Spelling assistance for compound words

by Rudolf Frisch
Antonio Zamora

This paper describes a method for providing spelling assistance for Germanic compound words. The technique systematically analyzes an unknown word to determine its components, using a dictionary which associates word components with codes that describe their compounding characteristics. Language-specific morphological transformations are used to take into consideration common intraword elision patterns. Special dictionary entries, heuristic rules, and lexical distance measures are used to provide the best possible replacement compound words. The method is fast and provides spelling assistance and hyphenation support in an interactive environment.

Introduction

Spelling verification and assistance are now considered essential components of word-processing packages even for personal computers. Spelling verification is the process of highlighting the misspellings of a document, whereas spelling assistance involves displaying a set of correctly spelled words which could potentially replace a misspelling. The level of support provided by these programs depends on the capabilities of the computer and on the sophistication of the software.

In general, all spelling-verification programs work by referencing a dictionary of correctly spelled words [1–4]. Interactive spelling verification can be supported by encoding the dictionaries as hashcodes to achieve the necessary response time, but such dictionaries cannot be used to support spelling aid because the words cannot be reconstructed from the hashcodes. The latest generation of software uses compressed dictionaries which take into consideration the frequency of occurrence of the words to achieve the desired speed and to have the reversibility necessary to supply words from the dictionary as spelling-aid candidates [4–6].

The basic technique used for providing spelling-aid candidates is to scan a word list (or part thereof), associate a figure of merit with each word in the list, and present a number of words with the best figures of merit as replacement candidates. The figures of merit can be obtained by using a measure for string similarity which determines how many error operations are required to change one word to another [5–7].

Spelling-support technology has not evolved for all languages with the same ease as it has for English. The morphological simplicity of written English and strong economic marketing factors are responsible for the rapid maturation of the English technology. The spelling-support technology for English can be used for other languages with few changes, but there are some languages, such as Finnish, which have an elaborate inflectional system and cannot use the same technology [8]. There are also languages (e.g., German) which can use most of the technology, but require special compound-word handling.

Because word agglutination is not a very productive word-formation mechanism in written English, it is possible to

include a very large percentage of compound words in a computerized word list. This is not feasible for other Germanic languages due to the exceedingly large combinatorial compounding possibilities which they allow. If one were to create a large list of compound words—e.g., for German—this would require an enormous amount of storage, but it still would not guarantee that scanning the list would result in the retrieval of suitable candidates for misspelled compound words or that correctly spelled compound words would match against it.

The "compound word" concept

Compound words are a common occurrence in the Germanic-language family. Present-day Germanic languages are generally divided into the North Germanic and West Germanic groups. The main languages of the former include Danish, Icelandic, Norwegian, and Swedish, whereas the latter include Afrikaans, Dutch, English, and German.

Compound words can be formed through

a. A simple word sequence such as *salt water, sugar cube, snow removal equipment repair facility.*
b. A sequence of words joined by required hyphens, e.g., *mother-in-law, able-bodied.*
c. A sequence of words directly agglutinated, such as *homemaker, housewife,* or the German word *Gesundheitsamt* ('health department').

It is important to notice that a component of a compound word formed by mechanisms b or c might not be usable as a word by itself, e.g., "bodied" in b, or "Gesundheits" in c. The morphological mechanism of word compounding in English has been examined by Botha [9].

In this paper, the concept of "compound word" will be restricted to compound words formed by mechanism c above, since this paper deals only with the problem of providing spelling assistance for misspelled compound words that do not have internal punctuation delimiters. It should also be noted that although the approach presented here is generally applicable to the Germanic languages, it is not limited to them.

Dictionary features

Earlier work [10] on compound-word spelling verification provided the basic characteristics of the dictionary which were adapted for compound-word spelling aid. The dictionary consists, in essence, of a list of words each of which is associated with codes that indicate whether the word can be 1) stand-alone, 2) a front component, 3) a middle component, or 4) a back component of a compound word. Since these four attributes are independent, any of 15 possible codes can be associated with a dictionary word to indicate its compounding characteristics. In addition to these codes, a different set of codes can be used to specify language-dependent transformations [11].

The consequence of this coding scheme is that the dictionary can contain some morphemes that are only combining forms rather than stand-alone words. Such forms (e.g., in English, the form *Russo,* as in *Russo-Japanese*) cannot be presented as aid candidates outside of an appropriate compound-word context.

Words that are altered when combined in particular sequences are entered in the dictionary in their different forms. Such words are created by the occurrence of "fugen-characters" (binding morphemes) or letter sequences which are inserted at the junction of some word agglutinations. For example, in German the word *Achtung* ('attention'), which can be a stand-alone or back component, is transformed to the form *Achtungs* when it is used as a front or middle component. The inclusion in the dictionary of words with these binding morphemes (*s* in this case) improves both the reliability of the decompounding process and its speed.

Decomposition of compound words

The identification of the components of a compound word is the most important step for word verification and for providing spelling aid. The word-decomposition module starts by looking in the dictionary for words which are initial substrings of the compound word. As each word is found, its compounding attributes are checked to make sure that it can be a front component. If not, the word is rejected as a possible component, and the search through the dictionary continues [10]. Once all the possible initial components have been identified, the remaining portion of the compound word is subjected recursively to the same substring-matching procedure against the dictionary, but the compounding attributes must be those of a middle or back component (the latter only if the remaining portion of the compound matches exactly against a word in the dictionary).

Many ambiguous cases (*sun* + *glasses, sung* + *lasses*) will be resolved by the decomposition process on the basis of the compounding attributes found in the dictionary, but some words may have more than one set of acceptable components. Since the process is recursive and requires constant access to the dictionary, the computer time required to decompound a word depends on the degree of branching of the compound word. The degree of branching is proportional to the length of the compound word and to the length of the components in the dictionary. The degree of branching, and execution time, can be reduced by eliminating from the dictionary short words which can be front or middle components and which are frequently found as substrings in many words. Removal of these words from the dictionary is practical for those short words which occur only in combination with a relatively small number of other words; it is accomplished by adding to the dictionary all compound words containing the component.

Juxtaposition is not the only mechanism employed to create compound words; as stated earlier, binding morphemes sometimes occur between the components. By including components with their binding morphemes as lexical entries, decomposition of compound words can be accomplished with the same mechanism. However, in addition to inserting characters, some Germanic languages elide characters at component interfaces during compounding. In general, decomposition of words formed by elision of characters is accomplished by language-specific procedures which are applied at component boundaries when the dictionary look-up fails to find adequate dictionary words. In some languages elisions are defined strictly by the characters before and after the interface (for example, in Norwegian and Swedish, if the component before the junction terminates in two equal consonants and the component after the junction starts with the same consonant, one of these is always elided during compounding). Similar rules apply in German—for example, when the words *Schiff* ('ship') and *Fahrt* ('ride') are combined to form *Schiffahrt* ('navigation').

In some languages elisions have grammatical dependencies. For example, Afrikaans has an elision mechanism for word forms containing the binding morpheme *s* followed by another word starting with *s*. Some of these word forms can occur as stand-alone words when they happen to represent plurals. The decompounding algorithm can cope with such elisions if the appropriate codes are in the dictionary. Once the elision has been recognized, it is merely necessary to indicate that the remaining portion of the compound word starts at the last character of the preceding component, and the normal process is continued.

The identification of the components of a compound word provides a way to verify spelling and also makes it possible to hyphenate properly. Generally it is preferable to hyphenate at the boundary of two components, and if characters have been elided it is necessary to restore them. Thus, *Schiffahrt*, when hyphenated between its components, retains all the *ff*s, i.e., *Schiff-fahrt*. The information derived from word decomposition provides the major break points for the word. These are supplemented with the internal hyphenation points available in the dictionary for each word component.

Description of the general approach

An early prototype for compound-word spelling assistance investigated by the authors consisted in having the user identify the beginning and ending of the misspelled component of a compound word, after which the computer system would provide replacement candidates as for any other isolated word. Upon selection of one of the candidates, the computer system replaced the misspelled component and constructed the correct compound word. The system was cumbersome because it had poor human factors.

The fully automatic version has the advantage of having a single human interface for the spelling-aid mechanism for simple and compound words. The spelling-aid algorithm uses the decompounding algorithm because it needs to identify the components. It proceeds in three phases. First, the "unknown" component of a compound word is identified by locating components that precede or follow the unknown component. Then, spelling aid is invoked to retrieve a list of correctly spelled words from the dictionary which are most similar to the unknown component. Finally, plausible compound words are generated by using the leading components, the candidate replacements from the spelling-aid list, and the trailing components.

More specifically, the first phase starts by looking in the dictionary for words which are initial substrings of the compound word while checking the word attributes for consistency. The algorithm uses language-specific morphological transformations to take into consideration elision patterns at possible component junctions. However, whereas the decompounding algorithm terminates when no more known components are encountered, the aid algorithm goes further. It skips one character of the remaining portion of the compound word and attempts the substring-matching procedure against the dictionary. If this is not successful, another character is skipped and the remaining string is processed again until either a back component is found or there are no more characters to process.

The first phase, thus, isolates a single unknown component preceded and followed by leading and trailing strings which consist of zero or more components. If the leading string has zero components, the unknown component is at the beginning of the compound word; if the trailing string has zero components, the unknown component is at the end of the compound; otherwise, it is embedded within the compound word.

The second phase uses the traditional spelling aid for simple words with the unknown component as an argument. A list of spelling candidates and their corresponding compounding attributes is obtained from the dictionary.

The third phase generates compound words that meet the constraints implied by the compounding flags, and the resulting compound words are then ranked against the input word using a string-similarity measure. A list of compound words ranked according to this measure is presented to the user.

The compound-word spelling-assistance algorithm

This section provides details of the algorithm used to provide spelling assistance for compound words. Some language-specific features for German are included in this algorithm.

Step 1 Examine the input word (for which spelling aid has been requested) to find if the word is correctly spelled. If it is, display a message and exit.

Step 2 Invoke simple spelling aid for the input word, obtaining candidates and their figures of merit. If the figure of merit is within specified limits for at least one candidate (i.e., there is a very good fit between the candidate and the input word), display the candidates and exit.

Step 3 Check the length of the input word for (preset) upper and lower bounds. If the length is outside the range, display simple spelling-aid candidates, if any; if there are none, display a message. Exit.

Step 4 Change the first letter of the input word to uppercase and all other letters to lowercase. From this point forward this will be considered the input word.

Step 5 Examine the input word (now with the first letter in uppercase) to see if it is a valid compound word. If so, put the word into the candidate list and go to the last step.

Step 6 Examine the input word. If it is correctly spelled except for required elisions, make the required elisions, put the word into the candidate list, and go to the last step.

Step 7 Examine the input word without its last letter. If it is correctly spelled, put the word into the candidate list and go to the last step.

Step 8 Examine the input word without its last letter. If it is correctly spelled except for required elisions, make the required elisions, put the word into the candidate list, and go to the last step.

Step 9 Match the input word against the dictionary to obtain all possible initial strings containing one or more valid sequences of components (these are called "frontwords").

Step 10 Select a frontword obtained as a result of Step 9.

Step 11 Obtain all possible terminal strings containing one or more valid sequences of components (these are called "backwords"). This involves skipping characters until a valid verification is achieved on the remainder of the input string.

Step 12 Select the first backword, forming a frontword/backword pair.

Step 13 Invoke simple spelling aid for the characters delimited by the frontword/backword pair (this is the "unknown" component). The candidates obtained from simple-word spelling aid are called "aidwords."

Step 14 If the unknown word is shorter than a preset minimum length and both frontword and backword are not null, concatenate frontword and backword, obtain figure of merit, and post to the candidate list. If either the frontword or the backword is null, go to Step 16. If the unknown word equals or exceeds the minimum length and if no aidwords are found, go to Step 16; otherwise go to the next step.

Step 15 Concatenate the frontword, each aidword, and the backword; evaluate the figure of merit for each; and post to the candidate list.

Step 16 If the unknown word already has an elision letter as its first letter (as defined in Step 17) go to Step 18.

Step 17 Examine the frontword/unknown-word junction for the possibility of an elision. If this possibility exists, it is necessary to restore the elided letter and repeat Steps 13–16 for the modified unknown word. If this possibility does not exist, go to the next step.

Step 18 If there is another backword for this frontword, form a new frontword/backword pair and repeat Steps 13–18; otherwise go to the next step.

Step 19 If there is another frontword, repeat Steps 11–19 for this frontword; otherwise go to the next step.

Step 20 If no candidates have been found and if the first two characters of the input word are identical except for case, delete the second character of the input word and repeat Steps 5–20. (This is done only for the first two characters of the original word for which spelling assistance has been requested; if this has been done once, do not repeat.)

Step 21 Examine all candidates for required elisions and make them as required by the language.

Step 22 If no candidates are available from either simple spelling aid or compound spelling aid, display a message. Otherwise display candidates ranked by figure of merit.

Discussion of results

The spelling-assistance algorithm depends on the performance of its subordinate procedures. It is affected in particular by the decompounding procedure, the spelling-assistance procedure for simple words, and the codes for compounding stored in the dictionary. With regard to the decompounding procedure, both Type-I errors (flagging a correctly spelled word) and Type-II errors (not flagging a misspelled word) have been observed, but the error rate is small (less than one percent).

For the hyphenation function, the results have been extremely satisfactory. The only limitation found was the possibility of multiple decomposition of the compound word. For example, the German string *Staubecken* can be decomposed into *Stau* + *Becken* ('collection basin') or *Staub* + *Ecken* ('dust corners').

Whereas for compound words spelling verification and hyphenation are basically analytical procedures, spelling assistance deals with compound-word synthesis; therefore, syntactic and semantic criteria need to be used to judge the

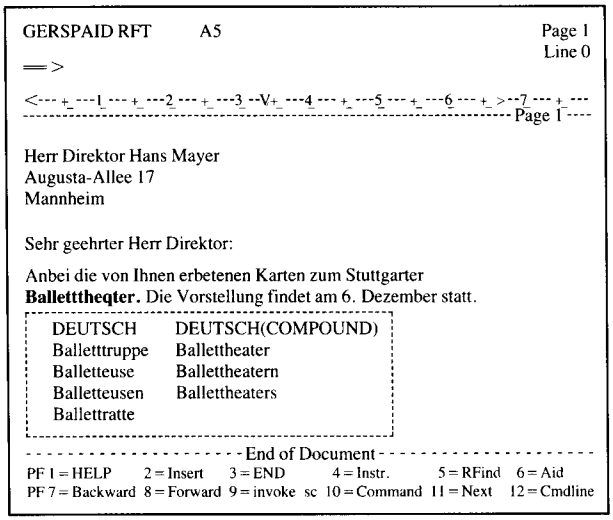
suitability of the generated words. As can be expected, compound-word spelling assistance is less efficient at finding the "correct" replacement candidate than is spelling assistance for simple words; but what is more important, in contrast to the latter, it can generate replacement candidates which are lexically odd or unacceptable (e.g., *churchgoer* and *housewife* are fine but *kitchengoer* and *homewife* are not [9]).

The algorithm was tested against a sample of 227 misspelled German compound words extracted from a large corpus (over 65000 words containing approximately 14000 unique words) of German test documents. Of these misspelled words, 85% had two components, 14% had three components, and 1% had four components. Analysis of the words generated as candidates showed that the results were very dependent on the efficiency of the spelling assistance for simple words (simple aid). If the "correct" candidate was not generated for the "unknown" component of the compound word, it was not possible to generate the correct compound word. The spelling assistance for compound words presented the correct compound in 70% of the test cases. In the remaining cases the correct compound was not in the list (19%) or no candidate list was given because there were no plausible candidates for the misspelled component (11%).

In addition, lexically unacceptable forms were generated when some of the candidates presented by the simple aid had the appropriate compounding attributes in the dictionary but were inappropriate for the context. The results were quantified as follows: If simple aid provided a list with the "correct" candidate X times out of Y, then the compound spelling aid presented the correct candidate approximately as the square of that ratio—i.e., $(X/Y)^2$. The number of lexically unacceptable candidates was relatively large, although not intolerable for German in a word-processing environment. When the candidates had two components, 34% were unacceptable, but 54% were unacceptable when there were more than two components. Overall, 38% of the compound aid candidates were unacceptable.

Further examination of the output candidates also indicated that a significant proportion of the unacceptable words had more components than the original word due to the occurrence of short character sequences common in the language which could also be compound-word components (e.g., German *ges*). However, such candidates generally occurred at the end of the candidate list because they had a worse figure of merit (less similarity to the input word).

The algorithm was modified on the basis of these results. As a first step, a limit was placed on the number of components in a replacement candidate, the number being a function of the length of the compound word. Additional constraints were imposed on the figure of merit required to generate a replacement candidate. Finally, the maximum number of compound candidates presented by the program



Example of combined simple- and compound-word spelling assistance in a word-processing system. The aid candidates are displayed in a window under the misspelled word.

was reduced from six to four. As a result of these improvements, the number of semantically unacceptable candidates was reduced by more than 20% without a practical effect on the efficiency of the compound-word spelling-assistance function.

Conclusion

Implementing a program to provide spelling assistance for misspelled compound words is a complex problem. On the one hand, there are difficult linguistic issues that require semantic resolution, and irregularities in the use of binding characters or in the elisions between components that require sophisticated algorithms and extensive dictionary look-up. On the other hand, the design of practical spelling-assistance programs has to take into consideration the computational efficiency and response time expected by the users. This may involve examination of statistical factors that have some bearing on the problem. For example, when looking for ways of improving program performance, we observed that compound words with more than three components are rare. Thus, in the implementation of the program we made sure that two- and three-component words were handled efficiently.

Another design consideration based on our observations was the identification of the "unknown" component in the compound word. Empirically, we have found that most misspellings have few errors and affect a single component. It happens that such cases are the ones for which an automatic approach has the greatest likelihood of succeeding. We have not found any reliable and efficient

ways of correcting words with more than one incorrect component.

The technique described here is an effective way of providing spelling assistance for on-line word-processing systems. Compound words, because of their length, are misspelled more frequently than simple words. For this reason, compound-word spelling assistance when combined with aid for simple words significantly improves the service to the user (see **Figure 1**). The speed with which the spelling candidates are presented is satisfactory (less than two seconds), but there are some linguistic problems which we are currently addressing. For example, the compounding codes associated with the dictionary entries are not completely effective in preventing invalid associations. This sometimes results in the incorrect identification of components during verification, or, worse, it can result in the generation of compound-word candidates which do not make semantic sense. One way in which we foresee this problem being solved is by including syntactic and semantic features in the dictionary, but this solution will not be an easy one to accomplish. Additional improvements can be achieved by improving the efficiency of the algorithm for simple-word spelling assistance, by studying short words which are frequent substrings of other words, and by carefully editing the compounding attributes for the words.

Acknowledgments

The authors would like to acknowledge the contributions of V. R. Bass, V. A. Bonebrake, J. K. Landis, M. S. Neff, R. J. Urquhart, and S. C. Williams toward the development of a method for verifying the spelling of compound words which provided the basis for this work.

References

1. A. Zamora, "Control of Spelling Errors in Large Data Bases," *The Information Age in Perspective, Proc. ASIS* **15,** 364–367 (1978).
2. A. Zamora, "Automatic Detection and Correction of Spelling Errors in a Large Data Base," *J. ASIS* **31,** No. 1, 51–57 (1980).
3. J. J. Pollock and A. Zamora, "Automatic Spelling Correction in Scientific and Scholarly Text," *Commun. ACM* **27,** No. 4, 358–368 (1984).
4. J. L. Peterson, "Computer Programs for Detecting and Correcting Spelling Errors," *Commun. ACM* **23,** No. 12, 676–687 (1980).
5. D. B. Convis, D. Glickman, and W. S. Rosenbaum, "Instantaneous Alpha Content Prescan Method for Automatic Spelling Error Correction," U.S. Patent 4,355,371, 1983.
6. D. B. Convis, D. Glickman, and W. S. Rosenbaum, "Alpha Content Match Prescan Method for Automatic Spelling Error Correction," U.S. Patent 4,328,561, 1982.
7. R. Lowrance and R. A. Wagner, "An Extension of the String-to-String Correction Problem," *J. ACM* **22,** No. 2, 177–183 (1975).
8. H. Jappinen and M. Ylilammi, "Associative Model of Morphological Analysis: An Empirical Inquiry," *Comput. Linguist.* **12,** No. 4, 257–272 (1986).
9. R. P. Botha, *Morphological Mechanisms. Lexicalist Analysis of Synthetic Compounding*, Language and Communication Library, Volume 7, Pergamon Press, Oxford, 1984.
10. V. R. Bass, V. A. Bonebrake, J. K. Landis, M. S. Neff, R. J. Urquhart, and S. C. Williams, (a) "Compound Word Suitability for Spelling Verification," U.S. Patent 4,672,571, 1987; (b) "Compound Word Spelling Verification," U.S. Patent Application Serial No. 06/664,183, filed October 24, 1984.
11. A. Zamora and R. Frisch, "Method for Verifying Spelling for Compound Words," U.S. Patent Application Serial No. MA9-86-011, filed March 12, 1987.

Received February 13, 1987; accepted for publication August 6, 1987

Rudolf Frisch *Sy Sims School of Business, Yeshiva University, 500 West 185th Street, New York, New York 10033.* Dr. Frisch teaches Management Information Systems and Management Science. In 1987 he retired from IBM, where he was an advisory staff member working on automation of linguistic support functions. He received B.S. degrees in physics and mathematics from São Paulo State University, São Paulo, Brazil, and C.E. and E.E. degrees from Mackenzie College, São Paulo. After working as a civil engineer, he applied his electrical engineering background to transmission optimization of microwave and cable-based networks. From 1968 to 1971 Dr. Frisch taught electrical engineering and computer science at New York University, where he received his doctorate in 1971. He worked as a consultant on minicomputers and data communications, and later joined Lever Brothers, where he worked to improve computer applications in business, manufacturing, and engineering. Dr. Frisch has published on topics in pure and applied sciences as well as in engineering. He has also served as a referee for the *Zeitschrift für Mathematik* and is a member of several professional and honor societies.

Antonio Zamora *IBM Corporation, 10401 Fernwood Road, Bethesda, Maryland 20817.* Mr. Zamora is a senior staff member currently working on the development of natural-language parsers and their application in multilingual word-processing environments. He received a Bachelor's degree in chemistry from the University of Texas and for many years worked in the research department of Chemical Abstracts Service, where he developed computer-based methods for handling structural chemical information and chemical literature. After receiving a Master's degree in computer science from Ohio State University in 1969, Mr. Zamora worked on automatic abstracting, automatic indexing, and automatic classification of documents based on syntactic and semantic attributes of natural language, as well as on spelling-error detection and correction. Mr. Zamora has always had a strong interest in processing natural language with computers; his research work has appeared in numerous publications and received an award from the American Society for Information Science in 1971. He has also served on the advisory board of the *Journal of Chemical Information and Computer Sciences* and is a member of a number of professional societies.