

Tratamiento Automático de Reglas Ortográficas para la Detección y Corrección de Errores

Brian Plüss Laura Pomponio

Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Departamento de Computación
{bpluss,pomponio}@fceia.unr.edu.ar

Resumen Este trabajo presenta una propuesta novedosa para la detección y corrección de errores ortográficos basada en la automatización de las reglas que rigen la ortografía de la lengua castellana. Para ello, se analiza un subconjunto de las reglas tal como aparecen en la literatura y se introducen los conceptos lingüísticos a los cuales las reglas hacen referencia. Se describen brevemente las técnicas computacionales existentes para el procesamiento de lenguajes naturales, explicando cómo éstas pueden ser utilizadas en la tarea de determinar si una palabra viola alguna de las reglas establecidas y corregir el error. La factibilidad de la propuesta es apoyada con la implementación de un prototipo escrito en Prolog y su utilidad expuesta con algunos ejemplos ilustrativos.

1. Introducción

La detección y corrección de errores ortográficos es un problema importante y cotidiano al que se enfrenta cualquier usuario al escribir un texto. En la actualidad, todos los procesadores de texto incluyen la utilidad de corrección ortográfica entre sus herramientas. También la encontraremos en navegadores, administradores de correo electrónico y prácticamente cualquier aplicación en la cual el usuario deba ingresar texto.

El enfoque, aparentemente universal, que tienen estas herramientas se centra en el uso de un diccionario. El método habitual comienza con la búsqueda de una palabra en el mismo y, si ésta es encontrada, se prosigue con la próxima; en caso contrario, se presenta al usuario una lista de palabras ordenadas de acuerdo a una determinada distancia entre la palabra analizada y las presentes en el diccionario, entre las cuales se espera esté la correcta.

Claramente, este método no hace uso alguno de las propiedades lingüísticas de la palabra y el contexto en que ésta es utilizada. Esto impide la detección de ciertos errores y produce falsos avisos cuando la palabra es correcta pero no se encuentra en el diccionario utilizado por el corrector. El primero de estos problemas fue descrito en [1] y se refiere a la imposibilidad de detectar el error cuando éste produce una palabra válida, distinta de la que el usuario deseaba escribir. Desde entonces, ha sido atacado, principalmente, mediante lo que se denomina *corrección ortográfica basada en contexto* [2,3] y otras técnicas como

aprendizaje automatizado [4] y, más recientemente, *cohesión léxica* [5]. Textos relevantes en este sentido son [6] y [7].

En castellano, el trabajo realizado al respecto no es tan voluminoso. En [8] se propone un entorno integrado para el tratamiento de textos en lengua castellana, pero la detección y corrección de errores ortográficos es tratada de la forma tradicional. Esta ausencia abre una posibilidad interesante, pues, a diferencia del inglés que es altamente irregular, la lengua castellana resuelve su ortografía con un número muy reducido de reglas que cubren la gran mayoría de los casos relevantes. Por ello, serán menos complejas las herramientas lingüísticas y computacionales necesarias para atacar el problema, con una tasa de éxito más elevada en los resultados.

En este trabajo proponemos la detección y corrección de errores ortográficos en castellano por medio de un conjunto de reglas que aprovechen las propiedades lingüísticas y contextuales de la palabra. Estas reglas no serán otras que las reglas ortográficas para la lengua castellana, que marcan una ortografía correcta según lo determina la Real Academia Española [9].

Se eligió *Prolog* para la implementación del prototipo que nos permite estudiar la viabilidad de la propuesta, en primer lugar, por la popularidad del lenguaje en el ámbito del *procesamiento de lenguajes naturales*¹ [10,11,12] y también por las virtudes que presenta la *programación lógica* para definir sistemas basados en reglas. Nuestro prototipo no apunta a brindar un sistema completo, sino que se restringe a mostrar cómo es posible atacar el problema en cuestión siguiendo la propuesta descripta, concentrándose sólo en su “núcleo”. Sin embargo, describiremos con algún detalle el resto del sistema, que estaría compuesto por un editor de textos, un analizador de formas de palabras, un separador en sílabas y una librería de manipulación de cadenas de caracteres. Para cada uno de ellos, daremos referencias a implementaciones disponibles y explicaremos brevemente su función y las interacciones con el resto del sistema.

En la siguiente sección nos referiremos a las reglas ortográficas. Analizaremos su naturaleza y los conceptos lingüísticos que las conforman. Además, definiremos el subconjunto de las mismas que consideraremos en el resto del trabajo. La sección 3 presentará, más en detalle, conceptos lingüísticos como palabra, forma de palabra, fonología, morfología, sintaxis, gramática, semántica y pragmática. Al mismo tiempo, se establecerá el rol que tendrá cada uno de estos conceptos en nuestra propuesta y demarcaremos el alcance del tratamiento que haremos de los mismos, relacionándolos con analizadores léxicos, sintácticos y semánticos ya desarrollados. La sección 4 expondrá el método global para la detección y corrección de errores ortográficos en un texto. Describiremos brevemente la arquitectura de un sistema que incluya un corrector ortográfico como el que proponemos y la forma en que éste invoca y se comunica con el motor de inferencia que realiza la corrección. En la sección 5 describiremos la implementación de las reglas seleccionadas y cómo se obtiene la información requerida en las premisas. Finalmente, las secciones 6 y 7 presentan algunos ejemplos y la conclusión, respectivamente.

¹ NLP, por sus siglas en inglés.

2. Reglas de Ortografía de la Lengua Española

La fuente por excelencia sobre la ortografía española es, naturalmente, “Ortografía de la Lengua Española” [9]. Este documento es revisado y actualizado por la Real Academia Española (RAE) en conjunto con las Academias de la Lengua Española de los demás países de habla castellana.

Una particularidad de esta publicación y las reglas que contiene es que las mismas están pensadas y formuladas para ser utilizadas por humanos. En general, esto constituiría un problema importante al momento de intentar formalizarlas en términos adecuados para su tratamiento computacional. En nuestro caso, por el contrario, éste es uno de los aspectos más interesantes de la formulación, pues pretendemos utilizarlas, tal y como están, como punto central de nuestra propuesta. Esto es, diseñar nuestro enfoque de forma que las mismas reglas utilizadas por humanos para detectar y corregir errores ortográficos, sirvan para que un programa lleve a cabo esa misma tarea.

2.1. Diccionarios vs. Reglas

Ponemos énfasis ahora, en la relevancia de la utilización de reglas en la corrección de errores ortográficos. La corrección mediante el uso de diccionarios parece ser más adecuada para la detección de errores tipográficos, en los cuales la palabra producida difícilmente se encuentre en el idioma en que se escribe. Por ejemplo, si en vez de “alcanzar” escribiéramos “aclanzar” la detección será exitosa en la inmensa mayoría de los casos. Sin embargo, este enfoque no ataca aquellas situaciones en que el error se debe a ambigüedades propias del idioma (por ejemplo, *halla*, *haya* o *allá*), o a la presencia de numerosas homofonías en la grafía (por ejemplo, *s-c-z*, *v-b*, etc).

Las reglas ortográficas apuntan directamente en esta dirección, resolviendo tales ambigüedades mediante el uso de información lingüística de contexto, que quien escribe conoce al momento de producir el texto. De esta forma, con el aprendizaje y aplicación de un número reducido de reglas, deja de ser necesario conocer de memoria cómo se escribe cada una de las palabras del idioma para producir textos ortográficamente correctos.

2.2. Estructura y Naturaleza de las Reglas

El documento de la RAE es completo en el sentido que cubre la totalidad de las reglas para la escritura del castellano. Posee capítulos para el uso de las diversas letras, de mayúsculas y minúsculas, de acentuación, de los signos de puntuación y de abreviaturas. Nos interesa centrar nuestro trabajo en el primero de estos grupos, por considerarlo más interesante y de mayor relevancia al momento de detectar posibles errores, a la vez que constituye un subconjunto bien delimitado sobre el cual basar nuestro estudio.

El uso de las letras se trata por grupos que conllevan ambigüedad por sonar de igual o similar forma al hablar, introduciendo así cierta distancia entre la grafía

del idioma y sus distintas versiones orales, que responde a motivos históricos, dialectales y de evolución. Los grupos se listan a continuación:

- Letras b, v, w
- Letras g, j
- Letras i, y, ll
- Letra ñ
- Letra r; dígrafo rr
- Letra x
- Letras c, k, q, z; dígrafo ch
- Letra h
- Letra m
- Letra p
- Letra t
- Peculiaridades de las voces de otras lenguas y de los nombres propios

Cada grupo comienza con una breve introducción a las dificultades presentadas por las letras involucradas al momento de su uso, algunas notas históricas y luego, por cada letra, una lista de reglas que pretenden servir de orientación para el uso correcto de la misma. Consideramos el primer grupo lo suficientemente interesante y rico como para ilustrar completamente el enfoque que proponemos. Es por esto que lo utilizaremos a lo largo de todo el trabajo e incluso es el subconjunto de reglas elegido para llevar a cabo la implementación. Transcribimos abajo la introducción y las reglas a utilizar.

2.1. Letras *b*, *v*, *w*

En la mayor parte de España y en la totalidad de Hispanoamérica, las letras *b*, *v*, y a veces la *w*, representan hoy el mismo fonema labial sonoro, lo que origina numerosas dudas sobre su escritura. Estas son aún mayores en el caso de las palabras homófonas, porque en ellas el empleo de una u otra letra diferencia significados (por ejemplo, *baca/vaca*). Caso aparte es el de los nombres propios, en los que el uso arbitrario de *b* o *v* parece un resto del trueque de estas letras en siglos pasados. Así, *Balbuena/Valbuena* o *Tobar/Tovar*.

2.1.1. Letra *b*

La letra *b* siempre representa el fonema labial sonoro de **barco**, **beso**, **blusa** o **abuelo**.

Notas orientadoras sobre el uso de la letra *b*

Se escriben con *b*:

- a. Los verbos terminados en *-bir*. Ejemplos: *escribir*, *recibir*, *sumbir*. Excepciones en voces de uso actual: *hervir*, *servir*, *vivir* y sus compuestos.
- b. Los verbos terminados en *-buir*. Ejemplos: *contribuir*, *atribuir*, *retribuir*.
- c. Los verbos *deber*, *beber*, *caber*, *saber* y *haber*.
- d. Las terminaciones *-aba*, *-abas*, *-ábamos*, *-abais*, *-aban* del pretérito imperfecto de indicativo de los verbos de la primera conjugación. Ejemplos: *cantaba*, *bajabas*, *amaban*.
- e. El pretérito imperfecto de indicativo de ir: *iba*, *ibas*, etc.

- f. Las palabras que empiezan por el elemento compositivo *biblio-* ('libro') o por las sílabas *bu-*, *bur-* y *bus-*. Ejemplos: *biblioteca*, *bula*, *burla*, *buscar*. Excepción: *vudú* y sus derivados.
- g. Las que empiezan por el elemento compositivo *bi-*, *bis-*, *biz-* ('dos' o 'dos veces'). Ejemplos: *bipolar*, *bisnieto*, *bizcocho*.
- h. Las que contienen el elemento compositivo *bio-*, *-bio* ('vida'). Ejemplos: *biografía*, *biosfera*, *anaerobio*, *microbio*.
- i. Las palabras compuestas cuyo primer elemento es *bien* o su forma latina *bene*. Ejemplos: *bienaventurado*, *bienvenido*, *beneplácito*.
- j. Toda palabra en que el fonema labial sonoro precede a otra consonante o está en final de palabra. Ejemplos: *abdicación*, *abnegación*, *absolver*, *obtener*, *obvio*, *subvenir*, *amable*, *brazo*, *rob*, *nabab*. Excepciones: *ovni* y algunos términos desusados. En las palabras *oscuro*, *subscribir*, *substancia*, *substitución*, *substraer* y sus compuestos y derivados, el grupo *-bs-* se simplifica en *s*. Ejemplos: *sustancia*, *sustantivo*, *oscuro*.
- k. Las palabras acabadas en *-bilidad*. Ejemplos: *amabilidad*, *habilidad*, *posibilidad*. Excepciones: *movilidad*, *civilidad* y sus compuestos.
- l. Las acabadas en *-bundo* y *-bunda*. Ejemplos: *tremebundo*, *vagabundo*, *abunda*.

2.1.2. Letra v

La letra *v* siempre representa el fonema labial sonoro de *vaso*, *vida*, *invadir* o *cavar*.

Notas orientadoras sobre el uso de la letra v

Se escriben con *v*:

- a. Las palabras en las que las sílabas *ad-*, *sub-* y *ob-* preceden al fonema labial sonoro. Ejemplos: *adviento*, *subvención*, *obvio*.
- b. Las palabras que empiezan por *eva-*, *eve-*, *evi-* y *evo-*. Ejemplos: *evasión*, *eventual*, *evitar*, *evolución*. Excepciones: *ébano* y sus derivados, *ebionita*, *ebonita* y *eborario*.
- c. Las que empiezan por el elemento compositivo *vice-*, *viz-* o *vi-* ('en lugar de'). Ejemplos: *vicealmirante*, *vizconde*, *virrey*.
- d. Los adjetivos llanos terminados en *-avo*, *-ava*, *-evo*, *-eva*, *-eve*, *-ivo*, *-iva*. Ejemplos: *esclavo*, *octava*, *longevo*, *nueva*, *aleve*, *decisiva*, *activo*. Excepciones: *suabo* y *mancebo*.
- e. Las voces llanas de uso general terminadas en *-viro*, *-vira*, como *decenviro*, *Elvira*, *triunviro*, y las esdrújulas terminadas en *-ívoro*, *-ívora*, como *carnívora*, *herbívoro*, *insectívoro*. Excepción: *víbora*.
- f. Los verbos acabados en *-olver*. Ejemplos: *absolver*, *disolver*, *volver*.
- g. Los presentes de indicativo, imperativo y subjuntivo del verbo *ir*. Ejemplos: *voy*, *ve*, *vaya*.
- h. El pretérito perfecto simple de indicativo y el pretérito imperfecto y futuro de subjuntivo de los verbos *estar*, *andar*, *tener* y sus compuestos. Ejemplos: *estuvo*; *desanduvo*; *retuvo*, *contuviese*.

2.3. ¿Qué Dicen las Reglas y Cómo lo Dicen?

Analicemos algunas reglas para ver qué información contienen y en términos de qué conceptos lingüísticos la presentan. Estos últimos sólo serán mencionados aquí y sobre ellos nos extenderemos en la próxima sección.

2.1.1. b. (Se escriben con *b*) Las terminaciones *-aba*, *-abas*, *-ábamos*, *-abais*, *-aban* del pretérito imperfecto de indicativo (copretérito, en la terminología de Andrés Bello) de los verbos de la primera conjugación. Ejemplos: *cantaba*, *bajabas*, *amaban*.

Esta regla hace referencia a terminaciones o inflexiones para la conjugación de los verbos en cierto tiempo. Estos son conceptos pertenecientes al ámbito de la **morfología**.

2.1.1. h. (Se escriben con *b*) Las que contienen el elemento compositivo *bio-*, *-bio* ('vida'). Ejemplos: *biografía*, *biosfera*, *anaerobio*, *microbio*.

Si bien aquí la morfología también está presente al mencionarse el concepto de elemento compositivo, debemos notar la referencia al significado del mismo ('vida'). Aquí se habla de **semántica**.

2.1.1. j. (Se escriben con *b*) Toda palabra en que el fonema labial sonoro precede a otra consonante o está en final de palabra. Ejemplos: *abdicación*, *abnegación*, *absolver*, *obtener*, *obvio*, *subvenir*, *amable*, *brazo*, *rob*, *nabab*. Excepciones: *ovni* y algunos términos desusados. En las palabras *oscuro*, *subscribir*, *substancia*, *substitución*, *substraer* y sus compuestos y derivados, el grupo *-bs-* se simplifica en *s*. Ejemplos: *sustancia*, *sustantivo*, *oscuro*.

Aquí sólo se hace referencia a posiciones relativas de fonemas entre sí y dentro de la palabra. Estas cuestiones conforman el objeto de estudio de la **fonología**.

Por último, veamos conjuntamente estas dos reglas:

2.1.1. g. (Se escriben con *b*) Las que empiezan por el elemento compositivo *bi-*, *bis-*, *biz-* ('dos' o 'dos veces'). Ejemplos: *bipolar*, *bisnieto*, *bizcocho*.

2.1.2. c. (Se escriben con *v*) Las que empiezan por el elemento compositivo *vice-*, *viz-* o *vi-* ('en lugar de'). Ejemplos: *vicealmirante*, *vizconde*, *virrey*.

Aquí también aparece información morfológica, pues se trata de elementos compositivos, pero debemos notar que en ambas reglas están las sílabas homófonas *biz-/viz-* y *bi-/vi-*, con respectivas referencias a su semántica. Este tipo de ambigüedades, al momento de aplicar la regla, nos coloca en el campo de la **pragmática**, esto es, el significado en contexto o debido al uso.

3. Conceptos de la Lingüística General

En esta sección presentaremos uno a uno los conceptos lingüísticos relevantes al trabajo, junto a una breve reseña del estado del arte en cuanto al tratamiento computacional de los mismos. Para las definiciones utilizaremos la fuente seminal en lingüística general [13] y para el enfoque computacional nos basaremos levemente en [14].

3.1. Fonología

Muchas veces considerada una ciencia auxiliar de la lingüística, trata de la evolución histórica de los sonidos presentes en un idioma. También estudia los vínculos entre los sonidos de una lengua, diferenciando aquellas secuencias o conjuntos que están presentes en un idioma de los que no lo están.

A la fonología pertenecen los conceptos de *fonema* y *sílaba*. El primero es la unidad básica de conformación de la lengua. El segundo es un conjunto de fonemas cuya interrelación los coloca como pertenecientes a una determinada lengua.

El concepto de sílaba es fundamental al momento de observar las reglas ortográficas. También es, en muchos casos, la base con la que operan los afijos y elementos compositivos de la morfología, como veremos en seguida. No creemos conveniente entrar en demasiado detalle sobre teoría fonológica y formación o análisis silábico. Basta mencionar que la *estructura silábica* es el conjunto de restricciones propias de cada lengua sobre qué fonemas o grupos de fonemas pueden ocupar ciertas posiciones. La división en sílabas es un fenómeno acústico que tiene que ver con la métrica del lenguaje, aunque en casos como el inglés, en que la grafía y los sonidos del idioma son tan dispares, existe una división silábica escrita y otra oral. Este fenómeno no ocurre en castellano, cuya estructura silábica es, además, relativamente simple, contándose con numerosas herramientas capaces de separar en sílabas cualquier palabra, por ejemplo [15].

Si bien, la separación en sílabas constituiría un paso ineludible previo a la aplicación de las reglas ortográficas, por razones de tiempo y alcance, no incluimos un silabeador automático en nuestro trabajo. Sin embargo, sería posible utilizar cualquiera que contase con una interfase de programación de aplicaciones² a la cual llamar con la palabra bajo análisis, antes de someterla al corrector.

3.2. Morfología

Estudia la estructura interna de las palabras, sus diversas categorías (verbos, sustantivos, adjetivos, etc.), los distintos tipos de flexión (conjugación, declinación) y la formación de nuevas palabras por derivación o composición. Conceptos fundamentales aquí son los de *palabra* y *forma de palabra*.

Una palabra o *lema* es una unidad, una representación abstracta, canónica, cuyo uso concreto se da por medio de formas de palabras. Una forma de palabra se obtienen tanto por flexión como por derivación o composición, por lo cual resulta un concepto adecuado para referirnos a la familia completa de una palabra.

Existen herramientas que, dada una forma de palabra, obtienen su categoría y el lema al cual ésta está relacionada [16,17]. Estos procesos se denominan, respectivamente *categorización* y *lematización*. Varias de las reglas presentadas tienen que ver con la categoría de una palabra o si ésta está, o no, vinculada con

² API, por sus siglas en inglés.

otra. El proceso de lematización provee esa información al momento de aplicar las reglas.

3.3. Sintaxis

Se encarga del estudio de las reglas que determinan la combinatoria de formas de palabras en estructuras superiores como *sintagmas* u oraciones. Un sintagma es una unidad sintáctica, por ejemplo, frase nominal.

La sintaxis no aparece directamente en ninguna de las reglas presentadas, pero es fundamental al momento de analizar una oración tratando de determinar la clase de una forma de palabra en caso de ambigüedades. También, como veremos, es imprescindible para un correcto análisis semántico y pragmático.

En este caso existen también herramientas y técnicas que permiten analizar sintácticamente una oración, generalmente, determinando si es gramaticalmente correcta o no [18]. La información obtenida en estos casos puede ser útil para la aplicación de ciertas reglas o para la determinación de valores semánticos, pragmáticos o morfológicos.

3.4. Semántica

Es la ciencia de los significados literales. Hablamos de semántica *léxica* cuando se trata de una palabra y *composicional* cuando intervienen en la conformación del significado relaciones sintácticas. Computacionalmente, la semántica se trata del mapeo entre palabras (o composiciones gramaticalmente correctas de las mismas) y conceptos expresados en un lenguaje independiente del idioma, esto es, en un marco cognitivo determinado.

Naturalmente, para esta etapa se utilizan diccionarios que posibilitan el mapeo de base, pero las relaciones sintácticas y las transformaciones morfológicas van modificando ese significado tornándolo más específico. Las herramientas capaces de un procesamiento semántico completo aún constituyen un asunto pendiente y son objeto de estudio de la lingüística computacional [19,12]. En determinados dominios hay avances importantes y, de todos modos, nuestro problema no requiere un tratamiento total de la semántica. Bastará con poder determinar el significado de ciertas palabras, una vez lematizadas sus formas, como es el caso de las reglas vistas en las cuales las palabras deben estar relacionadas con conceptos como ‘vida’, ‘doble’ o ‘en lugar de’.

3.5. Pragmática

Es el estudio del uso de las expresiones ya sintáctica y semánticamente analizadas. Trata del lenguaje en uso, los conceptos relacionados entre sí conformando un contexto de interpretación, recíproco y de base. Es también en este punto donde la lingüística computacional incluye el *sentido común* o los conocimientos adquiridos históricamente. Constituye el desafío más grande en el tratamiento computacional, puesto que requiere modelos cognitivos complejos y a la vez consistentes.

El concepto central a la pragmática computacional es la *inferencia* [20]. Según [21], son cuatro los problemas inferenciales que constituyen el núcleo central del estudio actual en el área: *resolución de referencias*, interpretación y generación de *actos de habla* (o *ilocutivos*), interpretación y generación de *estructuras del discurso y relaciones de coherencia* y, por último, *abducción*. Referimos al lector a la bibliografía mencionada para ampliar cualquiera de estos conceptos. Basta notar aquí que todas estas tareas tienen que ver con la obtención de información que no está presente en la semántica asociada a una palabra o estructura sintagmática; información que permite determinar con mayor claridad a qué se refiere la palabra en cuestión.

El último ejemplo de las reglas citadas en la sección 2.3 muestra con claridad la utilidad de la información de contexto para la aplicación de las reglas. Mediante un análisis pragmático sería posible determinar si la palabra analizada está vinculada al concepto de ‘dos’ o al de ‘en lugar de’, permitiendo así la aplicación de la regla correspondiente.

4. Arquitectura Propuesta y *Workflow*

En esta sección presentaremos la arquitectura completa del enfoque, haciendo énfasis en la relación del corrector con el editor de textos y con las demás herramientas que lo asistirían en su trabajo. Para esto, consideremos la arquitectura bosquejada en la Figura 1.

En primer lugar, tenemos un **Editor de Textos** con el *buffer* conteniendo el texto a analizar. Puede tratarse de un editor ya implementado que soporte extensiones (*plugins*) o de uno nuevo que acceda explícitamente a las funciones del corrector. La primera opción es atractiva, pues prácticamente todos los editores populares brindan la posibilidad de anexar nuevas funcionalidades mediante la implementación de pequeños módulos (por ejemplo, el procesador de textos de OpenOffice.org, jEdit, los editores de la Plataforma Eclipse, EditPlus, emacs). Precisamente, ésta es la función del módulo llamado **Verificador**. En general, estará escrito en alguno de los lenguajes soportados para la creación de plugins en el editor que se quiera utilizar y podrá hacer de frontera entre este último y las demás funcionalidades. Será el encargado de solicitar los servicios del corrector, brindándole la información proveniente del buffer que el mismo requiera, en la forma adecuada. Para ello podrá utilizar herramientas auxiliares, como *tokenizers*, que eliminen signos de puntuación; alguna función de librería, que lleve las palabras a minúsculas; un silabeador, que separe en sílabas cada una de las palabras; por nombrar sólo algunas. Estas herramientas son independientes del editor de textos elegido y, de contar con vías de acceso adecuadas (APIs), también del lenguaje en que se programe el plugin.

El **Corrector** responderá los pedidos del verificador y utilizará las reglas en la **BC** para tratar cada palabra. Para aquellas reglas en las que necesite información lingüística compleja, podrá acceder a herramientas externas que se la provean. Para análisis fonológicos y morfológicos (lematización y categorización) bastará la palabra bajo inspección, pero en casos de desambiguación semántica

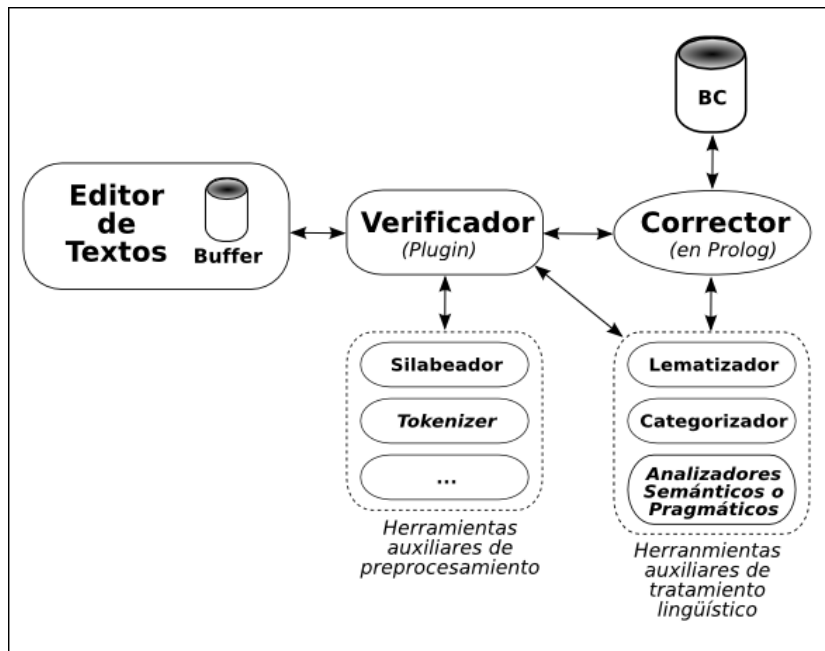


Figura 1. Arquitectura Propuesta

o pragmática, puede que las respectivas herramientas necesiten contexto, para lo cual podrán comunicarse con el verificador (posiblemente a través del corrector), pues éste tendrá acceso total al buffer con el texto.

5. Implementación del Prototipo

En esta sección describiremos la implementación del prototipo, limitándonos a tratar las reglas vinculadas a *b* y *v* presentadas anteriormente.

El sistema incluye la base de conocimiento (BC) con las reglas, un mecanismo de inferencia y sólo algunas de las herramientas auxiliares parciales, implementadas *ad-hoc*. Por razones de alcance, no se intentó vincular el mismo a ninguna de las herramientas de procesamiento lingüístico mencionadas en las secciones anteriores.

5.1. Clasificación de Reglas y Base de Conocimiento

Siguiendo el análisis de las reglas mostrado en 2.3, las mismas fueron clasificadas para la definición de la BC y para determinar la información que el sistema requerirá del usuario. Entre las que son puramente fonológicas, aquellas que tienen que ver con la posición relativa de los fonemas formarán parte de la BC del sistema sin que éste requiera del usuario más información, por ejemplo:

2.1.1. j. (Se escribe con *b*) Toda palabra en que el fonema labial sonoro precede a otra consonante o está en final de palabra. Ejemplos: *abdicación, abnegación, absolver, obtener, obvio, subvenir, amable, brazo, rob, nabab*. Excepciones: *ovni* y algunos términos desusados.

En cambio, al aplicar reglas que utilizan información morfológica, tanto aquellas en las que será necesario determinar si la palabra a tratar es un verbo, un adjetivo o un sustantivo, como en las que se debe saber si la palabra es derivada de alguna otra, será requerida la intervención del usuario para contar con más información y poder así tomar decisiones. Por ejemplo, considérese la regla:

2.1.2. g. (Se escribe con *v*) Los presentes de indicativo, imperativo y subjuntivo del verbo *ir*. Ejemplos: *voy, ve, vaya*.

Si la palabra a analizar fuese ‘baya’ podría tratarse de cierto sustantivo (que enuncia un fruto) y en tal caso no presentar errores, o bien podría tratarse de una conjugación del verbo ‘ir’ que debió ser escrita con *v*. En esta instancia, el sistema requerirá la intervención del usuario.

De la misma manera, para las reglas de la BC que vinculen aspectos fonológicos o morfológicos simples con información más compleja (del área de la semántica o la pragmática), el sistema podrá detectar los primeros pero efectuará consultas al usuario relativas a los últimos. Retomemos aquí el ejemplo visto anteriormente:

2.1.1. g. (Se escriben con *b*) Las que empiezan por el elemento compositivo *bi-*, *bis-*, *biz-* (‘dos’ o ‘dos veces’). Ejemplos: *bipolar, bisnieto, bizcocho*.

2.1.2. c. (Se escriben con *v*) Las que empiezan por el elemento compositivo *vice-*, *viz-* o *vi-* (‘en lugar de’). Ejemplos: *vicealmirante, vizconde, virrey*.

En este caso es evidente que, si una palabra comienza con *bi-*, *biz-*, *vi-* o *viz-*, el sistema lo detectará pero requerirá de más información para decidir si existe o no una falta ortográfica. Para esto, consultará al usuario sobre los aspectos pragmáticos del caso, solicitándole que determine si el elemento compositivo que inicia la palabra pretende significar ‘dos veces’, o bien, ‘en lugar de’. Reiteramos que esta información podría ser obtenida automáticamente utilizando herramientas de silabeo, lematización, categorización e, incluso, desambiguadores que hagan uso de contexto para cuestiones pragmáticas.

5.2. Descripción de Módulos

Nuestro prototipo está conformado por los archivos `corrector.pl`, `reglas.pl`, `excepciones.pl`, `letras.pl` y `util.pl`. La BC está constituida por `reglas.pl`, `excepciones.pl` y `letras.pl`, donde:

- `letras.pl` contiene predicados utilizados por las reglas, que indican qué letras son vocales y cuáles son consonantes (por ej., `vocal(u)`, `consonante(m)`).

- `excepciones.pl` contiene los predicados que definen las excepciones existentes para cada una de las reglas (por ejemplo, `excepcion(ébano)`, `excepcion(movilidad)`). Este conocimiento será el primero que utilizará nuestro sistema ante una palabra a analizar.
- `reglas.pl` contiene los predicados que indican qué es una regla asociada a b o a v y qué forma tiene cada una. Por ejemplo, estarán presentes `regla_b(rb4)`, que indica que `rb4` es una regla asociada a b , y la definición de `rb4` mostrada más adelante. De forma análoga, están aquellos predicados vinculados a las reglas de la v como `regla_v(rv7)` con su correspondiente definición.

La parte central del prototipo está en `corrector.pl` que utilizará `reglas.pl` y `excepciones.pl` para tratar cada palabra presente en los datos de entrada.

Es importante mencionar que el corrector es lo suficientemente genérico como para permitir que se agreguen o quiten reglas de los mismos grupos de letras; o bien, sus formas o nombres sean modificados, siendo esto totalmente transparente para él.

5.3. ¿Cómo Funciona el Corrector?

El sistema toma una lista de palabras y tratará aquellas en las que detecte la existencia de b o v , analizando y, de ser necesario, corrigiendo cada una, antes de pasar a la próxima. Cuando el corrector determina que debe tratar cierta palabra, primero verifica que no sea una excepción de alguna de las reglas. Si éste fuera el caso, la palabra es pasada por alto; en caso contrario, el sistema aplicará las reglas según correspondan. Si el corrector encuentra que existe una b y/o una v , y no está ante una excepción, asume a priori que estas letras han sido usadas en forma incorrecta e intentará determinar qué reglas se están violando para hacer la corrección. Ante una palabra escrita con v , se usarán las reglas de la b para ver si se está quebrantando alguna de ellas y lo análogo ocurrirá si se está ante la presencia de una b . En la Figura 2, vemos el código que implementa esta funcionalidad.

El motor de inferencia de *Prolog* unificará la palabra a tratar con X y ésta ya corregida con Q . De este modo, si por ejemplo existe una v en la palabra, el predicado `exist_v(X)` será válido; entonces cada nombre de regla, correspondiente a b , será unificado con `Regla` por medio de `regla_b(Regla)` (por ejemplo, `Regla` se unificará con `rb4`). Luego, se evaluará el predicado definido en `aplicar_reglas(Regla,Z,Y,T)`.

Una de las bondades de este lenguaje de programación lógica es la posibilidad de, en forma dinámica, construir y evaluar predicados. Así es que, la expresión `T =.. [Regla,Z,Y]` construye un predicado que se unifica con T , el cual luego será evaluado. Este predicado tendrá la estructura `Regla(Z,Y)` donde `Regla`, Z y Y serán unificados con ciertos valores. Por ejemplo, podrá construirse dinámicamente y evaluarse, el predicado `rb4(avalara, Y)` para aplicar la regla `rb4` a la palabra `avalara`. De esta manera serán construidos, durante el proceso de inferencia, los predicados asociados a las reglas que deban aplicarse en cada situación.

```

aplicar(X,Q):- (excepcion(X) -> (Q = X,
                                write('Es excepción : \'\"'),
                                write(Q),
                                write('\".'),nl,!
                                )
              )
;
((retract(buffer(_)) -> true; true),
 assert(buffer(X)),
 (exist_v(X) -> ((regla_b(Regla),
                  aplicar_reglas(Regla,Z,Y,T)
                  )
                ;
                (exist_b(X) -> ((regla_v(Regla),
                                  aplicar_reglas(Regla,Z,Y,T)
                                  )
                                ; %%Si exist_v y exist_b
                                buffer(Q)
                                )
                              ; %%Si exist_v y NO exist_b
                              buffer(Q)
                              )
                )
                ; %% Si NO exist_v
                (exist_b(X) -> ((regla_v(Regla),
                                  aplicar_reglas(Regla,Z,Y,T)
                                  )
                                ; %%Si NO exist_v y exist_b
                                buffer(Q)
                                )
                              ; %% Si NO exist_v y NO exist_b
                              Q=X
                              )
                )
              )
).

aplicar_reglas(Regla,Z,Y,T):- buffer(Z),
                              T =.. [Regla,Z,Y], T,
                              write('Aplicar regla a \\'\"'),write(Z),
                              write('\\" : '), write(T),nl,
                              retract(buffer(Z)),assert(buffer(Y)),fail.

```

Figura 2. Código de la función de aplicación de reglas

Focalizando ahora en cómo se implementaron las reglas ortográficas, damos como ejemplo la que sigue:

2.1.1. k. (Se escriben con *b*) Las palabras acabadas en *-bilidad*. Ejemplos: *amabilidad, habilidad, posibilidad*. Excepciones: *movilidad, civilidad* y sus compuestos.

En nuestro sistema esta regla tiene la forma:

```

rb4(X, Y):- ((sub_atom(X,_,_,0,'vilidad')-> H = bilidad);
             (sub_atom(X,_,_,0,'vildades')-> H = bilidades)
            ) -> (atom_length(H,N),
                  sub_atom(X,0,_,N,Pre),
                  atom_concat(Pre,H,Y));
              Y=X.

```

Cuando el corrector aplique *rb4* a la palabra, detectará si ésta termina en *-vilidad/-viedades* y la corregirá. En caso contrario, la palabra no será modificada y el sistema pasará a aplicar la siguiente regla asociada a la *b*.

Prolog evalúa los predicados en el orden que hayan sido definidos. Debido a esto, las reglas fueron ordenadas de forma tal que primero se usen aquellas que son puramente fonológicas y luego las morfológicas y semánticas o pragmáticas. Esta decisión de diseño fue tomada para retrasar la intervención del usuario tanto como fuese posible.

5.4. Fronteras del Sistema y Herramientas de Apoyo

Existen puntos en la implementación que requerirían cierta funcionalidad, provista por módulos externos cuya integración excede el alcance de este trabajo. No obstante, mencionaremos algunos de éstos para demarcar las fronteras que permiten se establezca separación de intereses y modularidad.

Como entrada, nuestro sistema espera una lista de palabras en *minúscula* debiéndose eliminar cualquier otro tipo de elemento (por ejemplo, signos de puntuación) del texto original. Para esto, sería necesario un módulo que se encargue de pre-procesar el texto y, luego, ante la lista devuelta por el corrector, reconstruirlo.

Por otra parte, algunas reglas se basan en el análisis de ciertas sílabas de la palabra. Por ejemplo:

2.1.1. f (parte). (Se escriben con *b*) Las palabras que empiezan por las sílabas *bu-*, *bur-* y *bus-*. Ejemplos: *bula*, *burla*, *buscar*. Excepción: *vudú* y sus derivados, además de otras voces caídas en desuso.

Para estos casos, un sistema completo contaría con un separador en sílabas que, ante un pedido del corrector, entregue la palabra silabada para que este último pueda analizarla.

De forma similar, conjugadores de verbos y lematizadores servirían de apoyo al corrector. Existen reglas que requieren la conjugación de verbos y, claramente, esta funcionalidad cae fuera de las fronteras del corrector. Sobre todo, teniendo en cuenta que existen herramientas y técnicas que efectúan estas tareas en forma independiente, como se dijo al momento de presentar los conceptos lingüísticos. Por ejemplo, para la regla:

2.1.1. c. (Se escriben con *b*) Los verbos *deber*, *beber*, *caber*, *saber* y *haber*.

es de esperar que en una implementación total se cuente con un lematizador que responda a las necesidades del corrector. En este caso, dada una conjugación de alguno de estos verbos, el lematizador retornaría la forma en infinitivo, permitiendo que el corrector aplique la regla sin inconvenientes. Lo mismo ocurre con palabras que son derivadas de aquellas que aparecen en el enunciado de la regla.

Es idéntico el caso para familias de palabras que derivan o se forman por composición de otras. Un lematizador utilizará conocimiento morfológico para, dada la palabra, determinar qué forma canónica (lema) le da origen. Esta funcionalidad es necesaria en varias reglas y sobre todo en lo que respecta a la

definición de excepciones. Por ejemplo, la siguiente regla establece que ‘ébano’ y sus derivados son excepciones:

2.1.2. b. (Se escriben con *v*) Las palabras que empiezan por *eva-*, *eve-*, *evi-* y *evo-*. Ejemplos: *evasión*, *eventual*, *evitar*, *evolución*. Excepciones: *ébano* y sus derivados, *ebionita*, *ebonita* y *eborario*.

6. Ejemplos

Presentamos aquí dos ejemplos que ilustran exactamente los puntos débiles del enfoque tradicional por diccionarios y que son resueltos con éxito por nuestro prototipo.

Consideremos primero la palabra “verificabilidad”. Claramente, no es una palabra que sea utilizada en la mayor parte de los dominios, pero es fundamental en ciertas áreas de las ciencias de la computación y es muy probable que aparezca en un texto técnico. Sin embargo, hemos encontrado que los diccionarios de varias de las herramientas populares de procesamiento de textos no la contienen y, por ello, la tratan como un error. En la Figuras 3 y 4 podemos ver algunas de estas respuestas.

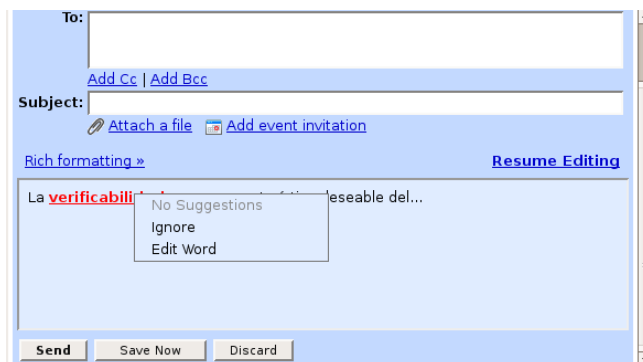


Figura 3. Falso error detectado por el corrector ortográfico de Gmail

La palabra sí está en el diccionario de la Real Academia Española, definida como

verificabilidad.

1. f. *Fil.* Cualidad de verificable.

Además, es claro que se obtiene de “verificable” por derivación morfológica.

Nuestro corrector aplica las reglas correspondientes a la *v* y determina que la palabra es ortográficamente correcta. Vemos esto en la sesión que muestra la Figura 5.

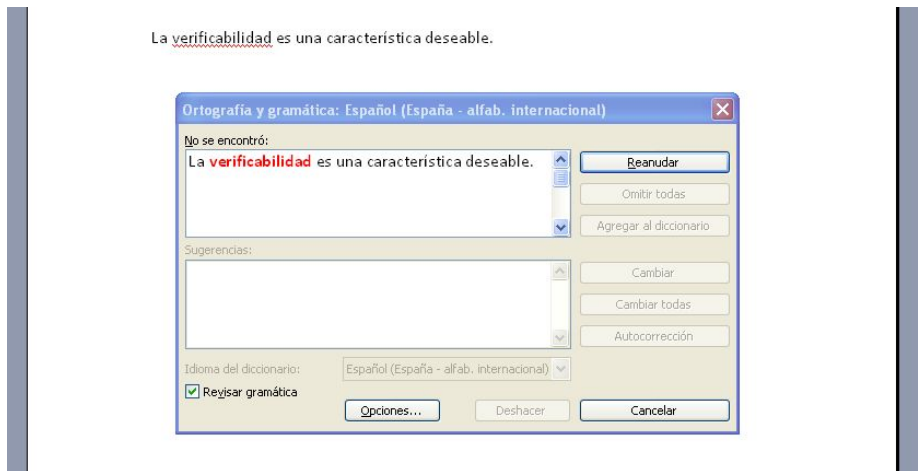


Figura 4. Falso error detectado por Microsoft Word 2003

Tres conjuntos de reglas son aplicados aquí, buscando alguna que detecte un error y lo corrija. En primer lugar, son aplicadas las reglas para *b*, sobre la primera *v* en “verificabilidad”, luego las de la *v* para la *b* y, por último, nuevamente las de la *v* para “deseable”. Ninguno de los grupos detecta errores y el resultado es igual al texto ingresado.

```

2 ?- corregir.
Ingrese una frase o una lista de palabras entre doble comilla " " y en letras minúsculas:
|: "la verificabilidad es una característica deseable".
Aplicar regla a "verificabilidad" : rb1(verificabilidad, verificabilidad)
...
Aplicar regla a "verificabilidad" : rb17(verificabilidad, verificabilidad)
Aplicar regla a "verificabilidad" : rv1(verificabilidad, verificabilidad)
...
Aplicar regla a "verificabilidad" : rv10(verificabilidad, verificabilidad)
Aplicar regla a "deseable" : rv1(deseable, deseable)
...
Aplicar regla a "deseable" : rv10(deseable, deseable)

El resultado de la corrección es:
"la verificabilidad es una característica deseable"

Yes

```

Figura 5. Sesión del corrector con la palabra “verificabilidad”

Más aún, si hubiésemos escrito “verificavilidad”, al aplicar la regla sobre *-bilidad* (**rb4**), el sistema detecta y corrige el error, como se ve en la Figura 6.

Como segundo ejemplo, retomemos las palabras homófonas “vaya” y “baya”. En este caso, ambas estarán en los diccionarios utilizados por los correctores

```
3 ?- corregir.
Ingrese una frase o una lista de palabras entre doble comilla " " y en letras minúsculas:
|: "la verificabilidad es una característica deseable".
Aplicar regla a "verificabilidad" : rb1(verificabilidad, verificabilidad)
...
Aplicar regla a "verificabilidad" : rb4(verificabilidad, verificabilidad)
...
Aplicar regla a "verificabilidad" : rb17(verificabilidad, verificabilidad)

El resultado de la corrección es:
"la verificabilidad es una característica deseable"

Yes
```

Figura 6. Sesión del corrector detectando el error en “verificabilidad”

tradicionales y el error pasará desapercibido. De hecho, esto puede verse en las Figuras 7 y 8.

En cambio, la corrección ortográfica por reglas utilizará más información al momento de intentar aplicar las mismas. En este caso puntual, deberá determinarse si “baya” es un verbo o un sustantivo. Utilizando reglas sintácticas muy simples, un analizador semántico podrá determinar fácilmente que la palabra “baya” en la oración presentada es un verbo y que, por ello, la versión correcta, aplicando la regla que refiere a conjugaciones del verbo “ir”, se escribe con *v*.

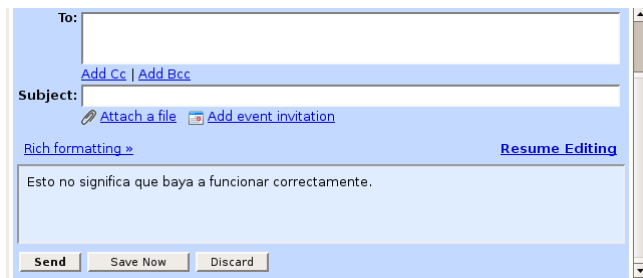


Figura 7. Error no detectado por el corrector ortográfico de Gmail



Figura 8. Error no detectado por Microsoft Word 2003

En efecto, nuestro corrector pedirá del usuario la información que necesita para determinar si se trata de una conjugación de este verbo, ante cuya respuesta corregirá el error de manera automática, tal como aparece en la Figura 9.

```
4 ?- corregir.
Ingrese una frase o una lista de palabras entre doble comilla " " y en letras minúsculas:
|: "esto no significa que baya a funcionar correctamente".
Aplicar regla a "baya" : rv1(baya, baya)
...
Aplicar regla a "baya" : rv8(baya, baya)

¿Ud. quiso escribir "vaya" en lugar de "baya"
donde "vaya" es presente subjuntivo del verbo "ir"?(s / n) : s.
Aplicar regla a "baya" : rv9(baya, vaya)
Aplicar regla a "vaya" : rv10(vaya, vaya)

El resultado de la corrección es:
"esto no significa que vaya a funcionar correctamente"

Yes
```

Figura 9. Sesión del corrector detectando el error en “baya”

7. Trabajos Futuros y Conclusiones

En esta instancia del trabajo, nos concentramos en el análisis de las reglas y de los conceptos lingüísticos utilizados para su definición, con el objetivo de obtener una representación en *Prolog* que permita algún grado de tratamiento automático. Por motivos de alcance, no analizamos en detalle ninguna de las herramientas que podrían liberar al usuario de proveer la información necesaria para la aplicación de las reglas (analizadores sintácticos, lematizadores, POS *taggers*, etc.). Tampoco realizamos pruebas estadísticas de la efectividad del método, comparado con las herramientas tradicionales. Estas actividades conforman un próximo paso fundamental en la constitución del método como una alternativa efectiva y viable. Para lo primero, debería seleccionarse un conjunto adecuado de las herramientas de soporte y analizar qué tan fácilmente cada una de ellas puede acoplarse al corrector. En cada caso, habrá que examinar qué tanta intervención del usuario continuaría siendo necesaria y cuáles serían las potenciales dificultades introducidas por el empleo de la herramienta (manejo de errores, respuestas falsas, etc.). Para las pruebas de efectividad, habría que elegir un *corpus* de acuerdo al conjunto de reglas implementado y diseñar *scripts* que, para la misma entrada, apliquen el método propuesto y algunos basados en diccionarios. Luego, se contarían las correcciones exitosas y erróneas en cada caso para llevar a cabo un análisis comparativo de los resultados. Una vez realizadas estas tareas, podría procederse a la construcción de un *plugin* que integre la funcionalidad a un editor existente.

En conclusión, presentamos en este trabajo una alternativa a las soluciones empleadas actualmente para la detección y corrección de errores ortográficos.

Acompañamos nuestra propuesta con un prototipo que, para un subconjunto de los posibles errores, muestra que nuestro punto es factible. También justificamos con base teórica la afirmación de que la corrección, tal como la proponemos, podría ser llevada a cabo de forma casi o totalmente automática, utilizando los avances más recientes en técnicas y herramientas de la lingüística computacional.

Sin embargo, este enfoque relativamente natural no ha sido abordado por las principales líneas de investigación y desarrollo de herramientas de procesamiento de texto. Es nuestro parecer que la razón para esto está en que el inglés, la lengua dominante tanto en ciencias como en prácticamente todo ámbito que exceda fronteras geográficas o culturales, no es pasible de recibir un tratamiento por reglas. La ortografía inglesa evolucionó de forma radicalmente distinta a la de muchas otras lenguas que sí poseen un conjunto pequeño de reglas que cubren casi la totalidad de los casos dudosos.

El punto anterior es mencionado en [8], notablemente uno de los pocos trabajos que abordan la problemática del tratamiento de lenguajes naturales que difieren lingüísticamente del inglés y en el cual se propone un entorno integrado para la composición de documentos en castellano utilizando fuertemente conceptos teóricos de la lingüística (aunque puntualmente en la corrección ortográfica sigue el enfoque tradicional por diccionarios). El trabajo data de la década de 1980 y no hemos encontrado publicaciones sobre la continuidad del mismo.

El tratamiento de lenguajes naturales desde la Inteligencia Artificial “dura”³ se transformó en un área poco visitada tras la burbuja de interés despertada por avances falaces tras la Segunda Guerra Mundial (por ejemplo, en traducción automática), que luego acabaron por dar pocos o nulos resultados concretos. Más tarde, con el crecimiento del poder computacional y la disminución de los costos, los métodos estadísticos y por fuerza bruta comenzaron a ser más utilizados (y a producir resultados concretos mucho más acertados), con lo cual la popularidad de los enfoques lingüísticos decreció casi hasta extinguirse.

Actualmente, la lingüística computacional ha avanzado tanto que ya se cuenta con herramientas que pueden resolver, al menos parcialmente, la mayor parte de los problemas auxiliares a la corrección de errores ortográficos, y a muchos otros de similar naturaleza. Esto debería despertar un nuevo interés en enfoques que intenten automatizar la resolución de problemas por las mismas vías que tomaría un humano para esa tarea. Finalmente, creemos que es la Inteligencia Artificial el lugar, técnica y filosóficamente adecuado, en el cual hacer confluir las distintas disciplinas que permitirían un ataque de estos problemas como el descrito en este trabajo.

Agradecimientos. A nuestra directora, Lic. Ana Casali, por su supervisión durante el desarrollo del trabajo, tanto en la cátedra que dirige como en las etapas de preparación de este artículo. También quisiéramos reconocer el esfuerzo y aporte de los revisores anónimos, quienes contribuyeron indudablemente a mejorar el resultado final.

³ *Strong AI*, entendida como aquella que pretende abordar los problemas mediante el entendimiento, de la misma forma en que lo haría un humano.

Referencias

1. Mitton, R.: Spelling checkers, spelling correctors and the misspellings of poor spellers. *Inf. Process. Manage.* **23**(5) (1987) 495–505
2. Mays, E., Damerau, F.J., Mercer, R.L.: Context based spelling correction. *Inf. Process. Manage.* **27**(5) (1991) 517–522
3. Jones, M., Martin, J.: Contextual spelling correction using latent semantic analysis. In: *Proceedings of the Fifth Conference on Applied Natural Language Processing, ANLP (1997)* 166–173
4. Zhao, Y., Truemper, K.: Effective spell checking by learning user behavior. *Applied Artificial Intelligence* **13** (1 December 1999) 725–742(18)
5. Hirst, G., Budanitsky, A.: Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering* **11**(1) (March 2005) 87–111
6. Kukich, K.: Techniques for automatically correcting words in text (abstract). In: *CSC '93: Proceedings of the 1993 ACM conference on Computer science, New York, NY, USA, ACM (1993)* 515
7. Mitton, R.: *English spelling and the computer.* Longman Group, Harlow, Essex (1996)
8. Casajuana, R., Rodriguez, C., Sopena, L., Villar, C.: Towards an integrated environment for spanish document verification and composition. In: *EACL. (1987)* 52–55
9. R.A.E.: *Ortografía de la Lengua Española.* Real Academia Española, Madrid (1999)
10. Pereira, F.C., Schieber, S.M.: *Prolog and Natural Language Analysis. Volume 10.* CSLI (1987)
11. Dahl, V.: Representing linguistic knowledge through logic programming. In: *ICLP/SLP. (1988)* 249–262
12. Blackburn, P., Bos, J.: *Representation and Inference for Natural Language. A First Course in Computational Semantics.* CSLI (2005)
13. Saussure, F.d.: *Cours de linguistique générale.* V.C. Bally and A. Sechehaye (eds.), Paris/Lausanne (1916) English translation: *Course in General Linguistics.* Peter Owen, London, 1960.
14. Hausser, R.: *Foundations of Computational Linguistics : Human-Computer Communication in Natural Language.* Springer-Verlag, Berlin (2001)
15. Magnasoft: *Silabeo.* <http://www.misosguar.com.ar/silabeo> (2005)
16. Grupo de Estructuras de Datos y Lingüística Computacional: *Flexionador y lematizador de palabras del español.* Universidad de Las Palmas de Gran Canaria, <http://www.gedlc.ulpgc.es/investigacion/scogeme02/lematiza.htm> (2006)
17. Porter, M.: *Spanish stemming algorithm.* Snowball. Tartarus.org, <http://snowball.tartarus.org/algorithms/spanish/stemmer.html> (2001)
18. Galicia-Haro, S., Gelbukh, A.F., Bolshakov, I.A.: Análisis sintáctico para el español basado en el formalismo de la teoría significado \leftrightarrow texto. *Procesamiento del Lenguaje Natural* (29) (2002) 81–88
19. Blackburn, P., Bos, J., Kohlhase, M., de Neville, H.: Inference and computational semantics. In Bunt, H., Thijsse, E., eds.: *Third International Workshop on Computational Semantics (IWCS-3).* (1999) 5–21
20. Bunt, H., Black, B.: The abc of computational pragmatics. In Bunt, H., Black, B., eds.: *Computational Pragmatics, Abduction, Belief and Context; Studies in Computational Pragmatics.* John Benjamins, Amsterdam (2000) 1–46
21. Jurafsky, D.: Pragmatics and computational linguistics. In Horn, L., Ward, G., eds.: *The Handbook of Pragmatics.* Blackwell, Oxford (2004) 578–604