UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ciencias y Humanidades

SISTEMA COMPUTARIZADO PARA LA DETECCION Y CORRECCION DE FALTAS ORTOGRAFICAS

MARTA JULIA FERNANDEZ RAVELO

Guatemala

1986

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ciencias y Humanidades

SISTEMA COMPUTARIZADO PARA LA DETECCION Y CORRECCION DE FALTAS ORTOGRAFICAS

MARTA JULIA FERNANDEZ RAVELO

Modelo de trabajo profesional
presentado para optar al grado académico de
Licenciada en Ciencias de la Computación

Guatemala

1986

Vo. Bo.

(f) O. Cordón

Licenciado Luis Octavio Cordón G.
Asesor

Comité del Trabajo Profesional

(f) Luis R. Furlán

Ingeniero Luis R. Furlán

(f) O. Cordón

Licenciado Luis Octavio Cordón G.

(f)

Licenciado David Alvarez

Fecha de aprobación: 1o. de octubre de 1986.

CONTENIDO

LISTA DE FIGURAS

Página

RESUMEN

En este trabajo se presenta un sistema computarizado para detectar y corregir los errores de ortografía que aparecen en textos. El sistema se implementó en una computadora Hewlett-Packard 3000, utilizando el lenguaje de programación PASCAL.

El proceso se divide en dos etapas:

Detección de un error

El texto que se está revisando debe tener un máximo de ochenta caracteres por línea y debe estar almacenado en código ASCII. Cada palabra en el texto se busca en una lista de palabras frecuentemente usadas y luego en un diccionario. Si no aparece en ninguna de las dos, la palabra se toma como mal escrita.

Corrección del error

A la palabra incorrecta se le aplican cuatro reglas, a manera de formar nuevas palabras que se espera estén en el diccionario. Se pueden obtener cualquiera de los resultados siguientes: ninguna palabra nueva aparece en el diccionario (la corrección es nula), una de las palabras nuevas sí aparece (y se supone que es la correcta), varias palabras aparecen en el diccionario (el usuario escoge cuál es la correcta).

El diccionario de palabras se implementó en una estructura llamada "trie", similar a la de un árbol. Al buscar una palabra, se recorre el "trie" desde la raíz (o sea la primera letra de la palabra) hasta llegar a una hoja (la última letra de la palabra). Si una palabra no está almacenada en el "trie", su búsqueda se interrumpe en un nivel N que corresponde a la N-ésima letra de la palabra. Es en esta letra donde se aplican las reglas siguientes:

1. transposición de las letras en las posiciones N y N+1
2. se sustituye la letra por las otras del alfabeto
3. se elimina la letra
4. se insertan, una a la vez, todas las letras del alfabeto

En teoría se pueden crear 2*N+1 (o sea 1+(N-1)+1+N) palabras nuevas usando un alfabeto de N letras. Cada una de estas palabras se busca en el diccionario.

Para que el lector tenga un conocimiento básico sobre el problema, también se presenta un historial de los estudios realizados y las soluciones planteadas.

I. INTRODUCCION

Durante los últimos años, el aprovechamiento de las computadoras ha aumentado en diversas áreas de aplicación. El "software" favorecido por personas expertas o no en la materia es aquél que puede manejarse sin mayor problema y que facilita la producción de tareas repetitivas. La preparación computarizada de documentos es una de las más atractivas tanto por su utilidad como por su sencillo manejo.

Los procesadores de palabras se han convertido en una herramienta indispensable para casi todos los usuarios de las computadoras. Brindan una ayuda excepcional para ingresar, editar e imprimir textos. Ofrecen tantas facilidades para procesar textos que la dependencia del usuario en esta clase de herramienta ha aumentado.

El propósito del presente trabajo es desarrollar un sistema para detectar y corregir errores ortográficos que aparecen en textos. Está principalmente enfocado hacia los errores mecánicos que se cometen al teclear información en un computador, usando un procesador de palabras u otro "software" similar.

Los errores de ortografía siempre han constituido un problema en el momento de escribir. Aún escribiendo con cuidado, difícilmente no se cometen errores. Buscar y

corregirlos forma parte integral de la preparación de documentos. Es un paso que suele ser largo y aburrido. No es sorprendente, pues, que este tipo de contratiempo sea motivo suficiente para encontrarle soluciones factibles.

Algunos errores ortográficos se deben a la ignorancia de la persona que escribe el texto, pues puede que ella, simple y sencillamente, desconozca cómo se escribe correctamente una u otra palabra. Por otro lado, es posible que la pronunciación de una palabra le cause confusión, pues no establece de ninguna manera la ortografía de la misma. Por ejemplo, en el idioma español hablado por los guatemaltecos, las letras 'v' y 'b', 'c' y 's' se pronuncian igual o lo bastante parecido como para que se utilicen indistintamente a la hora de transcribir su sonido. Errores comunes son escribir 'seción' por 'sesión', 'almoada' por 'almohada',etc.

El teclado de una terminal es uno de los medios más divulgados para introducir información al computador. Es de especial interés la manera en que éste se opere ya que su uso puede conducir a que se cometan errores ortográficos al ingresarse la información. La posición de las teclas, o el ordenamiento de los símbolos que ellas representan, determina la colocación de los dedos en las mismas. El movimiento de los dedos sobre el teclado debe ejecutarse con la mayor exactitud posible a manera de transcribir correctamente el

texto.* Si el operador de la terminal coloca mal los dedos sobre el teclado, obviamente sin darse cuenta de ello, no va a transmitir al computador la información que desea. Esta situación puede suceder, por ejemplo, cuando no coloca correctamente los dedos sobre las teclas guías 'A', 'S', 'D', 'F', y ';' ('Ñ' en el teclado español), 'L', 'K', 'J'. También puede ser que no presione bien las teclas al tratar de escribir rápido o al no leer con atención el texto que transcribe. Por ejemplo, si no presiona la tecla 'S' con la suficiente fuerza puede que escriba la palabra 'fraco' en vez de 'frasco', o en caso la presione por un tiempo relativamente más largo termine escribiendo 'frassco'.

El método que se presenta en este trabajo es el sistema "DCO". No pretende corregir el 100% de los errores ortográficos, ni mucho menos sustituir al usuario en el proceso de la revisión ortográfica de un texto. El objetivo que se persigue es proporcionar una ayuda efectiva al usuario a manera que él pueda prestar mayor atención a aspectos más relevantes como lo es el contenido del documento.

Aquellas personas que sin afán de lucro deseen tener a la vista el programa fuente del sistema "DCO", pueden solicitarlo a su autora a través de la Universidad del Valle de Guatemala.

---

*Según un estudio de F. L. Van Nes [20], "Analysis of keying errors", Ergonomics (Vol. 19, No. 2, 1976), el 40% de los errores en un texto se originan durante el tecleo del mismo. Y un 20% de los errores ortográficos se originan al presionar una tecla vecina.

II. BREVE RESEÑA HISTORICA

En las dos siguientes secciones se exponen algunos métodos para detectar y corregir errores que muestran la variedad de técnicas computacionales que se han desarrollado. A continuación se presenta un historial breve de métodos y estudios que han surgido para hacerle frente a diferentes aspectos del problema.

Uno de los primeros estudios que se realizaron fue la clasificación de errores ortográficos que hizo A. I. Gates [20] en 1937. Dividió los errores en cuatro clases:

1. sustitución de letras
2. transposición de letras
3. adición de letras extras
4. omisión de letras

Este estudio es quizás el más importante por su originalidad y por el hecho de que esta clasificación representa acertadamente los tipos de errores que suelen cometerse. Ha servido de base a trabajos posteriores que tratan sobre la corrección de estos errores.

La información que se maneja en numerosas aplicaciones debe ser lo más exacta a manera de que los errores en ella no interfieran en la ejecución de los procesos. La participación del ser humano en el ingreso de datos es una

fuente de error que ha sido objeto de estudio para encontrarle solución.

Algunas aplicaciones requieren que el usuario del computador ingrese una cantidad considerable de instrucciones. Tal es el caso de un sistema operativo, un editor de palabras o un manejador de bases de datos [7,11,20], en donde se mantiene un diálogo constante entre el usuario y la máquina. La detección de errores sintácticos en las instrucciones está bajo la responsabilidad de los programas de aplicación en sí. Muchos de estos errores pueden considerarse de carácter ortográfico. La implementación de un método para corregirlos es inmediata y relativamente fácil porque aparecen en un medio con vocabulario limitado y sintaxis rigurosa.

La generación de módulos de corrección en la fase de la compilación de lenguajes de programación siempre ha sido de interés especial [11,16]. N. Freeman [16] en 1963 realizó el primer trabajo para corregir automáticamente, durante la compilación de un programa, los errores ortográficos en las instrucciones del mismo. Específicamente diseñó el sistema para trabajar con código escrito en el lenguaje CORC (Cornell Computer Language).

Los datos almacenados en un computador usualmente están sujetos a consultas. El programa de aplicación que los maneja debe poder enfrentar la posibilidad de recuperar información mal ingresada. L. Davidson [5] presentó el caso

real de una línea aérea. La información de cada pasajero está en el computador, y se recupera en base al nombre de la persona. Lo especial del problema está en que los nombres pueden escribirse mal al ingresarlos por primera vez a la máquina, o a la hora de ser consultados. Un estudio más sencillo lo realizó C. Blair [3], quien presentó un método para recuperar 117 palabras aisladas, tomando en cuenta que pueden tener errores ortográficos. Otras personas han investigado los errores que aparecen en bases de datos, principalmente las que almacenan datos bibliográficos o genealógicos [20].

En las últimas dos décadas, el problema de los errores de ortografía ha recibido mayor interés especialmente cuando aparecen en textos corrientes. Se han desarrollado varios métodos para detectar y corregirlos; también han sido estudiados los tipos de errores que suelen cometerse y la frecuencia en que aparecen. El primer programa escrito más como una aplicación que como una investigación es SPELL [18]. En la actualidad, la mayoría de los procesadores de palabras incluyen un subsistema para corregir los errores de ortografía [13].

De igual forma han sido foco de atención los errores producidos por máquinas especiales que "leen" o "escuchan" textos escritos o hablados para almacenarlos en sus memorias. Estas máquinas tratan de identificar un caracter o sonido

comparándolo con una determinada forma de símbolo o sonido que ya "conocen" [26]. En el caso concreto de las máquinas de reconocimiento óptico de caracteres,* se reconocen palabras escritas a mano o a máquina. Naturalmente estas máquinas están limitadas a reconocer pocas figuras de caracteres. Si no reconoce bien un símbolo debido a una distorsión en él, la máquina lo asociará a la forma que más se le parezca. En este caso se puede cometer un error ortográfico al asociar el símbolo a una forma equivocada [26]. Los errores en textos procesados por estas máquinas se deben casi exclusivamente al sustituir una letra por otra [20].

Herbert T. Glantz [16,20] escribió en 1957 uno de los primeros artículos sobre la corrección de ortografía, enfocando el problema específicamente a las máquinas de reconocimiento óptico de caracteres. Su interés se centró en aparear de uno en uno los caracteres del texto original con los que produce la máquina al ir leyendo el texto. El proceso de apareamiento debe hacerse lo más exacto posible, a fin de no cometer errores al aparear equivocadamente una letra con otra. Si en el texto original aparecen letras unidas, la máquina debe tratar de separarlas. Srihari [26] menciona un método que detecta los errores causados por la mala separación de letras. T. Kohonen y E. Reuhkala [20] presentaron en 1978 un método para corregir textos producidos por medio del reconocimiento de la voz.

*"optical character recognition" (OCR), en inglés

III. METODOS DE DETECCION DE ERRORES ORTOGRAFICOS

El problema de poder encontrar un error ortográfico en una frase u oración reside en cómo decidir cuáles son las palabras mal escritas. Cada palabra es una cadena o secuencia de letras. No necesariamente todas las secuencias de letras constituyen palabras correctas.

El objetivo principal en el desarrollo de un método de detección es la minimización de los siguientes dos errores:

1. indicar que una palabra bien escrita tiene un error ortográfico, y
2. tomar una palabra mal escrita como buena.

También es importante considerar el volumen de palabras que normalmente se procesará, y si los textos se revisarán interactivamente o en tandas.

Un método sencillo para ayudar al usuario a detectar posibles errores ortográficos en un texto es el siguiente: Conforme el computador lee el texto se construye una lista con todas las palabras, sin repetirlas, que aparezcan en él. El usuario debe estudiar la lista y decidir cuáles palabras están mal escritas. Como se podrá suponer, los errores que aparezcan en la lista debido a la falta de conocimiento del usuario no se captarán fácilmente, si es la misma persona quien revisa la lista.

La revisión de la lista se puede mejorar si el computador la ordena alfabéticamente o por el número de veces en que aparecen las palabras en el texto. La ventaja que ofrece la segunda alternativa es que la mayoría de los errores mecanográficos no se repiten mucho [19]. Se espera que la mayoría de los errores aparezcan al principio o al final de esta lista según el orden, ascendente o descendente, que se haya efectuado en la misma. Sin embargo, este método puede resultar inadecuado pues el número de veces que se escribe mal una palabra determina su frecuencia [23], la que a su vez determina la posición de la palabra dentro de la lista.

El programa TYPO [18] produce una lista de palabras distintas similar a la del método anterior, pero ordenada por un índice de peculiaridad. Este índice mide la frecuencia de los diferentes "trigramas" que aparecen en un texto.

Un "trigrama" es una secuencia de tres letras; un "digrama", de dos. En un alfabeto de N letras pueden haber N*N diferentes cadenas de dos letras y N*N*N cadenas de tres. Para los propósitos del método de detección, se debe tener en cuenta que no todas las combinaciones de letras ocurren en el lenguaje usual. Por ejemplo, 'bdg' no aparece en ningún vocablo del idioma español. En cambio el "trigrama" 'cui' es correcto en la palabra 'cuidado'; pero si 'ciudad' aparece escrita como 'cuidad', la secuencia 'cui' no es válida. Es decir que las cadenas no tienen validez absoluta, pudiendo aparecer correctamente en una palabra, erróneamente en otra o

nunca ocurrir. El número de veces que aparece cada "digrama" o "trigrama" en los textos varía, pero en general se mantiene bajo. Peterson [18] menciona que en una muestra tomada de un texto, sólo aparecieron un 70% de todos los posibles "digramas" y un 25% de los "trigramas" (utilizando un alfabeto de 28 letras). En base a estas observaciones se puede deducir que si una palabra contiene una cadena de dos o tres letras que aparece pocas veces, es muy probable que esté mal escrita.

El programa TYPO, como se señaló anteriormente, calcula un índice para cada "trigrama" en el texto. Este valor representa la probabilidad de que el "trigrama" en una palabra junto con el resto del texto provengan de la misma fuente. En cierto modo puede decirse que mide la pertenencia de la palabra dentro del contexto del documento. El índice total que le corresponde a la palabra depende de los índices de los "trigramas" que la componen. En base a las listas producidas por este programa, se ha observado que las palabras mal escritas tienden a obtener un alto índice de peculiaridad.

La cantidad de palabras que aparecen en las listas producidas por este tipo de programas puede reducirse si se excluyen aquellas palabras que con certeza se sabe que están bien escritas. Para escoger automáticamente cuáles están correctas, se comparan contra un diccionario de palabras almacenado en el computador. Si la palabra en cuestión

aparece en el diccionario, entonces se asume que está bien escrita y se elimina de la lista. Si la lista con todas las palabras del texto se ordena en la misma forma que el diccionario, la búsqueda en el diccionario es más rápida. En 1975 Steve C. Johnson [1] escribió una versión del programa SPELL (ver sección anterior) que utiliza un diccionario de palabras conforme al siguiente algoritmo:

1. Se convierten todas las letras mayúsculas del texto a minúsculas; se forma una lista de palabras tomando los espacios en blanco como marcas que separan una palabra de otra.
2. La lista se ordena alfabéticamente.
3. Las palabras duplicadas se remueven.
4. Las palabras que quedan en la lista se buscan en el diccionario.
5. Se produce una nueva lista con las palabras que no aparecen en el diccionario.

La mayoría de los algoritmos que se han desarrollado para la detección automática de errores ortográficos usa un diccionario en manera similar a la anterior. El diccionario es básicamente una lista ordenada de palabras correctamente escritas. La diferencia principal entre uno y otro algoritmo reside en la forma en que se representan y buscan las palabras en el diccionario. Algunas opciones [20] que más adelante se exponen son:

- almacenar letra por letra cada palabra

- representar las palabras como cadenas de "N" letras ("digramas" por ejemplo) o como patrones de "bits"
- utilizar códigos de "hash" para el direccionamiento o la representación de las palabras
- recurrir al análisis de afijos de las palabras
- representar el diccionario usando una estructura de árbol u otra similar como la estructura de un "trie"

El almacenar en el computador cada una de las letras de la palabra tiene la ventaja de proporcionar una completa seguridad de que la palabra esté o no en el diccionario. En cambio si se abrevia o codifica la palabra se obtiene un ahorro considerable de memoria, pero sólo existe una gran posibilidad de que la palabra esté en el diccionario. Esto obedece a que los códigos, por más que se trate de evitar, no son del todo únicos para cada palabra. El número de veces que esto ocurra dependerá de la eficiencia del método de codificación.* El método que propone Blair [3] reduce cada palabra de cuatro o más letras a cuatro letras. Si dos palabras resultan con abreviaturas iguales se abrevian con

*En algunas aplicaciones no es de gran importancia la duplicación de códigos. En el caso de agregar una palabra al diccionario no es mayor problema. Sin embargo, a la hora de averiguar si la palabra está o no en el diccionario no se sabe realmente cuál de las palabras con el mismo código es la que se almacenó.

cinco letras, y el número de letras se sigue aumentando hasta que las abreviaturas sean únicas.

Davidson [5] desarrolló un método para recuperar la información de determinada persona. Con el nombre de la persona se forma una clave de cinco letras. Si existen varios registros en la computadora con la misma clave, se verifica otro tipo de información en los registros, como el número telefónico. Esta revisión ya no se lleva a cabo automáticamente, sino que se hace con la ayuda del usuario.

Damerau [4] presenta una técnica que facilita la búsqueda de palabras en un diccionario. Cada palabra se representa en un registro de 28 "bits". Los primeros 26 "bits" representan cada una de las 26 letras del alfabeto; los otros dos "bits" son para indicar la presencia de números y signos especiales, como los de puntuación, que forman parte de la palabra. Cada letra o símbolo que aparezca en la palabra se representa en este registro encendiendo el "bit" correspondiente. Antes de comparar dos palabras (la del diccionario y la que se está revisando) letra por letra,

1. se comparan sus registros respectivos para ver si son iguales (con el fin de averiguar si por lo menos las palabras contienen los mismos caracteres),
2. se revisa que contengan el mismo número de caracteres.

El propósito de comparar primero los registros es para luego sólo comparar detalladamente aquellas palabras que tengan suficientes características en común.

Otra manera para codificar las palabras se basa en el hecho de que las palabras, en un diccionario en orden alfabético, empiezan con las mismas letras que sus palabras predecesoras. La secuencia de letras que se tiene en común, empezando con la primera letra, se reemplaza por el número de letras en la secuencia [24]. Por ejemplo, la serie de palabras 'fuerte', 'fuerza', 'fuga', 'fugaz', aparecería como: 'fuerte', '4za', '2ga', '4z'. El tamaño físico del diccionario se reduce en un 50%. No hay necesidad de procesar en forma especial los prefijos y sufijos de las palabras, como se verá a continuación.

El diccionario también puede construirse si se quitan los afijos de las palabras y se almacenan en él solamente las raíces [2,19,20,23]. Por ejemplo, 'casas' no aparecería en el diccionario sino que sólo 'casa', y a la hora de buscarla se buscaría sólo la raíz 'casa'. La normalización de afijos permite una reducción considerable en el tamaño del diccionario (hasta un 15% en un diccionario en inglés [20]) y aumenta el número de palabras que el diccionario pueda detectar como correctas sin aumentar su tamaño físico. La desventaja que tiene este tipo de análisis es que no se pueden detectar los errores ortográficos en los prefijos o sufijos de las palabras. Una posible solución a este

problema es almacenar junto a la raíz los prefijos y sufijos que se pueden usar con esa raíz. Al buscar una palabra, se busca sólo su raíz y luego se revisa si el prefijo o sufijo es el correcto. Las reglas para desmenuzar la palabra en su raíz y sus posibles afijos son complejas y tienen excepciones. El ahorro en el almacenamiento del diccionario puede ser contrarrestado por el tiempo que tome identificar los afijos [19]. Una idea para mejorar la eficiencia de este método es incluir en el diccionario tantas variantes de las palabras como se necesite y normalizar sólo las palabras poco comunes.

En el caso del programa SPELL [18] antes mencionado, se utiliza una tabla encadenada de "hash". La función "hash" que se aplica a cada palabra es: (L1 * 26 + L2) * 10 + MIN(LG-2,9), donde L1 y L2 son los códigos numéricos de la primera y segunda letra de la palabra, LG es su longitud (el número de letras) y MIN es la función MINIMO. En cada fila de la tabla hay un apuntador que señala a una determinada cadena. Esta cadena está formada por todas las palabras a las que les corresponde el mismo valor de la función "hash". De este modo, las palabras que pertenecen a una cadena tienen en común las dos primeras letras y tienen igual longitud.

Para almacenar el diccionario, Nix [17] emplea otra técnica de "hash". Suponiendo que un diccionario consta de 1,000 palabras, éste se representa en una tabla T de 20,000 "bits", en donde cada palabra se representa por una serie de

diez "bits". Para procesar una palabra W en la tabla T, se calculan diez funciones "hash": H1(W),H2(W), ... H10(W), independientes una de la otra y que dependen directamente de las letras y el orden en que aparecen en la palabra W. Así, a cada palabra se le asignan diez valores entre 1 y 20,000. Para agregar la palabra W al diccionario deben encenderse diez "bits": T[H1(W)],T[H2(W)], ... T[H10(W)]. En forma similar, una palabra se busca en la tabla revisando que los diez "bits" respectivos estén encendidos. Ocasionalmente (el 0.1% de las veces) el método se equivoca al indicar que una palabra W está en la tabla cuando realmente no lo está.

Los métodos de "hashing" no almacenan las palabras letra por letra sino que en una forma más compacta. Ofrecen una alta probabilidad de que los códigos sean únicos para cada palabra.

Otra idea de "hashing" [2] usa una tabla de 2^27 "bits".* Si a la palabra 'mesa' le corresponde el "bit" 6, a 'volar' el 11, a 'triste' el 14, a 'cemento' el 19 y a 'pelota' el 23, entonces la tabla se representa con una lista de números que corresponden a los "bits" encendidos: 6,11,14,19,23. Una palabra W está en la tabla sólo si el valor de la función "hash" H(W) está presente en la lista de números de los "bits" encendidos. Este criterio es arriesgado pero con resultados bastante aceptables, tomando en cuenta que el número de colisiones (el número de palabras a las que les

*El símbolo '^' indica una operación de exponenciación: 2^3 equivale a dos elevado al cubo.

toca el mismo valor de la función H) es reducido en una tabla tan grande. El tamaño de la tabla implica que las direcciones de los "bits" pueden fluctuar entre 0 y 2^27 y que no todos los "bits" estarán encendidos. Para utilizar números más pequeños en la representación de los "bits" encendidos, se usan las diferencias entre los valores sucesivos (deben estar ordenados del "bit" menor al mayor) convirtiéndose la lista anterior en: 6,5,3,5,4. La búsqueda de una palabra se convierte en una búsqueda secuencial en esta lista hasta sumar al valor H(W).

Aún otra modalidad para usar funciones "hash" es la siguiente [6]. El diccionario contiene N palabras. Cada una se almacena en L "bits" en una tabla de M nodos. Para cada palabra K se producen tres valores "hash" H1(K), H2(K) y H3(K). Para insertar una palabra en la tabla se busca un nodo vacío según la secuencia [H1(K) + N * H2(K)] mod M, con N=0...M-1. En el nodo vacío se almacena el valor H3(K), representado en L "bits". Para buscar una palabra se repite la misma secuencia hasta encontrar el nodo con el valor H3(K) respectivo.

James [11] representa el diccionario en un árbol, cuya forma refleja específicamente la sintaxis del lenguaje FORTRAN. La estructura del árbol no es estática sino que cambia según la frecuencia con que se usen las palabras claves del lenguaje. Las palabras en las instrucciones más

usadas se van moviendo hacia la raíz del árbol, a manera de optimizar la búsqueda de las palabras claves.

Otra estructura de datos que se puede utilizar para almacenar el diccionario es un "trie" [18,26], similar a la estructura del árbol. Tomando en cuenta que cada letra de una palabra puede ser cualquiera de las N letras de un alfabeto, cada nodo del "trie" se bifurca en N nodos. Los caminos que van de la raíz del "trie" a los nodos hijos representan las primeras letras de todas las palabras. Los caminos que salen de estos nodos a sus propios nodos hijos representan las segundas letras de las palabras, y así sucesivamente hasta llegar a la última letra de las palabras: los nodos sin hijos. La búsqueda de una palabra de N letras hace necesario que se recorra el "trie" en N niveles.

La estrategia de la búsqueda puede realizarse en varias etapas a manera de hacerla más eficiente. Por ejemplo, pueden haber dos listas (o diccionarios) de palabras totalmente independientes la una de la otra en lo que se refiere a contenido: una pequeña con las palabras más comunes del idioma, y la otra con un volumen considerablemente mayor de palabras. Toda palabra en el texto se busca primero en la lista pequeña. Si no aparece en ella, entonces se busca en la otra.

También pueden combinarse distintas estrategias para detectar los errores. Un método auxiliar al principal podría ser el siguiente: seleccionar como bien escritas todas las

palabras que aparecen más de N (usualmente dos) veces [23]. Puede que en textos muy grandes esta estrategia falle, pues hay mayor posibilidad de que un error se repita muchas veces. Pero si este método se usa en textos pequeños puede que brinde buenos resultados. Es difícil que en un texto corto se repitan los mismos errores de ortografía.

IV. METODOS DE CORRECCION DE ERRORES ORTOGRAFICOS

El método de corrección debe proporcionar la forma correcta de la palabra mal escrita o, en forma alternativa, la o las formas que más se le parezcan en base a algún criterio de similitud.

Los trabajos realizados para corregir la ortografía de las palabras imponen algunas restricciones en el tipo de error que procesan con éxito. Esto se debe a la variedad de los tipos de errores ortográficos que pueden aparecer en un texto, siendo imposible que un método pueda cubrirlos todos. Aún así, estas técnicas proporcionan resultados bastante aceptables. En general, establecen una similitud entre la palabra incorrecta y una o algunas de las palabras de un diccionario. El encontrar una única palabra similar a la incorrecta y que sea la forma correcta determina la eficiencia del método de corrección. Los métodos presentados en esta sección analizan las palabras como secuencias ordenadas de caracteres, olvidándose del significado de las mismas.

Un impulso inicial para corregir las palabras es programar las reglas de ortografía del idioma. Llevar a cabo este procedimiento es costoso, pues basta pensar en el número de excepciones que se dan para la mayoría de las reglas y en

que, además, se podría usar solamente en textos escritos en el idioma para el cual se hizo.

Otra solución al problema es que, para cada una de las palabras en el diccionario, se almacenen las formas mal escritas que suelen ocurrir y que no dan lugar a ambigüedades. Por ejemplo, si la palabra 'entonses' suele aparecer frecuentemente y sólo como un error de la palabra 'entonces', puede transformarse con certeza a 'entonces' cada vez que se encuentre. Este proceso ofrece resultados óptimos a largo plazo. La elaboración, mantenimiento y operación de semejante diccionario son muy trabajosos; y su tamaño es considerablemente mayor. Debe llevarse un control de los errores que se cometen frecuentemente y verificar que estos errores se deriven siempre de la misma palabra. Se ha calculado que este método puede corregir un 10% de todos los errores [20,23]. Este porcentaje tan bajo se debe a que la mayoría de los errores no se repiten en textos de tamaño normal [22,23], y a que constantemente aparecen nuevos errores.

La mayoría de los métodos codifican las palabras para encontrar la palabra similar en un diccionario, la cual se toma como la correcta. Si el código de una palabra mal escrita es igual al de una correcta, entonces se asume que ésta última es la forma correcta.

La idea original de trabajar con códigos de palabras proviene del método SOUNDEX [3,8]. Este método sirve para archivar documentos de personas en base a la pronunciación de sus nombres. El código se forma con la primera letra del nombre y un número de tres dígitos; este número depende de la pronunciación del nombre y el orden en que aparecen las consonantes. Las codificaciones de este tipo son ideales para corregir errores de origen fonético, como corregir "colejio" a "colegio" pues a estas dos palabras les corresponde el mismo código fonético.

Lo importante en el método de codificación es que debe retener únicamente la información necesaria en forma concisa. Así se logra mantener una relación cercana con la palabra original, y el código no se modifica fácilmente por algún error de ortografía que haya en ella. Una característica ideal del código es que sea único y propio para cada palabra. La codificación debe basarse en la identidad de la palabra y en la interrelación de las letras que la componen.

Blair [3] presenta un método de corrección donde utiliza una lista de palabras y sus códigos. La abreviatura (el código) se forma tomando N número de letras con el mayor peso en la palabra. El peso depende de:

(1) la letra que es (las consonantes tienen mayor importancia que las vocales); y

(2) la posición de la letra dentro de la palabra (la primera letra siempre se retiene en la abreviatura).

En el caso de que a dos palabras en la lista les toque abreviaturas idénticas, se forma otra abreviatura para cada una utilizando ahora N+1 letras. Mientras N sea mayor, menos serán las veces en que dos o más palabras tengan la misma abreviatura; sin embargo,

(1) ocupan más espacio de almacenamiento,

(2) toma más tiempo compararlas

(3) son más sensibles a cambiar por un error de ortografía.

En el ejemplo que proporciona Blair, las abreviaturas eran de cuatro letras para trabajar con una lista de 117 palabras correctas; en uno o dos casos aparecieron palabras con la misma abreviatura. El método de corrección funciona así:

1. Se busca en la lista la palabra cuya abreviatura es igual al de la palabra mal escrita.
2. Si se encuentra, se asume que esa palabra es la correcta; si no la encuentra, no se hace nada.

Este método falla si el error de ortografía está en una de las letras que forman la abreviatura, pues la asociará con otra palabra. Sin embargo, es efectivo cuando el error se comete en una letra que no está en la abreviatura.

Para buscar el registro de una persona en una lista de pasajeros Davidson [5] usa, en forma similar a la anterior, abreviaturas para los nombres (ver sección anterior). La diferencia más notoria con el método de Blair consiste en que, al no encontrar en la lista de pasajeros el nombre que corresponde a la abreviatura que se busca, calcula un punteo de cero a cuatro. El punteo se basa en la similitud de las letras entre la abreviatura del nombre que se busca y la abreviatura del nombre que está en la lista, tomando en cuenta las letras que no coinciden. Un punteo de cero indica que las abreviaturas no tienen nada en común y un punteo de cuatro corresponde a abreviaturas iguales. Se escogen los registros de los nombres cuyas abreviaturas tienen el punteo más alto.

Otra manera para encontrar la palabra correcta es invirtiendo el proceso a través del cual se cometió el error de ortografía, partiendo de la palabra mal escrita para llegar a la correcta. Lowerance y Wagner (1975) [8,19] propusieron un algoritmo de orden M x N para calcular la "distancia" entre dos palabras de M y N longitudes cada una. La "distancia" es una medida de la similitud entre las dos palabras. Se mide en base al número de operaciones necesarias para transformar una palabra a otra. El valor que se calcula en este algoritmo corresponde a la probabilidad de que una palabra sea la forma correcta de otra. La palabra con el índice más alto se selecciona como la correcta.

En 1937, A. I. Gates [21] clasificó los errores de ortografía en cuatro grupos, basándose en los que suelen aparecer en un texto [20,22].

1. Sustitución de una letra por otra (15-20% de todos los errores)
2. Transposición de dos letras adyacentes (10-15%)
3. Adición de una letra (25-35%)
4. Omisión de una letra (30-40%)

Con esta clasificación, varios métodos [4,7,16,19] proceden a corregir un error suponiendo que éste se creó en la palabra correcta de un diccionario. La idea es transformar la palabra en duda a la palabra correcta aplicando una sola operación de letras.

1. Si dos palabras tienen el mismo número de letras y difieren en una sola posición se asume que son iguales. En este caso se cometió un error de ortografía al haber sustituido una letra por otra, por ejemplo escribir 'alfebeto' en vez de 'alfabeto'.
2. Si las palabras son de la misma longitud y difieren en dos posiciones adyacentes, estas dos letras se intercambian entre sí. Si ahora son iguales, el error se debió a la transposición de estas dos letras. Así, 'panfelto' se transforma en 'panfleto'. Si después del intercambio de las dos letras, las

palabras aún no son iguales, no se hace nada al respecto.

3. Si la palabra incorrecta tiene una letra de más, la longitud de la palabra del diccionario es M+N, y tanto las primeras M letras como las últimas N letras de ambas palabras son iguales, se descarta la letra M+1 en la palabra incorrecta. Por ejemplo, si se tiene 'coritina', y 'cortina' está en el diccionario, la única diferencia entre ellas está en la cuarta letra y si se elimina, las dos palabras ya son iguales.

4. Si la palabra del diccionario tiene una letra de más y se diferencia en una sola letra con la palabra incorrecta, se elimina esta letra. Así, si se busca 'frazda' y en el diccionario está 'frazada', se elimina la quinta letra de 'frazada' para resultar con 'frazda'. Como ahora las dos cadenas ya son iguales se asume que 'frazada' se escribió incorrectamente como 'frazda'.

Con estas cuatro operaciones pueden rehacerse los pasos que probablemente se tomaron para cometer el error. Por ejemplo, al escribir 'caudermo' puede decirse que:

1. se transpusieron las letras 'a' y 'u'
2. se sustituyó la 'm' por la 'n'

y entonces se obtiene una nueva palabra: 'cuaderno', que posiblemente es la correcta. Debe quedar claro que estas operaciones en las letras para tratar de invertir el error no proporcionan un único camino para llegar a la palabra correcta. Su utilidad se debe más que nada a la correspondencia [19] que tienen con las operaciones que ocurren durante el tecleo del texto.

Un método de corrección a emplearse en compiladores o en sistemas operativos [16] tiene la ventaja de contar con:

1. un número limitado de palabras en el vocabulario,
2. reglas rigurosas de sintaxis que fijan el uso de palabras claves, y la semántica que maneja las palabras definidas por el usuario,
3. tablas de símbolos donde se guardan las palabras claves y las variables definidas por el usuario; estas tablas reemplazan a la lista de palabras o al diccionario que se usa en otros métodos de aplicación más general.

El propósito de implementar un sistema de esta naturaleza es el mantener una comunicación flexible entre el usuario y la máquina. Freeman [16] presenta una técnica para corregir los errores ortográficos que aparecen en los programas fuentes escritos en el lenguaje CORC (Cornell Computer Language). Se estima la probabilidad de que una palabra con un posible error de ortografía sea realmente otra palabra que se sabe está correcta. La probabilidad se calcula en base a:

a. el número de letras que tienen iguales las dos palabras
b. el número de letras que tienen iguales las dos palabras después de transponerles o sustituirles letras, y
c. el número de letras que tienen iguales las dos palabras si no se hace distinción alguna entre mayúsculas y minúsculas.

Este método tiene el defecto de que deben calcularse un gran número de probabilidades para llevar a cabo una corrección.

El empleo de las probabilidades ha sido de gran utilidad en el desarrollo de otros métodos de corrección. Srihari [26] presenta un algoritmo que corrige específicamente errores cometidos al sustituir unas letras por otras, especialmente en textos producidos por máquinas de reconocimiento óptico de caracteres (ver Sección II). El algoritmo propuesto integra el uso de las probabilidades para trabajar la palabra de letra en letra y el uso de un diccionario donde se procesa la palabra como una unidad. Las probabilidades se manejan así:

1. Tomando en consideración el conjunto de letras que aparecen en el texto antes de determinada letra, se calcula la probabilidad que ésta tiene para que ocurra en ese lugar del texto.
2. Conociendo las características de la fuente por la cual se introduce el error ortográfico, se trabaja con una tabla de probabilidades de

confusión. Las máquinas de reconocimiento óptico de caracteres tienden a producir errores ortográficos de sustitución. Se calcula la probabilidad de que la máquina asigne una letra por otra, posiblemente la misma y entonces no se comete ningún error, debido a la ambigüedad en la forma de la letra.

Uno de los métodos de corrección más completos es SPEEDCOP [19,20,21,22,23]. En este proyecto se diseñó un algoritmo para corregir errores en las palabras de textos científicos y técnicos almacenados en siete bases de datos. La palabra en duda se busca en un diccionario de errores comunes. Si no se encuentra, se genera una llave de similitud poco propensa a cambiar por error. Se ordenan las consonantes de la palabra según el orden de la lista 'JKQXZVWYBFMGPDHCLNTSR', donde 'J' es la letra que menos tiende a ser omitida y 'R' la que más. Las vocales de la palabra se agregan a la llave según el orden en que aparecen (a 'molécula' le corresponde la llave 'MCLOEUA'). Con esta llave se buscan llaves similares en un diccionario usando el método para invertir el proceso por el cual se cometió el error. (Se aplican las cuatro operaciones de error que Gates [21] clasificó.) Si se encuentran dos o más palabras similares, se escoge una de ellas usando una rutina que normaliza sufijos, y otra rutina donde se estudia la posibilidad de que la palabra es realmente la concatenación de una proposición con un artículo ('delos' a 'de los').

Este tipo de error corresponde a sólo un 1 o 2% del total de errores, pero la rutina que los capta proporciona resultados bastante exactos.

V. METODO EMPLEADO

Una característica muy importante del método que se presenta a continuación es su aplicación a cualquier texto legible almacenado en un computador. Dada la variedad de símbolos que pueden aparecer en el texto, el método define una palabra como cualquier secuencia de hasta veinte letras continuas, en mayúsculas o minúsculas que pertenezcan al conjunto {'A'...'Z','a'...'z'}, y que en los extremos están limitadas por cualquier otro caracter. La secuencia puede estar separada por un guión sólo si la primera parte junto con el guión está al final de una línea del texto, y la otra parte está al principio de la línea siguiente.

El método revisa la ortografía de cada palabra que encuentre en el texto, comparándola contra una lista de palabras correctamente escritas. Este proceso es el más práctico para detectar un posible error ortográfico. Si la palabra está en la lista, se toma como buena. Si no, se considera como mal escrita. El método señala al usuario cada error que detecte y le presenta una lista con las palabras que pueden ser la forma correcta. A continuación se detalla este proceso.

Primero se busca la palabra en una lista, relativamente pequeña, de palabras que suelen aparecer frecuentemente en

los textos. Cada palabra de la lista se almacena literalmente y se lleva un control del número de veces que el método la ha encontrado hasta en ese momento (en lo que va del texto y en textos anteriores). La posición de cada palabra en la tabla (o lista) se calcula según la función P=(L1+L2+LONG-197) mod N, donde L1 y L2 son las dos primeras letras de la palabra, LONG el número de letras que tiene, y N un número primo que corresponde al máximo número de filas en la tabla. Siguiendo la definición de esta función, a todas las palabras que empiezan con las mismas dos letras y que tienen igual número de letras les corresponde la misma posición en la tabla. Viendo la necesidad de tomar en cuenta estas colisiones, se dispone de un área adicional en la tabla reservado para este tipo de situaciones. En el caso supuesto que una palabra W1 está en la posición P de la tabla, y a otra palabra W2 le toca la misma posición, entonces W2 se guarda en el área de colisiones si aún queda espacio libre.

Inicialmente, la tabla está vacía. Al ir revisando los textos las palabras bien escritas, las que sí aparecen en el diccionario, tienen oportunidad de entrar a la tabla. Una palabra nueva W1 se coloca en la tabla sólo si se cumple una de las tres situaciones siguientes:

1. La posición P que le corresponde no está ocupada por otra palabra.
2. La posición P está ocupada por otra palabra W2 cuya frecuencia es, en ese instante, menor que un número N arbitrario. Entonces la palabra W2

se remueve de la tabla y se pone en su lugar la nueva.

3. La posición P está ocupada por otra palabra pero hay espacio libre en el área de colisiones. Entonces W1 se coloca en este espacio.

En vista de que la información en la tabla se maneja frecuentemente y el espacio que ocupa es relativamente pequeño (4228 "bytes" de memoria), se mantiene en la memoria principal del computador durante la ejecución del programa. La operación y el manejo de esta tabla son totalmente transparentes al usuario.

Si una palabra no aparece en la lista de las palabras más usadas, entonces se busca en un diccionario con un volumen mayor de palabras. Al iniciar por primera vez el sistema, el diccionario también está totalmente vacío. El usuario puede construir tantos diccionarios como desee. Y puede agregar las palabras una por una o durante el proceso de la corrección, como se explica en la Sección XVIII. De esta forma el diccionario crece a la usanza del usuario. Durante las primeras veces que se use un diccionario nuevo, muchas palabras se señalarán como incorrectas pues aún no han sido incluidas en el diccionario.

La estructura de datos en que se implementa el diccionario es de gran importancia ya que de ella depende la

eficiencia de los algoritmos de detección y corrección. Debe adecuarse fácilmente a los siguientes circunstancias:

1. el conjunto de datos que se va a procesar (un diccionario de palabras)
2. el tipo de cada dato (una cadena de caracteres de longitud variable, donde dichos caracteres pertenecen a un vocabulario finito)
3. las operaciones que se van a efectuar sobre el conjunto y sobre cada uno de los datos: actualización (insertar y borrar), búsqueda de un elemento (básico para la detección y corrección), y manipulación del orden en que aparecen los caracteres (necesario para el método de corrección)
4. las memorias, principal y auxiliares, disponibles

El diccionario se implementó en una estructura similar a la de un árbol llamada "trie" [25].* Todas las palabras parten de un mismo punto (la raíz) y se van ramificando según cambien las letras que las componen. Las palabras 'hábil', 'hábito', 'hambre', 'helado', 'helecho', 'hiel', 'hielo', 'hierba', y 'mecha' se representan en un "trie" según se muestra en la Figura 1.

El signo '(' señala el final de cada palabra. Los números entre paréntesis corresponden a las longitudes (el

*ver Secciones VI y VII

número de letras) de las palabras que están en esa sub-rama. Conocer la longitud de las palabras disminuye el número de comparaciones necesarias para buscar una palabra y facilita el método de corrección, como se explicará más adelante.

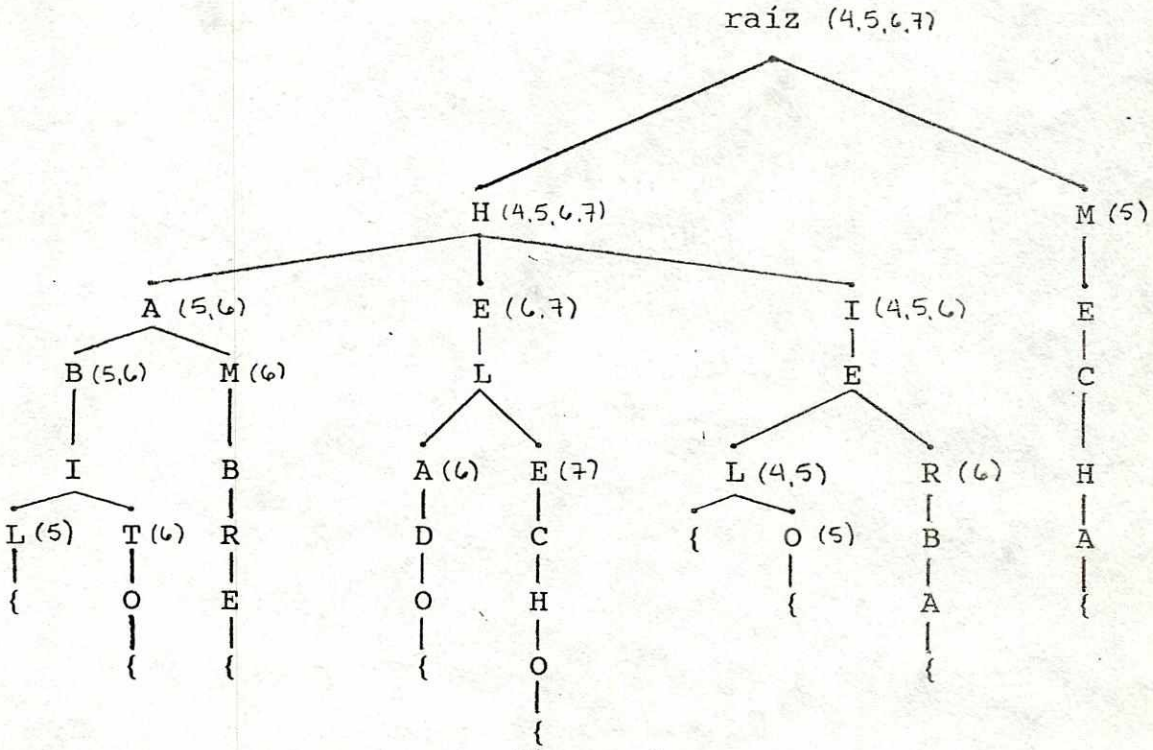

Figura 1.
Diagrama de un "trie"

La búsqueda de una palabra se inicia en la raíz del "trie". Las letras de la palabra, y el orden en que aparecen de izquierda a derecha, determinan el camino a recorrer en el "trie". Durante el recorrido de la búsqueda se revisa que en ese camino existan palabras con la misma longitud de la que se busca. Así, si se busca 'hemisferio' en el "trie" de la Figura 1, basta con revisar sólo la raíz para saber que no existe pues no hay ninguna palabra con diez letras. De esta manera se evita tener que recorrer el camino de 'H' a 'E' para saber que no existe el ramal que parte de la 'E' a la

'M'. Para buscar 'hermano' se tiene que pasar por la rama 'H'-'E' (que forma parte del camino en donde hay palabras con siete letras) para poder saber si hay un camino que siga a la 'R'.

Toda palabra del texto que aparece en el diccionario se considera como bien escrita. Si una de ellas no se encuentra en el diccionario entonces se asume que la palabra tiene una y solamente una de las siguientes cuatro faltas ortográficas.

1. una letra transpuesta

Premisa: El orden de dos letras adyacentes es incorrecto.

Solución: Transponerlas.

Ejemplo: revsita ---> revista

2. una letra mala

Premisa: Una letra en la palabra está incorrecta.

Solución: Sustituirla por otra letra del alfabeto.

Ejemplo: prebio ---> previo

3. una letra adicional

Premisa: Una letra está de más.

Solución: Eliminarla.

Ejemplo: congereso ---> congreso

4. una letra ausente

Premisa: Falta una letra.

Solución: Insertar una letra.

Ejemplo: imagnar ---> imaginar

Estas cuatro reglas de transformación se aplican, una a la vez, a cada palabra que no esté en el diccionario. La

palabra transformada se considera como una posible corrección si está en el diccionario. No se busca desde la primera letra (la raíz del "trie"), sino que a partir de la letra donde se paró la búsqueda original. Si la nueva palabra está en el diccionario, se agrega a la lista de posibles correcciones. Si no, se deshecha.

El error se supone que está en la i-ésima letra de la palabra, posición en la cual se suspendió la búsqueda. Por ejemplo, la palabra 'hireba' se recorre sin ningún problema por el ramal 'H'-'I' del "trie" de la Figura 1. Al no encontrar el camino de la 'I' a la 'R', se asume que hay un error en la tercera letra ('R'). Aplicando la primera regla, se transpone la 'R' con la letra siguiente obteniéndose la palabra 'hierba'. Se continúa recorriendo el "trie", ahora buscando la rama que va de la 'I' a la 'E'.

Después de aplicar las cuatro reglas, se obtiene una lista con todas las transformaciones de la palabra original que se encontraron en el diccionario. Cada una de estas palabras son candidatas a ser la forma correcta. Al usuario se le presenta esta lista para que él escoja la indicada. La corrección queda bajo el control total del usuario. De esta manera el programa, por sí solo, no puede dañar el texto. Para facilitar la decisión del usuario y que él pueda visualizar la palabra dentro del contexto, siempre se presentan en la pantalla tres líneas continuas del texto. La línea de en medio contiene la palabra en cuestión subrayada.

Además se presentan al usuario las palabras candidatas a ser la correcta. Hay cinco opciones que el usuario puede escoger:

1. no hacer nada y continuar revisando el texto
2. no hacer ninguna corrección, agregar la palabra subrayada al diccionario
3. corregir la palabra, escogiendo en su lugar a una de las que aparece en la lista de candidatas
4. corregir la palabra, tecleando la palabra correcta directamente al computador
5. corregir la palabra y agregarla al diccionario, tecleando la palabra correcta directamente al computador

La opción 2 es de gran importancia para el crecimiento del diccionario pues permite que éste refleje el uso de las palabras del usuario. Además ofrece la ventaja para que el diccionario crezca dinámicamente.

Se puede observar fácilmente que las palabras cortas, de una a tres letras, restan eficiencia al método de corrección. Después de aplicarles las cuatro reglas se obtendría una lista muy larga de palabras candidatas. Esto se debe a que las palabras cortas se diferencian de otras al cambiar una sola letra. Por ejemplo, 'ola' puede transformarse en 'la', 'bola', 'loa', 'oda', etc., aplicando una regla a la vez. Por lo tanto, el método sólo revisa palabras con cuatro a veinte letras.

VI. ORGANIZACION LOGICA DEL DICCIONARIO

El diccionario de palabras se implementó en una estructura conocida como "trie". Esta estructura fue propuesta por E. Fredkin [25]. El nombre proviene de la palabra inglesa "retrieval" (recuperación). Su forma es básicamente como la estructura de un árbol de múltiples niveles. En cada nodo se almacenan apuntadores que señalan a los hijos del mismo. La estructura especial del "trie" permite representar con facilidad conjuntos de cadenas de caracteres (u objetos ordenados de cualquier otro tipo de dato). La raíz representa por sí sola al conjunto vacío, o sea una cadena nula. Todo camino entre la raíz y una hoja (un nodo sin hijos) corresponde a una única cadena.

El diccionario es un conjunto de palabras. Una palabra es una cadena de caracteres que pertenecen a un alfabeto finito. Para representar las palabras del diccionario en el "trie", el alfabeto se restringió al conjunto de letras minúsculas 'a','b',...'z'. (La mayoría de teclados soporta este alfabeto.) No se diferencian mayúsculas de minúsculas pues, para los propósitos de este método, no importa distinguir entre 'Ayer' y 'ayer', pero sí entre 'ayer' y 'hayer'. Las letras mayúsculas en el texto que se revisa se convierten a minúsculas antes de chequear la ortografía de las palabras en donde aparecen. La ventaja de trabajar con

letras minúsculas es que, generalmente, en un documento hay menos mayúsculas que minúsculas.

Para recorrer el camino de una palabra, se codifican cada una de las letras de la palabra de la manera siguiente. A la letra 'a' le corresponde el valor 1, a la 'b' el 2, a la 'c' el 3, y así sucesivamente hasta la 'z', a la que se le asigna un valor de 26. En base a esta secuencia se estableció una función para calcular el valor F de una letra x: F(x)= (código ASCII de la letra x) - (número constante), donde el número constante tiene un valor de 96 y x pertenece a {'a'...'z'}. Este alfabeto ofrece la ventaja de que el código ASCII para cada una de las letras es numéricamente consecutivo, empezando la 'a' con 97 y terminando la 'z' con 122. Cada nodo del "trie" puede tener 26 hijos (por las 26 letras en el alfabeto). El valor F de la N-ésima letra nos indica cuál apuntador de los 26 que hay en el nodo señala al nodo hijo, nodo en el cual se ha de efectuar la misma operación para la letra N+1. Se continúa este proceso hasta llegar al fin del camino de la cadena de letras.

Representando de esta forma las palabras, surge la necesidad de marcar el final de cada una de ellas para diferenciar el fin del camino. Se escogió el caracter '{' como la marca final ya que, según el código ASCII es el símbolo que le sigue al caracter 'z', ajustándose perfectamente a la función F(x): F('{')=27. Para distinguir el camino que corresponde a "capa" del de "capaz", es más

sencillo trabajar con las cadenas "capa{" y "capaz{". La representación de estas dos palabras en el "trie" es:

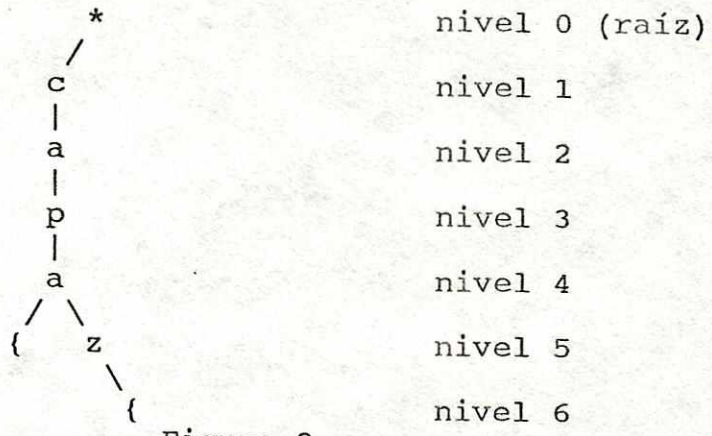

Figura 2.
Ejemplo de un "trie"

Como se puede observar en la figura, cada rama señala al siguiente caracter de la cadena. El nodo del nivel N corresponde al N-ésimo caracter de esa cadena. En otras palabras, el camino que va de la raíz a un nodo del nivel N representa al conjunto de todas las cadenas que empiezan con esta secuencia de N caracteres. El "trie" proporciona la facilidad de determinar si una cadena dada es prefijo (tiene las primeras letras) de una palabra. Esto es de suma importancia en el caso de la representación de un diccionario, donde hay una gran cantidad de grupos de palabras que empiezan con las mismas secuencias de letras, o prefijos. Por ejemplo, las palabras que empiezan con 'a' tienen en común el prefijo 'a' y tendrán en común este primer nodo. El "trie" ofrece un ahorro en el almacenamiento de información repetitiva entre las palabras, como lo son los prefijos. Vale la pena hacer notar que el número de prefijos

distintos en una palabra es siempre menor que la longitud total de ella.

La búsqueda de una palabra es más eficiente si para encontrarla no es necesario recorrer el árbol desde la raíz hasta una hoja. Se agregó un campo a la estructura del nodo que contiene las longitudes de las palabras representadas en el camino que pasa por el nodo.* Si la longitud de la palabra que se busca no está indicada en este campo, entonces la búsqueda no será exitosa y no hay razón para continuar recorriendo el "trie". Una palabra puede tener una longitud máxima de veinte letras.

La estructura del nodo queda constituida por dos campos:

1. 27 apuntadores: Los 26 primeros representan las 26 letras del alfabeto. El último apuntador indica el fin del camino de una cadena.
2. secuencia de 20 banderas: La N-ésima bandera encendida es señal de que ese camino contiene una palabra de N letras.

Cada nodo puede tener un máximo de 27 hijos. Las hojas del "trie" corresponden a las marcas finales ('{') de las cadenas.

La mayoría de los nodos tendrán menos de 27. Por ejemplo, la secuencia "bp" tiene un bajo índice de ocurrencia en el idioma español. Lo más posible es que nunca aparece. Mientras no ocurra, el apuntador del nodo 'b' que señale a

*Ver sección anterior.

"p" estará vacío. Se activará (el nodo hijo se creará) sólo cuando esta secuencia se presente en una cadena que se desee insertar en el "trie".

VII. ORGANIZACION FISICA DEL DICCIONARIO

El diccionario constituye una gran masa de información. Sería ideal que el diccionario completo residiera en la memoria principal; así las operaciones de lectura y escritura serían rapidísimas. En realidad la memoria disponible es limitada debido a que existen otros procesos simultáneos, en un sistema multiusuario, que también hacen uso de ella. Siendo así, lo más práctico y factible es mantener sólo una parte del diccionario en la memoria principal.

La estructura del diccionario se divide principalmente en páginas, habiendo a lo sumo una página en la memoria principal. Cada nodo del "trie" se almacena en un único registro de una página específica.

Los apuntadores de un nodo señalan a los nodos hijos. Es necesario, entonces, que cada apuntador activo indique tanto el número de página donde se encuentra el nodo hijo como su posición dentro de la página. Esta información se almacena en una palabra (16 "bits") de la computadora. Los primeros diez "bits", de izquierda a derecha, indican el número de la página en notación binaria, y los seis restantes representan el número de registro dentro de la página. De esta forma, se puede tener un máximo de 1023 (2^10-1) páginas, habiendo en una página 63 (2^6-1) nodos. En total, el "trie" puede estar formado hasta por 64,449 (1023x63) nodos.

La utilización de "bits" para almacenar el direccionamiento de los nodos facilita:

1. aumentar el tamaño del "trie". Para este caso se tiene que cambiar la estructura del apuntador incrementando su almacenamiento a dos palabras, por ejemplo, y jugar con los 32 "bits" resultantes. Con esta nueva cantidad de "bits" pueden aumentarse: el número máximo de páginas y/o el número de nodos en cada página.
2. crear nodos nuevos, con la única limitación del máximo número de páginas permitido.
3. dividir el "trie" en bloques (páginas) manejables
4. administrar el direccionamiento de apuntadores

Para representar la secuencia de banderas que indican las longitudes de las cadenas, un vector de "bits" es el más indicado. Un "bit" en cero significa que la bandera está apagada y viceversa. El número de banderas es veinte, asumiendo que la mayoría de las palabras tienen menos de 21 letras. Una secuencia de 20 "bits" necesita por lo menos dos palabras de almacenamiento, quedando 12 "bits" sin usarse. Como ejemplo: si una palabra tiene cinco letras el "bit" 5 (empezando con el 1 de derecha a izquierda) se enciende.

El diccionario se implementó en PASCAL como un archivo corriente cuyos registros físicos corresponden a cada una de

las páginas. Para implementar la estructura del "trie" se definieron estos tipos de datos:

1. ENTERO_SIMPLE=-32768..32767; De esta manera se tiene un tipo de dato que ocupa una palabra de almacenamiento.
2. NODOTRIE=RECORD Un nodo es un tipo de dato complejo que consta de 16 "bits" en PUNTERO : ARRAY[1..27,1..2] OF 0..255; y 32 "bits" en LONGITUD : INTEGER;
3. BLOQUE=ARRAY[1..63] OF NODOTRIE; Corresponde a una página del diccionario

En cada operación de lectura o escritura se lee un bloque, o sea 63 nodos. El factor de bloqueo físico es 1, y el lógico es 63. Para crear nodos nuevos, es indispensable saber cuál es la última página del "trie" sin usar, y el último registro que se haya usado. Esta información se guarda cada vez que se cierra el diccionario en el apuntador No. 27 del primer nodo del "trie".

Un apuntador nulo tiene el valor cero, y ninguna rama sale de él. Para indicar el fin del camino de una cadena, el veintisieteavo apuntador del nodo se señala a sí mismo.

La representación de un "trie" con una única cadena 'barco{' se muestra en la Figura 3. Si se agrega la cadena 'barrio{', el "bit" 6 del campo LONGITUD se enciende para los dos primeros nodos (ver Figura 4).

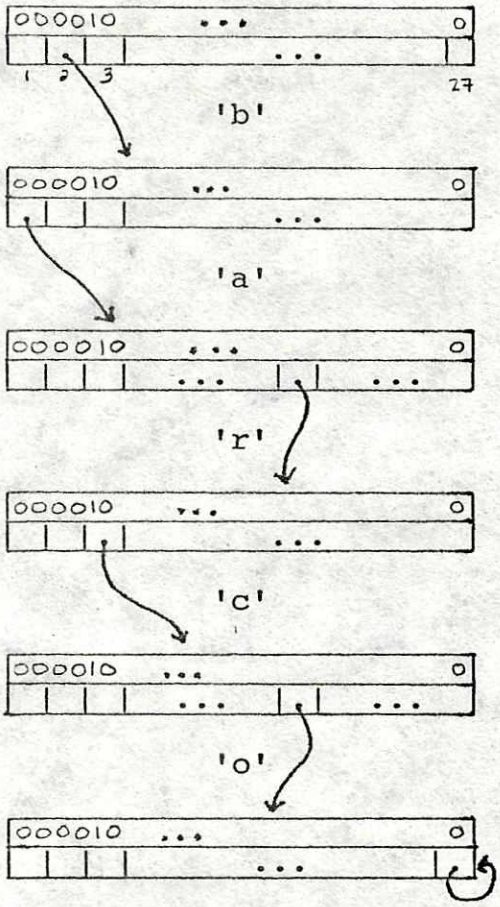

Figura 3.
Representación de 'barco' en un "trie"

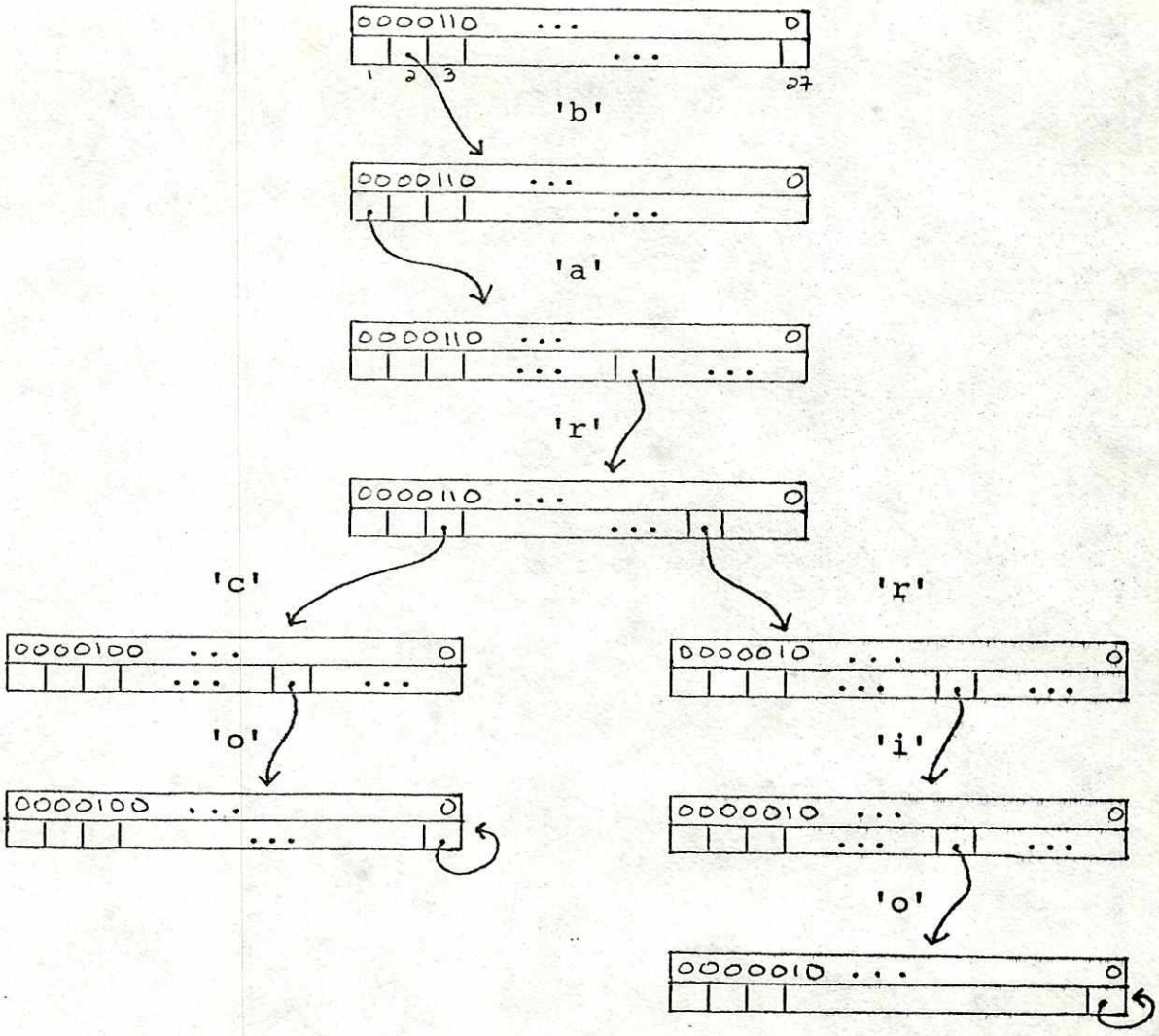

Figura 4.
Representación de 'barco' y 'barrio' en un "trie"

VIII. MANUAL DE USO PARA EL SISTEMA "DCO"

El usuario debe estar familiarizado con las instrucciones básicas del sistema operativo MPE-V en una computadora HP3000. El sistema funciona en cualquier pantalla que soporte "video" subrayado, y preferiblemente "video" inverso donde los caracteres aparecen oscuros sobre un fondo claro. El sistema necesita procedimientos especiales que deben aparecer en una librería segmentada. Asegúrese de que usted tenga acceso a esta librería. Si tiene dudas pregunte al encargado del centro de computación. Al final de esta sección aparecen las figuras a las que se hace referencia en los párrafos siguientes.

A. INICIO DEL SISTEMA

1. Entre a sesión en la computadora como acostumbra hacerlo.

2. Entre al sistema "DCO" tecleando :RUN DCO. Si es necesario, debe especificar el nombre del grupo y de la cuenta donde está localizado el programa. Generalmente estará en la PUB.SYS. Si es así, entonces teclee :RUN DCO.PUB.SYS

3. A continuación aparecerá en la pantalla la información de la Figura 5. Antes de seguir usando el sistema, usted debe seleccionar el diccionario con que va a

trabajar. El nombre del diccionario puede ser hasta de ocho caracteres alfanuméricos, el primero debe ser alfabético. Si es necesario, se deben especificar el grupo y la cuenta donde está el diccionario. En caso el diccionario no exista, el programa se lo indicará (Figura 6). Entonces, usted puede decidir si crea un diccionario nuevo o si da el nombre de otro.

B. MENU PRINCIPAL

El menú principal del sistema "DCO" se muestra en la Figura 7. En la primera línea de la pantalla aparece la identificación del sistema y abajo el nombre del diccionario que está en uso. El detalle de las cinco funciones en el Menú Principal se expone más adelante. En la esquina superior izquierda de los menús de cada función aparece la identificación del subsistema. En el otro lado aparece el nombre del diccionario activo. Para salir de un subsistema, se debe escoger la opción FIN (siempre aparece de último). El sistema regresa automáticamente al Menú Principal.

C. OPCION 1: AGREGAR PALABRAS AL DICCIONARIO

Al escoger la Opción 1 de la Figura 8 aparece una ventanilla en donde usted puede escribir cualquier palabra de 4 a 20 letras para así agregarla al diccionario. No hay ningún problema si la palabra ya está en el diccionario. Al terminar de insertar la nueva palabra (por ejemplo,

'arboleda' en la Figura 9), se emite un mensaje abajo de la ventanilla para indicar que todo fue un éxito.

D. OPCION 2: REVISAR PALABRAS EN EL DICCIONARIO

Con esta opción, se puede averiguar si una palabra está o no en el diccionario. Si está, aparece en la pantalla un mensaje como en la Figura 10. Si no está, aparece un mensaje similar indicando que no está.

E. OPCION 3: REVISAR LAS PALABRAS EN UN TEXTO

Lo primero que a usted se le pregunta (Figura 11) es el nombre del texto que desea procesar. El texto debe estar almacenado en código ASCII en su grupo y cuenta. Cada línea debe tener un máximo de 80 caracteres.

A continuación aparecerá la pantalla de la Figura 12. En la esquina superior izquierda aparece el nombre del texto que se está procesando. Luego aparecen tres líneas del texto, a manera de que la línea de en medio contiene la palabra que el sistema "DCO" ha detectado como incorrecta. Esta palabra aparece subrayada y también aparece en una ventanilla aparte. Dependiendo del número de palabras similares que el sistema encuentre en el diccionario, a usted se le presentarán diferentes opciones. Sin embargo, siempre se le presentarán las cinco opciones de la Figura 12, la cual corresponde al caso en que el diccionario no proporciona ninguna corrección.

IGNORAR: El sistema ignora la ortografía de la palabra y procede a revisar la palabra siguiente.

IGNORAR/AGREGAR: Igual que la opción anterior, sólo que la palabra se agrega al diccionario. Es muy útil para que el diccionario vaya creciendo.

DAR CORRECCION: Usted es quien proporciona la corrección. Aparece una nueva ventanilla en donde puede escribirla (ver Figura 13).

DAR CORRECCION/AGREGAR: Igual que la opción anterior, sólo que la palabra que usted ingresa se agrega al diccionario.

FIN: El sistema regresa al Menú Principal

En caso el sistema encuentre una sola corrección, ésta se muestra a la derecha del menú (Figura 14). También aparece otra opción:

CORREGIR CON SIMILAR: La palabra subrayada se sustituye con la similar.

Si el sistema encuentra de dos a un máximo de veinte palabras similares, entonces se emite una nueva opción.

LISTAR SIMILARES: La lista de similares se muestra de siete en siete (Figura 15) para que usted pueda ver el número de la palabra similar con la que sustituirá la incorrecta, o para simplemente revisar esta lista. Si usted desea ver las demás, responda 'S' a la pregunta 'CONTINUO?'.

CORREGIR CON SIMILAR: Usted proporciona el número de la palabra similar que va a sustituir a la palabra subrayada (Figura 16).

F. OPCION 4: MANEJO DE DICCIONARIOS

Con el menú de esta opción (Figura 17), usted puede cambiar de diccionario (Figura 18) o listar un conjunto de sus palabras (Figura 19). Para listar debe ingresar a partir de y hasta cuál palabra desea que se tome en cuenta. Según el ejemplo de la Figura 19, usted estaría listando todo el diccionario. El listado puede salir en la pantalla o en la impresora bajo el nombre de DICLIST.

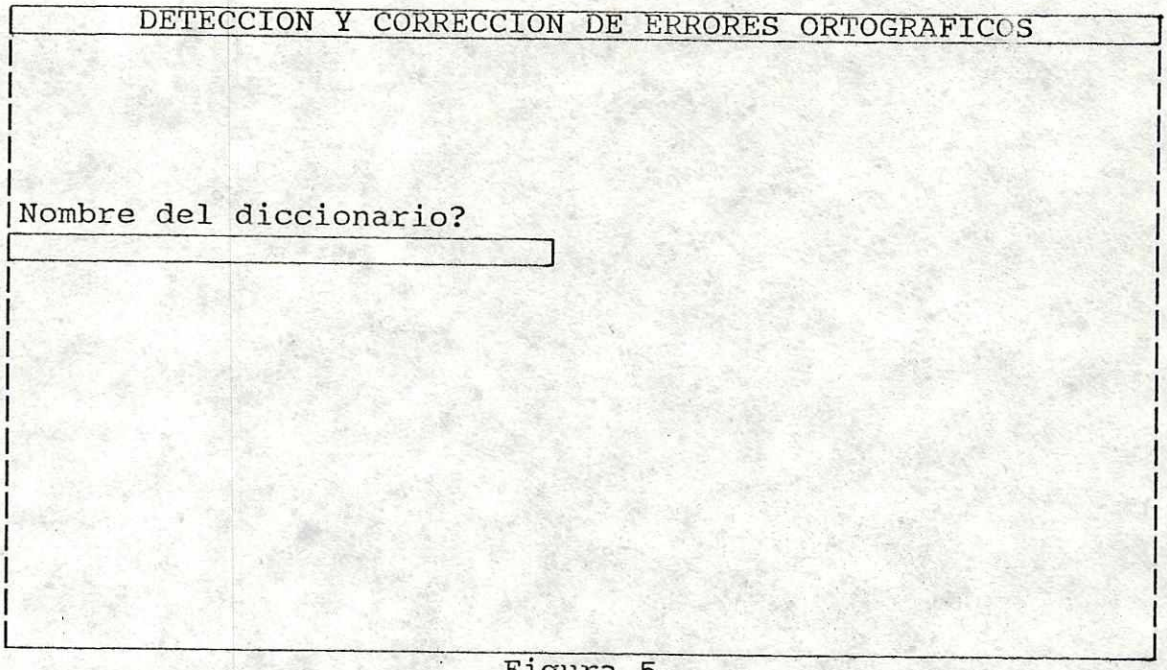

Figura 5.
Inicio del sistema DCO

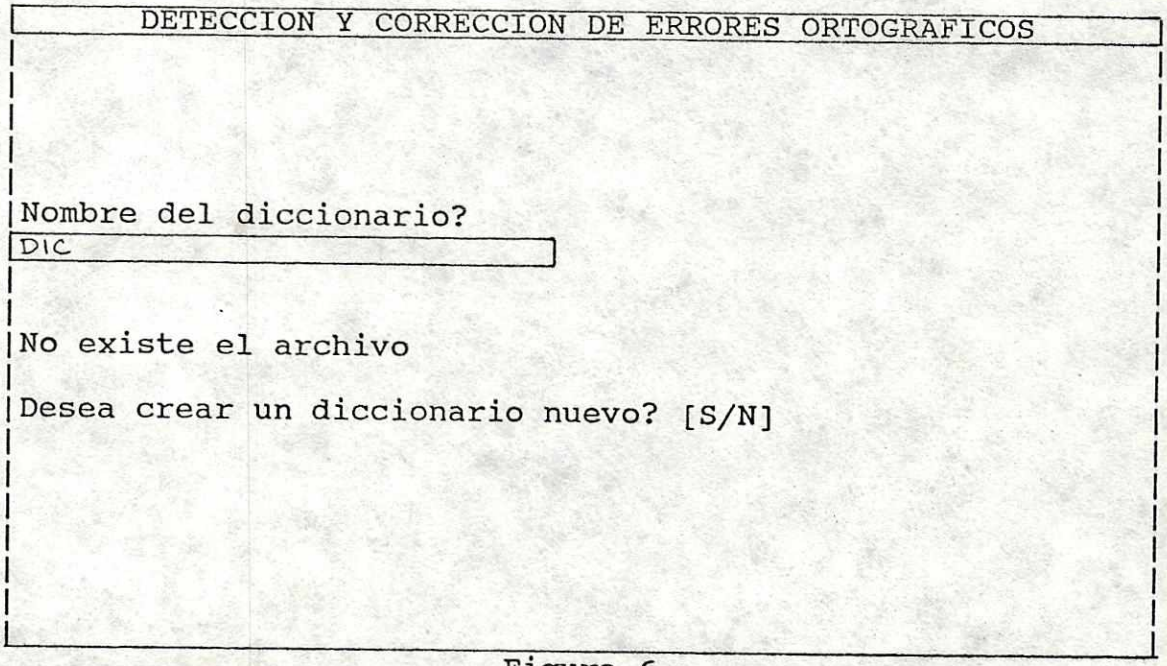

Figura 6.
Opción para crear un diccionario nuevo

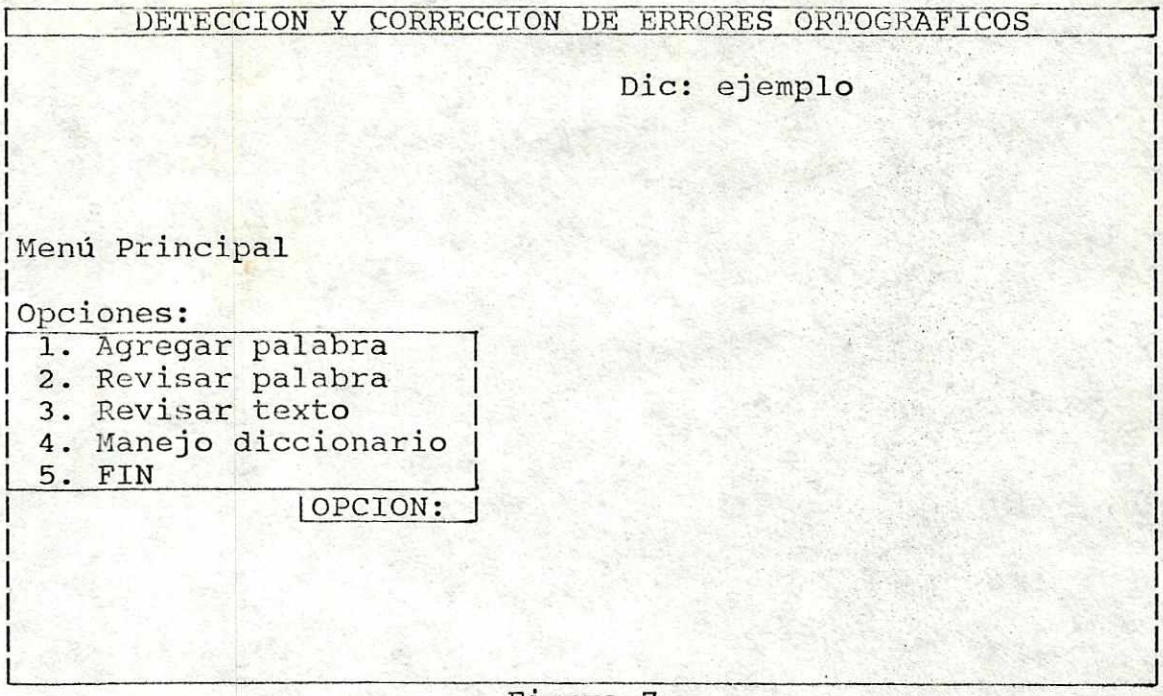

Figura 7.
Menú Principal

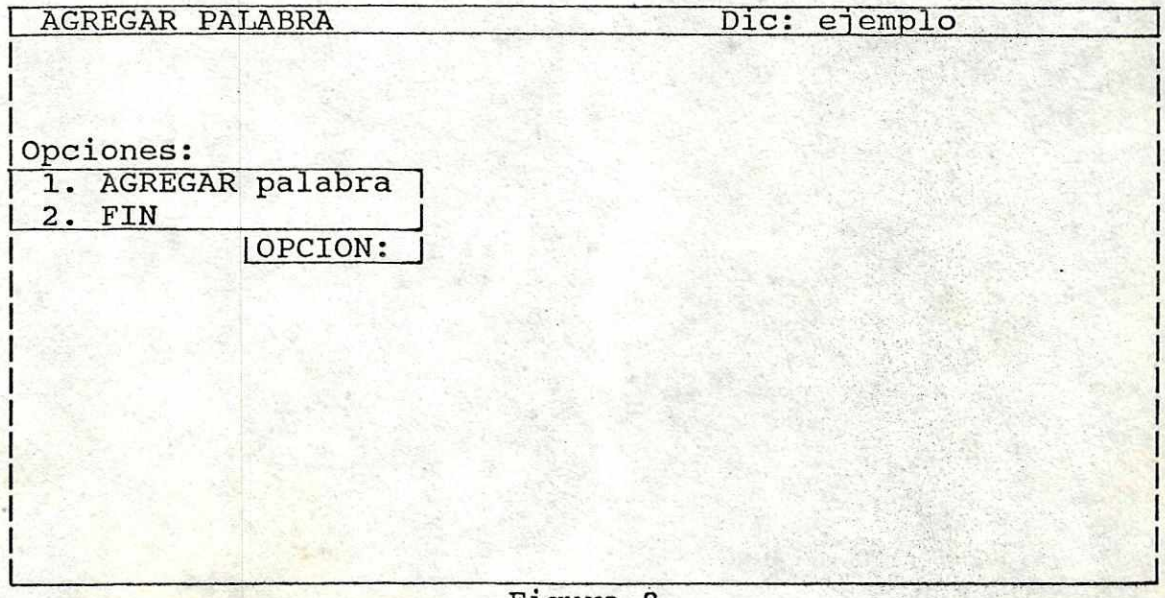

Figura 8.
Menú para agregar palabras

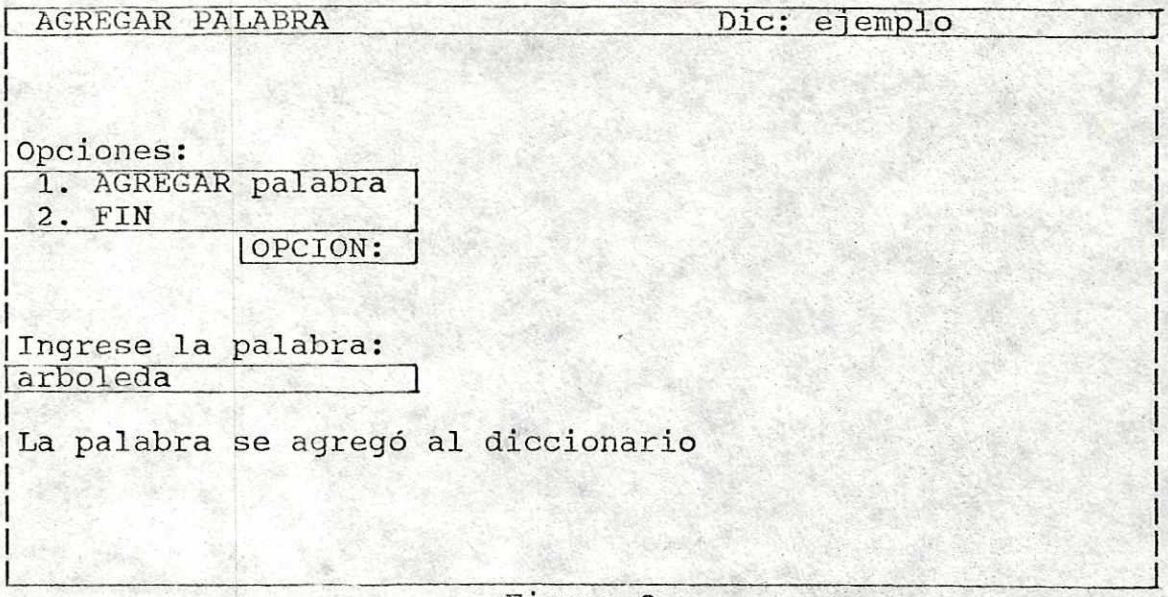

Figura 9.
Inserción de 'arboleda' al diccionario

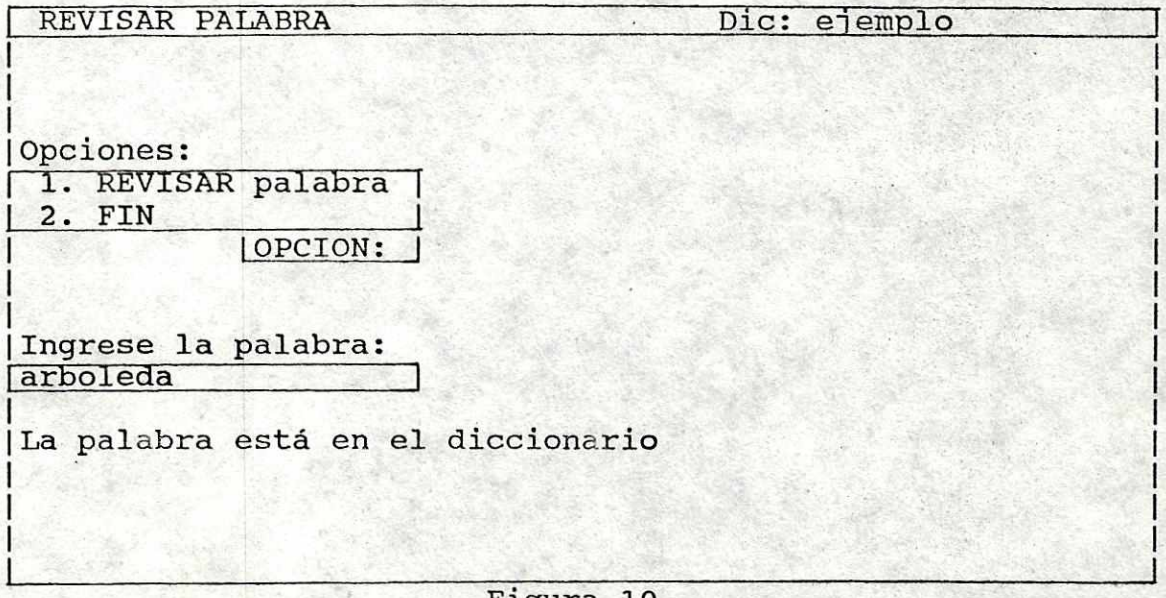

Figura 10.
Verificación de si 'arboleda'
está en el diccionario

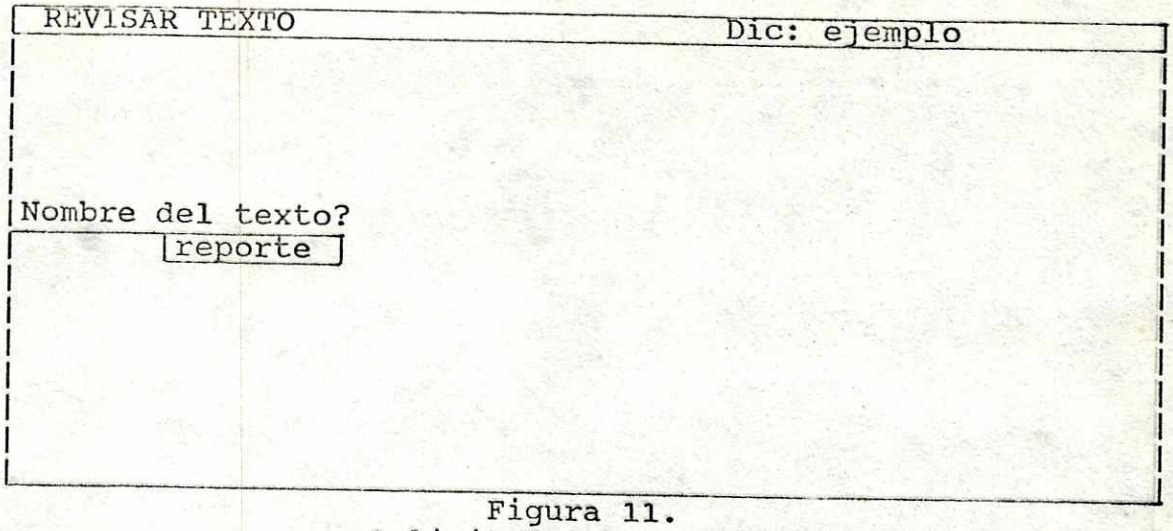

Figura 11.
Solicitud del texto que
se va a revisar

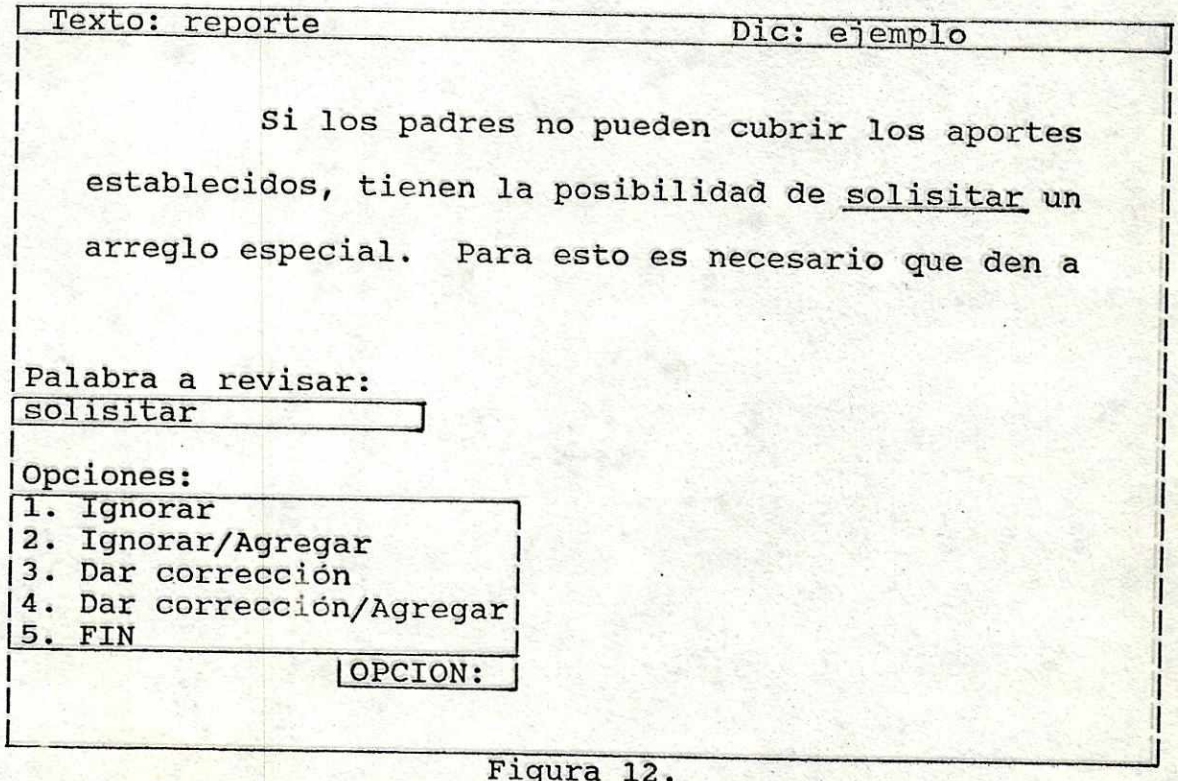

Figura 12.
Revisión de un texto: El sistema no encontró
la corrección para una palabra incorrecta

```
Texto: reporte                                  Dic: ejemplo

          Si los padres no pueden cubrir los aportes
   establecidos, tienen la posibilidad de solisitar un
   arreglo especial.  Para esto es necesario que den a


Palabra a revisar:                    Ingrese la palabra:
solisitar

Opciones:
1. Ignorar
2. Ignorar/Agregar
3. Dar corrección
4. Dar corrección/Agregar
5. FIN
                        OPCION:3
```

Figura 13.
Ingreso manual de la corrección

```
Texto: reporte                                  Dic: ejemplo

          Si los padres no pueden cubrir los aportes
   establecidos, tienen la posibilidad de solisitar un
   arreglo especial.  Para esto es necesario que den a


Palabra a revisar:
solisitar

Opciones:                             Palabra(s) similar(es):
1. Ignorar                            1. solicitar
2. Ignorar/Agregar
3. Dar corrección
4. Dar corrección/Agregar
5. Corregir con similar
6. FIN
                        OPCION:
```

Figura 14.
El sistema encuentra una corrección

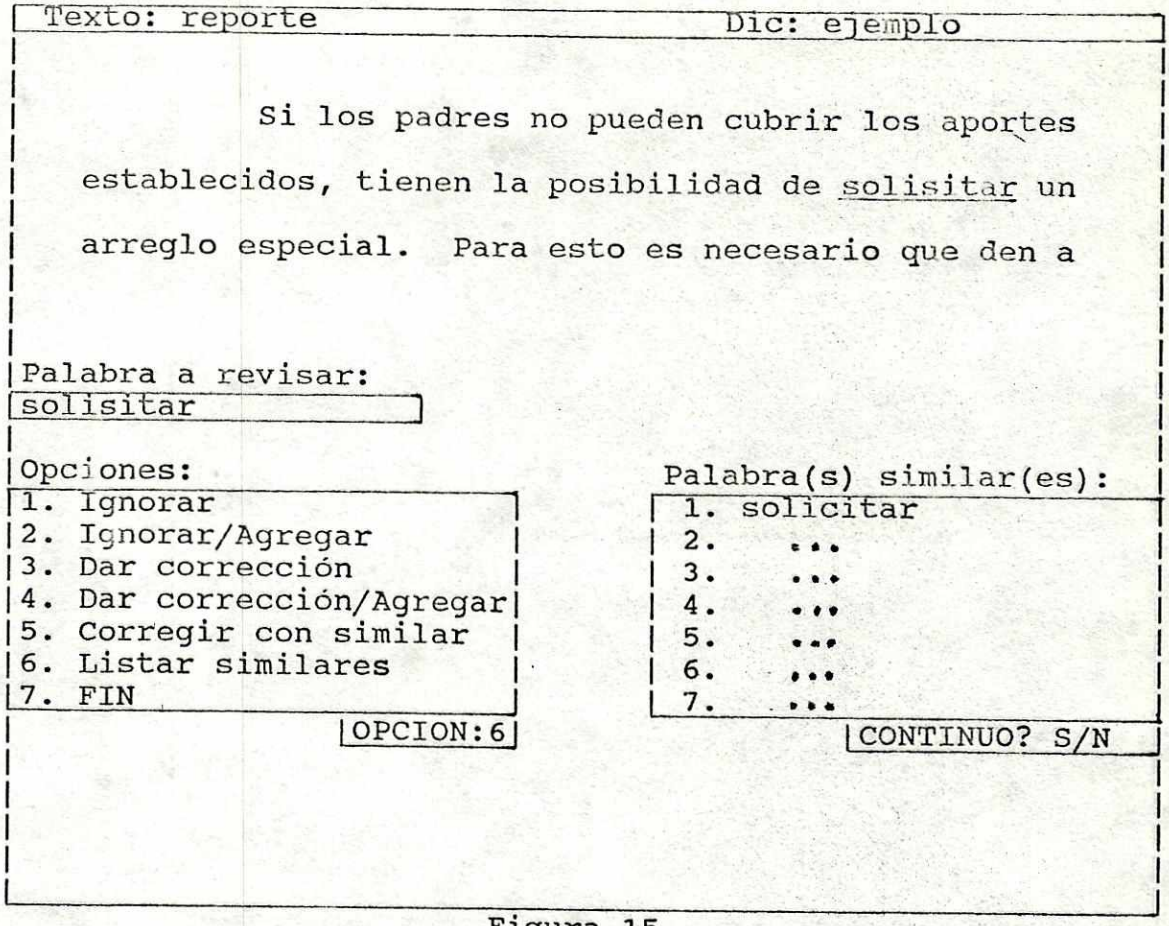

Figura 15.
El sistema encuentra varias posibles correcciones.

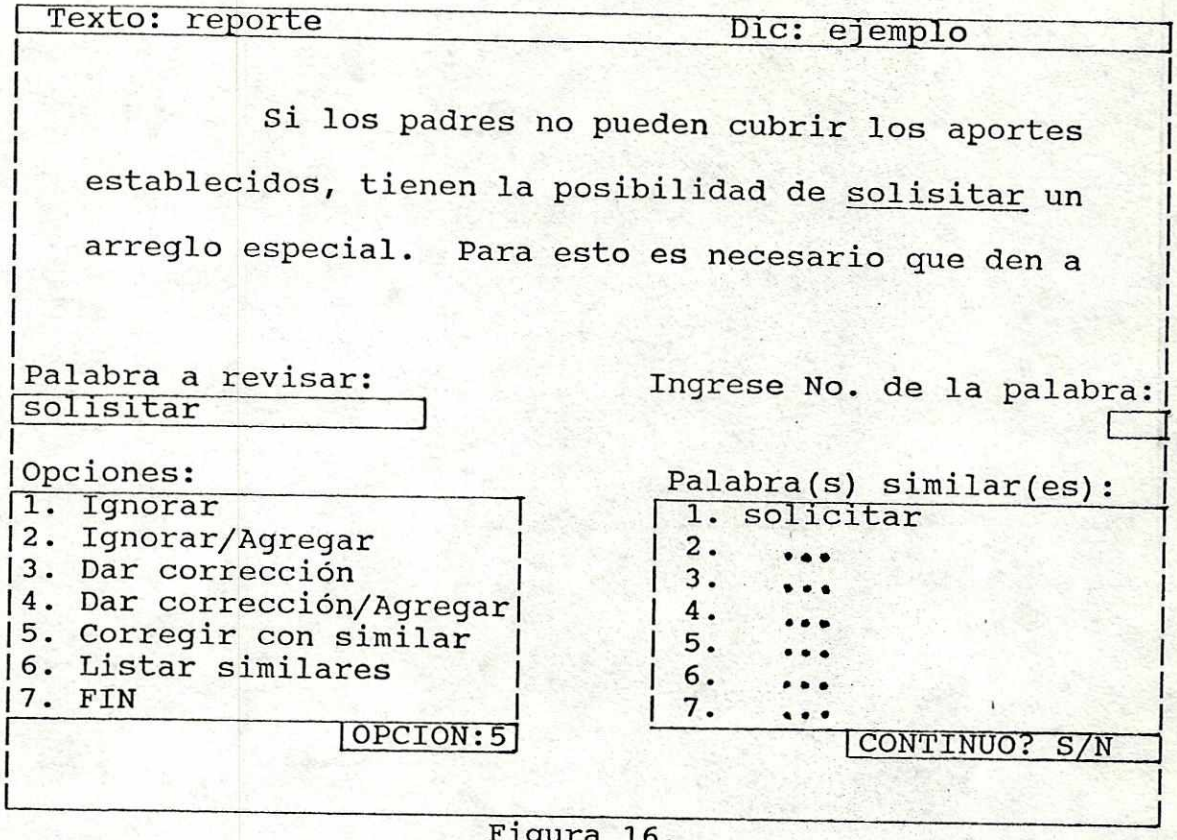

Figura 16.
Selección de una palabra similar como la correcta.

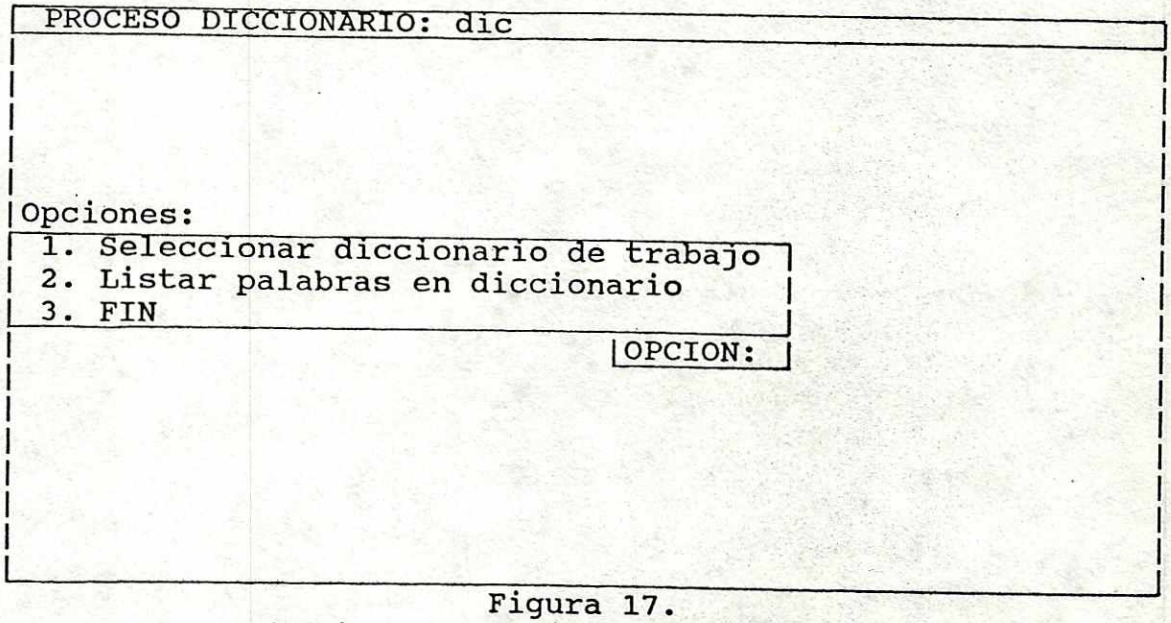

Figura 17.
Menú para manejar el diccionario

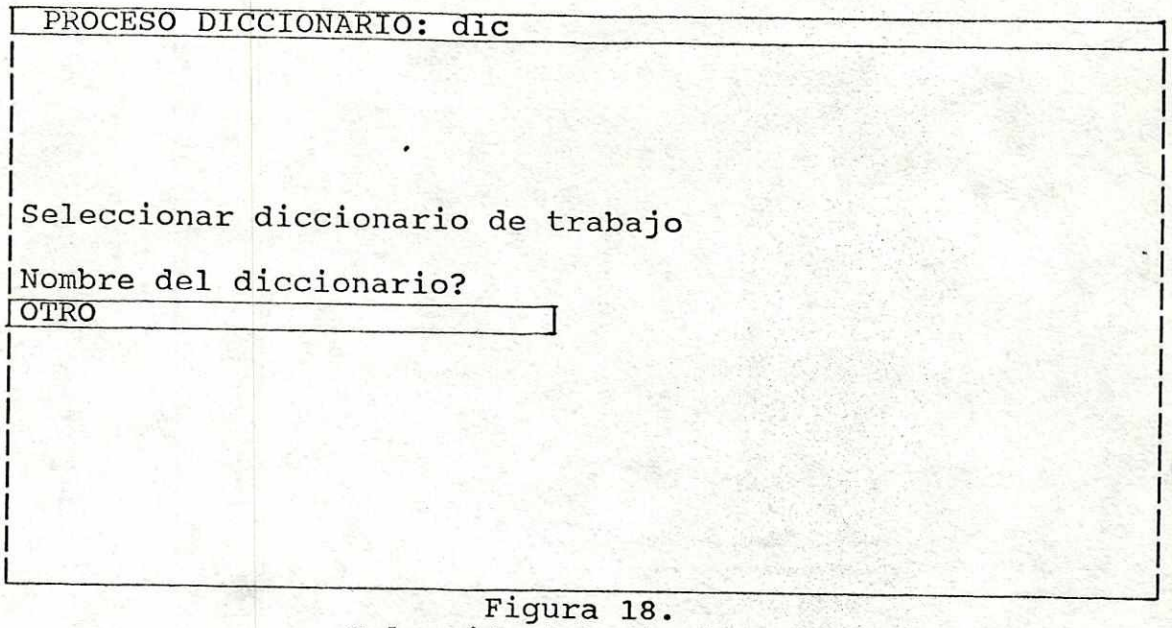

Figura 18.
Selección del diccionario

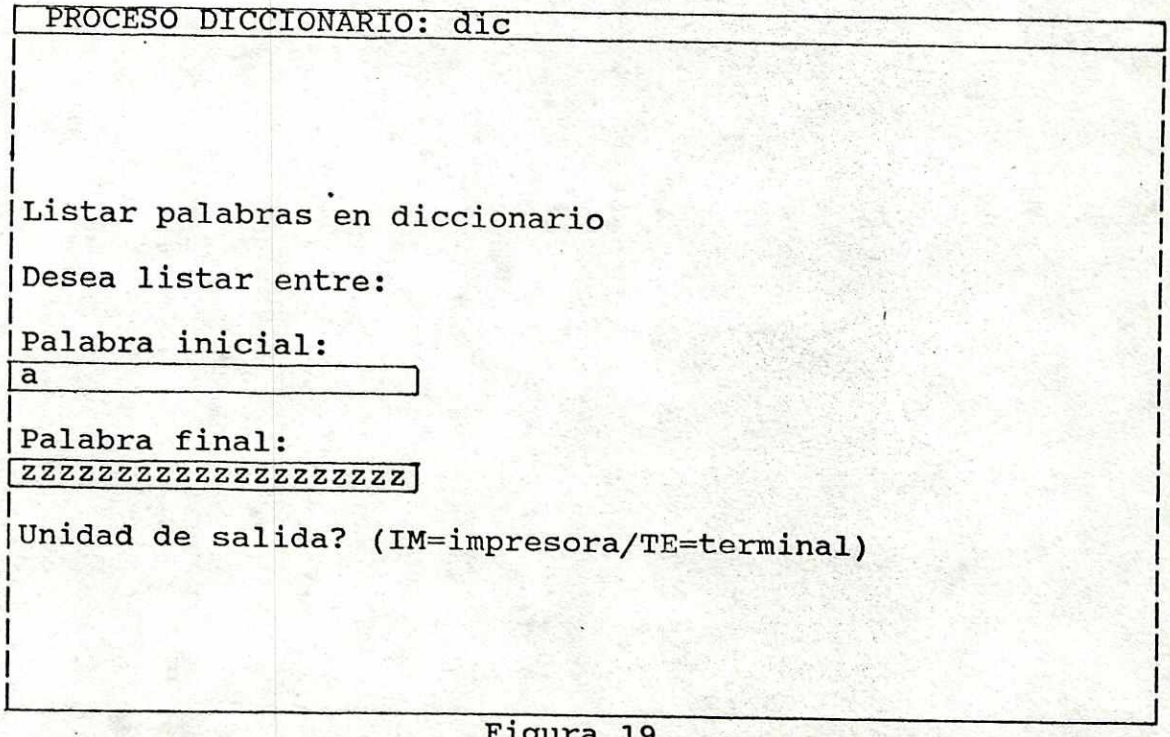

Figura 19.
Listado de palabras del diccionario

IX. CONCLUSIONES Y RECOMENDACIONES

La mayoría de los procesos que se realizan en un computador se reducen a simples comparaciones entre un objeto y otro. El grado de similitud entre ellos determina la acción que se debe tomar. El método de detección del sistema "DCO" se reduce a comparar la palabra en duda contra una lista de palabras escogidas, y luego contra las palabras de un diccionario. Si no se encuentra una palabra igual, la palabra en duda se toma como incorrecta. La evolución de las listas de palabras bien escritas se hace en base a comparaciones también, a manera de mantenerlas en orden y para que sean de fácil acceso.

El método de corrección que se utilizó en este sistema se basa en la idea de que una palabra mal escrita contiene cualquiera de los siguientes errores: sustitución, adición u omisión de una letra, o transposición de dos letras. Se ha comprobado que un 90 a 95% de los errores que se cometen al teclear información contienen un sólo error de éstos [22].

Si el acto de cometer un error produce una palabra que sí existe en el diccionario, este error pasa desapercibido. Errores de esta clase se consideran como parte de un problema de mal uso de la palabra y no como un error ortográfico.

Siempre va a haber más de alguna palabra mal escrita que el método no podrá corregir porque no encuentra la palabra supuestamente correcta en el diccionario. Aprovechando esta situación, se considera una buena táctica que el método de corrección deje escapar un número relativamente pequeño de errores, pues muchas palabras pueden convertirse en otras con una sola de las cuatro operaciones de error antes mencionadas. También es conveniente que el tamaño del diccionario no sea muy grande, para que no contenga palabras arcaicas o poco usadas. Es más probable que el usuario escriba mal una palabra a que él la use intencionalmente como una palabra poco común. En caso de que se cumpla esta última condición, es preferible que el usuario sea quien corrija los errores "difíciles". Si se deja crecer el diccionario demasiado, puede que el método reporte menos errores (pues encuentra la mayoría de las palabras del texto en el diccionario) pero es igualmente posible que produzca un número considerable de correcciones equivocadas. La variedad en los resultados del método de corrección depende más que nada del tamaño del diccionario que se use y no de la estrategia que se emplee.

El método propuesto puede ampliarse a manera de que la fase de la detección se apoye en más información que la que proporciona un diccionario corriente, y que la corrección pueda aplicar distintos criterios para encontrar las palabras correctas. En algunas oportunidades puede ser deseable que el método ignore las instrucciones para los procesadores de

palabras que aparecen en algunos textos, o que no tome en cuenta siglas de organizaciones o nombres propios.

Si bien es importante que el usuario controle el proceso de la corrección, también es necesario que se le consulte lo menos posible. Las decisiones que el usuario tome deben ser reusadas automáticamente por el programa, en todas las ocasiones donde sea posible. Si la palabra 'tormnta' aparece varias veces en el texto y ha sido corregida a 'tormenta', el programa debe, en lo sucesivo, corregirla automáticamente.

Para procesar textos escritos en español, es conveniente realizar estudios sobre: la frecuencia de las letras y sílabas en este idioma, los errores que se cometen corrientemente, en qué parte de la palabra suelen aparecer estos errores, etc. Con esta información puede estructurarse un proceso para enfrentar específicamente los errores ortográficos en el idioma español.

Con algunos cambios, el sistema "DCO" puede tomar en cuenta palabras tildadas o con eñe. No se hizo así desde un principio porque son pocas las terminales que soportan estos caracteres.

X. BIBLIOGRAFIA

1. AHO, Alfred V., John E. HOPCROFT, Jeffrey D. ULLMAN. The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Mass., 1983.

2. BENTLEY, John. "A Spelling Checker". Communications of the ACM, Vol. 28, No. 5, mayo 1985. pp 456-462.

3. BLAIR, Charles R. "A Program for Correcting Spelling Errors". Information and Control, Vol. 3, No. 1, marzo 1960. pp 60-67.

4. DAMERAU, Fred J. "A Technique for Computer Detection and Correction of Spelling Errors". Communications of the ACM, Vol. 7, No. 3, marzo 1964. pp 171-176.

5. DAVIDSON, Leon. "Retrieval of Misspelled Names in an Airlines Passenger Record System". Communications of the ACM, Vol. 5, No. 3, marzo 1962. pp 169-171.

6. DODDS, D. J. "Reducing Dictionary Size by Using a Hashing Technique". Communications of the ACM, Vol. 25, No. 6, junio 1982. pp 368-370.

7. DURHAM, Ivor, David A. LAMB, James B. SAXE. "Spelling Correction in User Interfaces". Communications of the ACM, Vol. 26, No. 10, octubre 1983. pp 764-773.

8. FALOUTSOS, Christos. "Access Methods for Text". ACM Computer Surveys, Vol. 17, No. 1, marzo 1985. pp 49-74.

9. GOOD, Phillip. "Spelling Checkers: Electronic Dictionaries for your Computer". Popular Science, abril 1983. pp 145-148.

10. GROSS, J. "On Spelling Error Detection". (preguntas al artículo de J. Peterson). Communications of the ACM , Vol. 24, No. 5, mayo 1981. pp 331-332.

11. JAMES, E. B., D. P. PARTRIDGE. "Tolerance to Innacuracy in Computer Programs". The Computer Journal, Vol. 19, No. 3, agosto 1976. pp 207-212.

12. KNUTH, Donald E. The Art of Computer Programming, Volume 3: Sorting and Searching. Addison-Wesley, Reading, Mass., 1973.

13. LEMMONS, Phil. "Five Spelling-Correction Programs for CP/M-Based Systems". Byte, Vol. 6, No. 11, noviembre 1981. pp 434-448.

14. McKEAN, Kevin S. "Las computadoras que corrigen lo escrito". International Management, Vol. 37, No. 7, julio 1982. pp 31-33.

15. MEILACH, Dona Z. "The Random House Proofreader". Interface Age, mayo 1983. pp 24,160-163.

16. MORGAN, Howard. L. "Spelling Correction in Systems Programs". Communications of the ACM, Vol. 13, No. 2, febrero 1970. pp 90-94.

17. NIX, Robert. "Experience with a Space Efficient Way to Store a Dictionary". Communications of the ACM, Vol. 24, No. 5, mayo 1981. pp 297-298.

18. PETERSON, James L. "Computer Programs for Detecting and Correcting Spelling Errors". Communications of the ACM, Vol. 23, No. 12, diciembre 1980. pp 676-687.

19. POLLOCK, Joseph J. SpeedCop. Final Report. Chemical Abstracts Service, Columbus, Ohio, septiembre 1981.

20. POLLOCK, Joseph J. "Spelling Error Detection and Correction by Computer: Some Notes and a Bibliography". Journal of Documentation, Vol. 38, No. 4, diciembre 1982. pp 282-291.

21. POLLOCK, Joseph J., Antonio ZAMORA. "Automatic Spelling Correction in Scientific and Scholarly Text". Communications of the ACM, Vol. 27, No. 4, abril 1984. pp 358-368.

22. POLLOCK, Joseph J., Antonio ZAMORA. "Collection and Characterization of Spelling Errors in Scientific and Scholarly Text". Journal of the American Society for Information Science, Vol. 34, No. 1, enero 1983. pp 51-58.

23. POLLOCK, Joseph J., Antonio ZAMORA. "System Design for Detection and Correction of Spelling Error in Scientific and Scholarly Text". Journal of the American Society for Information Science, Vol. 35, No. 2, marzo 1984. pp 104-109.

24. ROBINSON, Peter, Dave SINGER. "Another Spelling Correction Program". Communications of the ACM, Vol. 24, No. 5, mayo 1981. pp 296-297.

25. SEVERANCE, Dennis G. "Identifier Search Mechanisms: A Survey and Generalized Model". ACM Computer Surveys, Vol. 6, No. 3, septiembre 1974. pp 175-194.

26. SRIHARI, Sargur N., Jonathan J. HULL, Ramesh CHOUDHARI. "Integrating Diverse Knowledge Sources in Text Recognition". ACM Transactions on Office Information Systems, Vol. 1, No. 1, enero 1983. pp 68-87.