Un lenguaje de apoyo para tareas de preprocesamiento de textos

Erika Hernández[1], David Pinto[2], Héctor Jiménez-Salazar[3], Jesús Lavalle[4]

Facultad de Ciencias de la Computación,
Benemérita Universidad Autónoma de Puebla,
14 sur y Av. San Claudio, Ciudad Universitaria, Edif. 135
Puebla, Pue., México, C.P. 72570
Tel. (+52-222)2295500 Ext. 7212, Fax (+52-222)2295672
[1]erikab@mail.cs.buap.mx, {[2]dpinto, [3]hjimenez, [4]jlavalle}@cs.buap.mx

Resumen. El tratamiento de textos en Internet para su uso posterior en Sistemas de Recuperación de Información(SRI), es de vital importancia. Es un hecho que los resultados de precisión y evocación en los SRI dependen en gran medida de la fase de preprocesamiento. En este proyecto se propone un lenguaje computacional que sirva como herramienta de apoyo en tareas de preprocesamiento de textos. Se ha definido una gramática e implementado el proyecto mediante los lenguajes Java y C (a través de flex y bison). Se han realizado experimentos sobre diversos *corpora:* TREC-5, EuroGOV y otras colecciones compiladas manualmente. Los resultados obtenidos muestran una aceleración y mejora en la etapa de preprocesamiento.

Palabras clave: Procesamiento del Lenguaje Natural, Preprocesamiento, *Corpora.*

1 Introducción

Con el fin de desarrollar sistemas robustos para el Procesamiento de Lenguaje Natural(PLN), es necesario emplear algoritmos que comúnmente requieren que los documentos fuente sean preprocesados antes de utilizarlos en alguna técnica de análisis de información. Adicionalmente, es posible verificar que los conjuntos de información existentes carecen de una estructura y simbología homogénea que permita crear un programa *ad-hoc* para esta tarea. El preprocesamiento universal de *corpora* es una tarea difícil y aun por resolver; de hecho, investigadores en el área de PLN requieren programar métodos de preprocesamiento diferentes, cada vez que necesitan aplicar alguna técnica de análisis de información a nuevos *corpora.*

De manera general, el preprocesamiento puede ser planteado de la manera siguiente: dado un texto D, denotamos con D' al que se obtiene por eliminar las palabras cerradas, símbolos y etiquetas [1].

En algunos trabajos, como el presentado en [2], se contemplan las siguientes operaciones de preproceso: separación de palabras utilizando el carácter de espacio, detección de nombres simples, detección de nombres formados con la inclusión "de/de la" y la eliminación de palabras cerradas. Por otro lado, en [3], se

Avances en la ciencia de la computación en México
Research on Computing Science 13, 2005, pp. 91-98

utiliza un vocabulario formado por las palabras clave que describen a todos los vocablos de un glosario, eliminando de éste, las palabras cerradas y aplicando el algoritmo de Porter adaptado al español para realizar truncamiento.

El tiempo empleado en programar métodos de preprocesamiento para *corpora* heterogéneos crece de manera sustancial, produciendo una inversión de tiempo considerable en la fase de preprocesamiento, el cual podría reducirse para incrementar el periodo de pruebas. Es por ello que el presente trabajo expone un lenguaje con la capacidad de preprocesar una gran variedad de *corpora*, con la finalidad de convertirlos en textos planos preprocesados y listos para tareas específicas de PLN. La gramática brinda una sintaxis casi en Lenguaje Natural(LN), lo que hace de éste un lenguaje de fácil aprendizaje.

El resto del artículo se organiza de la siguiente manera. En la sección 2 se describe el lenguaje desarrollado, exponiendo la gramática como parte fundamental en el diseño del lenguaje. En la sección 3 se presenta la implementación de la gramática. En la sección 4 se muestra un conjunto de pruebas aplicadas sobre diferentes *corpora*. Finalmente las conclusiones del trabajo son presentadas.

2 Diseño de la gramática

2.1 Descripción del lenguaje

En la herramienta diseñada, se han incluido tres archivos (*stopwords.txt, symbols.txt y labels.txt*), los cuales contienen elementos iniciales que, se espera, faciliten la labor de preprocesamiento de información. Estos contienen palabras cerradas[1], símbolos[2] y etiquetas de formato[3], los cuales cuentan con un tamaño inicial de 20KB, 9KB y 7KB, respectivamente. Se han diseñado diversos mecanismos para administrar dichos archivos. A continuación se puntualiza sobre cada uno de ellos, así como también se hace mención de los servicios soportados por el lenguaje.

Los servicios que el lenguaje proporciona se han clasificado de la siguiente manera: servicios a nivel de primitivas y servicios a nivel de preprocesamiento.

Servicios a nivel de primitivas: Fueron denominados así por el hecho de permitir únicamente el manejo de las operaciones básicas, efectuadas sobre las colecciones de palabras cerradas, símbolos y etiquetas de formato. A continuación se describen las operaciones brindadas por este servicio.

La herramienta proporciona tres archivos o colecciones[4], permitiendo a los usuarios agregar, eliminar y consultar elementos de estas colecciones.

[1] También conocidas como *stopwords*, son palabras que aparecen son frecuencia entre los documentos y no son buenas para la recuperación de información (artículos, preposiciones, conjunciones, etc).

[2] Se consideran signos de puntuación, caracteres ASCII y algunos otros símbolos localizados en los textos.

[3] Conjunto de marcas que forman parte de la estructura de algunos *corpora* por ejemplo: <title>,< \ title> <p>, < \ p>, <TEXT>, <\ TEXT>, etc.

[4] *stopwords.txt, symbols.txt* y *labels.txt*.

Servicios a nivel de preprocesamiento: Estos tipos de servicios permiten la preparación de *corpora*, mediante la invocación de los servicios primitivos cada vez que sea necesario.

Operaciones sobre corpora: El lenguaje permite la creación de un nuevo *corpus* como un contenedor de archivos. Acepta la eliminación de un *corpus* ya existente en caso que así se requiera. Proporciona un listado de archivos pertenecientes a un determinado *corpus*. Muestra el contenido de un archivo perteneciente a un *corpus*, así como las propiedades[5] del mismo. Permite la inserción de nuevos archivos a diferentes *corpus*.

Operaciones de preproceso: El lenguaje permite preprocesar un determinado archivo perteneciente a un *corpus*, bajo diferentes niveles de preprocesamiento tales como: eliminación de palabras cerradas, eliminación de símbolos, eliminacion de etiquetas de formato, así como cualquier combinación de las operaciones anteriores. Extrae sólo partes de un determinado archivo perteneciente a un *corpus*, mediante la inclusión de cadenas de inicio y fin. Permite la sustitución de una determinada etiqueta por otra, mediante la inclusión de cadenas de inicio y fin.

Para hacer uso de los servicios, a nivel primitivo y a nivel de preproceso, es necesario contar con una gramática que verifique las sentencias introducidas por el usuario para aceptarlas o en su defecto rechazarlas, por lo que en la siguiente sección se introduce la estructura de la misma.

2.2 Estructura de la gramática

El establecimiento de las reglas gramaticales que el lenguaje debe de cumplir es una de las situaciones más complejas para su desarrollo. La forma en que se describe la gramática en este trabajo está dada por la notación BNF (Backus-Naur-Form).

La definición de la gramática libre de contexto (GLC) está dada por:

$$G = (V_n, V_t, S, P)$$

donde
V_n = Un conjunto de símbolos no terminales,
V_t = Un conjunto de símbolos terminales,
S = Símbolo inicial de la gramática,
P = Un conjunto de producciones.

A continuación se presenta la GLC del lenguaje desarrollado.

S=<SENTENCIA>

$$V_n = \left\{ \begin{array}{l} \text{<SENTENCIA>, <CREATE>, <DROP>, <ADD>, <DELETE>} \\ \text{<LIST>, <INSERT>, <DISPLAY>, <PREPARE>, <REMOVE>} \\ \text{<REPLACEALL>, <EXTRACT>, <ID>, <USING> , <CICLOS>} \\ \text{<LETRA>, <DIGITO>, <NOMARCH>, <STRING>, <SIMBOLO>} \\ \text{<DICCIONARIOS>} \end{array} \right\}$$

[5] Tales como: ruta, tamaño del archivo y fecha de la última modificación.

$$V_t = \left\{ \begin{array}{l} \text{create,corpus,drop,list,display,add,from,delete,insert,put,prepare} \\ \text{and,extract,to,replaceall,for,acutes,a,b,c,d,e,f,g,h,i,j,k,l,m,0,p,q,r,s,t,u,v,w,x,y} \\ \text{z,A,B,C,D,E,F,G,H,I,J,K,L,M,,O,P,Q,R,S,T,U,V,W,X,Y,Z,), (, /,*,ascii,\,} \end{array} \right\}$$

$$P = \left\{ \begin{array}{l} \text{<SENTENCIA>::=<CREATE>|<DROP>|<LIST>|<DISPLAY>|} \\ \quad \text{<ADD>|<DELETE>|<INSERT>} \\ \text{<PREPARE>|<EXTRACT>|<REPLACEALL>|<REMOVE>} \\ \text{<CREATE>::=create corpus <ID >;} \\ \text{<DROP>::=drop corpus <ID >;} \\ \text{<LIST>::=list * from <ID>;} \\ \text{<DISPLAY>::=display <ID> . <NOMARCH >;} \\ \text{<ADD>::=add <DICCIONARIOS> from <NOMARCH >; |} \\ \quad \text{add <DICCIONARIOS> ({<STRING> \,});} \\ \text{<DELETE>::=delete <DICCIONARIOS> from <NOMARCH > ; |} \\ \quad \text{delete <DICCIONARIOS> ({ <STRING> \,});} \\ \text{<INSERT>::=insert <ID> . <NOMARCH >;} \\ \text{<PREPARE>::=prepare<ID>.<NOMARCH > <USING>} \\ \quad \text{put <NOMARCH>;} \\ \text{<USING>::=(<CICLOS>) |<DICCIONARIOS>} \\ \text{<CICLOS>::=<DICCIONARIOS> { <DICCIONARIOS> and}} \\ \text{<EXTRACT>::=extract <STRING> to <STRING> <ID> .} \\ \quad \text{<NOMARCH > put <NOMARCH>;} \\ \text{<REPLACEALL>::=replaceall<STRING>for<STRING> <ID> .} \\ \quad \text{<NOMARCH> put <NOMARCH>;} \\ \text{<REMOVE>::=remove acutes <ID> . <NOMARCH> put} \\ \quad \text{<NOMARCH>;} \\ \text{<ID>::=<LETRA> { <LETRA> \ |<DIGITO> }} \\ \text{<LETRA>::=a| b|c...z|A|B|C...Z} \\ \text{<DIGIT>::=0| 1|2...9} \\ \text{<NOMARCH>::=<ID> . <ID>} \\ \text{<STRING>::=<LETRA> { <LETRA> } |<DIGITO> { <DIGITO>} |} \\ \quad \text{<DIGITO> { <LETRA>} |<SIMBOLO> { <LETRA> |} \\ \quad \text{< } SIMBOLO \text{ >}\} \\ \text{<DICCIONARIOS>::=STOPWORDS | } SYMBOLS \text{ | } \{LABELS\} \\ \text{<SIMBOLO>::=ascii} \end{array} \right\}$$

Después de presentar el diseño del lenguaje, en la siguiente sección procedemos a mostrar la implementación del mismo.

3 Implementación de la gramática

3.1 Implementación en Java

Se ha elegido a Java como lenguaje de programación para la implementación de la gramática y del lenguaje en general por varias razones. Entre las más importantes se tienen las siguientes: 1) Está diseñado para ser independiente de la plataforma y 2) proporciona clases para el manejo de expresiones regulares. Con el uso de las expresiones regulares se tiene un amplio abanico de posibilidades principalmete para hacer búsquedas, para sustituir ocurrencias y para comprobar formaciones

de cadenas. Con la inclusión de las expresiones regulares, Java se convierte en un lenguaje de programación tan flexible como otros más tradicionales: awk, perl, etc. El lenguaje de preprocesamiento cuenta con un intérprete de comandos. Este programa tendrá inmerso un analizador léxico, sintáctico, semántico y por último un cargador de operaciones primitivas.

El analizador léxico lee los caracteres uno a uno desde la entrada y forma grupos de caracteres con alguna relación entre sí denominados *tokens*, que constituirán la entrada para la siguiente etapa del lenguaje; algunos *tokens* válidos para el lenguaje son: **prepare, drop, in, put, from, insert, punto coma, etc.** El analizador sintáctico, toma como entrada los *tokens* que recibe del analizador léxico y comprueba si éstos llegan en orden correcto (orden con respecto a la sintaxis del lenguaje). A continuación se muestran algunos ejemplos de construcciones sintácticas validas y aceptadas por el lenguaje de preprocesamiento:

```
create corpus trec;
add stopwords from temp.txt;
insert trec . ism001.txt;
display trec . introduccion.txt;
list * from trec;
prepare trec . ism001.txt using(stopwords and symbols)
put salida.txt;
drop corpus trec;
remove all acutes trec . uno.txt put temp.txt;
```

La función que realiza el analizador semántico es verificar que las sentencias tecleadas por el usuario tengan significado en el lenguaje y no sean un absurdo. Por último, la función del cargador de funciones es colocar en memoria las colecciones de palabras cerradas, símbolos y etiquetas cada vez que se invoca al lenguaje de preprocesamiento.

El haber programado la gramática en el lenguaje Java, proporcionó un mayor control sobre el código generado, sin embargo se invirtio una mayor cantidad de tiempo para implementar las técnicas de compilación.

3.2 Implementación en Flex y Bison

Existe un par de herramientas que hacen más fácil el desarrollo de compiladores a partir de la especificación de los lenguajes en expresiones regulares y de reglas BNF. Estos programas, llamados *flex* y *bison*, fueron utilizados para comprobar si la gramática fue construida correctamente.

Se tradujo la gramática diseñada en el formato requerido por *flex* y la salida obtenida de éste fue usada como archivo de entrada para *bison*. El programa *bison* a su vez generó un archivo de salida con código fuente en lenguaje C que, después de ser compilado, generó un programa con la capacidad de verificar la sintaxis del lenguaje de preprocesamiento.

4 Pruebas

4.1 *Corpora* de prueba

A continuación se muestra una descripción detallada de las características que tienen los *corpora* utilizados como recursos de prueba.

Corpus TREC-5: Es una colección de la agencia de noticias del diario de Guadalajara Jal. formada por un total de 58,062 documentos (198 MB).

Corpus EuroGOV: Está compuesto de documentos Web obtenidos de sitios gubernamentales Europeos. El *corpus* tiene cerca de 3 millones y medio de documentos en diferentes idiomas provenientes de diversos paises o dominios de Internet (fr, es, uk, pt, etc). El tamaño aproximado es de 70 Gigabytes.

Corpus Medicina: Textos del área de Medicina, compilados en un solo archivo de 5.17MB, construido por un alumno de la FCC Benemérita Universidad Autónoma de Puebla.

Corpus Economía: Documentos del área de Economía, compilados en un solo archivo de 3.18MB, construido por un alumno de la FCC Benemérita Universidad Autónoma de Puebla.

Estas dos últimas colecciones carecen de formatos complejos, únicamente poseen un delimitador simple entre documentos.

4.2 Ejemplos de prueba

Las figuras 2 y 4 muestran los resultados arrojados después de preprocesar los *corpora* TREC–5 y EuroGOV (se usa el programa generado en java).

A continuación se muestra el conjunto de sentencias introducidas por el usuario para realizar el preprocesamiento del *corpus* TREC–5, EuroGOV.

```
create corpus trec;
insert trec . ism001.txt;
list * from trec;
display trec . introduccion.txt;
prepare trec . ism001.txt using(stopwords and symbols and labels)
put salida.txt;

create corpus eurogov;
insert eurogov . in.txt;
list * from eurogov;
display trec . in.txt;
remove acutes eurogov . in.txt put out.txt;
prepare eurogov . out.txt using stopwords
put final.txt;
```

```
<DOC>
<DOCNO> SP94-0202041 </DOCNO>
<ARTNUM> 0202041 </ARTNUM>
<HEADLINE>
 Para viajar en avión
</HEADLINE>
<TEXT>
hay que tener buen oido
El NORTE / Especial
        Como consecuencia de los cambios en la presión atmosférica que llevan
aparejados, los viajes en avión pueden producir serias alteraciones en el oido
medio, sobre todo si estaba previamente lesionado.
      En condiciones normales, gracias a la compensación que se produce
mediante la entrada de aire a través de la trompa de Eustaquio ésta comunica el
oido medio con la nasofaringe la presión que existe a ambos lados del timpano es
expandirse.
</TEXT>
</DOC>
```

Figura 1. *Corpus* TREC-5 sin preprocesar

```
Para viajar avión
tener buen oido
NORTE / Especial
        Como consecuencia cambios presión atmosférica llevan
aparejados viajes avión pueden producir serias alteraciones oido
medio sobre todo si estaba previamente lesionado
      condiciones normales, gracias compensación produce
mediante entrada aire través trompa Eustaquio comunica
oido medio nasofaringe presión existe ambos lados timpano
expandirse
```

Figura 2. *Corpus* TREC-5 preprocesado

```
<P><FONT FACE="Arial">1. Con un a&ntilde;o de antelaci&oacute;n a la finalizaci&oacute;n
        del plazo establecido para la prestaci&oacute;n del servicio universal
        en una determinada zona, el Ministerio de Fomento realizará&aacute; una
        consulta p&uacute;blica, para determinar si existen operadores interesados
        en prestarlo y en qu&eacute; condiciones. A estos efectos, dichos operadores
        comunicarán&aacute;n al Ministerio de Fomento el &aacute;mbito territorial,
        per&iacute;odo y condiciones en que estarían dispuestos a llevarlo
        a cabo.</FONT></P>
      <P><FONT FACE="Arial">En las zonas en las que ning&uacute;n operador manifieste
        su inter&eacute;s en prestar el servicio, será&iacute; de aplicaci&oacute;n
        lo establecido en el <A HREF="#a20">art&iacute;culo anterior</A>.</FONT>
        <li>ISLAS ATL&Aacute;NTICAS DE GALICIA
        <i>En fase de elaboraci&oacute;n.</i></li>
      </P>
```

Figura 3. *Corpus* EuroGOV sin preprocesar

```
<FONT FACE="Arial">1. a&ntilde;o antelación finalización
        plazo establecido prestación servicio universal
        determinada zona, Ministerio Fomento realizará
        consulta pública, determinar existen operadores interesados
        prestarlo condiciones. efectos, dichos operadores
        comunicarán Ministerio Fomento ámbito territorial,
        periodo condiciones estarian dispuestos llevarlo
        cabo.</FONT>
      <FONT FACE="Arial"> zonas ningún operador manifieste
        interes prestar servicio, aplicacion
        establecido <A HREF="#a20">articulo anterior</A>.</FONT>
        <li>ISLAS ATLÁNTICAS DE GALICIA
        fase elaboración.</i></li>
```

Figura 4. *Corpus* EuroGOV preprocesado

5 Conclusiones

Hemos presentado una nueva alternativa para el preprocesamiento de textos. La herramienta desarrollada permitirá al investigador de PLN evitar programar procedimientos de textos *ad-hoc* para sus *corpora*. Tal vez el mayor aporte de este trabajo sea el hecho de aplicar procedimientos estándar para la etapa de proprocesamiento, facilitando de esta manera el análisis comparativo de las técnicas de PLN, sin importar el investigador que lo realice.

Referencias

1. Pinto David y Jiménez Héctor *Recuperación de Información*, Notas de Academia (2003).
2. Reyes Berenice, Moyolt Edgar, Jiménez Héctor *Reducción de términos índice usando el punto de transición*, Benemerita Universidad Autónoma de Puebla (2003).
3. Fraguela Liset y Jiménez Héctor *Uso de lattices para la Recuperación de términos*, Benemerita Universidad Autónoma de Puebla, Puebla, México (2003).