *nurse*, without the need for an explicit, handcrafted, formal semantic model. Indeed, all SLMs are compiled automatically. One limitation of both word-based and POS-based bigram and trigram SLMs is that they are confined to representing only local relations (i.e., two or three contiguous terms).

It is not immediately clear how successful various SLMs might be at either detecting or correcting real-word errors. Three recent studies, one using a POS bigram model [Atwell and Elliott 1987], one using a word bigram model [Gale and Church 1990; Church and Gale 1991a], and one using a word trigram model [Mays et al. 1991], were designed specifically to shed some light on these questions. In the first study, researchers attempted to determine how well a POS bigram model could detect real-word errors. In the second study, the ability of a word bigram model to improve the accuracy of a nonword corrector was tested. In the third study, the ability of a word trigram model both to detect and correct real-word errors was tested. The results of all three studies are reviewed here.

3.3.1 Detecting Real-Word Errors via Part-of-Speech Bigram Probabilities

One successful application of statistical-language modeling to a natural language text-processing problem is the syntactic tagging and parsing project undertaken by the University of Lancaster's Unit for Computer Research on the English Language (UCREL) [Garside et al. 1987]. The UCREL prototype system includes a probabilistic word tagger called CLAWS (Constituent Likelihood Automatic Word-tagging System) that makes use of a syntactic-tag set consisting of 133 part-of-speech categories and a part-of-speech bigram SLM. POS bigram transition probabilities were estimated from a tagged corpus of one million words. CLAWS first assigns one or more syntactic tags to each word in a sentence by looking up a word's potential parts-of-speech in a dictionary. Approximately 65% of all words (i.e., tokens) receive only one tag when processed in this manner. Remaining ambiguities are resolved by selecting the tag of highest probability according to POS bigram transition probabilities. This is accomplished by computing the path of maximum probability for a sequence of one or more ambiguously tagged words bounded at either end by unambiguously tagged words. CLAWS ultimately assigns an unambiguous part-of-speech tag to every word with an overall accuracy rate of 96–97%.

Two UCREL researchers, Atwell and Elliott [1987] set out to explore the utility of CLAWS for detecting and correcting real-word errors. Prior to experiments, they studied a small sample of errors to determine what proportion fell into each of the following categories: (1) nonword errors; (2) real-word errors in locally invalid syntactic contexts; (3) real-word errors in globally invalid syntactic contexts; and (4) syntactically valid real-word errors in invalid semantic contexts. They hypothesized that CLAWS might be helpful in detecting and correcting errors in the second and perhaps third categories. The results of their brief study (shown in Table 2 in Section 3.1) indicated that nearly half of the errors they found in published text were real-word errors, and over three-quarters of those were due to either local or global syntactic violations. So they went on to test the ability of CLAWS to detect these types of errors.

By manually examining a corpus of 13,500 words of text taken from four different sources. Atwell and Elliott [1987] and their colleagues identified 502 real-word errors. Of those, 420 errors fell into categories 2 or 3, and 82 fell into category 4. Since category 4 errors would be impossible to detect via syntactic analysis alone, these errors were separated

from the others. Note that category 3 errors, which turn on global syntactic relations, are also unlikely to be detected by POS bigram probabilities, but they were not separated from the others.

Next, Atwell and Elliott [1987] ran the CLAWS tagger to assign syntactic-category tags based on POS bigram transition probabilities to the entire 13,500-word corpus and to record the resulting POS bigram transition probabilities. They hypothesized that a sequence of two low POS bigram probabilities might indicate an occurrence of an error, and they set about determining an optimal threshold setting for those probabilities such that the number of errors detected would be maximized while the number of false alarms would be minimized. The optimal setting they found resulted in a detection rate of 62% and a precision rate of 35%. That is, 260 of the 420 errors were detected, and 753 words in all were flagged as errors. Unfortunately, changing the thresholds slightly only raised the precision rate to 38% while sacrificing the detection rate to the level of 47%.

These results indicate that raw POS bigram transition probability scores alone perform relatively poorly as a diagnostic tool because it is impossible to choose a threshold that ensures the detection of a high number of errors without permitting an unreasonable number of false alarms. However, the original implementation of CLAWS did not assign probabilities to the candidate tags for words (except for marking some tags as extremely rare). So it might be possible to add candidate tag probabilities, as in some other POS n-gram models [Church 1988; Derouault and Merialdo 1984a], thereby improving the error-detecting capabilities of CLAWS. Nevertheless, there is an inherent limitation in the correction power of POS bigram probabilities. That is, if multiple candidates share the same preferred POS tag, the tag loses its discrimination power.

Atwell and Elliott [1987] considered their results to be reasonably promising, but they acknowledged a need to incorporate other sources of contextual knowledge into a real-word spelling corrector. They have proposed a hypothetical context-dependent spelling correction scheme in which an isolated-word spelling corrector would be used to generate first a relatively small candidate set ($\sim$ 5 words) for every word in a sentence. Following that, a relative likelihood rating would be ascribed to each candidate by considering five factors: (1) the degree of similarity of the candidate to the typed word; (2) part-of-speech bigram transition probabilities; (3) probabilities of individual words; (4) collocation probabilities; and (5) domain-dependent lexical preferences. They did not try to implement this scheme because they deemed it to be too computationally expensive.

3.3.2 Improving Nonword Correction Accuracy via Word Bigram Probabilities

A more general (though still local) probabilistic approach to word correction relies on the use of word bigram or trigram probabilities rather than just POS bigram probabilities. Although the word n-gram approach has the disadvantage of requiring far more data (e.g., a 50,000-word dictionary gives rise to $50{,}000^2$ word bigrams vs. 133^2 POS bigrams), it has the potential for detecting and correcting not only syntactic anomalies but also semantic anomalies that may be syntactically valid. Word n-gram approaches to context-dependent spelling correction have been explored in at least two studies. One study, by Church and Gale [1991a], demonstrated the potential of word bigrams to improve the accuracy of isolated-word spelling correctors. It also demonstrated the importance of obtaining careful estimates of word bigram probabilities [Gale and Church 1990]. Another, by Mays et al. [1991], demonstrated the detection/correction power of word trigrams. It also identified an optimum setting for an a priori likelihood estimate that the actual word typed was the correct word.

The Church and Gale [1991a] study did not address the detection problem. Instead, it tested a context-sensitive technique for improving the accuracy of the *correct* isolated-word spelling correc-

tion program (described in Section 2.2.5). Church and Gale's technique took as input a nonword together with its immediate left and right contextual neighbors. First, *correct* was used to generate a set of correction candidates and their probabilities for the nonword. Next, word bigram probabilities were used to estimate left and right context conditional probabilities for each candidate. Finally, candidates were scored and ranked by computing a simple Bayesian combination of the three probabilities for the candidate itself and its left and right context conditional probabilities. Results obtained from the context-sensitive ranking were then compared to results obtained from the *correct* ranking alone. Experimental results indicated that, for a test set of 329 errors, the context-sensitive ranking did in fact achieve an improvement in correction accuracy over the context-free ranking. The improvement was from 87% to almost 90%, which the authors found to be significant.

Gale and Church [1990] also found that the performance of the context-sensitive technique was highly sensitive to the method used to estimate conditional probabilities given word unigram and bigram frequencies. Their unigram and bigram frequencies were tabulated from a 44 million-word corpus of AP news wire text. Of all the estimation techniques they tried, including various combinations of maximum-likelihood estimation, expected-likelihood estimation, and a Good-Turing estimation [Good 1953; Church and Gale 1991b], the only combination to achieve a significant improvement in performance was one that combined a Good-Turing estimation with an expected-likelihood estimation.

3.3.3 Detecting and Correcting Real-Word Errors via Lexical Trigram Probabilities

The study by Mays et al. [1991] addressed the simultaneous detection and correction capability of word trigrams. It employed a word trigram SLM based on a 20,000-word lexicon that was compiled originally for IBM's speech recognition experiments. It also employed a set of 100 sentences containing only words found in that lexicon. The researchers created a set of 8,628 erroneous sentences, each containing one real-word error, by generating every possible single-error real-word misspelling of each word in all 100 sentences and by substituting "all of the misspellings for all of the words exactly once." This error set, together with the 100 original sentences, was used to test the ability of word trigrams to detect and correct real-word errors.

Their combined detection/correction technique consisted of computing a Bayesian probability score for each sentence in a cohort set, where the cohort set comprised one of the 8,728 test sentences together with all other sentences generated by replacing each of its words with all of its potential real-word misspellings, including the correct spelling. Given a cohort set, they could determine how often some other sentence had a higher probability than the erroneous sentence (error detection) and how often the correct sentence from the original 100 had the highest probability (error correction).

A Bayesian probability score for each sentence was computed by taking the product of the word trigram probabilities of each word in the sentence multiplied by an a priori probability for each word. The a priori probability represents the likelihood that a typed input word is actually the intended word. The set of potential real-word alternatives to the typed input word is called the confusion set and is made up of all possible single-error real-word transformations of the typed input word. The authors assume that the a priori probability for a typed input word is some constant α, and that the remaining probability $(1 - \alpha)$ is divided equally among the other words in the confusion set. If α is set too high the result will be a tendency to retain typed input words even if they are incorrect; if α is set too low the result will be a tendency to change typed input words even if they are correct. The authors tested a range of values for α and found that the optimum value was between 0.99 and 0.999.

Mays et al. [1991] obtained overall test results that yielded a detection score of 76% and a correction score of 74%. These results are informative because they provide an approximate upper bound on the capacity of lexical trigram SLMs to perform real-word error detection and correction on sentences containing single real-word errors consisting of single-error typos. A system that could detect 76% of real-word errors would probably be very useful. However, it is not yet clear what level of performance could be achieved in practice because it is unclear to what extent genuine error data conform to the one-error-per-sentence, one-typo-per-word random error distribution of the artificially generated test. Furthermore, the computational requirements of the technique were manageable under the artificial test conditions since sentences could be pregenerated and Bayesian probability scores could be precomputed. But computational requirements would certainly be problematic if cohort sets for multitypo, multiword errors had to be generated and their Bayesian scores computed on the fly. Certainly, candidate sets for individual words could be pregenerated and ranked, but it somehow seems excessive to generate a complete cohort set of all possible sentence alternatives in order to detect a possible error, especially if multiword errors are allowed. Intuition suggests that localizing the error somehow before generating the complete sentential cohort set would be useful. Might word trigrams perhaps be more successful than POS n-grams for localizing errors? Might unigram probabilities (based on word frequencies) be factored into the a priori probabilities ascribed to input words and the words in their confusion sets to better reflect error likelihoods? Obviously, there is still room for further investigation and invention.

3.3.4 Related Neural Net Research

One other area of research that has barely begun to be explored for context-dependent word recognition and error correction is that of neural net (NN) modeling. Neural net models have much in common with statistical language models in that both attempt to capture contextual expectations by modeling the conditional probability distributions that represent the strengths of associations between terms. Their main difference is that in SLMs expectations are *explicitly* represented as conditional probabilities, whereas in NNs they are *implicitly* represented as weights in the nets. Another non-negligible difference is the intense computational storage and processing requirements of neural nets. For SLMs, hundreds of thousands of conditional probabilities for lexicons of many thousands of words can be incrementally compiled and stored on disks. In contrast, for NNs, thousands of nodes representing words and hundreds of thousands of weights representing strengths of associations between words must be fully contained in working memory during training and processing. Unfortunately, those demands make NN models impractical for vocabularies larger than a few thousand words at most, at least until specialized neural net hardware and software or clever neural net modularization techniques become available.

Nevertheless, some peripherally relevant neural net experiments have been carried out by restricting NN models to represent only syntactic parts of speech or by severely limiting vocabulary size. One of the earliest such experiments was performed by Hanson and Kegl [1987] who used the syntactically tagged Brown corpus [Kucera and Francis 1967] to successfully train a neural net to predict the next part-of-speech based on the parts-of-speech of previous words in the sentence. More recently, Allen and Kamm [1990] trained a net to recognize words in an input stream of continuous phonemes, using a limited vocabulary of 72 words. Most recently, Gallant [1991] proposed a neural net model that employs a context vector representing the strengths of associated terms for use in a word sense disambiguation task. These experiments and proposals only hint at the possibili-

ties for neural nets to represent and exploit contextual information. However, those possibilities will not be fully explored until more practical neural net computational tools and techniques become available.

3.4 Summary of Context-Dependent Word Correction Work

Research in context-dependent spelling correction techniques is in its infancy. Despite limited empirical studies of the pervasiveness and character of real-word errors in text-processing applications, the need for context-dependent spelling correction and word recognition techniques is clear. The few data points that do exist suggest that real-word errors account for anywhere from 25% to well over 50% of all textual errors depending on the application. One study indicates further that as much as 75% of real-word errors involve syntactic violations. This latter finding, which begs further empirical confirmation, suggests the value of exploring syntactic techniques for detecting and correcting such errors.

The two main approaches being explored for exploiting contextual information are traditional NLP (Natural Language Processing) and SLM (Statistical Language Modeling). For practical use on unrestricted domains, NLP techniques are currently limited to syntax-driven approaches. One research prototype NLP system, EPISTLE, offered accurate or useful corrections 82%, 87%, and 41% of the time when tested on corpora containing genuine errors in the genres of business correspondence, student essays, and professional text, respectively. Another syntactic rule-based parser/corrector for the Dutch language detected and accurately corrected 72 of 75 errors, including 14 real-word errors, on a standard secretarial typing exam, without generating any false alarms.

SLM techniques may be explicitly syntactic in their approach, as when they exploit part-of-speech statistics, or they may be implicitly semantic, as when they exploit word co-occurrence statistics which tend to reflect latent semantic relations. One study demonstrated the effectiveness of word bigram statistics for improving the accuracy of a nonword corrector. But the problem of detecting real-word errors remains a significant challenge. One real-word error detection study demonstrated that a syntactic part-of-speech bigram model was able to detect 62% of 420 real-word errors that involved local or global syntactic-constraint violations, but not without generating nearly twice as many false alarms. Another study demonstrated that a word trigram model accurately detected real-word errors 76% of the time and accurately corrected those errors 74% of the time, but not without significant computational overhead resulting from the need to generate and test a large number of alternative sentences for each potentially erroneous input sentence. Further research is needed to devise computationally feasible real-word error detection techniques that are sensitive enough to attain high correction rates and discriminating enough to maintain low false alarm rates.

Neural net approaches will remain constrained to limited vocabularies until efficient neural net hardware and software become available. Most importantly, these and other developments now point toward a host of new techniques that could add significant intelligence to our computational text and speech-processing systems. Hence the topic of the epilogue.

FUTURE DIRECTIONS

Words are a basic unit of communication common to all natural language text and speech-processing activities. But the physical signals that embody words, whether realized in electronic, acoustic, optical, or other forms, frequently arrive at their destinations in imperfect condition. Therefore, automatic spelling correction, or, more generally, automatic word recognition, is an essential, nontrivial, practical problem at the heart of all computational text- and speech-proces-

sing applications. Before any text understanding, speech recognition, machine translation, computer-aided learning, optical character, or handwriting recognition system can achieve a marketable performance level, it must tackle the pervasive problem of dealing with noisy, ill-fitting, novel, and otherwise unknown input words.

Now is a particularly exciting time to be studying the problem of recognizing and correcting words in text and speech. In addition to the variety of techniques that are beginning to show potential, including rule-based, statistical, and neural net techniques, various efforts are underway, under the auspices of the Association for Computational Linguistics, to accumulate a "critical mass" of sharable text, including large textual corpora, machine-readable dictionaries, transcriptions of speech, bilingual corpora, etc., and to develop tools for annotating that text [Walker 1991; Liberman and Walker 1989]. Past research has shown that by bootstrapping from manually annotated corpora (with part-of-speech tags, for example), automatic techniques can be developed for annotating ever larger corpora, which can then be used for exploring higher-level text-processing techniques. Some manual efforts are underway, and some automatic tools are already being developed for some of the lower- and mid-level tasks involved in text processing, including formatting, tagging, aligning, and extracting lexical, syntactic, and even latent semantic statistics. These tools and their resultant annotated corpora and statistical language models will pave the way for developing hybrid rule-based, statistical, and neural net systems for higher-level text-processing tasks, including real-word error detection and correction, parsing, generation, semantic understanding, and even pragmatic and discourse modeling. As the increasing body of tools, annotated corpora, and statistical language models becomes generally available, researchers will be able to build on each other's results for ever larger domains of discourse. In short, the synergistic effects resulting from the shared use and combination of these techniques and resources could be explosive.

So what are some of the possible error detection and correction techniques that might be explored? Hybrid rule-based/statistical parsers capable of probabilistic error detection and correction are inevitable. The prerequisite for such systems is the general availability of lexical and part-of-speech n-gram statistical language models. As mentioned in Section 3.4, a variety of potential statistical techniques has yet to be explored for localizing errors. These include the use of word n-grams and the use of a combination of part-of-speech trigram statistics and part-of-speech probabilities for individual words. Other creative combinations of statistics will certainly be invented. Abandoning the assumption that the word is the basic unit of processing will open up a host of word recognition, error correction, and other processing possibilities. Not only might lexical or part-of-speech n-grams be treated as basic units, but so also might complex units built of small syntactic structures that are fully or only partially lexically instantiated.[12] Statistical language models as well as annotated dictionaries based on such syntactic structures will certainly become available as corpora-tagging efforts progress.

In the long term, the arrival of high-speed hardware and software for large-scale neural net processing and the availability of sufficiently large tagged training corpora could mark a watershed in computational language processing. Neural nets could be trained to capture and adaptively reflect the changing lexical, syntactic, and error probability statistics of given discourse domains or user populations. Even in the short term, the possibility exists for hybrid neural net/statistical techniques to improve the error detection capabilities of existing statistical techniques by addressing the

[12] See Joshi's [1985] Tree-Adjoining-Grammar (TAG) formalism, for example.

A Context-Dependent Word Recognition Model

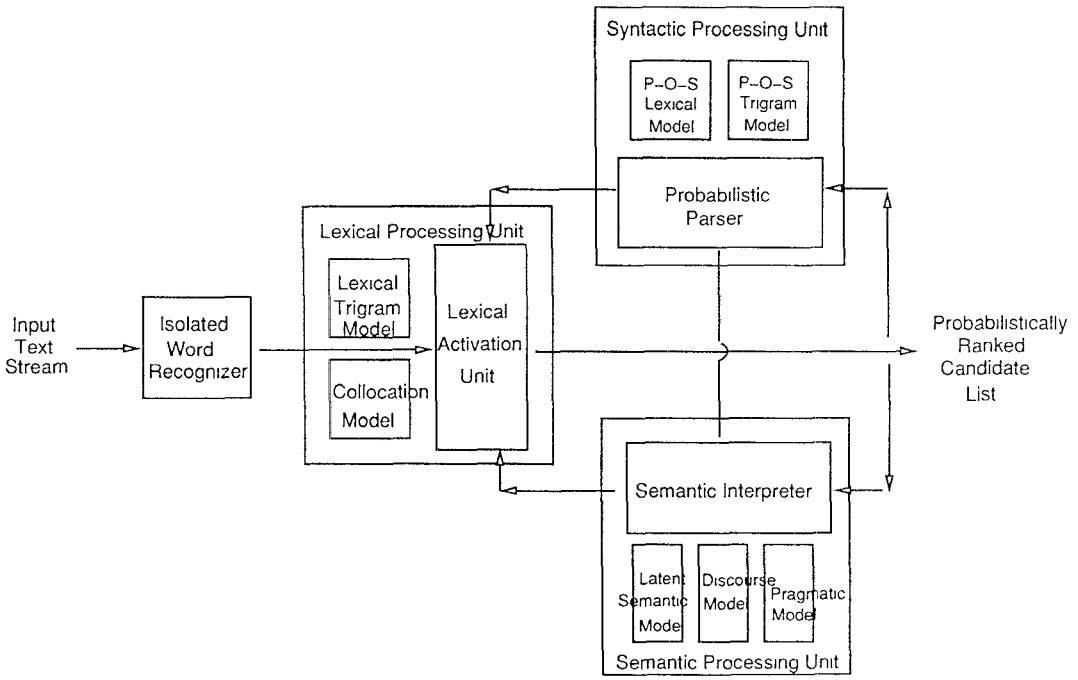

Figure 1.

precision-recall problem. For example, a neural net might be trained to improve the discrimination capability of part-of-speech transition probability statistics by distinguishing between genuine errors and false alarms.

A rough diagram for a hypothetical, large-scale, expectation-based word recognition/error detection system that would exploit many of the sources of linguistic knowledge available to humans might look something like the model in Figure 1.

The central premise, hence the central component, of the model is the postulation of a lexical processing unit whose job is to intelligently combine probabilistic estimates of the identity of a word in a text input stream. The estimates come from three primary sources: the input signal itself, feedback from a syntactic processing unit, and feedback from a semantic processing unit.

In the first phase of processing, a token from the text input stream, which may be a correct word, a nonword, or a real-word error, is fed to the Isolated-Word Recognizer. This unit produces a probabilistically ranked list of word candidates based on orthographic, phonetic, and visual similarities as well as error probabilities. Thus, it is the locus of nonword spelling error detection, and it is also the source of candidate corrections for real-word errors. The output of this unit could be thought of as a large vector representing the entire lexicon, or it could be treated as a truncated list of the top-ranked lexicon entries given the input signal and the orthographic and phonemic characteristics of the lexicon and the error probability characteristics of the linguistic domain. In either case, that output becomes input to the Lexical Processing Unit.

The Lexical Processing Unit contains a

Lexical Trigram Model, a Collocation Model, and a Lexical Activation Unit. The latter retains some short-term memory for the past few words that have been processed which it uses in combination with conditional probabilities from the lexical trigram and collocation models to rerank the candidate list to take lexical context into account. The output of the Lexical Processing Unit is a new list of word candidates whose probabilistic ranking is dependent on lexical context.

That output can be simultaneously sent to both a Syntactic Processing Unit and a Semantic Processing Unit. These units are the locus of real-word error detection capabilities. Input to the Syntactic Processing Unit goes to a Probabilistic Parser. The parser makes use of a Lexical Part-of-Speech Model to ascribe part-of-speech probabilities to the ranked input word candidates. It also employs some short-term memory of the parts of speech of previously processed words so that it can exploit a Part-of-Speech Trigram Model during parsing. In the process of parsing it recomputes the probabilities of the words in the candidate list based on syntactic considerations. A potential error is signaled if the top-ranked candidate changes.

A similar process is carried out by the Semantic Processing Unit. Its Semantic Interpreter may employ a Latent Semantic Association Model that was precomputed from lexical statistics, Local and Global Discourse Models that retain short- and long-term memories of the topic and focus of the discourse represented by previously processed terms, and a Pragmatic Model that represents plans and goals of the system and user. Like the Syntactic Processing Unit, the Semantic Processing Unit also recomputes the probabilities of the words in the candidate list. Again, a change in the top-ranked candidate may signal a potential real-word error.

The output streams of ranked candidates and their probabilities from both the Syntactic Processing Unit and the Semantic Processing Unit are fed back into the Lexical Processing Unit to be recombined with the unchanged ranked list from the Isolated-Word Recognizer. The newly ranked candidate list produced by the Lexical Processing Unit is sent again to both the Syntactic and Semantic Processing Units which in turn recompute their probability rankings and feed them back to the Lexical Processing Unit. This process is repeated until all three units agree on the top-ranked candidate. Of course, on rare occasions unresolvable conflicts may occur, much like the conflicts of meaning and form found in Escherian paintings, so some mechanism must be incorporated to detect such conflicts.

The individual components of the model might be implemented separately using different techniques. For example, a combination of rule-based and case-based techniques might be appropriate for the Semantic Interpreter; a combination of rule-based and stochastic techniques might be appropriate for the Probabilistic Parser; and neural net technology might someday be appropriate for the Lexical Activation Unit.

The intent of the model in Figure 1 is only to spark the imagination. Many other possible organizations and realizations of the modules in this rough model might be explored, and many practical implementation problems will pose interesting research problems. Indeed, the future of research on real-word recognition and error correction holds much promise for a wide array of creative solutions and useful applications.

ACKNOWLEDGMENTS

This article has benefited significantly from the contributions (and patience) of the editors, Sal March and Dick Muntz, from the helpful input of friends and colleagues who reviewed it, including Vladimir Cherkassky, Fred Damerau, Bill Gale, Joel Gannett, Don Gentner, Roger Mitton, Frank Smadja, and Lynn Streeter, and from many pleasant discussions and/or collaborations with Steve Abney, Al Aho, Alex Bocast, Dave Burr, Vladimir Cherkassky, Ken Church, David Copp, Scott Deerwester, Sue Dumais, Bill Gale, Jonathan Grudin, Stu Haber, Cheryl Jacques, Mark Jones, Len Kasday, Mark Kernighan, Tom Landauer, Michael

Littman, Pam Troy Moyer, Sarvar Patel, Debbie Schmitt, Frank Smadja, Scott Stornetta, Jim Tobias, Jean Veronis, and Don Walker.

REFERENCES

ABNEY, S. 1990. Rapid incremental parsing with repair. In *Proceedings of the 6th New OED Conference: Electronic Text Research* (Waterloo, Ontario, Oct. 1990).

AHO, A. V. 1990. Algorithms for finding patterns in strings. In *Handbook of Theoretical Computer Science*, J. Van Leeuwen, Ed. Elsevier Science Publishers, B. V., Amsterdam.

AHO, A. V., AND CORASICK, M. J. 1975. Fast pattern matching: An aid to bibliographic search. *Commun. ACM 18*, 6 (June), 333–340.

AHO, A. V., AND PETERSON, T. G. 1972. A minimum distance error-correcting parser for context free languages. *SIAM J. Comput.* 1, 4 (Dec.), 305–312.

ALBERGA, C. N. 1967. String similarity and misspellings. *Commun. ACM 10*, 302–313.

ALLEN, R. B., AND KAMM, C. A. 1990. A recurrent neural network for word identification from continuous phoneme strings. In *Advances in Neural Information Processing Systems*, vol. 3. R. P. Lippmann, J. E. Moody, D. S. Touretzky, Ed. Morgan Kaufmann Publishers, San Mateo, Calif.

ALM, N., ARNOTT, J. L., AND NEWELL, A. F. 1992. Prediction and conversational momentum in an augmentative communication system. *Commun. ACM 35*, 5 (May), 46–56.

ANGELL, R. C., FREUND, G. E., AND WILLETT, P. 1983. Automatic spelling correction using a trigram similarity measure. *Inf. Process. Manage. 19*, 255–261.

ATWELL, E., AND ELLIOTT, S. 1987. Dealing with ill-formed English text (Chapter 10). In *The Computational Analysis of English: A Corpus-Based Approach*. R. Garside, G. Leach, G. Sampson, Ed. Longman, Inc. New York.

BAHL, L. R., BROWN, P. F., DESOUZA, P. V., AND MERCER, R. L. 1989. A tree-based statistical language model for natural language speech recognition. *IEEE Trans. Acoust. Speech Sig. Process. 37*, 7, (July), 1001–1008.

BAHL, L. R., JELINEK, F., AND MERCER, R. L. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. Patt. Anal. Machine Intell. PAMI-5*, 2 (Mar.), 179–190.

BENTLEY, J. 1985. A spelling checker. *Commun. ACM 28*, 5 (May), 456–462.

BICKEL, M. A. 1987. Automatic correction to misspelled names: A fourth-generation language approach. *Commun. ACM 30*, 3 (Mar.), 224–228.

BLAIR, C. R. 1960. A program for correcting spelling errors. *Inf. Contr.* 3, 60–67.

BLEDSOE, W. W., AND BROWNING, I. 1959. Pattern recognition and reading by machine. In *Proceedings of the Eastern Joint Computer Conference*, vol. 16, 225–232.

BOCAST, A. K. 1991. Method and apparatus for reconstructing a token from a Token Fragment. U.S. Patent Number 5,008,818, Design Services Group, Inc. McLean, Va.

BOIVIE, R. H. 1981. Directory assistance revisited. AT & T Bell Labs Tech. Mem. June 12, 1981.

BROWN, P. F., DELLA PIETRA, V. J., DESOUZA, P. V., AND MERCER, R. L. 1990a. *Class-Based n-Gram Models of Natural Language*.

BROWN. P., COCKE, J., DELLA PIETRA, S., DELLA PIETRA, V., JELINEK, F., MERCER, R., AND ROOSIN, P. 1990b. A statistical approach to machine translation. *Comput. Ling.* 16, (June), 79–85.

BROWN, P., DELLA PIETRA, S., DELLA PIETRA, V., AND MERCER, R. 1991. Word sense disambiguation using statistical methods. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics* (Berkeley, Calif., June), ACL, 264–270.

BURR, D. J. 1983. Designing a handwriting reader. *IEEE Trans. Patt. Anal. Machine Intell. PAMI-5*, 5 (Sept.), 554–559.

BURR, D. J. 1987. Experiments with a connectionist text reader. In *IEEE International Conference on Neural Networks* (San Diego, Calif., June). IEEE, New York, IV:717–724.

CARBERRY, S. 1984. Understanding pragmatically ill-formed input. In *Proceedings of the 10th International Conference on Computational Linguistics*. ACL, 100–206.

CARBONELL, J. G., AND HAYES, P. J. 1983. Recovery strategies for parsing extragrammatical language. *Amer. J. Comput. Ling. 9*, 3–4 (July-Dec.), 123–146.

CARTER, D. M. 1992. Lattice-based word identification in CLARE. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics* (Newark, Del., June 28–July 2). ACL, 159–166.

CHERKASSKY, V., AND VASSILAS, N. 1989a. Back-propagation networks for spelling correction. *Neural Net. 1*, 3 (July), 166–173.

CHERKASSKY, V., AND VASSILAS, N. 1989b. Performance of back-propagation networks for associative database retrieval. *Int. J. Comput. Neural Net.*

CHERKASSKY, V., RAO, M., AND WECHSLER, H. 1990. Fault-tolerant database retrieval using distributed associative memories. *Inf. Sci. 46*, 135–168.

CHERKASSKY, V., FASSETT, K., AND VASSILAS, N. 1991. Linear algebra approach to neural associative memories and noise performance of neural classifiers. *IEEE Trans. Comput. 40*, 12, 1429–1435.

CHERKASSKY, V., VASSILAS, N., BRODT, G. L., AND

WECHSLER, H. 1992. Conventional and associative memory approaches to automatic spelling checking. *Eng. Appl. Artif. Intell. 5*, 3.

CHERRY, L., AND MACDONALD, N. 1983. The Writer's Workbench software *Byte*, (Oct.), 241–248.

CHOUEKA, Y. 1988. Looking for needles in a haystack. In *Proceedings of RIAO*, 609–623

CHURCH, K. W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the 2nd Applied Natural Language Processing Conference* (Austin, Tex., Feb.). ACL, 136–143.

CHURCH, K. W., AND GALE, W. A. 1991a. Probability scoring for spelling correction. *Stat. Comput. 1*, 93–103.

CHURCH, K. W., AND GALE, W. A. 1991b. Enhanced Good-Turing and cat-cal Two new methods for estimating probabilities of English bigrams. *Comput. Speech Lang.* 1991.

COHEN, G. 1980. Reading and searching for spelling errors. In *Cognitive Processes in Spelling*. Uta Frith, Ed. Academic Press, London.

COKER, C. H., CHURCH, K. W., AND LIBERMAN, M. Y. 1990. Morphology and rhyming: Two powerful alternatives to letter-to-sound rules for speech synthesis. In *Proceedings of the Conference on Speech Synthesis*. European Speech Communication Association.

CONTANT, C., AND BRUNELLE, E. 1992 Exploratexte: Un analyseur a l'affut des erreurs grammaticales. In *Actes du colloque lexiques-grammaires compares*, Universite du Quebec a Montreal. In French.

CUSHMAN, W. H., OJHA, P. S., AND DANIELS, C. M. 1990. Usable OCR: What are the minimum requirements. In *CHI-90 Conference Proceedings, Special Issue of the ACM SIGCHI Bulletin* (Seattle, Wash., Apr 1–5.) ACM, New York, 145–151.

DAHL, P., AND CHERKASSKY, V. 1990. Combined encoding in associative spelling checkers. Univ. of Minnesota EE Dept. Tech. Rep.

DAMERAU, F. J. 1990. Evaluating computer-generated domain-oriented vocabularies. *Inf Process. Manage. 26*, 6, 791–801.

DAMERAU, F. J. 1964. A technique for computer detection and correction of spelling errors. *Commun ACM 7*, 3 (Mar.), 171–176.

DAMERAU, F. J., AND MAYS, E. 1989. An examination of undetected typing errors. *Inf. Process. Manage. 25*, 6, 659–664.

DAVIDSON, L 1962. Retrieval of misspelled names in an airline passenger record system. *Commun. ACM 5*, 169–171.

DEERWESTER, S., DUMAIS, S. T., FURNAS, G. W., LANDAUER, T K., AND HARSHMAN, R. 1990. Indexing by Latent Semantic Analysis. *JASIS 41*, 6, 391–407.

DEFFNER, R., EDER, K, AND GEIGER, H. 1990a. Word recognition as a first step towards natural language processing with artificial neural nets. In *Proceedings of KONNAI-90*.

DEFFNER, R., GEIGER, H., KAHLER, R., KREMPL, T., AND BRAUER, W. 1990b. Recognizing words with connectionist architectures. In *Proceedings of INNC-90-Paris* (Paris, France, July), 196.

DEHEER, T. 1982. The application of the concept of homeosemy to natural language information retrieval. *Inf. Process. Manage.* 18, 229–236.

DELOCHE, G., AND DEBILI, F. 1980. Order information redundancy of verbal codes in French and English: Neurolinguistic implications. *J. Verbal Learn. Verbal Behav. 19*, 525–530.

DEMASCO, P. W., AND MCCOY, K. F. 1992. Generating text from compressed input: An intelligent interface for people with severe motor impairments. *Commun. ACM 35*, 5 (May), 68–78.

DEROUAULT, A.-M., AND MERIALDO, B. 1984a. Language modeling at the syntactic level. In *Proceedings of the 7th International Conference on Pattern Recognition* (Montreal, Canada, July 30–Aug. 2), 1373–1375.

DEROUAULT, A.-M., AND MERIALDO, B. 1984b. TASF: A stenotypy-to-French transcription system. In *Proceedings of the 7th International Conference on Pattern Recognition* (Montreal, Canada, July 30–Aug. 2), 866–868.

DUNLAVEY, M. R 1981. On spelling correction and beyond. *Commun. ACM 24*, 9 (Sept.), 608.

DURHAM, I., LAMB, D A., AND SAXE, J B. 1983. Spelling correction in user interfaces. *Commun. ACM 26*, 10 (Oct.), 764–773.

EASTMAN, C. M., AND MCLEAN, D. S. 1981. On the need for parsing ill-formed input. *Amer. J Comput. Ling. 7*, 4, 257.

ELLIOTT, R. J. 1988. Annotating spelling list words with affixation classes. AT & T Bell Labs Int. Mem. Dec. 14.

ELLIS, A. W. 1979 Slips of the pen. *Vis. Lang. 13*, 265–282.

ELLIS, A. W. 1982. Spelling and writing (and reading and speaking). In *Normality and Pathology in Cognitive Functions*, A. W Ellis, Ed. Academic Press, London.

FASS, D., AND WILKS, Y. 1983. Preference semantics, ill-formedness, and metaphor *Amer J. Comput. Ling. 9*, 3–4 (July-Dec), 178–189.

FINK, P. K., AND BIERMANN, A. W. 1986. The correction of ill-formed input using history-based expectation with applications to speech understanding. *Comput. Ling. 12*, 1 (Jan.-Mar.), 13–36.

FORNEY, G. D., JR. 1973. The Viterbi algorithm. *Proc. IEEE 61*, 3 (Mar.), 268–278.

FOX, E. A., CHEN, Q. F., AND HEATH, L. S. 1992. A faster algorithm for constructing minimal perfect hash functions. In *Proceedings of the*

15th Annual International SIGIR Meeting, SIGIR'92 (Denmark, June). ACM, New York, 266–273.

FREDKIN, E. 1960. Trie memory. *Commun ACM 3*, 9, (Sept.), 490–500.

FROMKIN, V., ED. 1980. *Errors in Linguistic Performance: Slips of the Tongue, Ear, Pen and Hand*. Academic Press, New York, 1980.

GALE, W. A., AND CHURCH, K. W. 1990. Estimation procedures for language context: Poor estimates are worse than none. In *Proceedings of Compstat-90* (Dubrovnik, Yugoslavia). Springer-Verlag, New York, 69–74.

GALLANT, S. I. 1991. A practical approach for representing context and for performing word sense disambiguation using neural networks. *Neural Comput. 3*, 293–309.

GARRETT, M. 1982. Production of speech: Observations from normal and pathological language use. In *Normality and Pathology in Cognitive Functions*, A. W. Ellis, Ed. Academic Press, London.

GARSIDE, R., LEACH, G., AND SAMPSON, G. 1987. *The Computational Analysis of English: A Corpus-Based Approach*. Longman, Inc., New York.

GENTNER, D. R., GRUDIN, J., LAROCHELLE, S., NORMAN, D. A., AND RUMELHART, D. E. 1983. Studies of typing from the LNR typing research group. In *Cognitive Aspects of Skilled Typewriting*, W. E. Cooper, Ed. Springer-Verlag, New York.

GERSHO, M., AND REITER, R. 1990. Information retrieval using self-organizing and heteroassociative supervised neural networks. In *Proceedings of IJCNN* (San Diego, Calif. June).

GOOD, I. J. 1953. The population frequencies of species and the estimation of population parameters *Biometrika 40*, 3 and 4 (Dec.), 129–264.

GORIN, R. E. 1971. SPELL: A spelling checking and correction program. Online documentation for the DEC-10 computer.

GOSHTASBY, A., AND EHRICH, R. W. 1988. Contextual word recognition using probabilistic relaxation labeling. *Patt. Recog. 21*, 5, 455–462.

GRANGER, R. H. 1983. The NOMAD system: Expectation-based detection and correction of errors during understanding of syntactically and semantically ill-formed text. *Amer. J. Comput. Ling. 9*, 3–4 (July-Dec.), 188–196.

GRUDIN, J. 1983. Error patterns in skilled and novice transcription typing. In *Cognitive Aspects of Skilled Typewriting*, W. E. Copper, Ed. Springer-Verlag, New York.

GRUDIN, J. 1981. The organization of serial order in typing. Ph.D. dissertation Univ. of California, San Diego.

HALL, P. A. V., AND DOWLING, G. R. 1980. Approximate string matching. *ACM Comput. Surv. 12*, 4 (Dec.), 17–38.

HANSON, S. J., AND KEGL, J. 1987. PARSNIP: A connectionist network that natural language grammar from exposure to natural language sentences. In *Proceedings of the Cognitive Science Conference*.

HANSON, A. R., RISEMAN, E. M., AND FISHER, E., 1976. Context in word recognition. *Patt. Recog. 8*, 35–45.

HARMON, L. D. 1972. Automatic recognition of print and script. *Proc. IEEE 60*, (Oct.), 1165–1176.

HAWLEY, M. J. 1982. Interactive spelling correction in Unix: The METRIC Library. AT & T Bell Labs Tech. Mem., August 31.

HEIDORN, G. E. 1982. Experience with an easily computed metric for ranking alternative parses. In *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics* (Toronto, Canada). ACL, 82–84.

HEIDORN, G. E., JENSEN, K., MILLER, L. A., BYRD, R. J., AND CHODOROW, M. S. 1982. The EPISTLE text-critiquing system. *IBM Syst. J. 21*, 3, 305–326.

HENSELER, J., SCHOLTES, J. C., AND VERDOEST, C. R. J. 1987. The design of a parallel knowledge-based optical character recognition system. Master of Science Theses, Dept. of Mathematics and Informatics, Delft Univ. of Technology.

HINDLE, D. 1983. User manual for Fidditch, a deterministic parser. Tech. Mem. 7590–142, Naval Research Lab.

HO, T. K., HULL, J. J., AND SRIHARI, S. N. 1991. Word recognition with multi-level contextual knowledge. In *Proceedings of IDCAR-91* (St. Malo, France), 905–915.

HOTOPF, N. 1980. Slips of the pen. In *Cognitive Processes in Spelling*, Uta Frith, Ed. Academic Press, London.

HULL, J. J. 1987. Hypothesis testing in a computational theory of visual word recognition. In *Proceedings of AAAI-87, 6th National Conference on Artificial Intelligence*. vol. 2 (Seattle, Wash., July 13–17). AAAI, 718–722.

HULL, J. J., AND SRIHARI, S. N. 1982. Experiments in text recognition with binary *n*-gram and Viterbi algorithms. *IEEE Trans. Patt. Anal. Machine Intell. PAMI-4*, 5 (Sept.), 520–530.

JELINEK, F., MERIALDO, B., ROUKOS, S., AND STRAUSS, M. 1991. A dynamic language model for speech recognition. In *Proceedings of the DARPA Speech and Natural Language Workshop* (Feb. 19–22), 293–295.

JENSEN, K., HEIDORN, G. E., MILLER, L. A., AND RAVIN, Y. 1983. Parse fitting and prose fixing: Getting a hold on ill-formedness. *Amer. J. Comput. Ling. 9*, 3–4 (July-Dec.), 147–160.

JOHNSTON, J. C., AND MCCLELLAND, J. L. 1980. Experimental tests of a hierarchical model of word identification. *J. Verbal Learn. Verbal Behav. 19*, 503–524.

JONES, M. A., STORY, G. A., AND BALLARD, B. W. 1991. Integrating multiple knowledge sources in a Bayesian OCR post-processor. In *Proceedings of IDCAR-91* (St Malo, France), 925–933.

JOSHI, A. K. 1985. How much context-sensitivity is necessary for characterizing structural descriptions—Tree Adjoining Grammars In *Natural Language Processing—Theoretical, Computational and Psychological Perspectives*, D. Dowty, L. Karttunen, A. Zwicky, Ed. Cambridge University Press, New York.

KAHAN, S , PAVLIDIS, T., AND BAIRD. H. S. 1987. On the recognition of characters of any font size *IEEE Trans Patt. Anal. Machine Intell. PAMI-9*, 9, 274–287

KASHYAP, R. L . AND OOMMEN, B. J. 1981 An effective algorithm for string correction using generalized edit distances. *Inf Sci 23*, 123–142.

KASHYAP, R. L., AND OOMMEN, B. J. 1984. Spelling correction using probabilistic methods. *Patt Recog. Lett. 2*, 3 (Mar.), 147–154.

KEELER, J., AND RUMELHART, D. E. 1992. A self-organizing integreted segmentation and recognition neural net. In *Advances in Neural Information Processing Systems*, vol. 4, J. E. Moody, S. J. Hanson, R. P. Lippmann, Ed. Morgan Kaufmann, San Mateo, Calif., 496–503.

KEMPEN, G., AND VOSSE, T. 1990. A language-sensitive text editor for Dutch. In *Proceedings of the Computers and Writing III Conference* (Edinburgh, Scotland, Apr)

KERNIGHAN, M. D. 1991. Specialized spelling correction for a TDD system AT & T Bell Labs Tech. Mem., August. 30.

KERNIGHAN, M. D., AND GALE, W. A. 1991. Variations on channel-frequency spelling correction in Spanish. AT & T Bell Labs Tech. Mem., September.

KERNIGHAN, M. D., CHURCH, K. W., AND GALE, W. A. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of COLING-90, The 13th International Conference on Computational Linguistics*, vol. 2 (Helsinki, Finland). Hans Karlgren, Ed. 205–210.

KNUTH, D. E. 1973. *The Art of Programming*. Vol. 3, *Sorting and Searching*. Addison-Wesley, Reading, Mass.

KOHONEN, T. 1980. *Content Addressable Memories* Springer-Verlag, New York.

KOHONEN, T. 1988. *Self-Organization and Associative Memory*. Springer-Verlag, New York.

KUCERA, H., AND FRANCIS, W. N. 1967. *Computational Analysis of Present-Day American English* Brown University Press, Providence, R.I.

KUKICH, K. 1988a. Variations on a back-propagation name recognition net. In *Proceedings of the Advanced Technology Conference*, vol 2 (May 3–5). U.S. Postal Service, Washington D.C., 722–735.

KUKICH, K. 1988b. Back-propagation topologies for sequence generation. In *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1 (San Diego, Calif., July 24–27). IEEE, New York, 301–308.

KUKICH, K. 1990 A comparison of some novel and traditional lexical distance metrics for spelling correction. In *Proceedings of INNC-90-Paris* (Paris, France, July), 309–313.

KUKICH, K. 1992. Spelling correction for the telecommunications network for the deaf. *Commun ACM 35*, 5 (May), 80–90.

LANDAUER, T. K , AND STREETER, L. A. 1973. Structural differences between common and rare words. *J. Verbal Learn. Verbal Behav. 12*, 119–131.

LEE, Y.-H., EVENS, M., MICHAEL, J. A., AND ROVICK, A. A. 1990. Spelling Correction for an intelligent tutoring system. Tech. Rep., Dept. of Computer Science, Illinois Inst. of Technology, Chicago

LEVENSHTEIN, V I. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl. 10*, (Feb), 707–710.

LIBERMAN, M. Y., AND WALKER, D. E. 1989. ACL Data Collection initiative: First release. *Finite String 15*, 4 (Dec.), 46–47.

LOWRANCE, R., AND WAGNER, R. 1975. An extension of the string-to-string correction problem. *J. ACM 22*, 2 (Apr.), 177–183.

MATAN, O., BURGES, C. J. C , LECUN, Y , AND DENKER, J. S. 1992. Multi-digit recognition using a space displacement neural network. In *Advances in Neural Information Processing Systems*, vol. 4, J. E Moody, S. J. Hanson, R. P. Lippmann, Ed. Morgan Kaufmann, San Mateo, Calif , 488–495.

MAYS, E., DAMERAU, F. J., AND MERCER, R L 1991. Context based spelling correction. *Inf. Process. Manage. 27*, 5, 517–522.

MCCLELLAND, J. L., AND RUMELHART. D. E. 1981 An interactive activation model of context effects in letter perception. *Psychol. Rev. 88*, 5 (Sept.), 375–407.

MCCOY, K. F. 1989 Generating context-sensitive responses to object-related misconceptions. *Artif. Intell.* 41, 157–195

MCILROY, M. D 1992. Development of a spelling list. *IEEE Trans. Commun. COM-30*, 1 (Jan.), 91–99.

MEANS, L. G. 1988. Cn yur cmputr raed ths. In *Proceedings of the 2nd Applied Natural Language Processing Conference* (Austin, Tex, Feb.). ACL, 93–100.

MINTON, S., HAYES, P. J., AND FAIN J. 1985. Controlling search in flexible parsing. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan Kaufman, San Mateo, Calif., 786–787.

MITTON, R. 1987. Spelling checkers, spelling correctors, and the misspellings of poor spellers. *Inf. Process. Manage. 23*, 5, 495–505.

MITTON, R. 1986. A partial-dictionary of English in computer-usable form. *Lit. Ling. Comput. 1*, 4, 214–215.

MITTON, R. 1985. A collection of computer-readable corpora of English spelling errors. *Cog. Neuropsychol. 2*, 3, 275–279.

MOR, M., AND FRAENKEL, A. S. 1982a. Retrieval in an environment of faulty texts or faulty queries. In *Proceedings of the 2nd International Conference on Improving Database Usability and Responsiveness* (Jerusalem), P. Scheuerman, Ed. Academic Press, New York, 405–425.

MOR, M., AND FRAENKEL, A. S. 1982b. A hash code method for detecting and correcting spelling errors. *Commun. ACM 25*, 12 (Dec.), 935–938.

MORGAN, H. L. 1970. Spelling correction in systems programs. *Commun. ACM 13*, 2 (Feb.), 90–94.

MORRIS, R., AND CHERRY, L. L. 1975. Computer detection of typographical errors. *IEEE Trans. Profess. Commun. PC-18*, 1, 54–63.

MUTH, F. E., JR., AND THARP, A. L. 1977. Correcting human error in alphanumeric terminal input. *Inf. Process. Manage. 13*, 329–337.

NEMHAUSER, G. L. 1966. *Introduction to Dynamic Programming*. Wiley, New York.

NIELSEN, J., PHILLIPS, V. L., AND DUMAIS, S. T. 1992. Retrieving imperfectly recognized handwritten notes. *Behav. Inf. Tech.*

ODELL, M. K., AND RUSSELL, R. C. 1918. *U.S. Patent Numbers, 1,261,167 (1918) and 1,435,663 (1922)*. U.S. Patent Office, Washington, D.C.

OKUDA, T., TANAKA, E., AND KASAI, T. 1976. A method of correction of garbled words based on the Levenshtein metric. *IEEE Trans. Comput. 25*, 172–177.

OSHIKA, T., MACHI, F., EVANS, B., AND TOM, J. 1988. Computational techniques for improved name search. In *Proceedings of the 2nd Annual Applied Natural Language Conference* (Austin, Tex, Feb.). ACL, 203–210.

PARSAYE, K., CHIGNELL, M., KHOSHAFIAN, S., AND WONG, H. 1990. Intelligent databases. *AI Expert*, (Mar.), 38–47.

PETERSON, J. L. 1980. Computer programs for detecting and correcting spelling errors. *Commun. ACM 23*, 12, (Dec.), 676–684.

PETERSON, J. L. 1986. A note on undetected typing errors. *Commun. ACM 29*, 7 (July), 633–637.

POLLOCK, J. J., AND ZAMORA, A. 1983. Collection and characterization of spelling errors in scientific and scholarly text. *J. Amer. Soc. Inf. Sci. 34*, 1, 51–58.

POLLOCK, J. J., AND ZAMORA, A. 1984. Automatic spelling correction in scientific and scholarly text. *Commun. ACM 27*, 4 (Apr.), 358–368.

RAMSHAW, L. A. 1989. Pragmatic knowledge for resolving ill-formedness. Tech. Rep. No. 89-18, BBN, Cambridge, Mass.

RHYNE, J. R., AND WOLF, C. G. 1991. Paperlike user interfaces. RC 17271 (#76097), IBM Research Division, T. J. Watson Research Center, Yorktown Heights, N.Y.

RHYNE, J. R., AND WOLF, C. G. 1993. Recognition-based user interfaces. In *Advances in Human-Computer Interaction*, vol. 4, H. R. Hartson and D. Hix, Ed. Ablex, Norwood, N.J.

RICHARDSON, S. D., AND BRADEN-HARDER, L. C. 1988. The experience of developing a larger-scale natural language text processing system: CRITIQUE. In *Proceedings of the 2nd Annual Applied Natural Language Conference*, (Austin, Tex. Feb.). ACL, 195–202.

RISEMAN, E. M., AND HANSON, A. R. 1974. A contextual postprocessing system for error correction using binary *n*-grams. *IEEE Trans. Comput. C-23*, (May), 480–493.

ROBERTSON, A. M., AND WILLETT, P. 1992. Searching for historical word-forms in a database of 17th-century English text using spelling-correction methods. In *Proceedings of the 15th Annual International SIGIR Meeting, SIGIR'92* (Denmark, June). ACM, New York, 256–265.

ROSENFELD, A., HUMMEL, R. A., AND ZUCKER, S. W. 1976. Scene labeling by relaxation operations. *IEEE Trans. Syst. Man Cybernet. SMC-6*, 6, 420–433.

RUMELHART, D. E., AND MCCLELLAND, J. L. 1982. An interactive activation model of context effects in letter perception. *Psychol. Rev. 89*, 1, 60–94.

RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Ed. Bradford Books/MIT Press.

SALTON, G. 1989. Automatic text transformations. In *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, Reading, Mass.

SAMPSON, G. 1989. How fully does a machine-usable dictionary cover English text. *Lit. Ling. Comput. 4*, 1, 29–35.

SANKOFF, D., AND KRUSKAL, J. B. 1983. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, Mass.

SANTOS, P. J., BALTZER, A. J., BADRE, A. N., HENNEMAN, R. L., AND MILLER, M. S. 1992. On handwriting recognition system performance: Some experimental results. In *Proceedings of the Human Factors Society 36th Annual Meeting* (Atlanta, Ga., Oct. 12–16). Human Factors Society.

SCHANK, R. C., LEBOWITZ, M., AND BIRNBAUM, L. 1980. An integrated understander. *Am. J. Comput. Lng. 6*, 1, 13–30.

SHEIL, B. A. 1978. Median split trees. A fast look-up technique for frequently occurring keys. *Commun. ACM 21*, 11 (Nov.), 947–958

SHINGHAL, R., AND TOUSSAINT, G. T 1979a Experiments in text recognition with the modified Viterbi algorithm. *IEEE Trans Patt. Anal. Machine Intell. PAMI-1*, 4 (Apr), 184–193.

SHINGHAL, R., AND TOUSSAINT, G. T. 1979b. A bottom-up and top-down approach to using context in text recognition. *Int. J. Man-Machine Stud. 11*, 201–212.

SIDOROV, A. A. 1979. Analysis of word similarity on spelling correction systems. *Program. Comput. Softw 5*, 274–277.

SINHA, R. M. K., AND PRASADA, B. 1988. Visual text recognition through contextual processing. *Patt. Recog. 21*, 5, 463–479.

SITAR, E. J. 1961. Machine recognition of cursive script: The use of context for error detection and correction. Bell Labs Tech. Mem.

SLEATOR, D. D., AND TEMPERLY, D. 1992. *Parsing English with a Link Grammar*. Source code via internet host: spade.pc.cs.cmu.edu:/usr/sleator/public. Carnegie-Mellon Univ., Pittsburgh, Pa.

SMADJA, F. 1991a From *n*-grams to collocations: An evaluation of XTRACT. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics* (Berkeley, Calif., June). ACL, 279–284.

SMADJA, F. 1991b. Extracting collocations from text. An application: Text Generation. Ph.D. dissertation, Columbia Univ., New York.

SMADJA, F., AND MCKEOWN, K. 1990. Automatically extracting and representing collocations for language generation. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, (Pittsburgh, Pa., June). ACL, 252–259.

SPENKE, M., BEILKEN, C., MATTERN, F., MEVENKAMP, M., AND H. M. 1984. A language independent error recovery method for LL(1) parsers. *Softw. Pract. Exp. 14*, 11.

SRIHARI, S., ED. 1984. *Computer Text Recognition and Error Correction*. IEEE Computer Society Press, Piscataway, N.J

SRIHARI, S. N., HULL, J. J., AND CHOUDHARI, R. 1983. Integrating diverse knowledge sources in text recognition. *ACM Trans. Office Inf. Syst. 1*, 1 (Jan.), 68–87.

SURI, L. Z. 1991. Language transfer: A foundation for correcting the written English of ASL signers. Tech. Rep. No. 91-19, Dept. of Computer and Information Sciences, Univ. of Delaware, Newark, Del.

SURI, L. Z., AND MCCOY, K. F. 1991. Language transfer in deaf writing: A correction methodology for an instructional system. Tech. Rep. No. 91-20, Dept. of Computer and Information Sciences, Univ. of Delaware, Newark, Del.

TAYLOR, W D. 1981. GROPE—A spelling error correction tool. AT & T Bell Labs Tech. Mem.

TENCZAR, P., AND GOLDEN, W. 1972. CERL Report X-35. Computer-Based Education Research Lab., Univ of Illinois, Urbana, Ill.

THOMPSON, B. H. 1980. Linguistic analysis of natural language communication with computers. In *Proceedings of the 8th International Conference on Computational Linguistics* (Tokyo, Japan), 190–201.

TOUSSAINT, G T. 1978. The use of context in pattern recognition. *Patt Recog. 10*, 189–204.

TRAWICK, D J. 1983. Robust sentence analysis and habitability. Ph.D dissertation, California Inst. of Technology, Pasadena, Calif.

TROY, P. L. 1990 Combining probabilistic sources with lexical distance measures for spelling correction. Bellcore Tech Memo., Bellcore, Morristown, N.J.

TSAO, Y. C. 1990. A lexical study of sentences typed by hearing-impaired TDD users. In *Proceedings of the 13th International Symposium on Human Factors in Telecommunications* (Turin, Italy, Sept), 197–201.

TURBA, T. N. 1981. Checking for spelling and typographical errors in computer-based text. *SIGPLAN-SIGOA Newslett.* (June), 51–60.

ULLMANN, J. R. 1977 A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words. *Comput J. 20*, 141–147.

VAN BERKEL, B., AND DESMEDT, K. 1988 Triphone analysis: A combined method for the correction of orthographical and typographical errors. In *Proceedings of the 2nd Applied Natural Language Processing Conference* (Austin, Tex., Feb.). Association for Computational Linguistics (ACL).

VERONIS, J. 1988a. Computerized correction of phonographic errors. *Comput. Hum. 22*, 43–56.

VERONIS, J 1988b. Morphosyntactic correction in natural language interfaces. in *Proceedings of the 12th International Conference on Computational Linguistics* (Budapest, Hungary), 708–713

VOSSE, T. 1992. Detecting and correcting morpho-syntactic errors in real texts. In *Proceedings of the 3rd Conference on Applied Natural Language Processing* (Trento, Italy, Mar. 31–Apr.3). ACL, 111–118.

WAGNER, R. A. 1974. Order-n correction for regular languages. *Commun. ACM 17*, 5 (May), 265–268.

WAGNER, R. A., AND FISCHER, M. J 1974. The string-to-string correction problem. *J ACM 21*, 1 (Jan.), 168–178.

WALKER, D. E. 1991. The ecology of language. In *Proceedings of the International Workshop on Electronic Dictionaries* (Feb.). Japan Elec-

tronic Dictionary Research Institute, Tokyo, 10–22.

WALKER, D. E., AND AMSLER, R. A. 1986. The use of machine-readable dictionaries in sublanguage analysis. In *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*. Lawrence Erlbaum, Hillsdale, N.J., 69–83.

WALTZ, D. L. 1978. An English language question answering system for a large relational database. *Commun. ACM 21*, 7, 526–539.

Webster's New World Misspeller's Dictionary. Simon and Schuster, New York.

WEISCHEDEL, R. M., AND SONDHEIMER, N. K. 1983. Meta-rules as a basis for processing ill-formed input. *Amer. J. Comput. Ling. 9*, 3–4 (July–Dec.), 161–177.

WING, A. M., AND BADDELEY, A. D. 1980. Spelling errors in handwriting: A corpus and distributional analysis. In *Cognitive Processes in Spelling*, U. Frith, Ed. Academic Press, London.

WONG, C. K., AND CHANDRA, A. K. 1976. Bounds for the string editing problem. *J. ACM 23*, 1 (Nov.), 13–16.

WRIGHT, A. G., AND NEWELL, A. F. 1991. Computer help for poor spellers. *Brit. J. Educ. Tech. 22*, 2 (Feb.), 146–148.

YANNAKOUDAKIS, E. J., AND FAWTHROP, D. 1983a. An intelligent spelling corrector. *Inf. Process. Manage. 19*, 12, 101–108.

YANNAKOUDAKIS, E. J., AND FAWTHROP, D. 1983b. The rules of spelling errors. *Inf. Process. Manage. 19*, 2, 87–99.

YOUNG, C. W., EASTMAN, C. M., AND OAKMAN, R. L. 1991. An analysis of ill-formed input in natural language queries to document retrieval systems. *Inf. Process. Manage. 27*, 6, 615–622.

ZAMORA, E. M., POLLOCK, J. J., AND ZAMORA, A. 1981. The use of trigram analysis for spelling error detection. *Inf. Process. Manage. 17*, 6, 305–316.

ZIPF, G. K. 1935. *The Psycho-Biology of Language*. Houghton Mifflin, Boston.

Received February 1992; accepted June 1992