Desarrollo de un Componente de Stemming para el Idioma Castellano

Leonardo Castiglioni[1] y Lucila Chiarvetto Peralta[1,2]

[1]Departamento de Ciencias e Ingeniería de la Computación y [2]Laboratorio de Investigación y Desarollo en Computación Científica,
Universidad Nacional del Sur
{leonardocastiglioni, lucila.cp}@gmail.com

Resumen. El lexema es el elemento que contiene la significación de una palabra, llamado raíz, base o stem. El objetivo del stemming es mejorar la performance de los sistemas de recuperación de información agrupando bajo un mismo stem todas las formas en que varía una palabra que comparten significado común. Es posible realizar el stemming mediante un algoritmo que use reglas gramaticales de derivación morfológica. En este trabajo se plantea extender el uso de un stemmer algorítmico mediante el empleo de una lista de excepciones (diccionario). Se implementó un componente de software en Java utilizando un desarrollo previo llamado Snowball. Para la evaluación se usó un corpus de páginas web en español al cual se le aplicó stemming. Se empleó luego el clasificador naïve bayesiano, mediante el cual se comparó el poder de categorización del mismo sobre el corpus procesado vs. no procesado. En los resultados obtenidos se observó una significativa reducción en el tamaño de los índices tras la aplicación de los algoritmos de stemming sin que esto implicara un detrimento de la capacidad de clasificación.

Palabras clave: stemming, español, Snowball, stem, lexema.

Introducción

Se define al stemming (en castellano, lematización) como el proceso de representar mediante un único término (stem o lexema) todas las posibilidades flexivas de una palabra. El lexema es el elemento que contiene la significación de la palabra, tradicionalmente llamado raíz o base. El lexema porta el significado léxico básico y es común a las demás palabras de su misma familia. La Real Academia Española (RAE, 2005) define a la flexión como la variación que experimentan las palabras a través de desinencias que expresan contenidos gramaticales. En RAE (2005) se define la desinencia como el segmento final que se añade a la raíz de una palabra. En los sustantivos, adjetivos y algunos pronombres se usa principalmente para indicar género y número; en los verbos su uso señala: persona, número, tiempo y modo.

La recuperación de información (RI), es la ciencia de la búsqueda de información en documentos, búsqueda de los mismos documentos, la búsqueda de metadatos que describan documentos, o también la búsqueda en bases de datos, ya sea a través de

Internet o intranets, de textos, imágenes, sonido o datos de otras características, de manera pertinente y relevante. Los motores de búsqueda tienen sus orígenes en los sistemas RI, los cuales preparan un índice de palabras claves para un dado corpus y responden a consultas por una palabra clave con una lista de documentos ordenada según relevancia. Los buscadores son algunas de las aplicaciones más populares de la recuperación de información. El objetivo del stemming es mejorar la performance de los sistemas de RI agrupando bajo un mismo stem todas las formas en que varía una palabra que comparten significado común. El stemming no es un concepto aplicable a todos los idiomas, por ejemplo el chino no admite la aplicación de stemming.

Es posible realizar el stemming mediante un algoritmo que use reglas gramaticales de derivación morfológica. Un ejemplo del stemming mediante utilización de reglas gramaticales es el algoritmo de Porter para el idioma inglés (Porter, 1980). Se puede encontrar también una implementación basada en el uso de diccionarios, asociando a cada forma su lexema representante. El enfoque basado en diccionario se considera superior, pero padece las desventajas de construir y mantener la estructura de datos. Se puede construir un stemmer híbrido en el que se combina un algoritmo con el uso de diccionario. En este trabajo se plantea extender el uso de un stemmer algorítmico mediante el empleo de una lista de excepciones (diccionario).

Definiciones Lingüísticas Preliminares

Las definiciones de esta sección fueron adaptadas principalmente de Garcia Negroni et al. (2004), en menor medida se consultó definiciones provenientes de RAE (2001, 2005).

Tilde

El acento prosódico, también llamado acento de intensidad, tónico o fonético, es la mayor fuerza espiratoria con que se pronuncia una sílaba dentro de una palabra. El acento ortográfico, también llamado tilde, es la representación gráfica del acento prosódico.

Sustantivos y Adjetivo

Desde el punto de vista semántico, el sustantivo es una clase de palabra que denota individuos o un conjunto de individuos o de entidades que poseen rasgos en común. Desde el punto de vista morfológico, el sustantivo es una clase de palabra variable o flexional, dotado de género y número. El sustantivo impone estas categorías que le son propias a las otras palabras que se agrupan torno a él, como el artículo y el adjetivo. Los adjetivos flexionan derivativamente como los sustantivos, por lo que están sujetos a las mismas reglas que guían la flexión sustantiva.

El género y el número son dos categorías morfosintácticas que caracterizan al sustantivo que se manifiestan por medio de desinencias que se agregan a la raíz. Mientras que la flexión de género y número completa la palabra mediante información relevante para la sintaxis, la derivación permite formar nuevas palabras. A diferencia de los sufijos flexivos que poseen un significado estrictamente gramatical, los afijos derivativos poseen significados léxicos muy variados. Se

denomina afijo al morfema que se antepone (prefijo) o pospone (sufijo) a la palabra primitiva o base. En algunas ocasiones se combinan unos sufijos con otros, formando, por ejemplo, diminutivos de aumentativos (sala: salón, saloncillo). En el **Apéndice A** y **Apéndice B** se encuentra el inventario de los sufijos para sustantivos y adjetivos, respectivamente, utilizados para elaborar el algoritmo.

Verbo

El verbo es una clase léxica o de contenido descriptivo inherente que presenta, junto al significado léxico, las categorías morfológicas de tiempo, modo, aspecto, número y persona. Estos significados gramaticales se manifiestan a través de marcas desinenciales que se combinan con el lexema.

Los verbos irregulares son aquellos que en su conjugación sufren alteraciones respecto de los modelos representados por cualquiera de las tres conjugaciones regulares (amar, temer y partir). Las irregularidades pueden aparecer en el lexema, en las desinencias o en ambas partes, y también en ocasiones, en la vocal temática. La vocal temática o incremento es la letra o letras que van entre el lexema y el sufijo. Esta letra o letras han sido añadidas por eufonía.

Cuando una irregularidad es compartida por un conjunto de verbos, éstos reciben el nombre de verbos de irregularidad común. Cuando la irregularidad se produce en la raíz, como así también en las desinencias (y no es compartida por ningún otro verbo), estamos en presencia de los llamados verbos de irregularidad propia. En el **Apéndice C** se encuentran listados los modelos de conjugación verbal. Se denominan verbos tíldicos o diátonos a aquellos que, apartándose en alguna de sus formas del paradigma regular, les correspondería tomar acento o diéresis en vocales distintas a las de tal paradigma regular.

Pronombres Enclíticos

El Diccionario Panhispánico de Dudas (RAE, 2005) define los pronombres personales como los que hacen referencia a alguna de las tres personas gramaticales. Pueden ser átonos: me, te, se, nos, os, lo(s), la(s), le(s); o tónicos: yo, tú, vos, él, ella(s), ello(s), usted(es), nosotros/as, vosotros/as, mí, ti, sí.

Los pronombres personales átonos, se pronuncian necesariamente ligados al verbo, con el que forman una unidad acentual. Estos pronombres, llamados "clíticos", aunque cuando anteceden al verbo se llaman "proclíticos"; cuando siguen al verbo se llaman "enclíticos". Por tratarse de formas átonas ligadas al verbo, los clíticos deben aparecer inmediatamente antepuestos o inmediatamente pospuestos a éste. Cuando van antepuestos (proclíticos), se escriben como palabras independientes. Cuando van pospuestos (enclíticos), se escriben necesariamente soldados al verbo: Dímelo.

Adverbio

Desde el punto de vista morfológico, el *adverbio* es una palabra invariable, por lo que no puede ser definida a partir de propiedades flexionales. El adverbio carece de género y número. Algunos adverbios derivados de adjetivo admiten los morfemas derivacionales *-ito* y *-simo*, característicos de la formación de diminutivos y superla-

tivos (por ejemplo: poco, poquito, poquísimo). Los adverbios terminados en *-mente* provienen de adjetivos calificativos constituyen una clase abierta y se forman a partir de la forma femenina o indiferente de la base adjetival. El idioma español no admite la aplicación de un adverbio en *-mente* a otro, como lo admite el inglés.

Otros Antecedentes para el Idioma Castellano

No existen en la bibliografía muchos antecedentes de herramienta similares para el idioma castellano. Figuerola et al. presentan un algoritmo de stemming como una máquina de estados finitos que intenta reconocer stems cuando los mismos tienen sufijos. Otro herramienta similar se encuentra en la web disponible en virtud del desarrollo hecho por parte del Grupo de Estructuras de Datos y Lingüística Computacional del Departamento de Informática y Sistemas de la Universidad de Las Palmas de Gran Canaria (*http://www.gedlc.ulpgc.es/investigacion/scogeme02/lematiza.htm)*.

Recursos

En la derivación del algoritmo de stemming se utilizaron diccionarios de palabras en español disponibles en el sitio web *http://es.openoffice.org/programa/diccionario.html*. Estos conjuntos de palabras correctas en su ortografía fueron exportados a una base de datos Microsoft SQL Express Edition© que fue la herramienta principal en la derivación de las reglas que conforman el algoritmo.

Para el desarrollo del componente se empleó Java J2SE (*http://java.sun*); Netbeans IDE 6.8, como entorno de desarrollo (*http://netbeans.org*); MySQL como soporte para la base de datos de excepciones (*http://www.mysql.com*); se empleó el driver MySQLConnector/J 5.1.10 para la conexión con MySQL (*http://www.mysql.com/productos/connector*) y el paquete HTMLParser como soporte de procesamiento de documentos HTML (http://htmlparser.sourceforge.net).

Para la evaluación se empleó conjunto de datos extraído del directorio DMOZ (*http://www.dmoz.org*); el clasificador WEKA versión 3.6 (*http://www.cs.waikato.ac.nz/ml/weka/*) y el paquete HTMLParser como soporte de procesamiento de documentos HTML. Las evaluaciones se corrieron en el sistema operativo GNU/Linux Mint 7 (Gloria) (*http://www.linuxmint.com*) y un equipo con procesador Intel Core2Duo E4500 @ 2.20 Ghz, memoria principal 3 GB DDR2 @ 667 Mhz.

Derivación del Algoritmo

El algoritmo de stemmming propuesto consta de 6 etapas de transformación. Cada etapa de transformación realiza una variación en la palabra hasta llegar finalmente al lexema. El desarrollo del algoritmo comenzó definiendo el flujo de control entre las etapas de transformación que un stemmer en castellano debía seguir, en la Fig. 1 se muestra este flujo.

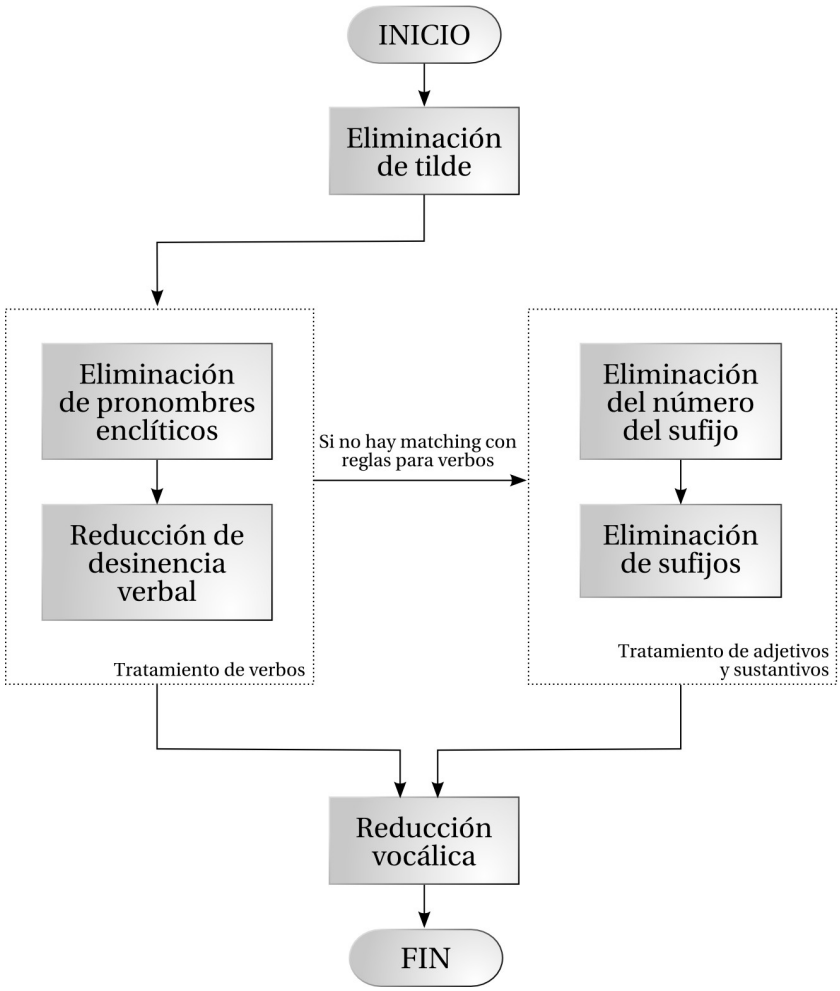

Fig. 1. Flujo de control del algoritmo de stemming

La primera de estas etapas es la eliminación de la tilde. Si bien la tilde en algunos casos puede implicar que cambie el significado de la palabra, en general se corresponden a casos que las palabras contienen el mismo lexema. Como segunda transformación se realiza la búsqueda de algún pronombre enclítico. A continuación se buscan desinencias verbales para remover. Si no se encontró ni un pronombre enclítico ni una desinencia verbal, es el caso de que la palabra es un sustantivo o adjetivo. Si se trata de un sustantivo o adjetivo, el algoritmo reescribe la palabra a su forma en singular de ser necesario (eliminación del número del sufijo, por ejemplo de *casas* a *casa*). Se continúa buscando y removiendo sufijos hasta no hallar ninguno más. Como última tarea, el algoritmo debe remover la vocal temática, esto es realizado en la etapa llamada "reducción vocálica".

Cada etapa de transformación está compuesta por una secuencia de reglas. Cada regla realizará una transformación en la palabra si las condiciones definidas en la regla se cumplen. Las condiciones de las reglas están definidas en base a regiones que se definen en la palabra.

El algoritmo de stemming utiliza las regiones: R1, R2 y RV. R1 es la región de la palabra luego de la primera consonante seguida de una vocal o es una región vacía al final de la palabra si no existe dicha vocal. R2 es la región dentro de R1 que comienza luego de la primera consonante seguida de una vocal o es una región vacía en caso contrario. La definición de RV resulta más compleja; si la segunda letra es una consonante, RV se define como la región luego de la próxima vocal. En el caso de que las dos primeras letras son vocales, RV es la región luego de la próxima consonante, en caso contrario RV es la región luego de la tercer letra. Pero RV es el fin de la palabra si ninguna de estas posiciones pueden ser encontradas.

En el **Apéndice D** se encuentra el algoritmo de stemming utilizado por el componente. El algoritmo de stemming no fue originalmente escrito en ningún lenguaje de programación en particular. La implementación del algoritmo comenzó con una investigación en la web en búsqueda de algún desarrollo previo, donde se encontró el proyecto Snowball (*www.snowball.tartarus.org*). En este sitio se presentan stemmers en diferentes lenguas, la mayoría de ellos escritos por el Dr. Martin Porter, autor del algoritmo de stemming en inglés (Porter, 1980).

El proyecto Snowball incluye, además, la definición de un pequeño lenguaje de manipulación de cadenas de caracteres de propósito específico en el cual están escritos los stemmers provistos en el sitio. Este lenguaje le da nombre al proyecto, Snowball, fue diseñado por el Dr. Martin Porter y es el lenguaje utilizado para implementar el algoritmo de stemming propuesto en el apéndice. Si bien Snowball tiene características muy minimales, posee funcionalidad muy afín al propósito de su creación. Por ejemplo, para el caso de la sufijación, se vuelve muy útil poder procesar una cadena de derecha a izquierda en lugar de izquierda a derecha (el recorrido por defecto). Actualmente Snowball no incluye un compilador propio, en su lugar, se provee un traductor de código Sowball para ANSI C y Java.

El enfoque del stemming aplicado en el proyecto Snowball es el de stemmers puramente algorítmicos. Este enfoque le confiere simpleza a la implementación, permitiendo además, aprender acerca de la estructura morfosintáctica de los componentes del idioma a tratar. Snowball, como lenguaje de programación, tiene características particulares basadas en el lenguaje SNOBOL (Farber et al., 1964; Griswold et al., 1968), con el cual comparte el concepto de emitir señales basándose en patrones de strings que alteran el flujo de control en el programa. La ejecución de un programa Snowball corresponde a la manipulación implícita de una única cadena de caracteres.

El flujo de control propuesto en la Fig. 1. no pudo ser reproducido en la implementación del algoritmo dada la complejidad de la codificación de las señales de control de flujo en Snowball. El algoritmo implementado sigue un flujo de control secuencial. Luego de que a la palabra se le realicen las transformaciones correspondientes al tratamiento de verbos se continúa con las transformaciones correspondientes al tratamiento de adjetivos y sustantivos.

El proyecto de Snowball provee un algoritmo de stemming para el idioma español. Sin embargo, luego de observar que no había un tratamiento lo suficientemente completo en líneas generales (conjugaciones de los verbos, tanto regulares como irregulares, desinencias verbales, flexión nominal y derivativa), se optó por escribir nuevamente el algoritmo conservando sólo algunas estructuras y rutinas menores. Esta redefinición del algoritmo implicó la derivación de las reglas de reescritura que componen las etapas de transformación.

Las observaciones anteriores hechas al algoritmo ofrecido por el proyecto de Snowball, nos llevaron naturalmente a la idea de extender el enfoque algorítmico mediante el concepto de *lista de excepciones*. La idea detrás de este modelo es la de tratar de capturar los aspectos más uniformes del idioma mediante el algoritmo, mientras que los casos excepcionales se tratan independientemente mediante un mecanismo de excepciones que asocia a una palabra con su lexema. Las palabras consideradas excepciones no son procesadas por el stemmer.

En este enfoque híbrido de implementación propuesto *la completitud del manejo de las irregularidades del idioma dependerá de la completitud del listado de excepciones del sistema.*

Desarrollo del Componente

De las dos traducciones posibles para el código Snowball, este trabajo emplea la versión a código Java. Luego de la traducción, se obtiene una jerarquía de clases que modelan un módulo de stemming.

Este módulo se toma como base para la construcción de un componente denominado *ExtendedStemmer*. El nombre surgió de considerar a la lista de excepciones como una extensión agregada a la capacidad de procesamiento del propio algoritmo de stemming.

El componente permite realizar las siguientes operaciones:

- Realizar stemming sobre una palabra
- Realizar stemming sobre un string de palabras
- Realizar stemming sobre un archivo HTML
- Realizar stemming sobre un archivo de texto plano.

Estos cuatro servicios pueden aplicarse tanto en forma algorítmica como en forma extendida, dando en total ocho servicios. Nos referimos a stemming en forma extendida al stemming realizado utilizando el soporte de base de datos para las excepciones en forma conjunta con el algoritmo.

El diseño

La Fig. 2 es un esquema simplificado del diseño y la comunicación entre las clases del componente ExtendedStemmer.

Es importante mencionar que cambiar el algoritmo de stemming por otro distinto no requiere de grandes esfuerzos. A tal fin, podría integrarse sin muchos cambios en el

código actual un patrón de diseño *Strategy* (Gamma et al., 1994), lo cual fácilmente permitiría probar el componente con distintas versiones de un algoritmo de stemming.

Para más información sobre el código, consultar la documentación Javadoc.

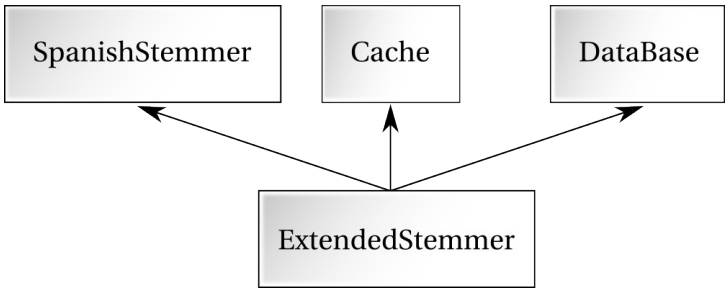

Fig. 2. Modelo simplificado de la comunicación de las clases del componente

Modalidad Algorítmica

La modalidad algorítmica es la más sencilla. Cualquiera de los cuatro servicios funcionan empleado únicamente el algoritmo de stemming generado por la traducción de Snowball. Este último recibe como entrada una cadena de caracteres, la procesa y devuelve su stem.

Modalidad Extendida y Uso de Caching

Para el manejo de excepciones se definió una base de datos sobre MySQL. La misma permite almacenar en una tabla las palabras que se constituyen en excepciones junto con el stem asociado a cada una.

Con el objetivo de reducir el impacto de los accesos a disco en la modalidad de stemming extendido, se introduce un buffer de cache debajo de los dos elementos del componente capaces de producir stems, la base de datos y el stemmer. De este modo, cada par <palabra, stem> proveniente tanto de la base de datos como del stemmer, es insertada en la cache de stems para un posible posterior uso.

El flujo de control para obtener un stem en la modalidad extendida es el siguiente:

1. Pedir la palabra en cache. Si se encuentra, ir a 5. Sino, continuar.
2. Pedir la palabra en la base de datos. Si se encuentra ir a 4. Sino, continuar.
3. Realizar el stemming utilizando el algoritmo.
4. Incorporar el par <palabra, stem> en cache.
5. Retornar el stem y terminar.

La cache está implementada como una tabla hash de tamaño fijo. Para más información, consultar la documentación Javadoc.

Plugins de Excepciones: Dominios de Aplicación

La base de datos definida también incluye una tabla que soporta el concepto de Plugins de excepciones. Mediante esta idea es posible categorizar aquellas excepciones que pertenecen a diferentes grupos o familias de palabras y poder tratarlas de forma independiente. Por ejemplo, podría agregarse el plugin "Nombres propios" que incluya nombres de personas femeninos y masculinos. Como lo nombres propios no deberían ser transformados por stemming, este plugin tiene la particularidad de que cada palabra y su stem son iguales.

En ciertas aplicaciones resulta de interés tratar ciertos términos de manera especial, aplicando transformaciones propias del dominio en cuestión. Un ejemplo clásico es en el área de minería de textos médicos o biológicos, donde suele ser necesario normalizar nombres de genes (Cohen, 2005). En tales casos, distintos nombres que son utilizados para referirse a un único gen suelen mapearse a un único término, pero la aplicación directa de algoritmos de stemming resulta inapropiada. En este sentido, el plugin de excepciones puede ajustarse para tratar al problema en cuestión de manera natural, definiendo reglas simples que involucren listas de nombres de genes y sus respectivas notaciones normalizadas.

Para proporcionar un uso y control más amigables sobre los plugins de excepciones, se escribió una pequeña aplicación llamada *Plugins Manager*. La aplicación está organizada mediante cuatro solapas, cada una proveyendo funcionalidad separada:

- **Carga de plugins**: para cargar un nuevo plugin, es necesario definir un archivo CSV que contenga exactamente dos columnas donde cada fila representa un par <palabra, stem>. Un nuevo plugin contiene un nombre, una descripción y, naturalmente, una lista de excepciones.
- **Eliminación de plugins**.
- **Visualización de plugins**: aquí podremos ver el nombre del plugin, su descripción, cantidad de palabras y un listado del contenido del mismo.
- **Agregado/eliminación de excepciones de usuario**: durante una fase de testeo y evaluación del stemmer es posible que surjan palabras que no fueron previamente consideradas como excepciones en los plugins. Estas palabras pueden ser agregadas en esta sección del programa. Las mismas pasarán a formar parte de un plugin predefinido llamado "user".

Línea de Comandos

El componente admite la posibilidad de ser empleado mediante línea de comandos. Admite los ocho servicios mencionados anteriormente con el agregado de que la salida puede ser mostrada en pantalla o redirigida a un archivo. La sintaxis es muy sencilla y, de ser necesario, se provee también ayuda en línea para su uso.

Aplicación: StemmerApp

Se provee además, junto al componente y la aplicación Plugins Manager, una segunda aplicación denominada *StemmerApp*. Esta aplicación permite cargar un corpus desde el disco, lo analiza, lo lista y le aplica stemming. Luego permite al usuario clickear

sobre uno de esos documentos para visualizar el conjunto de stems asociado al mismo. También es posible mostrar una visualización rudimentaria de un documento HTML haciendo doble click sobre él.

StemmerApp no es sólo un ejemplo de aplicación cliente del componente de stemming, sino que además permite realizar una auditoría sobre el funcionamiento general del componente.

Evaluación

Para la evaluación del desarrollo de este trabajo fue necesario utilizar un corpus de páginas web en español. Este corpus fue obtenido a través de la fuente de datos DMOZ. Se consideraron las siguientes categorías:

- URBANISMO (156)
- CICLISMO (92)
- FENÓMENOS PARANORMALES (156)
- MATERIALES Y SUMINISTROS (146)
- GASTRONOMÍA (76)

El número de documentos web descargados para cada categoría se muestra entre paréntesis. El pre-procesamiento aplicado al corpus, consistió en generar un archivo de texto plano por cada documento web. Este archivo estaba formado por las palabras de la página, para ello se utilizó la clase StringExtractor del paquete HTMLParser.

La descarga de estos documentos no pudo ser realizada de forma automática haciendo uso de un crawler. Las páginas debieron ser revisadas y luego descargadas manualmente ya que muchas de ellas estaban programadas con Adobe Flash. Estas últimas debieron ser descartadas ya que el texto se haya embebido en la aplicación Flash y es inaccesible para el parser HTML que utiliza el componente.

Debido a que el objetivo del stemming es mejorar la performance de los sistemas de recuperación de información, se optó por evaluar el desempeño de un clasificador comparando los resultados obtenido al tomar como entrada el corpus tras la aplicación del algoritmo de stemming puro y el algoritmo de stemming extendido. Para trazar una línea base de evaluación también se analizó el uso del corpus sin la aplicación de ningún algoritmo en particular.

Resultados y Discusiones

Weka es una colección de algoritmos de aprendizaje automatizado para tareas de minería de datos. Este framework contiene herramientas para el pre-procesamiento de datos, clasificación, regresión, clustering, reglas de asociación y visualización.

Se utilizó para la evaluación el clasificador *naïve bayesiano* incluido dentro de Weka. Sobre el corpus pre-procesado descripto anteriormente se aplicó stemming; se obtuvieron dos conjuntos distintos. Posteriormente, cada conjunto de datos fue transformado (mediante un filtro incluido en *Weka)* a la representación TF/IDF (Term

Frequency/Inverse Document Frequency), sobre esta representación se realizó la clasificación. La comparación de términos no consideró mayúsculas (case unsensitive) y se eliminaron números y símbolos.

Se midió de forma empírica el tiempo de ejecución consumido por el stemming del corpus mediante las versiones algorítmica y extendida. En el caso del stemming extendido, también se efectuaron las mismas corridas empleando dos tamaños diferentes de caché para observar sus efectos en el tiempo final de las ejecuciones. La base de datos sólo contenía las formas irregulares de aquellos verbos de los paradigmas fundamentales de irregularidad.

	Stemming	Stemming extendido	
	algoritímico	Caché de 1583 términos	Caché de 10007 términos
Tiempo (segundos)	4	95	84

Tab. 1. Tiempo de ejecución consumido en el stemming

Se registró el tiempo consumido por las distintas versiones en el stemming de los conjuntos, los valores se encuentran expresados en segundos. En la Tab. 1 se muestran los tiempos de ejecución obtenidos. Los resultados muestran lo excesivo del costo del modo extendido en comparación con el modo puramente algorítmico. Definimos como localidad léxica a la cardinalidad del conjunto de palabras de los documentos que se están inspeccionando. Si el tamaño de la caché no puede contener la localidad léxica se observará un menor desempeño del módulo de stemming.

En la Tab. 2 se muestran los tamaños de los índices producidos. Estas medidas corresponden a la cantidad de términos que producen los distintos enfoques en la representación TF/IDF. En todos los casos en los que se utilizó stemming, se observó una disminución del 15% en promedio del tamaño del índice respecto al texto plano. La versión extendida muestra una disminución despreciable del tamaño de los índices en comparación con la versión algorítmica. Esta diferencia en el tamaño de los índices, es atribuible al contenido del módulo de excepciones, dicho esto en términos lingüísticos, se debe al impacto de los verbos irregulares en el corpus.

El uso de bases de datos para manejo de excepciones es prohibitivamente costoso en tiempo y recursos para corpus grandes. No se observan ventajas significativas en la inclusión del manejo de la irregularidad verbal en el sistema de excepciones mediante base de datos.

En la Tab. 3 se muestra el desempeño obtenido en la clasificación de los documentos del corpus. En los casos en los que se utilizó stemming se observó un aumento de la cantidad de instancias clasificadas correctamente, 2.87% en promedio. No se observa diferencia en el desempeño de la versión algorítmica respecto de la versión extendida.

El uso de stemming **no muestra un detrimento de la capacidad de clasificación**, por el contrario, presenta una mejoría del 3.4%. Se debe considerar el hecho de que los índices a los cuales se aplicó stemming tienen un tamaño 15% menor respecto del producido para el texto plano, aun así, el clasificador no perdió poder de categorización.

	Texto plano	Stemming	
	(sin stemming)	Algorítmico	Extendido
Tamaño del índice	3587	3062	3033
Porcentaje de reducción (respecto al texto plano)		14,63%	15,44%

Tab. 2. Tamaño de los índices producidos

	Texto plano	Stemming	
	(sin stemming)	Algorítmico	Extendido
Instancias clasificadas correctamente	83,23%	86,10%	86,10%
Documentos (unidades)	521	539	539

Tab. 3. Desempeño en un clasificador naïve bayesiano

Conclusiones

Debe cuestionarse el uso de Snowball como lenguaje de implementación del algoritmo de stemming. Se trata de un lenguaje sin compilador ni debugger propio, motivo por lo cual resulta muy difícil asegurar el correcto comportamiento del programa luego de su traducción. El lenguaje tampoco permite analizar los tiempos de ejecución tomados por el algoritmo. Debe considerarse la opción de escribir un algoritmo de forma directa en un lenguaje con compilador o bien definir un nuevo lenguaje acorde a necesidades especificas.

El castellano es un idioma muy complejo, con un vocabulario amplio y una gran cantidad de mecanismos de formación de palabras y numerosos modelos de conjugaciones verbales. Estos hechos hacen que un buen stemmer requiera de un análisis profundo y minucioso del lenguaje.

Dentro de los aspectos independientes de la implementación podemos mencionar el buen uso de la ortografía. No se asegura un stemming correcto sobre palabras mal escritas o ajenas al idioma, ya que el módulo funciona a partir de un algoritmo con reglas basadas en formación de palabras ortográficamente correctas y de una base de datos con palabras válidas del castellano. Se asume al módulo de stemming como una

herramienta capaz de realizar procesamiento sobre palabras ortográficamente válidas del castellano.

Agradecimientos

Les agradecemos su ayuda en nuestra tesis a Eugenia Castiglioni, Lidia Gambón de Catalini y Celia Peralta. Queremos darle un agradecimiento especial a nuestra directora de tesis, Dra. Ana Maguitman, quien nos brindó su apoyo y la posibilidad de desarrollar este trabajo con suma libertad.

Bibliografía

Farber, D.J., Griswold, R.E, Polonsky I.P.: SNOBOL, a string manipulation language. Journal of the Association for Computing Machinery. 11, 21-30 (1964).

Griswold, R.E., Poage J.F., Polonsky I.P.: The SNOBOL4 programming language. Prentice-Hall, New Jersey (1968).

del Río, N.: Apunte de catédra de gramática española. Universidad Nacional del Sur (1977).

Porter, M.F.: An algorithm for suffix stripping. Program 14 no. 3, 130-137 (1980).

Riso de Sperber, E., Zafaroni, L.: Cuaderno de la lengua. Editorial Estrada (1981).

Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison Weslex, primera edición (1994).

Real Academia Española: Diccionario de la Lengua Española. Versión disponible en el sitio web *http://www.rae.es*, vigesima segunda edición (2001).

Figuerola, C. G., Gómez, R., Zazo Rodríguez, A. F., y

Berroca, J. L. A.: Stemming in spanish: A first approach

to its impact on information retrieval. Universidad de Salamanca,

España (2001).

Garcia Negroni M.M.,Pérgola L., Stern M.: El arte de escribir bien en español. Colección Instrumentos. Santiago Arcos (2004).

Cohen, A.: Unsupervised gene/protein entity normalization using automatically extracted dictionaries. In Proc: BioLINK2005 Workshop Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics, pp. 17-24. Association for Computational Linguistics, Detroit (2005).

Real Academia Española: Diccionario panhispánico de dudas. Versión disponible en el sitio web *http://www.rae.es*, primera edición (2005).

Cano, F.: Lengua y Literatura 2. Tinta Fresca (2006).

Patto, H.: Apunte de catédra de gramática española. Universidad Nacional del Sur (2006).

Apéndices

La recopilación de sufijos contenidos en los Apéndice A y Apéndice B fueron obtenidos de las siguientes fuentes: del Rio (1977), Riso de Sperber y Zafaroni (1981), Garcia Negroni et al. (2004), Cano (2006), Patto (2006).

La lista de modelos de conjugación verbal fue obtenida del Instituto De Verbologia Hispánica en su página web titulada "Árbol de formas y conjugaciones" (*http://www.verbolog.com/arbol.htm*).

A. Lista de sufijos para sustantivos

- Situación, cargo, jurisdicción, dignidad: *-ado -ato -atura -azgo -ción -ía -erio -cia*
- Abstracto: -anza -dad -edad -edumbre -ez -eza - ía -ia -icia -idad -bilidad-ncia -or -tud -ud -ura -ancia -encia
- Acción o efecto : -ado/a -aje/o -azgo -ción -ión -eo - ía -ido -miento -ón -torio/a -ura
- Actividad, doctrina, actitud: *-ismo -ante -ente -iente -ero*/*a -ista -dor*/*a -ario*/*a*
- Adicto a, partidario: *-ista*
- Animado, persona, con matices de agente, profesión, empleo, que recibe acción, de positario: *-ado*/*a -al - án*/*a -ante -anta -ario*/*a -tario*/*a -dor*/*a -ero -era -esa - ín*/*a -isa -isto*/*a - ón -or -tor*/*a -triz*
- Aumentativo: -ón -ona -ote -ota -ona -aza -azo -uda -udo -al
- Diminutivo: -ete -ezno/a -ezuelo/a -ico/a -cico/a -ecico -cecico -illo/a -cillo/a -ecillo/a -cecillo -ito/a -cito/a -ecito/a -cecito - ín/a -ote -uco/a -uelo/a -elo/a
- Despectivo: -achón/a -ejo/a -ajo -aja -stro/a -ucha/o -uja -uza -uca/o -orrio
- Cantidad contenida en: *-ada*
- Golpe: *-ada -azo*
- Grupo, conjunto, abundancia, colectivo: -ada -aje -al -ar -areda -ario -eda -edal -ela -ena -encia -erío -ería - ío -menta -aje

B. Lista de sufijos para adjetivos

- Aproximación, semejanza: *-áceo*/*a -ecino*/*a -izo*/*a -usco*/*a -uzco*/*a*
- Aspecto, cualidad, semejanza, estado: -aco/a -ado/a -eo/a -esco/a -estre -ico/a -ido/a -iego/a -il -ino/a - ío -oso/a -tico/a
- Aumentativo y despectivo: *-ote*/*a*
- Diminutivo: -illo/a -cillo/a -in/a -ito/a
- Capaz de, utilizado para: *-icio*/*a -ticio*/*a -til -torio*/*a*
- De, propio de, relacionado con: -aico/a -al - áneo/a -ano/a -ar -ense -icio/a -orio/a -uno
- Fácil o susceptible de, destinado a, apto para: *-dizo*/*a -ivo*/*a -tivo*/*a*
- Hecho con, de naturaleza de: *-eño*/*a -lento*/*a*
- Merecimiento, posibilidad: *-ble*
- Naturaleza, origen, procedencia (gentilicios): *-aco*/*a -aico*/*a - án*/*a -ano*/*a -eno*/*a -ense -eño*/*a -eo*/*a -ás*/*a -ico*/*a -ino*/*a -io -ita -ol -es*
- Partidario de: *-ano*/*a*

- Presencia, con, abundante en: *-ado*/*a -iente*/*a -iento -udo*/*a*
- Relación, pertenencia, tendencia o matiz: *-az -bundo*/*a*

C. Lista de modelos de conjugación verbal

C.1 Modelos de primera conjugación (39 arales, terminados en -ar)

Abuhar - Acertar - Actuar - Adecuar - Ahijar - Ahumar - Aislar - Amar - Andar - Aunar - Adeudar - Autoinflar - Averiguar - Bailar - Cambiar - Causar - Cazar - Colgar - Comenzar - Contar - Dar - Degollar - Desahuciar - Descafeinar - Desosar - Engoitar - Enviar - Errar - Estar - Forzar - Jugar - Lloudar - Pagar - Peinar - Regar - Rehilar - Rehusar - Sacar - Volcar

C.2 Modelos de segunda conjugación (30 erales, terminados en -er)

Agradecer – Caber – Caer - Cocer - Deber - Entender - Entrever - Fosforescer (fosforecer) - Haber Hacer - Leer - Mover - Oler - Placer - Poder - Poner - Proteger - Querer - Raer - Roer - Saber - Satisfacer - Ser - Tañer - Tener - Traer - Valer - Vencer - Ver - Yacer

C.3 Tercera conjugación (32 modelos; de los cuales 29 son irales, terminados en -ir, y 3 son íricos, terminados en -ír)

Argüir - Asir - Bendecir - Conducir - Construir - Contradecir - Counir - Decir - Delinquir - Dirigir - Discernir - Distinguir - Dormir - Elegir - Embaír (írico) - Erguir - Esparcir - Ir - Lucir - Mullir - Oír (írico) - Pedir - Prohibir - Reñir - Reunir - Salir - Seguir - Sentir - Sonreír (írico) - Venir – Vivir

D. Algoritmo de Stemming

Paso 1. Eliminar acentos ortográficos.

Paso 2, reglas de transformación para verbos.
Si mientras se procesan las reglas que corresponden a esta etapa no se produce ningún match con alguna de ellas, continuar con el Paso 3; en caso contrario, continuar con el Paso 4.

Paso 2.1, eliminación del primer pronombre enclítico.
Buscar al más largo de los siguientes sufijos:

la lo los las le les

y borrarlo en caso de que alguno de los siguientes esté contenido en RV:

nos me	segundo pronombre enclítico correspondiente a la primer persona
te os	segundo pronombre enclítico correspondiente a la segunda persona
se	segundo pronombre enclítico, forma reflexiva
ando endo	gerundio verbos regulares e irregulares
ar er ir	infinitivo

Reglas para el modo imperativo, persona: tu

ma	para los verbos modelos de flexión regular: amar
be	para los verbos modelos de flexión regular: deber
ve	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: vos

ma	para los verbos modelos de flexión regular: amar
be	para los verbos modelos de flexión regular: temer
is	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: vosotros

ad	para los verbos modelos de flexión regular: amar
ed	para los verbos modelos de flexión regular: temer
id	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: ustedes

en	para los verbos modelos de flexión regular: amar
an	para los verbos modelos de flexión regular: deber o partir

Paso 2.2, eliminación del segundo pronombre enclítico.
Buscar al más largo de los siguientes sufijos:

nos me

y borrarlo en caso de que alguno de los siguientes esté contenido en RV:

te os	segundo pronombre enclítico correspondiente a la segunda persona
se	segundo pronombre enclítico, forma reflexiva
ando endo	gerundio verbos regulares e irregulares
ar er ir	infinitivo

Reglas para el modo imperativo, persona: tu

ma	para los verbos modelos de flexión regular: amar
be	para los verbos modelos de flexión regular: deber
ve	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: vosotros

ad	para los verbos modelos de flexión regular: amar
ed	para los verbos modelos de flexión regular: temer
id	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: ustedes

en	para los verbos modelos de flexión regular: amar
an	para los verbos modelos de flexión regular: deber o partir

Paso 2.3, eliminación del tercer pronombre enclítico.
Buscar al más largo de los siguientes sufijos:

te os

y borrarlo en caso de que alguno de los siguientes esté contenido en RV:

se	segundo pronombre enclítico, forma reflexiva
ando endo	gerundio verbos regulares e irregulares
ar er ir	infinitivo

Reglas para el modo imperativo, persona: tu

ma	para los verbos modelos de flexión regular: amar
be	para los verbos modelos de flexión regular: deber
ve	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: vos

ma	para los verbos modelos de flexión regular: amar
be	para los verbos modelos de flexión regular: temer
is	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: vosotros

ad	para los verbos modelos de flexión regular: amar
ed	para los verbos modelos de flexión regular: temer
id	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: ustedes

en	para los verbos modelos de flexión regular: amar
an	para los verbos modelos de flexión regular: deber o partir

PASO 2.4, eliminación de la forma reflexiva.
Buscar el siguiente sufijo:

se

y borrarlo en caso de que alguno de los siguientes esté contenido en RV:

ando endo	gerundio verbos regulares e irregulares
ar er ir	infinitivo

Reglas para el modo imperativo, persona: tu

ma	para los verbos modelos de flexión regular: amar
be	para los verbos modelos de flexión regular: deber
ve	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: vos

ma	para los verbos modelos de flexión regular: amar
be	para los verbos modelos de flexión regular: temer
is	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: vosotros

ad	para los verbos modelos de flexión regular: amar
ed	para los verbos modelos de flexión regular: temer
id	para los verbos modelos de flexión regular: partir

Reglas para el modo imperativo, persona: ustedes

en	para los verbos modelos de flexión regular: amar
an	para los verbos modelos de flexión regular: deber o partir

PASO 2.5, eliminación de desinencia verbal.

Primera conjugación ar (amar), regular
Modo indicativo
mos ais eis an bas ba bamos bais ban ste steis ron re ras ra remos reis ran ria rias riamos riais rian
Modo subjuntivo
ais eis mos an ra se ras ses ramos semos rais seis ran sen re res remos reis ren
Modo imperativo
ad en aos

Segunda conjugación er (temer o deber), regular
Modo indicativo
mos eis iais en ste steis ron re ras ra remos reis ran ria rias riamos riais rian
Modo subjuntivo
mos ais eis en ra se ras ses ramos semos rais seis ran sen re res remos reis ren
Modo imperativo
ed en eos

Tercera conjugación ir (partir o vivir), regular
Modo indicativo
mos is iais eis eis en ste steis ron re ras ra remos reis ran rias ria riamos riais rian
Modo subjuntivo
mos ais an ra se ras ses ramos semos rais seis ran sen re res remos reis ren
Modo imperativo
id an as

PASO 3, reglas de transformación para sustantivos y adjetivos.

PASO 3.1, eliminación del número del sufijo.
Buscar al más largo de los siguientes sufijos y realizar el reemplazo indicado.

acas → aca	*aceas → acea*	*aceos → aceo*	*aces → az*	*achonas → achona*
acos → aco	*aicas → aica*	*aicos → aico*	*ajas → aja*	*achones → achon*
anas → ana	*aneas → anea*	*aneos → aneo*	*ajos → ajo*	*amenes → amen*
anos → ano	*antes → ante*	*antas → anta*	*anes → an*	*ancias → ancia*
ares → ar	*aturas → atura*	*anzas → anza*	*atos → ato*	*aredas → areda*
azas → aza	*azgos → azgo*	*azos → azo*	*bles → ble*	*bilidades → bilidad*
cicos → cico	*cicas → cica*	*cillas → cilla*	*cias → cia*	*bundas → bunda*
citas → cita	*ciones → cion*	*cillos → cillo*	*citas → cita*	*bundas → bunda*
dades →	*dizas → diza*	*dizos → dizo*	*das → da*	*bundos → bundo*

dad

dores → dor	*doras → dora*	*durias → duria*	*dos → do*	*edumbres → edumbre*
edas → eda	*ecicos → ecico*	*ecilla → ecilla*	*eas → ea*	*encias → encia*
edos → edo	*ecinas → ecina*	*ecillos → ecillo*	*enas → ena*	*edales → edal*
ejas → eja	*ecinos → ecino*	*ecitas → ecita*	*enos → eno*	*ezuelas → ezuela*
ejos → ejo	*ecitos → ecito*	*edades → edad*	*eños → eño*	*ezuelos → ezuelo*
eñas → eña	*entes → ente*	*enses → ense*	*eos → eo*	*idades → idad*
erias → eria	*escas → esca*	*esas → esa*	*eras → era*	*iegas → iega*
erios → erio	*escos → esco*	*ezas → eza*	*eros → ero*	*ientas → ienta*
etes → ete	*eznas → ezna*	*icias → icia*	*eses → es*	*ientes → iente*
icas → ica	*eznos → ezno*	*icios → icio*	*ias → ia*	*iegos → iego*
iles → il	*inas → ina*	*iones → ion*	*icos → ico*	*ientos → iento*
illas → illa	*ines → in*	*ios → io*	*idas → ida*	*ismos → ismo*
illos → illo	*inos → ino*	*isas → isa*	*idos → ido*	*istas → ista*
istos → isto	*itas → ita*	*itos → ito*	*ivas → iva*	*lentas → lenta*
izales → izal	*izas → iza*	*izos → izo*	*ivos → ivo*	*lentos → lento*
ncias → ncia	*orias → oria*	*onas → ona*	*oles → ol*	*mentas → menta*
orios → orio	*orrios → orrio*	*osas → osa*	*ones → on*	*mentos → mento*
osos → oso	*otas → ota*	*otes → ote*	*ores → or*	*mientos → miento*
rias → ria	*stras → stra*	*tecas → teca*	*ticos → tico*	*tarias → taria*
tivas → tiva	*stros → stro*	*ticas → tica*	*tiles → til*	*tarios → tario*
tivos → tivo	*toras → tora*	*tores → tor*	*ucas → uca*	*torias → toria*
tudes → tud	*uchas → ucha*	*trices → triz*	*ucos → uco*	*torios → torio*
udos → udo	*uchos → ucho*	*udes → ud*	*udas → uda*	*uelas → uela*
uelos → uelo	*uzcas → uzca*	*unos → uno*	*uzas → uza*	*uscas → usca*
ujas → uja	*uzcos → uzco*	*uscos → usco*		

adas → ada, si '*as*' cae en RV
ados → ado, si '*os*' cae en RV
ajes → aje, si '*ajes*' cae en RV
ales → al, si '*es*' cae en RV
arias → aria, si '*rias*' cae en RV
arios → ario, si '*rios*' cae en RV

Paso 3.2, eliminación de sufijo: repetir esta subetapa hasta que no se puedan remover más sufijos.

Eliminar los siguientes sufijos en caso de que caigan en R1 y RV:
uno achona achon erio eria ado ato atura azgo anza dad edad ez eza ia icia icio idad or tud ud eo ea ido ida miento ismo ante anta ista isto ario aria dor ero era esa triz tor tora osis esis asis itis areda eda io menta amen duria edo ina orio oria dora ecino

ecina usco usca uzco uzca aco aca esco esca estre iego iega aneo anea il dizo diza ivo iva eño eña lento lenta ble az bundo bunda stro stra uzo uza ete

Eliminar los siguientes sufijos en caso de que caigan en R2:
eno tica tico ena teca taria tario torio toria cion cia aje ion al an mente ente isa ar mento ria izo iza oso osa ino aceo acea ense tivo tiva ano ote es ol ana iente iento ienta ezno ezna

Eliminar el siguiente sufijo si cae en R2 o en RV:
bilidad

Eliminar los siguientes sufijos en caso de que caigan en R1:
til ncia ura ancia encia osis izal aico aica

Eliminar los siguientes sufijos, indepedientemente de las regiones:
edumbre cillo

Efectuar las siguientes eliminaciones en caso de que la porción indicada caiga en R2:
cete: '*ete*'
ezuelo: '*uelo*'
ezuela: '*uela*'

Efectuar los siguientes reemplazos en caso de que el sufijo caiga en R1 y RV:

ote → o	*ota → a*	*aza → a*	*azo → o*	*uda → a*	*udo → o*
acho → o	*acha → a*	*eja → a*	*ejo → o*	*ajo → o*	*aja → a*
ucha → a	*ucho → o*	*ujo → o*	*uja → a*	*uco → o*	*uca → a*
orro → o	*orrio → o*	*arro → a*	*ecillo → o*	*ecilla → a*	*illo → o*
illa → a	*uelo → o*	*uela → a*	*cica → a*	*cico → o*	*ecica → a*
ecico → o	*ito → o*	*ita → a*	*cito → o*	*cita → a*	*ecito → o*
ecita → a					

Efectuar los siguientes reemplazos en caso de que el sufijo caiga en R2:
ada → a *on → o* *ona → on*
ico → o *ica → a* *in → o* *ina → a*

Consideraciones especiales
edal: si '*al*' cae en R1 y en RV
entonces Borrarlo
sino si '*eda*' cae n R1 y en RV
entonces Borrar '*dal*'
cilla: si '*cilla*' cae en R2
entonces borrarlo
sino si ('*cilla*' cae en R1 y en RV) y no('*ecilla*' cae en R1 y en RV)
entonces borrarlo

PASO 4. Reducción vócalica. Buscar el más largo de los siguientes sufijos en RV y realizar la acción indicada.

as os a o i

Borrarlo.

e

Borrarlo.
Si está precedido por '*gu*' con la letra '*u*' dentro de RV, entonces borrar también la *u*.