ISSN 1870-4069

Un sistema experto para corrección de textos usando reglas ortográficas

Manuel Cristóbal López Michelone

Universidad Nacional Autónoma de México,
México

morsa@la-morsa.com

Resumen. Hoy en día los procesadores de palabras nos son familiares y cotidianos. Además de permitir a los usuarios crear documentos, estos pueden ser revisados y corregidos por software incluido que normalmente permite revisar si las palabras están correctamente escritas. El enfoque más usado es depender de un diccionario con cientos de miles de palabras para hacer la corrección ortográfica. Aquí presentamos un sistema experto que saca ventaja de las características del idioma español para hacer la corrección ortográfica sin necesidad de usar un diccionario. Además, el enfoque puede servir para aprender y entender el uso de muchas reglas comunes del idioma español.

Palabras clave: Corrección ortográfica, sistemas expertos, reglas ortográficas, algoritmos.

An Expert System for Proofreading Using Spelling Rules

Abstract. Word processors are very familiar to us. In addition to allowing users to create documents, a word processor can review and correct the documents trying to find spelling mistakes. The most widely used approach is to rely on a dictionary with hundreds of thousands of words to do the spell checking. Here we present an expert system that takes advantage of the characteristics of the Spanish language to do the spell checking without the need of a dictionary. In addition, this approach can serve to learn and understand the use of many common rules of the Spanish language.

Keywords: Text proofreading, expert systems, spelling rules, algorithms.

1. Introducción

Los correctores ortográficos son parte de muchos paquetes de software para la oficina, entre los que se cuentan los procesadores de palabras. LibreOffice, OpenOffice y Microsoft Word son tres ejemplos de programas que contienen un procesador de palabras con corrector ortográfico, el cual en su forma más general, utiliza un enorme

diccionario de términos en donde se busca que las palabras estén correctamente escritas. Este tipo de software hace búsquedas muy rápidas y en general trabaja adecuadamente.

Sin embargo, en el idioma español existe un número grande de reglas ortográficas las cuales definen, sin ambigüedad, cómo deben escribirse muchas palabras. Un buen corrector ortográfico, sin embargo, bien puede apoyarse de otras técnicas para hacer su tarea. Esto es, debe ser capaz de:

– Usar un diccionario con unos 500,000 términos.
– Usar diccionarios personalizados.
– Usar otras tecnologías para buscar errores, como patrones equivocados de letras.
– Usar un diccionario de verbos ya conjugados.
– Usar reglas ortográficas.

Un sistema híbrido sin duda es mejor que un sistema que sólo sigue una técnica en particular. Nosotros nos concentraremos en el sistema experto de corrección basado en reglas ortográficas, aunque mencionaremos al final del artículo la implementación de las otras posibilidades.

2. Sistemas expertos

Un sistema experto es "un programa de computadora inteligente que usa conocimiento y procedimientos de inferencia para resolver problemas que son los suficientemente difíciles para requerir la expertez significativa de un ser humano para su solución" [6]. Dicho de otra manera, un sistema experto es un sistema que **emula** las habilidades de decisión de un experto humano. Los sistemas expertos trabajan en dominios restringidos y en general contienen tres grandes apartados:

1. Interfaz con el usuario.
2. Base de conocimientos.
3. Sistema de inferencia.

En el caso de la corrección ortográfica, considerando el número de reglas sobre el uso de las diferentes palabras, se puede utilizar un sistema experto que, a través de la aplicación de las reglas ortográficas conocidas, pueda decidir (sin necesidad de consultar un diccionario de palabras necesariamente) si las palabras están bien escritas o no.

El idioma español consigna centenares de reglas [10], las cuales son la parte fundamental de la base de conocimientos del sistema experto. Por lo que se refiere al sistema de inferencia, se usará el que utiliza por definición Prolog, que es encadenamiento hacia atrás, backward chaining, el cual se usa en un número de sistemas expertos.

Finalmente, la interfaz del usuario se plantea como un proceso por lotes (batch), (en Prolog), en donde el usuario le indica al sistema que revise el texto escrito para ver si hay errores ortográficos, es decir, no se hace la corrección en tiempo real. Sin embargo, también se ataca el problema de hacer corrección después de que el usuario escribe alguna palabra en el procesador de palabras (tiempo real), incluso usando aplicaciones comerciales como Word de Microsoft.

3. Trabajo relacionado

La verificación y corrección de textos es parte cotidiana en el mundo digital. Por años se ha trabajado en el tema, particularmente en el idioma inglés. Se ha encontrado que los errores ortográficos son consecuencia de diferentes factores:

1. **Errores tipográficos**: Ocurren cuando el usuario escribe mal una palabra por error. Aquí no se puede adjudicar el error a un criterio lingüístico sino a errores en la escritura. En [3] se muestra que el 80 % de los errores tipográficos caen en alguna de estas categorías:
   a) Inserción extra de una o más letras.
   b) Borrado de una o más letras.
   c) Sustitución de una letra por otra.
   d) Transposición de dos letras adyacentes.
2. **Errores cognitivos**: Ocurren cuando el usuario no conoce la manera en cómo se escribe las palabras (simplemente ignorancia). En muchos casos esto tiene que ver la manera en cómo el usuario pronuncia las palabras [8].

Existen dos tipos de software: los verificadores ortográficos y los correctores ortográficos. Un verificador tiene una tarea relativamente sencilla: debe identificar, dado un archivo de texto (entrada), las palabras que están mal escritas. En cambio, un corrector ortográfico tiene que hacer dos cosas: detectar las palabras mal escritas y tratar de hallar cómo se escriben dichas palabras correctamente [12].

Existen muchas técnicas para verificar y corregir un texto. Una de las más usadas es la búsqueda de las palabras en un diccionario (dictionary lookup) [2], cuya única dificultad consiste en tener un diccionario de palabras lo suficientemente grande de forma que pueda abarcar gran parte del idioma de los usuarios escritores. Normalmente se usa un algoritmo de búsqueda binaria, por ser muy rápido y eficiente [13], aunque también hay esquemas como las búsquedas por hash [4].

Por otra parte, la idea de utilizar reglas ortográficas no es nueva, pero aparentemente sólo se ha aplicado a idiomas como el inglés [9] o incluso sueco [1, 5]. Esto ocurre igualmente en lo que se refiere a crear sistemas expertos para esquemas de corrección ortográfica [14, 11].

4. El sistema experto de correción ortográfica por reglas

Si consideramos que un sistema experto puede validar la idea de hacer corrección ortográfica usando reglas, entonces debemos describir los elementos de lo que consta el software en cuestión.

4.1. Descripción de las reglas en Prolog (base de conocimientos)

Tenemos tres posibilidades en las reglas ortográficas[1].

[1] https://www.urosario.edu.co/Centro-Multicultural-y-Multilingue-old/Centro-de-Lectura-y-Escritura-en-Espanol/Material-de-Apoyo/Material-de-Apoyo/Material-de-apoyo-Uso-de-las-letras-B-y-V/

Palabras que **contienen** una o varias letras específicas:

Se escriben con *B*, las palabras que lleven *rr* en su escritura.

Excepciones: ferroviario, corrosivo, verruga, correctivo, verrojo.
Ejemplos: barrer, arrabal, borrador, becerro, berrear, burro.

Palabras que **terminan** con una o varias letras específicas:

Se escriben con *B*, las terminaciones en *bundo, bunda, bilidad*.

Excepciones: movilidad, civilidad.
Ejemplos: vagabundo, nauseabunda, amabilidad, afabilidad, habilidad.

Palabras que **empiezan** con una o varias letras específicas:

Se escriben con *B*, las palabras que comienzan con *bur, bus, buz*.

Ejemplos: burla, buzo, buscar, buzón, burócrata, busto.

Esto nos hace pensar que en muchas ocasiones no hay necesidad de revisar cada palabra (candidata a una corrección), usando un diccionario, sino que bien podríamos aplicar la regla ortográfica para hallar si está bien o mal escrita. El español, como todo lenguaje humano es cambiante.

Palabras, expresiones y reglas que se usaban en el pasado son obsoletas ahora y considerando esto, se decidió mantener las reglas ortográficas en un archivo especial, el cual es consultado cada vez que el sistema se ejecuta. Este archivo, en el software creado, se denomina Reglas.DB, y contiene alrededor de 260 reglas ortográficas de uso común en el idioma español. Las reglas pueden ser editadas con cualquier procesador de palabras (o editor de textos) que utilice el formato ASCII sin caracteres de control o símbolos especiales.

Por ejemplo, el block de notas. Cada una de las reglas ocupa un renglón en el archivo Reglas.Db. Así entonces, escribir una regla nueva significa, finalmente, agregar una línea más al archivo ya mencionado. Las reglas ortográficas, para que puedan ser entendidas por el programa, requieren estar en un formato específico para poder ser leídas por el sistema. Las reglas ortográficas tienen tres posibles alternativas, las cuales pueden aplicar a:

– Prefijo (p) de la palabra analizada (parte inicial de la palabra en cuestión).
– Sufijo (s) de la palabra analizada (parte final de la palabra en cuestión).
– Subcadena (sb) de la palabra analizada (en cualquier parte de la palabra en cuestión).

Las letras “p”, “s”, y “sb” corresponden respectivamente a prefijo, sufijo y subcadena, y estas letras serán usadas para informarle al corrector en qué parte de la palabra se aplica la regla ortográfica.

La regla entonces sigue el siguiente formato:

letra palabra clave lista

Cuya descripción puede verse aquí:

- **letra**. Indica la letra a la cual se aplica la regla. Por ejemplo, si la regla ortográfica es sobre el uso de la v, éste es el parámetro que se utiliza. (Véanse los ejemplos más adelante).
- **palabra**. Indica la palabra que no cumple precisamente con la regla que está siendo definida, es decir, muestra el ejemplo de la contraposición a la regla ortográfica misma.
- **clave**. Es exactamente lo que indica el alcance de aplicación de la regla (prefijo, sufijo o subcadena). Utilícese solamente las siguientes palabras claves (entre doble comillas: p, s, sb).
- **lista**. Se refiere a la lista de palabras que son la excepción a la regla en cuestión. Tales palabras deben aparecer entrecomilladas y separadas por comas.

Algunos ejemplos ilustrativos. Considérese la siguiente regla:

Las palabras que empiezan con env se escriben siempre con v

La regla en el archivo Reglas.DB se escribirá entonces así:

v enb p

El software reconoce la regla de la siguiente manera: Las excepciones a las palabras que empiezan con *env* nada más pueden ser aquellas que comienzan con *enb*, que en este caso no hay tales excepciones a la regla y el rango de aplicación de la misma es al inicio (por eso le denominamos prefijo) de las palabras. Ahora considérese la siguiente regla:

Todas las palabras que terminan con ave se escriben con v.

Esta regla puede expresarse en el lenguaje descrito de la siguiente forma:

v abe s árabe jarabe

La cual puede ser descrita de la siguiente manera: Las excepciones a las palabras que terminan en *ave* nada más pueden ser aquellas que terminan con *abe*, las cuales son, *árabe, jarabe* (y nada más), y el rango de aplicación de la regla son los sufijos de las palabras. Por último, un ejemplo de una regla que se refiera a subcadenas:

Las palabras que tienen dentro de ellas (en cualquier parte de la misma) las letras ilv se escriben siempre con v.

La cual se traduce en el archivo de reglas de la siguiente forma:

v ilb sb bilbao

Y se interpreta de la siguiente manera: Las excepciones a las palabras que tienen como subpalabra *ilv* se escriben siempre con *v*, excepto la palabra *bilbao*. Un fragmento del archivo de reglas es este (ya con la sintaxis legal de Prolog):

```
ex(“v”,“enb”,[],“p”)
ex(“v”,“db”,[],“sb”)
ex(“v”,“nb”,[],“sb”)
ex(“v”,“lb”,[“elba”],“sb”)
ex(“v”,“ilb”,[“bilbao”],“sb”)
ex(“v”,“clab”,[],“sb”)
ex(“v”,“mob”,[“mobiliario”],“sb”)
ex(“v”,“seb”,[“sebo”,“sebastián”,“sebastopol”],“p”)
```

Las excepciones se encuentran como una lista de Prolog tradicional, entre paréntesis cuadrados.

4.2. Mecanismo de inferencia

En un sistema de reglas, como en el caso que nos ocupa, el mecanismo de inferencia determina cómo utilizar las mismas. El mecanismo natural en Prolog es el encadenamiento hacia atrás, el cual se basa en plantear una hipótesis (lo que queremos demostrar), para así llegar a la conclusión deseada.

Por ejemplo, en un sistema de diagnóstico médico, un síntoma podría ser que el paciente tenga congestión nasal, lo cual se toma como la hipótesis. A partir de esto el sistema podría llegar a la conclusión de que el paciente tiene gripa si se demuestran otros síntomas, es decir, si se satisfacen los hechos en las reglas [6].

Para el caso de las reglas ortográficas, al recibirse la palabra a corregir, podemos empezar por la hipótesis (la regla), de que dicha palabra está mal escrita a partir de la primera regla en el archivo Reglas.DB. Si no cumple con las características de esa regla particular, el esquema de inferencia la desecha y continúa con la siguiente regla. Prolog siempre hace una búsqueda exhaustiva con todas las reglas definidas.

4.3. Interfaz con el usuario

El sistema experto de reglas incluyó originalmente un editor con las características del antiguo programa WordStar[2], el cual era el estándar de facto y que, además, era un predicado que venía incluido en Turbo Prolog 2.0[3]. Este software no trabaja en tiempo real, sino por lotes (batch), de forma que las correcciones quedan grabadas todas en un archivo de texto, el cual contiene las palabras ofensoras (o equivocadas), y lo que debe hacerse para corregirlas. Sin embargo, el sistema contempla una opción para hacer una pausa cada vez que se genera una corrección.

[2] WordStar fue un procesador de textos, incluido en las computadoras Osborne 1. En particular, WordStar fue el último procesador de textos comercial para el sistema operativo CP/M y fue lanzado en septiembre de 1978.

[3] Lo que quiere decir que un predicado de Turbo Prolog genera todo el editor de textos, similar al que se usa en el IDE de este sistema.

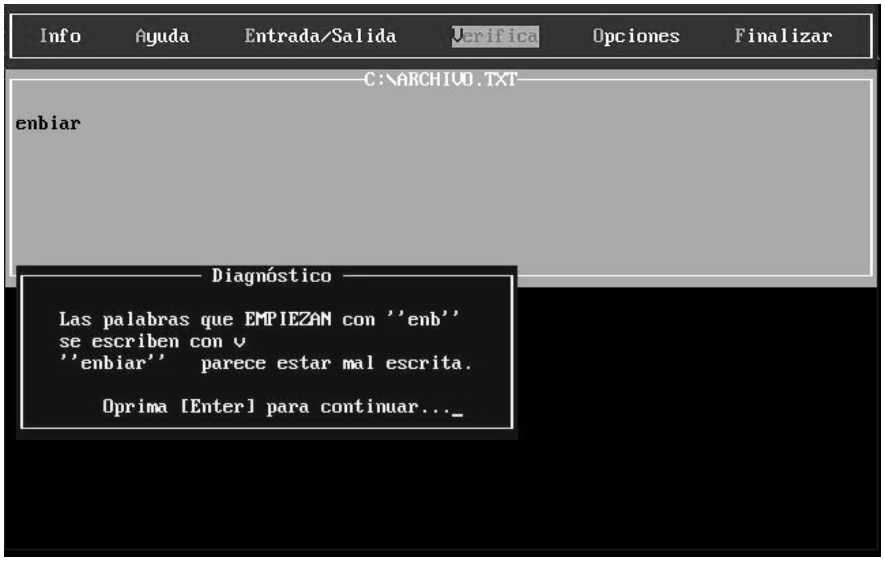

Fig. 1. La corrección que hace Prolog, indicando la regla usada.

Sin embargo, una vez que la prueba de concepto mostró el funcionamiento del corrector ortográfico usando reglas, se decidió implementar una versión que fuese capaz de trabajar con el estándar actual, Microsoft Word, a través de la comunicación entre procesos. Esto puede verse en la sección de resultados.

5. Resultados preliminares

Para las primeras pruebas, se escribió un programa en Turbo Prolog 2.0[4], el cual funciona bajo MsDOS (usando DOSBox)[5], pero que da algunas facilidades para depurar fácilmente el software. De hecho, todo el sistema experto de reglas ortográficas trabaja sobre archivos de texto ASCII sin caracteres de control. El software lee ese archivo de texto y lo separa, palabra por palabra, el cual se pasa como parámetro a la base de conocimiento de las reglas.

Entonces, mediante el encadenamiento hacia atrás, Prolog revisa la palabra y busca saber si está bien o mal escrita. En caso de hallar que la palabra esté mal escrita, el sistema mandará un mensaje que indica el error –de acuerdo a la regla– y cómo debe solucionarse. En caso de que la regla no se cumpla, a través del mecanismo del backtrack, se buscará si cumple con la siguiente regla y así sucesivamente.

Prolog revisa todos los nodos del árbol solución.

[4] Turbo Prolog 2.0 es un sistema completo de desarrollo de software que incluye un compilador y un entorno de desarrollo integrado (IDE) para el lenguaje de programación PROLOG, desarrollado por Borland.

[5] DOSBox es un emulador que recrea un entorno parecido al sistema DOS con el objetivo de poder ejecutar programas y videojuegos originalmente escritos para el sistema operativo MS-DOS de Microsoft en computadoras más modernas o en diferentes arquitecturas.

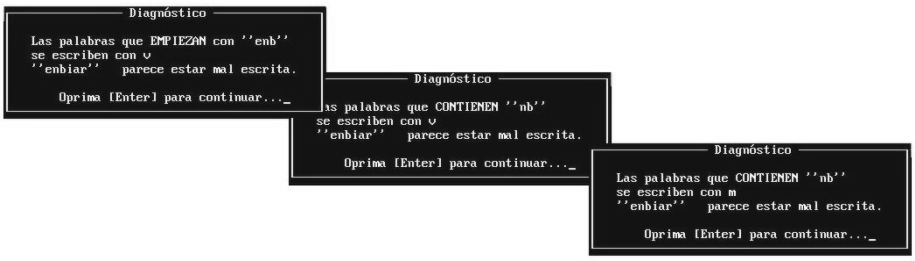

Fig. 2. Más de una regla se dispara con la palabra “enbiar”.

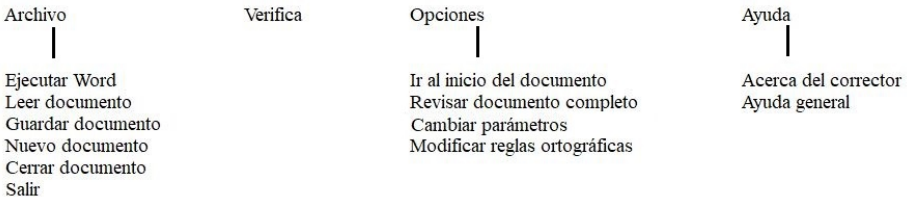

Fig. 3. La estructura del programa de corrección por reglas, en Windows.

La figura 1 muestra cómo el sistema hace la corrección ortográfica, indicando incluso el uso de la regla: Cabe señalar que más de una regla puede ser desplegada. Por ejemplo, en la figura 2 puede verse que la palabra “enbiar” consulta más de una regla.

5.1. Enlazando el corrector con MsWord en Windows

Escribir un procesador de palabras que contenga un corrector ortográfico es una labor que requiere de mucho trabajo. Además, es difícil que las personas cambien de software pues ya hay un apego y costumbre a usar lo que ya han aprendido. Hoy los usuarios procesan textos con algún programa de una suite de oficina.

El estándar es Microsoft y la mayoría de los usuarios usan Word para Windows. Considerando esto, la idea de probar el corrector ortográfico por reglas (bautizado como “Lapsus”), debiese hacerse enlazando éste a Word de alguna manera.

Afortunadamente Microsoft ofrece un mecanismo de comunicación entre su suite de oficina y aplicaciones de terceros. De esta manera, podemos pedirle a Word que mande las palabras a analizar al corrector ortográfico y este último le responde precisamente con los mensajes de corrección, palabra por palabra, en tiempo real si se desea.

Esto se realiza a través de la automatización OLE (Object Linking and Embedded), que es el mecanismo de comunicación entre dos procesos que corren bajo Windows. Para nuestro propósito, usamos OLE y Delphi[6]. Para ello nos basamos en los artículos de Ron Gray [7].

[6] Delphi es un entorno de desarrollo de software diseñado para la programación de propósito general con énfasis en la programación visual, basado en Object Pascal, ya con 26 años de existencia.

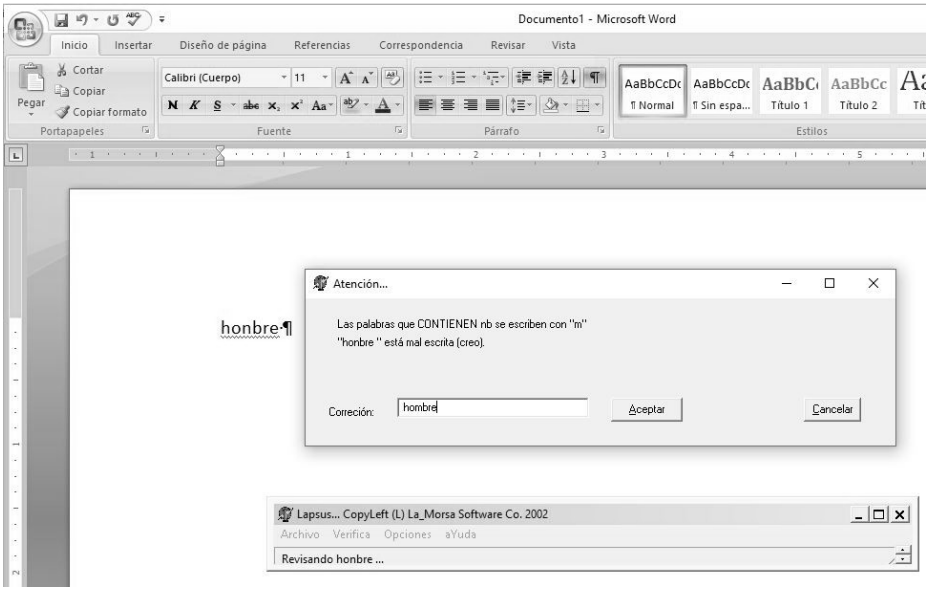

Fig. 4. Lapsus detectando el error en la palabra “honbre”.

En la figura 3, se describe la estructura del corrector Lapsus, el cual es un programa escrito en Delphi que se comunica con Word a través de la automatización OLE:

Hay que señalar que la implementación del sistema experto en Delphi requiere emular las características que tiene Prolog [15], de buscar exhaustivamente la solución, moviéndose por todos los nodos en el árbol de búsquedas, en este caso, en todas las reglas definidas en el sistema. Una vez implementada esta parte, se hicieron las pruebas correspondientes y en la figura 4 puede verse el corrector trabajando sobre la palabra “honbre”, evidentemente mal escrita.

6. Resultados, conclusiones y trabajo futuro

Se utilizó una computadora AMD Ryzen 5 PRO 1600 con seis procesadores, corriendo a 3.20 GHz y con 16 GB de RAM.

6.1. Resultados usando DosBox y Turbo Prolog 2.0

Para el primer lote de pruebas se usó DosBox (que emula MsDOS) y Turbo Prolog 2.0. Por ende, la memoria disponible no sobrepasa los 640 Kbytes y no hace uso de la memoria extendida. Se midieron tiempos de corrección en diferentes escenarios. Usando el texto:

“La noche estaba muy oscura. Tenía la ventana abierta y por ella me llegaba el resplandor de un anuncio de cerveza Superior. No se veían las estrellas, no había luna, nada más el rostro sonriente de la rubia de categoría, y fugaz intermitente, la luz roja

de una frase: 'La rubia que todos quieren es cerveza Superior'. La habitación, entre los ires y venires de la luz exterior, quedaba abandonada al humor pajizo de las menguadas veladoras. Del otro lado, en el cuarto contiguo, solamente se escuchaba el radio (ahora un locutor decía con voz trémula: 'Su programa favorito, Serenaataaa. Su estación, la B grande de México.'), de Sealtiel Alatriste[7], el sistema tardó 21.2 segundos, usando solamente un diccionario de 500 mil palabras, es decir sin utilizar las reglas ortográficas.

El sistema pudo revisar unas 5.09 palabras por segundo, teniendo el texto 107 palabras. Usando el mismo texto y solamente las reglas ortográficas, el sistema tardó 13.02 segundos, lo que representa 8.29 palabras revisadas por segundo.

Para probar la efectividad del corrector usando solamente reglas ortográficas, se encontró que el sistema pudo analizar –de un texto con 65 palabras mal escritas a propósito (todas con problemas en su construcción ortográfica)– en 12.03 segundos, logrando una efectividad de 5.40 palabras por segundo.

Sin embargo, haciendo el mismo proceso incluyendo el diccionario de 500 mil términos, el sistema tardó 27.62 segundos, logrando unas 2.35 palabras por segundo analizadas. Cabe decir que el total de reglas contempladas es de alrededor de 260, las cuales son aplicadas a cada palabra a corregir.

6.2. Resultados usando Windows 10, MsWord y Lapsus en Delphi

En el segundo lote de pruebas, usando el sistema escrito en Delphi (dentro de Windows 10), y conectado vía OLE con MsWord, se encontró que, para el texto de Alatriste –que consta de 107 palabras– el sistema pudo revisarlo en 5.04 segundos, usando solamente las reglas ortográficas. Esto muestra una eficiencia de 21.23 palabras analizadas por segundo.

Para el texto con 65 palabras mal escritas, es decir, que no cumplen con alguna de las reglas ortográficas, Lapsus tardó 2.77 segundos en hallar todos los problemas ortográficos usando solamente su conjunto de reglas. Se logró una eficiencia de unas 23.4 palabras por segundo en la versión para Windows.

Cabe señalar que en la versión de Lapsus para Windows, no se utiliza el diccionario de 500 mil términos, pues se usa el que viene integrado en Word. Se esperaba –como ocurrió– que este segundo lote de pruebas presentara mayor velocidad pues el sistema no está limitado por la memoria como en el caso de DosBox.

6.3. Conclusiones

La corrección de textos no es un asunto trivial, pues los idiomas contienen no solamente muchísimas palabras, sino que también hay giros idiomáticos, palabras que pueden considerarse erróneas (por ejemplo, "revolver" y "revólver", en donde dependiendo del contexto son palabras correctamente escritas), acentos diacríticos, etcétera. No obstante, lo que queda claro es que la corrección ortográfica basada solamente en un diccionario con cientos de miles de palabras no es suficiente.

[7] http://www.materialdelectura.unam.mx/index.php/cuento-contemporaneo/13-cuento-contemporaneo-cat/241-111-sealtiel-alatriste?start=4

Este es un primer enfoque que soluciona dificultades pero que requiere entender que se puede sacar ventaja de la estructura del propio idioma para hacer un corrector mucho más robusto. La corrección con reglas ortográficas para el idioma español es un primer paso a futuros correctores mucho más inteligentes.

6.4. Trabajo futuro

Un esquema híbrido parece ser mucho mejor idea: combinar un diccionario de muchísimos términos, además de diccionarios especializados (por ejemplo, de medicina, de términos legales), además de un esquema de verificación basándose en sílabas válidas en el español e incluso, el uso de diccionarios con palabras que no existen en el español (por ejemplo, nombres de empresas), pueden hacer más robusto el sistema.

Igualmente los futuros correctores deben analizar no solamente las palabras, sino frases enteras y si es necesario, sugerir o indicar errores. Es probable que un análisis estadístico de cómo se usan las palabras y los contextos en donde aparecen, puede ser un camino nuevo en la corrección de textos. La corrección ortográfica es sin duda un tema abierto en cómputo.

Referencias

1. Arppe, A.: Developing a grammar checker for swedish. In: Proceedings of NODALIDA, pp. 13–27 (1999)
2. Avarbuch, L. M.: Lookup: Interactive spell checker for single words. vol. 15, no. 1, pp. 33–34 (1991) doi: 10.1108/eb024363
3. Damerau, F. J.: A technique for computer detection and correction of spelling error. Communication ACM, vol. 7, no. 3, pp 171–176 (1964) doi: 10.1145/363958.363994
4. Damgard, I. B.: A design principle for hash functions. In: Brassard, G. (eds) Advances in Cryptology - CRYPTO' 89 Proceedings, CRYPTO 1989, Lecture Notes in Computer Science, vol 435, pp. 416–427 (1990) doi: 10.1007/0-387-34805-0_39
5. Domeij, R., Knutsson, O., Carlberger, J., Kann, V.: Granska: An efficient hybrid system for Swedish grammar checking. In: Proceedings of the 12th Nordic Conference of Computational Linguistics (1999), pp. 49–56 (2000)
6. Gary, R., Giarratano, J.: Expert Systems, principles and programming. PWS Publishing Company (1998)
7. Gray, R.: Word control. Delphi Informant Magazine, vol. 6, no. 9 (2000)
8. Kumar, R., Bala, M., Sourabh, K.: A study of spell checking techniques for Indian languages. JK Research Journal in Mathematics and Computer Sciences, vol. 1, no. 1 (2018)
9. Naber, D.: A rule-based style and grammar checker. Diplomarbeit Technische Fakultät, Universität Bielefeld (2003)
10. Ortografía Lengua Española. Reglas y Ejercicios, Larousse (2001)
11. QasemiZadeh, B., Ilkhani, A., Ganjeii, A.: Adaptive language independent spell checking using intelligent traverse on a tree. In: IEEE Conference on Cybernetics and Intelligent Systems, pp. 1–6 (2006) doi:10.1109/ICCIS.2006.252325
12. Peterson, J. L.: Computer programs for detecting and correcting spelling errors. Communications of the ACM, vol. 23, no. 12, pp 676–687 (1980) doi: 10.1145/359038.359041

13. R. Nowak: Generalized binary search. In: 46th Annual Allerton Conference on Communication, Control, and Computing, pp. 568–574 (2008) doi: 10.1109/ALLERTON.2008.4797609
14. Sokele, M., Dembitz, Š., Knežević, P.: Developing a spell checker as an expert system. Journal of Computing and Information Technology, vol. 11, no. 4 (2003) doi: 10.2498/cit.2003.04.03
15. Sawyer, B., Foster, D.: Programming expert systems in Pascal. Wiley (1986)