EST 2012
XV CONCURSO DE TRABAJOS ESTUDIANTILES

Trabajos de Cátedra

Analizador Semántico:
una extensión para un corrector ortográfico en español

Audano Emmanuel Cravero Maximiliano
De Croce Mauro
{eaudano,macsee,maurodecroce}@gmail.com

Asignatura: Introducción a la Inteligencia Artificial
Docente: Dra. Ana Casali

Licenciatura en Ciencias de la Computación
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

Analizador Semántico:
una extensión para un corrector ortográfico en español

Resumen En la actualidad, la mayoría de los procesadores de texto incluyen alguna utilidad de corrección ortográfica entre sus herramientas. También pueden ser encontrados en navegadores, administradores de correo electrónico y prácticamente en cualquier aplicación que requiera el ingreso de texto. El enfoque, aparentemente universal de estas herramientas, se centra en el uso de un diccionario. Este método no hace uso de las propiedades lingüísticas (morfología, sintaxis, semántica) de la palabra ni del contexto en el que es utilizada. Esto impide la detección de ciertos errores, por ejemplo, cuando se utilizan palabras que existen en el diccionario, pero por estar mal escritas tienen un significado incorrecto, o cuando se usan neologismos que están bien escritos pero por no aparecer en el diccionario son considerados errores.
Este trabajo extiende el desarrollo de un prototipo de corrector ortográfico que utiliza reglas de la Real Academia Española, centrándose principalmente en automatizar aquellas reglas que necesitan de la información lingüística de las palabras para su aplicación.

1. Introducción

El uso masivo de medios electrónicos y de comunicación escrita lleva al uso cotidiano de correctores ortográficos, muchos de ellos basados en diccionarios, lo que ofrece una fácil implementación. No obstante, este tipo de enfoque trae algunas desventajas. Por ejemplo, la rápida evolución del lenguaje y la constante aparición de neologismos, hacen difícil mantener un diccionario que evolucione a la misma velocidad, perdiendo así la efectividad de este tipo de correctores.

Otra gran desventaja de los diccionarios es que no hacen uso alguno de las propiedades lingüísticas de la palabra y del contexto en el que ésta es utilizada. Esto impide la detección de ciertos errores y produce falsos avisos cuando una palabra es correcta pero por no encontrarse en el diccionario, es marcada como errónea. Esto también ocurre en caso de utilizar palabras que existen en el diccionario pero por estar mal escritas tienen un significado incorrecto, como por ejemplo referirse al verbo “vaya” como “baya”. Ambas son válidas, pero hacen referencia a diferentes cosas. Por lo tanto, debería considerarse el abordaje y desarrollo de asistentes que permitan automatizar dicha corrección aprovechando las propiedades lingüísticas de las palabras.

A diferencia del inglés, que es una lengua altamente irregular, el castellano resuelve su ortografía con un número muy reducido de reglas que cubren la gran mayoría de los casos relevantes. Por lo tanto, resulta natural atacar este problema desde el campo de la inteligencia artificial, planteando herramientas lingüísticas y computacionales que puedan aprovechar estos recursos. Por ejemplo, utilizando sistemas basados en reglas.

Tomando en cuenta esto úlitmo, el corrector iniciado por Plüss y Pomponio [1], implementa un sistema basado en las reglas ortográficas de la Real Academia Española (RAE) [7]. En dicho trabajo, fue introducida la arquitectura del sistema que sentaría las bases para las futuras actualizaciones, junto con algunas reglas a modo de ejemplo para demostrar el funcionamiento del mismo.

El presente trabajo se centra en el desarrollo de un módulo especialmente dedicado al tratamiento automático de las reglas de la RAE que requieren información semántica de las palabras para su aplicación. Por ejemplo, en el caso de una palabra mal escrita como "viología", si se considera el tratamiento con un corrector basado en diccionario, el éxito de la corrección dependerá de la existencia de la palabra en la base de datos del mismo. Sin embargo, si se utilizara un corrector basado en reglas, aplicando la regla de la RAE que establece que las palabras que contienen "bio" y hacen referencia a la vida o a los seres vivos deben escribirse con "b", la corrección se haría sin problemas ya que la regla se cumpliría. No obstante, el éxito de la corrección con este tipo de reglas depende fuertemente de poder identificar el significado de las palabras y es esto lo que se busca demostrar en este trabajo.

1.1. Trabajos anteriores

En [1] se presentó un enfoque basado en reglas, se propuso la arquitectura a utilizar (Fig. 1) y se implementaron algunas de esas reglas en lenguaje **Prolog**. La arquitectura propuesta consta de dos módulos principales, Corrector y Verificador. El primero responde los pedidos de este último y utiliza las reglas de la Base de Conocimiento para tratar cada palabra. Para aquellas reglas en las que necesite información lingüística compleja, se deben utilizar herramientas externas como un Silabeador, un Lematizador o un Analizador Semántico.

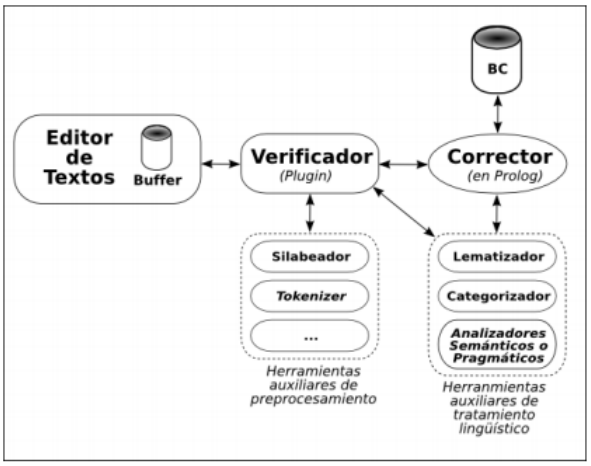

Figura 1. Arquitectura propuesta para el corrector tomada de [1]

El principio de funcionamiento está basado en asumir que las palabras están escritas con algún error ortográfico. La corrección tiene lugar cuando una de las reglas detecta que la subcadena que puede tratar se encuentra en alguna de las palabras introducidas por el usuario. Si el *matching* se produce, entonces la palabra se corrige según dicta la regla. Sin embargo, no todas las reglas son aplicables de manera automática. La mayoría de éstas palabras, por necesitar información lingüística compleja, como por ejemplo verbos en tiempos específicos, necesitan de la intervención del usuario.

En una segunda etapa se incorporó la posibilidad de analizar la morfología de las palabras mediante la utilización de un lematizador [2]. Con esto se consiguió identificar las raíces, los tiempos verbales, las personas y los infinitivos, permitiendo extender el corrector hacia un conjunto particular de reglas ortográficas que necesitaban del conocimiento morfológico para poder ser aplicadas de manera automática.

Posteriormente se incorporó un silabeador [3], como un módulo independiente del corrector. Esta nueva funcionalidad logró automatizar reglas de la RAE referidas a la fonología y morfología de las palabras.

Aún así, había reglas que todavía necesitaban de la intervención del usuario. Dichas reglas hacen uso de la información semántica de las palabras para poder ser aplicadas. Para mayor comodidad, a lo largo de este trabajo se las refiere como "reglas semánticas".

1.2. Objetivos

El trabajo original del corrector y las versiones que posteriormente le siguieron lograron demostrar que la herramienta posee un enorme potencial de uso aunque estuviese incompleta. Motivo por el cual resultó de interés extender la idea original con la intención de lograr una herramienta que pueda ser utilizada por el usuario corriente.

El presente trabajo parte de esta premisa aprovechando las características que se fueron añadiendo versión tras versión y tiene como objetivo principal acoplar un analizador semántico e integrarlo al resto del sistema, de manera de automatizar la aplicación de las reglas que hacen uso de la información semántica de las palabras.

Fueron considerados los siguientes objetivos:

- Estandarizar el modelo de reglas (situación que hasta el comienzo de este trabajo no se respetaba)
- Integrar este módulo adecuadamente para que pueda complementarse con los ya existentes, aprovechando las funcionalidades que ofrecen, logrando así una herramienta más potente.

El proyecto está orientado a lograr un corrector más robusto y abarcativo respetando las bases planteadas en sus orígenes.

2. Recursos lingüísticos y herramientas

A continuación se describe brevemente la teoría y las herramientas que fueron utilizadas como base de este trabajo.

2.1. Semántica

Es la ciencia que analiza los significados literales de las palabras. Se refiere a los aspectos del significado, sentido o interpretación del significado de un determinado elemento, símbolo, palabra, expresión o representación formal.

Computacionalmente, la semántica 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.

Generalmente, 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. De todos modos, en este trabajo, no es necesario un tratamiento total de la semántica. Bastará con poder determinar el signicado de ciertas palabras, una vez lematizadas sus formas, como es el caso de las reglas que se tratarán a continuación.

2.2. Reglas abordadas en el trabajo

Se trabajó con las siguientes reglas de la Real Academia Española:

- Se escriben con B:
  - Las palabras que contienen el elemento compositivo bio-, -bio (‘vida’).
    Ejemplos: biografía, biósfera, anaerobio, microbio.
  - Las palabras que empiezan por el elemento compositivo bi-, bis-, biz- (‘dos’ o ‘dos veces’).
    Ejemplos: bipolar, bisnieto, bizcocho.
  - Las palabras que empiezan por el elemento compositivo biblio- (‘libro’).
    Ejemplo: biblioteca.
- Se escriben con V:
  - Las que empiezan por el elemento compositivo vice-, viz- o vi- (‘en lugar de’).
    Ejemplos: vicealmirante, vizconde, virrey.
- Se escriben con G:
  - Las que empiezan por el elemento compositivo geo- (‘tierra’).
    Ejemplos: geógrafo, geometría, geodesia.

Estas reglas tratan palabras que refieren a conceptos como ‘vida’, ‘doble’ o ‘en lugar de’, los cuales se consideran como el o los significados de la palabra. Por lo tanto se precisaba de una cierta categorización de las mismas para que estas reglas puedan ser aplicadas. Naturalmente, era necesario la utilización de algún recurso lingüístico que posibilite la categorización semántica de base. La herramienta que se utilizó para lograrlo fue **WordNet** [4].

2.3. WordNet

WordNet es un sistema electrónico de referencia léxica, desarrollado en forma de base de datos en inglés. En la misma, sustantivos, verbos, adjetivos y adverbios se agrupan en conjunto de "sinónimos cognitivos", cada uno expresando un concepto distinto. Es un recurso lingüístico de uso libre [2], ideado para ser usado como complemento de aplicaciones de inteligencia artificial. La diferencia básica entre éste y otros proyectos de implementación de lexicones computacionales es que es el único proyecto a relativamente gran escala en el que se ha tenido como idea fundamental la organización del léxico en campos semánticos.

A diferencia de un diccionario tradicional, WordNet divide el lexicón en cinco categorías: nombres, verbos, adjetivos, adverbios y elementos funcionales.

Debido a la necesidad de cotejar las palabras con una gran base de datos para poder obtener una clasificación de la mismas, se optó por considerar la integración de WordNet con este trabajo para poder obtener rápidamente la categoría que engloba a una cierta palabra dada. Sin embargo, WordNet en su estado más puro resultaba inútil para este tabajo, ya que se necesitaba analizar texto en Español. Afortunadamente, el tamaño alcanzado por este proyecto ha logrado atravesar las fronteras idiomáticas y una gran cantidad de grupos de usuarios de diferentes lenguas han traducido la mayoría de las palabras a su idioma. La traducción al idioma Español se encuentra a cargo del grupo de investigación de procesamiento de lenguaje natural de la Universidad Politécnica de Catalunya [5].

3. Adaptación del prototipo

En el proyecto original se eligió utilizar a **Prolog**[3] como lenguaje de desarrollo, ya que las virtudes que la programación lógica presenta para definir sistemas basados en reglas, como así también su uso frecuente en el ámbito del procesamiento de lenguaje natural, lo hacen óptimo para esta tarea. Para continuar con el hilo del proyecto y facilitar la integración de todos los módulos con éste, se continuó con esta modalidad. Además, se dotó al corrector de una interfaz gráfica, permitiendo una interacción más amigable entre el usuario y el corrector.

Originalmente, cuando una palabra requería un poco más de análisis, de acuerdo con lo que se quería escribir, se consultaba al usuario, para de esta forma ayudar a la corrección de la palabra. Una vez ingresado el texto, el corrector iba analizando palabra por palabra y, dependiendo de las reglas que tuvieran que aplicarse, se requería que el usuario brindara la información semántica de la palabra para orientar la corrección ortográfica. Para esto, se generaban preguntas del tipo:

"Ud. hace referencia a biología, donde "BIO" es el elemento compositivo que indica VIDA?"

[2] Liberado bajo una licencia "BSD" (Berkeley Software Distribution).
[3] Se utilizó la versión 5.4.7 de SWI-Prolog. `http://http://www.swi-prolog.org/`

en cuyo caso el usuario debía responder “Si” o “No”. Este tipo de preguntas se producían, en este ejemplo en particular, porque la regla que trataba la palabra “biología” carecía de la información suficiente para llevar a cabo la corrección automática. Según dicta la RAE, las palabras que contienen el elemento compositivo “BIO” y hacen referencia a la vida, se escriben con “B”. Por lo tanto, al no disponer de esta información, se necesitaba consultar al usuario.

En contraposición, existen otras palabras que contienen “BIO” y se escriben con “B” aunque no hagan referencia a la vida o a los seres vivos. Un ejemplo de esto es la palabra “proverbio”, que si estuviera escrita como “provervio” se correspondería con la regla pero al no estar relacionada con la vida o los seres vivos no se aplicaría, con lo cual no habría corrección. Dado que este trabajo está centrado principalmente en implementar las reglas del manual de ortografía de la RAE que hacen referencia a la semántica de las palabras, este tipo de eventualidades corresponden que sean tratadas por otras reglas que no competen a este trabajo.

Para automatizar aún más el proceso, las consultas al usuario que se describieron más arriba fueron eliminadas. Esto se logró re-escribiendo las reglas que necesitaban conocer el significado de las palabras. Como primer paso, se asumió que las palabras a corregir tienen solamente errores ortográficos tratables por reglas que hacen uso de la información semántica. Teniendo en cuenta que las reglas corrigen de un error a la vez y son sensibles a la existencia de varios errores en la misma palabra, era necesario contar con un mecanismo que considere este tipo de casos. Para esto se decidió crear una jerarquía de reglas, aplicando primero todas aquellas que tratan lo sintáctico y lo morfológico [5]. Una vez hecho todo esto se aplican las reglas abordadas. Inclusive también se asumió que las palabras están acentuadas correctamente.

El corrector se compone de un predicado principal, llamado verificar, el cual es el encargado de ejecutar los distintos módulos desarrollados para llevar a cabo la corrección. Para lograr la jerarquía de reglas, se agregó el predicado `aplicar_semantico`. El mismo toma como argumento la lista de palabras que fueron modificadas previamente por el predicado `aplicar`, que es el encargado de aplicar todas las otras reglas implementadas que no requieren conocimiento lingüístico.

La nueva estructura del predicado `verficar` queda representado por el siguiente pseudocódigo:

```
verificar(Frase,Correccion) :-
            tratamiento(Frase,X),
            aplicar(X,Y),
            aplicar_semantico(Y,Correccion),
            write(Correccion).
```

De manera resumida, el predicado `verficar` toma la frase a corregir y devuelve el resultado de la corrección en la variable `Correccion`. El predicado

[5] Los diferentes tipos de reglas fueron analizados por Plüss y Pomponio en su trabajo [1].

`tratamiento` representa un conjunto de predicados encargados de convertir la entrada por teclado en un tipo de datos especial para el tratamiento de cadenas de caracteres en Prolog, junto con el análisis silábico necesario para aplicar algunas de las reglas que se ejecutan con el predicado `aplicar`. Una vez que `aplicar` ejecuta las reglas necesarias, todos los errores que no sean de tipo semántico quedan corregidos. El resultado de esta corrección parcial es almacenado en la variable `Y`, el cual es tomado por `aplicar_semantico` para ejecutar reglas etiquetadas como semánticas. Finalmente, el resultado se encuentra en la variable `Correccion`, que luego es mostrado por consola para mayor comodidad.

3.1. Estructura de las reglas semánticas

Las reglas semánticas se diferencian del resto ya que precisan información del significado de la palabra para poder ser aplicadas. Para obtener este tipo de información, lo más cercano es analizar la categoría a la que pertenece haciendo una consulta apropiada a la base de datos provista por WordNet.

En términos generales, la estructura de una regla semántica es la siguiente:

```
regla((X,XS)) :-
     detectar_patron_regla(X,patron)
     ->
     (
       reescribrir(X,P),              %P es la palabra reescrita
       insertar(['CATEGORIA'],XS,PS), %PS es la lista de categorías
       insertar_buffer(Buffer(P,PS))  %Buffer es el nombre del buffer
     ).
```

Cada vez que una regla semántica es invocada por `aplicar_semantico` para corregir una palabra `X`, en primer lugar se controla que el patrón de esa regla se cumple, es decir, si `X` contiene alguna de las subcadenas “BIO”, “GEO”, “BIBLIO”, “VICE”, “VIZ”, “VI”, “BIZ”, “BIS”, “BI”, pero mal escritas, (ya que no olvidemos que las reglas deben corregir justamente estos errores). Si esto ocurre, la palabra se reescribe correctamente de acuerdo con la regla, obteniéndose `P`. De lo contrario alguna otra regla es ejecutada.

A continuación se inserta la categoría `CATEGORIA` de esta regla en la lista `XS`, obteniendo una nueva lista `PS`. Por último, se inserta en `Buffer` la tupla formada por la palabra reescrita `P` y la lista de categorías `PS`. Una vez finalizado esto, se ejecuta el predicado `refiere_a(P,C)` por cada categoría `C` de la lista asociada a la palabra `P`. Este último predicado, mediante una serie de consultas a la base de datos de WordNet, verifica si una palabra `P` se encuentra bajo una categoría `C`. En caso de no cumplirse esta condición, se retorna la palabra original.

La elección de una lista para almacenar categorías fue una decisión de diseño tomada luego de considerar que, en contadas ocasiones, una palabra podía ser tratada por dos reglas semánticas al mismo tiempo. Esto generaría un árbol de prueba (Fig. 2), representado por `Buffer`, compuesto de todas las posibles

combinaciones de aplicación de las reglas que competen a la palabra. Se contempló también la posibilidad de no aplicar reglas, ya que podría ser necesario volver a un estado anterior producto de una mala corrección.

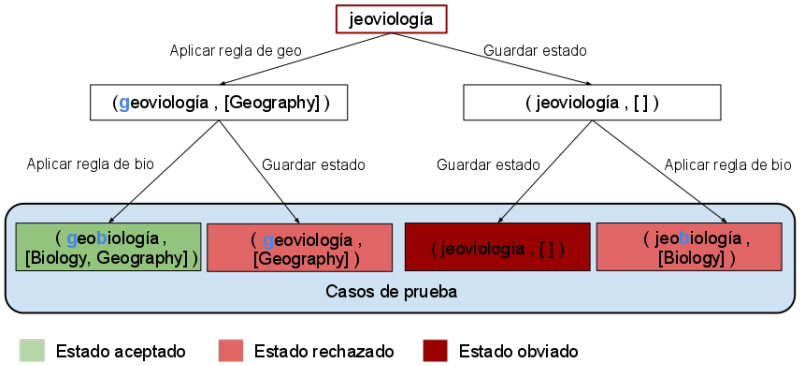

Figura 2. Árbol de prueba para la palabra "jeoviología"

En este árbol cada nodo tiene una tupla de la forma `(P,XS)`, donde `P` es la palabra corregida parcialmente y `XS` es la lista de las categorías que fueron insertadas por las reglas que trataron a la palabra hasta el momento. Cuando no es posible aplicar más reglas, se tienen los diferentes casos de pruebas. En cada uno de estos casos se debe comprobar que la palabra en la tupla se encuetra bajo todas las categorías de su lista asociada, aplicando el predicado `refiere_a`. Si esto ocurre en alguno de los casos de prueba, entonces se produce la corrección retornando la palabra que tuvo una comprobación exitosa. Caso contrario, se devuelve la palabra sin las modificaciones.

3.2. Uso del lematizador

Al usar una base de datos se impone una limitación sobre la información de la que se dispone. Así palabras como "microbio" y "microbios" por más que hagan referencia al mismo concepto, en la tabla solamente se encontrará una de las dos entradas. Para tratar con estos casos se decidió utilizar las ventajas que el lematizador [2] ofrece. De esta forma, al momento de buscar una palabra en WordNet, solamente se consulta su lema, y si éste se clasifica bajo la categoría correspondiente, se corrige la palabra original según la regla. De esta forma se logra mantener la base de datos compacta evitando redundancia y ofreciendo la misma funcionalidad.

4. La interfaz de usuario

Con el propósito de generar una herramienta independiente de Prolog y que disponga de una interfaz de usuario mucho más amigable, fue necesario crear una nueva aplicación desde otro lenguaje de programación que provea estas características y que permita utilizar todo lo hecho hasta el momento en el corrector. Se decició entonces utilizar Java en esta etapa ya que las ventajas que Prolog ofrecía para comunicarse con éste a través de su libreria especializada y la facilidad que brindaba para construir este tipo de interfaces lo hacían óptimo para esta tarea. Esta librería, llamada JPL [6], logra una comunicación bidireccional con Java. Esto significa que permite tanto embeber código Prolog, importar scripts y razonar desde un ambiente Java, como ejecutar código Java desde un entorno Prolog.

Gracias a esto, fue posible capturar la entrada de texto desde la nueva aplicación, realizar todo el razonamiento en Prolog, tomar la salida y mostrarla en la aplicación. De esta forma se mantiene intacta la escencia del trabajo original y se brinda una mejor presentación al usuario.

La parte grafica en sí, es decir, ventanas, botones, menús, etc, fue tomada de un template que ofrecía NetBeans, IDE que se utilizó en esta etapa, ya que era bastante sencillo y funcional.

En esta parte, la tarea principalmente se enfocó en crear los vínculos entre los lenguajes, a enviar el texto recibido desde la ventana del editor a Prolog para que lo procese, y a tomar las respuestas que éste generaba para poder mostrarlas en la misma ventana de manera adecuada. De esta manera, el uso de Prolog queda oculto para el usuario y el uso del corrector se vuelve más intuitivo.

Como todavía quedaban algunas reglas que continuaban precisando la asistencia del usuario, era necesario generar esas consultas en la nueva interfaz. La forma más elegante y fácil de lograr esto fue ejecutar código Java desde Prolog que creara una pequeña ventana con dos botones para permitir al usuario que elija la opción que considere más adecuada.

5. Conclusiones

La idea original de este proyecto presentó una alternativa a las soluciones empleadas en la actualidad para la detección y corrección de errores ortográficos. Este enfoque, aunque poco convencional, resulta más natural desde el punto de vista del tratamiento de los errores, ya que utiliza las reglas que rigen nuestra gramática tal y como la conocemos. Esto lo hace más poderoso, ya que cubre casos como neologismos y palabras homófonas que no son alcanzados por diccionarios convencionales. Las bases teóricas justifican la afirmación de que con esta perspectiva es posible la corrección casi automática de textos.

Con la intención de darle continuidad al proyecto, se proponen ideas para obtener mejoras en el prototipo, como el desarrollo de un *plugin* para algún editor de textos popular, como por ejemplo **LibreOffice**, que provea mayor portabilidad y facilidad de uso.

Otro desafío interesante en el campo del procesamiento de lenguaje natural, es tener la habilidad de reconocer el contexto. De esta forma se podrían automatizar aquellas reglas referidas a la pragmática, las cuales excedían el alcance de los trabajos realizados hasta el momento.

Por otra parte, pese a que WordNet es un recurso lingüístico ideado para ser usado como complemento de aplicaciones de inteligencia artificial, su versión en español resultó ser muy limitada para demostrar todo el potencial de nuestra herramienta. Motivo por el cual se tomó la decisión de agregar a WordNet algunas palabras y categorías que sirvieran como ejemplo para mostrar el total funcionamiento del corrector, respetando siempre su estructura original. Con esto, se llegó a la conclusión de que es necesario contar con recursos más flexibles, como por ejemplo Wikipedia [8], para obtener mejores resultados sin necesidad de producir alteraraciones.

Estamos convencidos que es desde la Inteligencia Artificial el lugar adecuado para abordar este tipo de problemáticas, ya que provee un tratamiento mucho más natural e intuitivo y de menor complejidad computacional.

Finalmente, creemos que este tipo de metodologías en la cual se consideran reglas sintácticas y semánticas del lenguaje natural que hasta el momento no eran tenidas en cuenta, darán lugar a una nueva generación de herramientas lingüísticas.

Agradecimientos. A nuestra directora, por su colaboración en este proyecto, y al Lic. Brian Plüss, por su dedicación y motivación brindadas durante todo el desarrollo del mismo. Cada una de estas personas contribuyó notablemente en este trabajo, sin ellos no hubieramos podido lograrlo.

Referencias

1. Plüss,B., Pomponio,L. - *Sistema para Corrección y Detección de Errores Ortográficos Basado en Conocimiento Lingüístico*, EST (2008)
2. Racca,P., Soriano,D., - *Sobre un Corrector*, Universidad Nacional de Rosario (2009)
3. Vanzetto, H. - *Parser silábico para la lengua española*, Universidad Nacional de Rosario (2010)
4. WordNet. - *A lexical database for English*, `http://wordnet.princeton.edu/`
5. NLP Research Group, Universidad Politécnica de Catalunya. - *Versión en Español de WordNet*, `http://nlp.lsi.upc.edu`
6. JPL - *A Java interface to Prolog*, `http://www.swi-prolog.org/packages/jpl/java_api/index.html`
7. Ortografía de la lengua española, Real Academia Española (2010), `http://www.rae.es/`
8. - Gabrilovich, E., Markovitch, S. - *Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis*, Proceedings of the 20th international joint conference on artificial intelligence (2007), Volumen 6.