Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Preprocesamiento y Normalización del Texto de un Sistema de Conversión Texto a Habla

Tesis de Licenciatura en
Ciencias de la Computación

Alumno: Ezequiel Saudino

Director: Agustín Gravano

Buenos Aires, marzo 2015

RESUMEN

Esta tesis consiste en el desarrollo e implementación de módulos que realizan el preprocesamiento y normalización de texto (front-end) de un sistema de conversión de Texto a Habla o TTS (Text-to-Speech) y su integración con sistemas Generadores de Habla (back-ends) existentes, logrando un sistema Texto a Habla completo para el español. El alcance definido para el sistema es el conjunto de artículos de *Wikipedia* en español. Se construyeron módulos que realizan las tareas de análisis de texto previas a la generación del habla: extracción del texto del artículo HTML de *Wikipedia*, separación del texto en oraciones, transformación de palabras en inglés para que se pronuncien adecuadamente en español (ej.: "green" a "grin"), normalización de números, abreviaturas, siglas y otros símbolos (ej.:"$3" a "tres pesos") e interacción con los sistemas generadores del habla (Festival y MARY TTS) para obtener el audio de cada oración y reproducirlo en forma fluida. Para el módulo de separación del texto en oraciones se obtuvo una tasa de error de 1,27 % sobre una muestra de diez artículos, comparable a las que encontramos en la literatura para el español y otros idiomas. Para el procesamiento de palabras del inglés se logró generar un procedimiento que construye su pronunciación en español, se evaluaron las 200 palabras del inglés con mayor frecuencia en *Wikipedia* en español con una tasa de aciertos del 90,5 % o 94,4 % si se pondera por la frecuencia. Para el módulo de normalización se obtuvo una tasa de aciertos de 97,3 % sobre una muestra de cinco artículos. También se alcanzó el objetivo de integración con los sistemas back-end logrando un sistema completo de Texto a Habla con una salida fluida del audio resultante.

Índice general

Agradecimientos

Al Dr. Agustín Gravano, por darme la oportunidad de trabajar con él, por su generosidad, paciencia y excelente predisposición.

A mi esposa Paula y a mis hijos: Sofía, Mercedes y Nicolás, por el tiempo que permitieron que dedique a la tesis.

A mi familia, sin su ayuda y apoyo no hubiese podido llegar hasta acá.

A mis amigos, por su apoyo e insistencia en que termine la carrera.

Capítulo 1

Introducción

Un sistema Texto a Habla (en inglés, text-to-speech o TTS) toma como entrada un texto y produce como salida un audio conteniendo el habla correspondiente. El sistema debe permitir la reproducción de cualquier palabra u oración del lenguaje, es decir, no alcanza con concatenar palabras u oraciones previamente grabadas, debe ser capaz de pronunciar cualquier expresión posible del lenguaje.

Algunos de los usos posibles de un sistema Texto a Habla son los siguientes:

- Aplicaciones educativas: es posible utilizar un sistema Texto a Habla como parte de un sistema educativo para niños o para la enseñanza de idiomas.
- Aplicaciones para personas con discapacidad: puede ayudar a personas con afasia motora/sensorial, a mejorar la comunicación con otras personas ingresando el texto por teclado u otro dispositivo. También puede ayudar a personas con discapacidad visual en la lectura de textos.
- Aplicaciones para conductores de vehículos: un sistema Texto a Habla permite escuchar la lectura de textos mientras se conduce un vehículo.
- Interfaces de dispositivos: junto a sistemas de reconocimiento del habla los sistemas Texto a Habla pueden ser usados como interfaz de cualquier dispositivo electrónico para la interacción con el usuario.

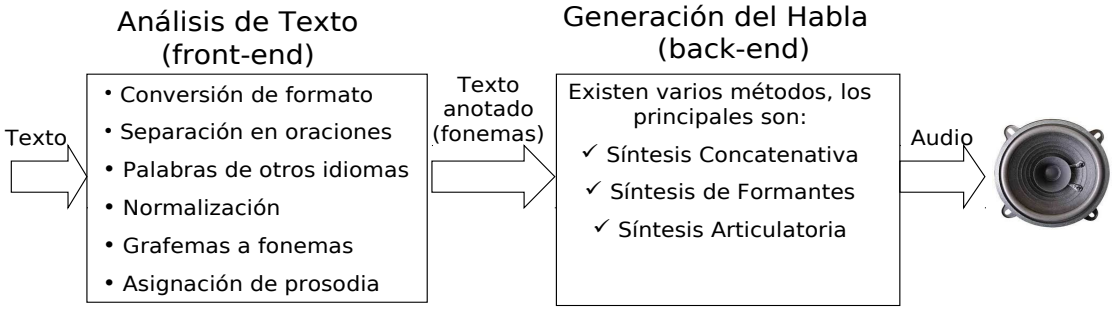

Figura 1.1: Arquitectura de un Sistema Texto a Habla

La figura 1.1 representa la arquitectura de un sistema Texto a Habla que, en general, se divide en dos módulos bien diferenciados:

1. Análisis de texto (front-end), comprende en general las siguientes tareas:
   - Conversión de formato: consiste en obtener texto plano de un origen en otro formato como la web, email, entre otros.
   - Separación del texto de entrada en oraciones.
   - Conversión de palabras de otros idiomas. (Ejemplos: *Washington* a *uashinton*, *Hollywood* a *joligud*).
   - Normalización del texto: consiste en expandir números, fechas, abreviaturas, etc.
   - Conversión de grafemas a fonemas (*zoquete* a *sokete*).
   - Asignación de la prosodia adecuada.
2. Generación del Habla (back-end).
   - Esta etapa toma como entrada una secuencia de fonemas y datos acerca de la prosodia y produce señales acústicas. Hay diferentes métodos para hacerlo: basada en HMM, síntesis de formantes, síntesis articularoria, entre otros.

Se puede encontrar una detallada descripción de las etapas de un Sistema Texto a Habla en el capítulo 8 de "Speech and Language Processing" de Daniel Jurafsky y James H. Martin[9].

1.1. Objetivo

Al momento de comenzar este trabajo si se quería implementar un Sistema Texto a Habla para el español se contaba con los recursos que se muestran en la figura 1.2.

Para el módulo de Análisis de Texto, dependiendo del dominio definido era necesario desarrollar algún módulo que extraiga y realice alguna conversión. *FreeLing*[12, 13, 4, 3], una librería para análisis de lenguaje, brinda funcionalidades para separar el texto en oraciones pero es necesario personalizar algunos parámetros y crear el módulo. Para el tratamiento de Palabras de Otros Idiomas no se encontraron recursos disponibles para el español. Para la tarea de Normalización de Texto, existen FSTs (traductores de estados finitos, en inglés *finite-state transducers*) desarrollados en la Tesis de Licenciatura de Verónica Pechersky[14] que traducen una palabra a su versión expandida, pero era necesario definir a qué expresiones del texto aplicar estas conversiones. Para las tares de Conversión de Grafemas a Fonemas, Asignación de Prosodia y Generación del Habla existen 2 sistemas disponibles: Festival[8] y MARY TTS[6] que cubren estas funcionalidades, para ambos sistemas también existen voces en español argentino creadas durante la Tesis de Licenciatura de Luisina Violante[19, 11]. Además de completar las tareas de procesamiento de texto que faltaban, era necesario integrar todas las herramientas y articular su funcionamiento para obtener un sistema completo. A partir de este diagnóstico se definió el objetivo de este trabajo.

El objetivo de esta tesis fue desarrollar e implementar un sistema TTS para el idioma español Argentino que sea capaz de leer artículos de *Wikipedia* en español. El trabajó se

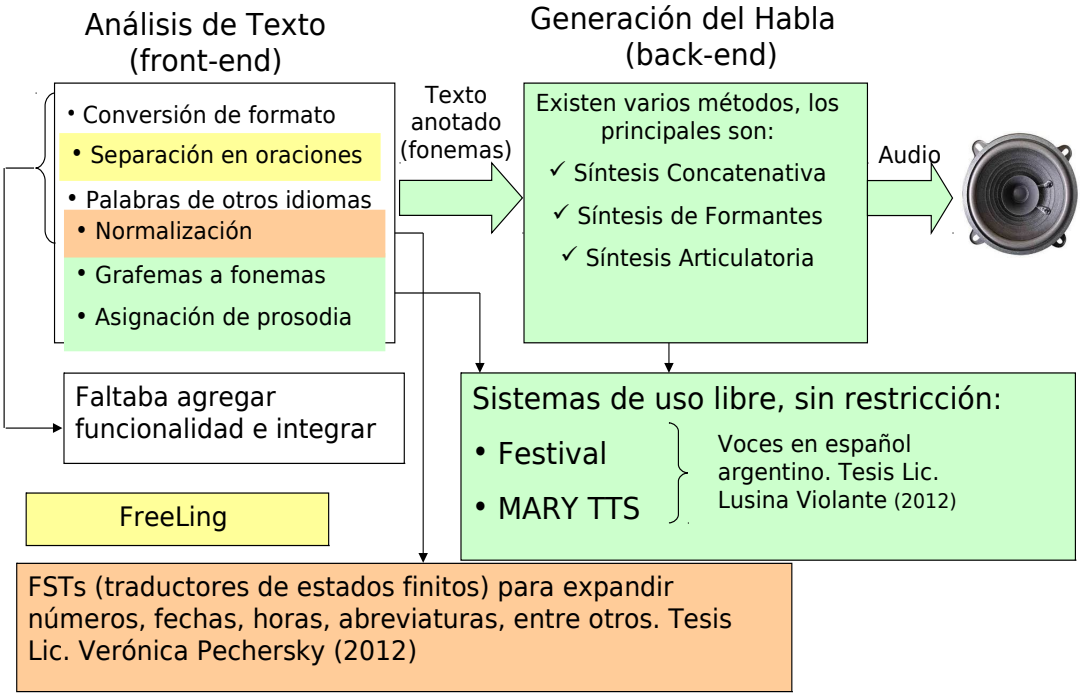

Figura 1.2: Recursos Disponibles para Implementar un Sistema TTS

focalizó en la construcción del sistema front-end que se conectó a dos sistemas back-end distintos: *Festival* y *Mary TTS* y de esta forma se llegó a un sistema completo TTS. El alcance definido del trabajo no incluyó las tareas de conversión de grafemas a fonemas y asignación de prosodia, dado que en este caso dichas funcionalidades están incluidas en los sistemas back-end. Además se definió que el diseño del sistema fuera modular, de manera que sus módulos puedan reemplazarse por nuevos desarrollos que implementen mejoras o que aborden el problema aplicando otras técnicas. También se estableció que los módulos fueran programados en c++ para el sistema operativo Linux.

1.2. Estructura de la Tesis

El funcionamiento del sistema construido se ilustra en la figura 1.3.

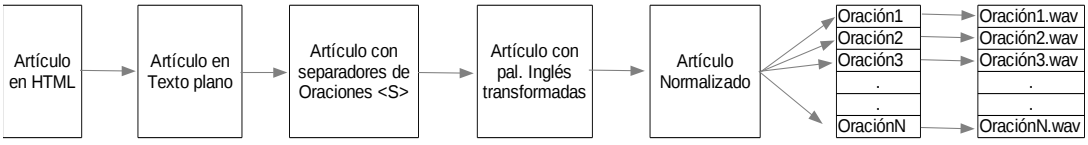

Figura 1.3: Esquema del funcionamiento del sistema

La estructura de esta tesis sigue el orden de acciones de esta figura dedicando un

capítulo a cada módulo del sistema. A continuación se describen brevemente:

- **Capítulo 2 - Extracción del texto de un artículo de Wikipedia**: en este capítulo se describe cómo se construyó el módulo 1 del sistema, que es el encargado de extraer el texto principal de un artículo de *Wikipedia* en HTML y dejar como resultado texto plano. En la figura 1.4 se puede ver un ejemplo sencillo de un fragmento de código HTML y el resultado luego de ejecutar el módulo 1.

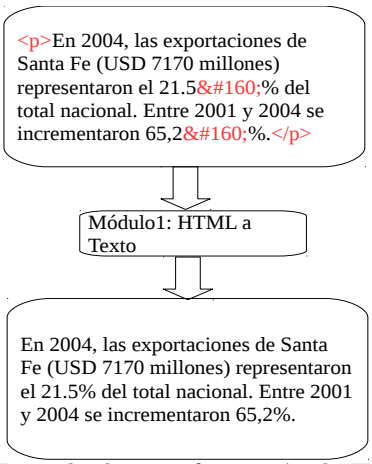

Figura 1.4: Ejemplo de transformación de HTML a Texto

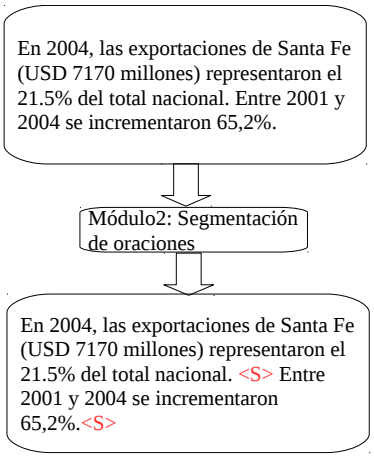

Figura 1.5: Ejemplo de segmentación de oraciones

- **Capítulo 3 - Segmentación de Oraciones**: En este capítulo se describe el módulo 2, que toma el texto y debe separarlo en oraciones. Como ejemplo, en la figura 1.5 se puede ver un fragmento de texto y el resultado esperado.
- **Capítulo 4 - Creación del Corpus de Datos**: En este capítulo se describe la construcción de una base de datos de artículos de *Wikipedia* en español que se utilizó para construir y evaluar reglas de normalización de texto.
- **Capítulo 5 - Palabras de Otros Idiomas**: En este capítulo se describe cómo se abordó el problema que presenta la existencia de las palabras de otros idiomas, el

inglés en particular, al momento de normalizar el texto. En la figura 1.6 se muestra un ejemplo del resultado que produce el módulo 3 del sistema.

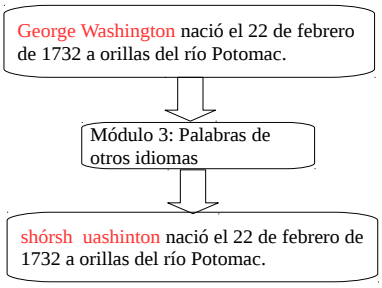

Figura 1.6: Ejemplo de transformación de palabras en inglés

- **Capítulo 6 - Normalización**: En este capítulo se describe la construcción del módulo 4 y las reglas de normalización generadas para detectar y expandir números, siglas, abreviaturas y otros símbolos que lo requieren. En la figura 1.7 se muestra un fragmento de texto de ejemplo y el resultado luego de la normalización.

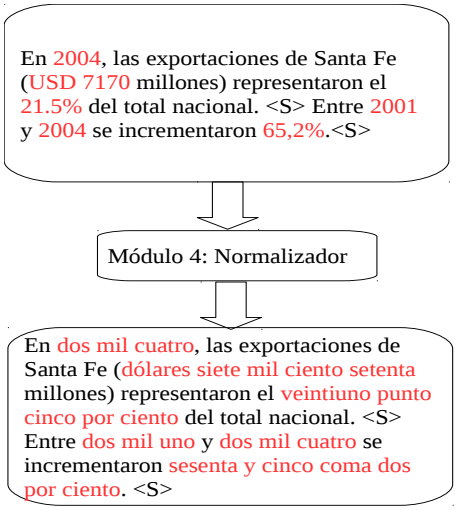

Figura 1.7: Ejemplo de normalización de texto

- **Capítulo 7 - Integración con el sistema back-end**: En este capítulo se describe la construcción del módulo 5 y 6 que transforman el texto e interactúan con los sistemas back-end para obtener el audio final del texto del artículo.
- **Capítulo 8 - Conclusiones**: En este capítulo se mencionan los resultados obtenidos y las posibles mejoras a futuro.
- **Apéndice A - Instalación y Uso del Sistema**: En este apéndice se incluyen instrucciones para la instalación y uso del sistema.

Capítulo 2

Extracción del texto de un artículo de Wikipedia

La primera acción del sistema es obtener el texto que se desea sintetizar, en este caso se trata de extraerlo de un artículo de Wikipedia en español a partir del ingreso de una URL, por ejemplo: http://es.wikipedia.org/wiki/UBA. En este capítulo se describe cómo se obtiene la página web del sitio de Wikipedia, se da una descripción sobre lo que se consideró como contenido principal del artículo y se explica el procedimiento que extrae el texto contenido en el documento HTML.

2.1. Obtención del la página web

Para obtener la página web de Wikipedia simplemente se usa el comando wget [1] de linux con las opciones: -q (quiet mode) modo silencio que no muestra la descripción de los pasos que realiza y -O- (output file) que especifica que el documento de salida se escribirá en standard output. Esta última opción permite concatenar procesos permitiendo que la salida de este proceso sea la entrada del siguiente.

Ejemplo:

wget -q-O- http://es.wikipedia.org/wiki/UBA

2.2. Contenido principal del artículo de Wikipedia

El alcance de este módulo del sistema está acotado a extraer el "texto principal" del artículo de *Wikipedia*. No existe una definición formal de este concepto y tampoco existe algún elemento en el código HTML que lo delimite en forma clara, por lo tanto el enfoque fue tomar todo el texto omitiendo elementos accesorios de la página como ser: índices, tablas, referencias, texto asociado a fotos o imágenes y enlaces externos.

En la Figura 2.1 se puede ver la captura de pantalla de un breve artículo de *Wikipedia* en español. Se marcaron con un recuadro rojo las secciones del documento que se consideran texto principal, su contenido formará parte del archivo de texto que se sintetizará.

[1] http://www.gnu.org/software/wget/manual/wget.html

Los elementos recuadrados en color azul son los que se omiten y por lo tanto su contenido se excluye de las futuras etapas del proceso.

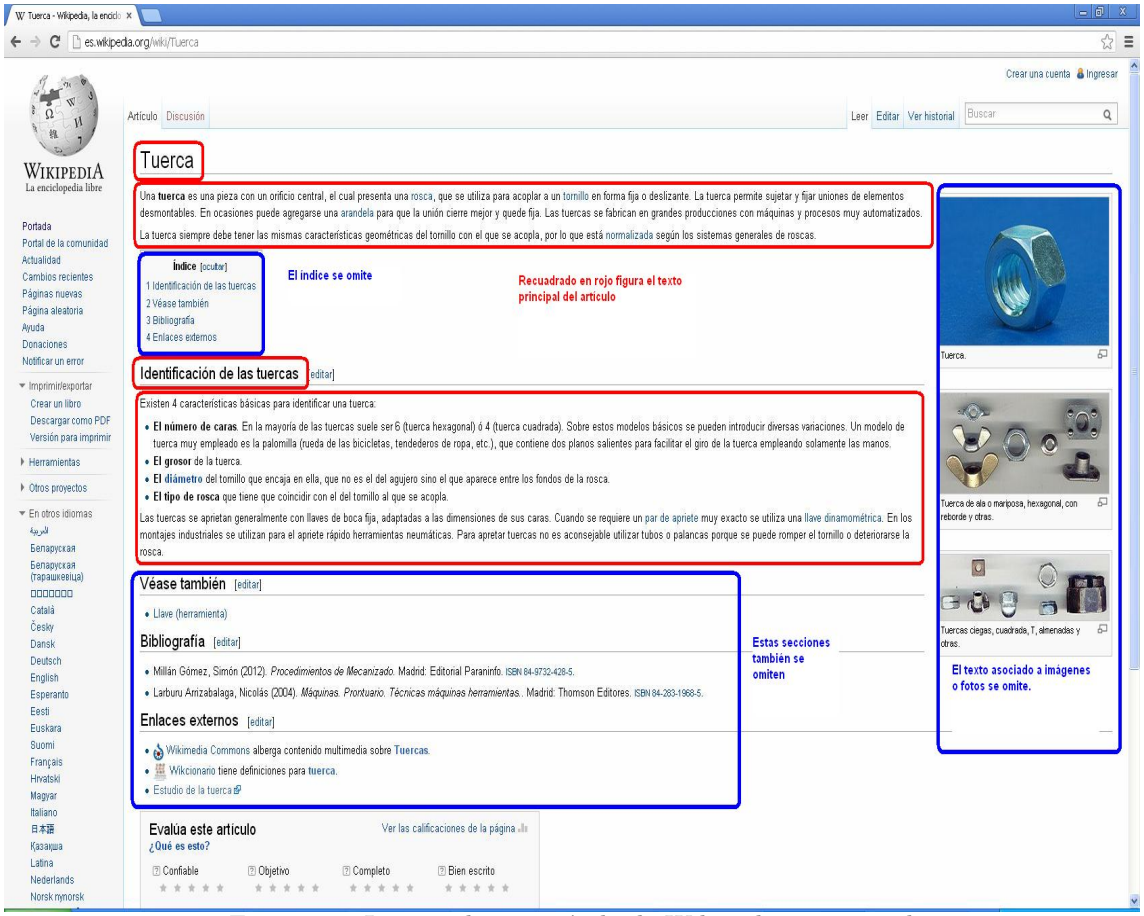

Figura 2.1: Imagen de un artículo de Wikipedia en español

2.3. Extracción del texto del documento HTML

Para realizar la transformación de la página web a un archivo de texto plano se utiliza la librería *htmlcxx*[2][5] que provee funcionalidad para construir un árbol que contiene los elementos de un documento HTML[3], que luego se puede recorrer en profundidad evaluando elemento por elemento y respetando el orden visual del documento.

Los elementos HTML que almacenan texto y que son los que se deben tener en cuenta para extraer el contenido del documento son:

- <p>: párrafos.
- <h1>,<h2>, ...,<h6>: títulos

[2]http://htmlcxx.sourceforge.net
[3]http://http://es.wikipedia.org/wiki/HTML

- ,<ul>: listas de items ordenadas y no ordenadas
- : items incluídos en listas

El contenido principal del documento está incluido en el elemento <body>, por lo tanto sólo se exploran los elementos incluidos dentro del mismo. El resto, por ejemplo: tablas <table>, código fuente <script>o cualquier otro elemento distinto a los enumerados anteriormente se ignoran.

```
<!DOCTYPE html>

<head> <!-- Encabezado, contiene datos para el navegador -->
........
</head>

<body> <!-- Este elemento contiene  el cuerpo principal del documento -->

        <h1>Tuerca</h1>  <!-- Título principal del artículo -->

        <p>Una <b>tuerca</b> es una pieza con un orificio central, el cual presenta una <a href="/wiki/Roscado"
               title="Roscado">rosca</a>.....</p> <!-- Párrafo con texto-->

         <table id="toc" class="toc"><!-- Tabla de contenido,  se omite-->
        .....
        .....

        </table>

        <h2>Identificación de las tuercas</h2> <!-- Título  -->

        <p>Existen 4 características básicas para identificar una tuerca:</p> <!-- Párrafo con texto  -->

        <ul> <!-- Lista no ordenada (bullets)  -->

                <li><b>El número de caras</b>.</li> <!-- Elemento de la lista  -->

                <li><b>El grosor</b> de la tuerca.</li> <!-- Elemento de la lista  -->

                <li><b>El <a href="/wiki/Di%C3%A1metro" title="Diámetro">diámetro</a></b> del
                       tornillo que encaja en ella..... </li> <!-- Elemento de la lista  -->

                <li><b>El tipo de rosca</b> que tiene que coincidir con el del tornillo al que se acopla.</li>
                <!-- Elemento de la lista  -->
        </ul>

        <p>Las tuercas se aprietan generalmente con llaves ...</p> <!-- Párrafo con texto  -->

        <h2>Véase también</h2> <!-- Título, a partir de acá se omite  todo el resto-->

</body>

</html>
```

Figura 2.2: Código HTML simplificado

La Figura 2.2 muestra el código HTML simplificado correspondiente al artículo de la Figura 2.1. El texto en color rojo corresponde a elementos de donde se extrae el texto del contenido principal: títulos, párrafos, listas. En azul figuran los que se omiten, en este ejemplo la tabla de contenido y el último título "Véase también", que corresponde a una sección de anexos muy frecuente en Wikipedia como lo son también: "Enlaces externos",

"Referencias" y "Estadísticas" que también se omiten.

El procedimiento para transformar el documento HTML a texto plano es el siguiente:

1. Se lee el código HTML
2. Se crea el árbol de análisis sintáctico
3. Se recorre el árbol en profundidad hasta encontrar el elemento <body>
4. Para cada elemento dentro de <body>:
   - Si el elemento es <p>, <h1-h6>, <ol>, <ul>,
     → Se extrae el texto de todos los nodos interiores de izquierda a derecha, se quitan referencias bibliográficas y etiquetas que no forman parte del contenido (ej.:"[12]", "[editar]") y se imprime en el archivo de salida
   - Parar si se encuentra un elemento que indique el final del contenido principal. Ej.: títulos: "Véase también", "Enlaces externos", "Referencias", "Estadísticas"
5. Fin

La librería *htmlcxx*[5] suministra la función text() que devuelve el texto de un nodo del árbol de parsing. Esta funcionalidad es muy útil, ya que extrae el texto considerando hipervínculos y de otros elementos anidados en el interior del nodo.

Ejemplo: dado el siguiente elemento, un item de una lista <li>:

```
<li><b>El <a href=/wiki/Di%C3%A1metro"title="Diámetro">diámetro</a></b>del
tornillo que encaja en ella..... </li>
```

La función text() devuelve el texto: "El diámetro del tornillo que encaja en ella...."

Por último, un detalle a mencionar es que antes de guardar el texto en el archivo de salida se quitan etiquetas de referencias bibliográficas que están definidas por un número entre corchetes: ej.: [10], [12]. También se quita la cadena "[editar]" que proviene de un hipervínculo de la web y que tampoco debe formar parte del texto a sintetizar.

2.4. Resultados y Trabajo Futuro

El resultado del módulo es texto plano que se puede grabar en un archivo de texto o enviarlo a la salida standard (*standard output*), para que el siguiente módulo del sistema lo capture. El módulo funcionó correctamente para todos los casos de prueba realizados, es decir, el texto de salida corresponde al contenido de la página web del artículo y el tiempo de respuesta es adecuado.

El trabajo a futuro podría consistir en extender el alcance, para que soporte otros sitios web en español y así se podría ampliar el espectro de uso de todo el sistema TTS.

Se realizaron pruebas con páginas de otros sitios web como, por ejemplo, los diarios Clarín[4] y La Nación [5]. Si bien todo el contenido figura en la salida, es necesario trabajar en filtrar los elementos que no forman parte del artículo, por ejemplo: nombres de secciones, texto de hipervínculos, títulos de artículos periféricos y texto de contenido legal.

[4] http://www.clarin.com.ar
[5] http://www.lanacion.com.ar

Capítulo 3

Segmentación de Oraciones

Luego de obtener el texto del artículo de *Wikipedia*, el siguiente paso es segmentarlo en oraciones que serán las unidades a sintetizar por el sistema. Este capítulo describe el módulo que realiza la demarcación del texto, indicando dónde termina cada oración. Se utiliza la librería *FreeLing*[12, 13, 4, 3], que ofrece funcionalidades para análisis de lenguaje.

3.1. Descripción del problema

Una oración es un conjunto de palabras con que se expresa un sentido gramatical completo. La segmentación de oraciones se basa en los signos de puntuación (. ? !) ya que ellos, la mayoría de las veces, marcan el final de las oraciones. Los signos de interrogación y exclamación son marcadores relativamente confiables del final de una oración, sin enbargo los puntos son más ambiguos. Ejemplo:

"*La empresa Y.P.F. anunció hoy en su sitio web www.ypf.com que la producción de gas natural ascendió 1.7 %* "

En el ejemplo vemos cómo el "." se utiliza no sólo como signo de puntuación delimitando oraciones, sino también en siglas, URLs y expresiones numéricas. Además, se lo puede encontrar en abreviaturas (etc., ej.), fechas (19.10.2013) y códigos (versión 10.1) entre otros.

Como vemos la segmentación de oraciones está estrechamente vinculada a la segmentación de palabras, por este motivo se suele abordar ambos problemas en forma simultánea. En general, los métodos de segmentación de oraciones aplican un clasificador binario (basado en una secuencia de reglas o en aprendizaje automático) que decide si un punto es parte de una palabra o es un signo de puntuación que actúa como delimitador de una oración.

Para nuestro módulo, primero segmentamos el texto en palabras aplicando un conjunto de reglas (expresiones regulares) que, entre otras cosas, nos permiten diferenciar los puntos que delimitan una oración de los que forman parte de una palabra. Luego, la segmentación de oraciones consiste en recorrer las palabras y ver si se trata de un delimitador de oración (. ? !).

3.2. FreeLing

FreeLing[1] es una librería de código abierto orientada a brindar herramientas para el análisis del lenguaje. Se distribuye bajo la Licencia Pública General (GPL) de GNU[2] de software libre. El proyecto *Freeling* es liderado por Lluís Padró, su objetivo es poner a disposición de la comunidad los logros alcanzados en el Centro de Investigación TALP[3] (Centro para el desarrollo de Tecnologías y Aplicaciones del Lenguaje y del Habla) de la UPC[4] (Universidad Politécnica de Cataluña). La librería fue desarrollada en C++ y, entre muchas funciones, ofrece un módulo denominado *Tokenizer*; que permite segmentar el texto en palabras (*tokens*). *FreeLing* también contiene el módulo *splitter*, que recibe una lista de palabras y la segmenta en oraciones. En este trabajo se utilizó la versión 3.1. El módulo desarrollado marca el final de cada oración con el marcador <S> aplicando el siguiente procedimiento:

1. Se lee el texto completo
2. Se invoca la función *Tokenizer::tokenize* que convierte el texto en una lista de palabras.
3. Se invoca la función *Splitter::split* que recibe la lista de palabras y la segmenta en oraciones obteniendo una lista de oraciones.
4. Se recorre la lista de oraciones y se imprime en el archivo todas las palabras que forman la oración, al finalizar se marca el fin de la oración con el marcador: <S>
5. Fin

Los módulos *Tokenizer* y *Splitter* se usaron casi como vienen en *Freeling*, se realizaron sólo algunos cambios menores. A continuación se describen el funcionamiento de ambos modulos y las modificaciones introducidas.

3.3. Tokenizer

El módulo *Tokenizer* de *FreeLing* convierte el texto en una lista de palabras (*tokens*) aplicando un conjunto de reglas. Estas reglas son expresiones regulares que se evalúan en orden, ante la primera coincidencia (*matching*) se extrae la subcadena que coincide con la expresión regular y se repite el proceso hasta extraer la última palabra del texto.

Las expresiones regulares del módulo *Tokenizer* están definidas en el archivo tokenizer.dat que se muestra en la Figura 3.1. El archivo esta dividido en tres secciones: <Macros>, <RegExps> y <Abbreviations>.

La sección <Macros> permite definir macros de expresiones regulares para facilitar la escritura posterior de las reglas. Las macros se definen con un nombre y una expresión regular con sintaxis POSIX. Ejemplo: MYALPHA [A-Za-z]

La sección <RegExps> define las reglas para separar las palabras. Se puede hacer referencia a las macros definidas anteriormente con su nombre entre llaves. Ejemplo: SOLOLETRAS {MYALPHA}+

[1] http://nlp.lsi.upc.edu/freeling/
[2] http://www.gnu.org/licenses/
[3] http://www.talp.upc.edu/
[4] http://www.upc.edu/

```
## macros to be used in RegExps rules
<Macros>
ALPHA       [[:alpha:]]
ALPHANUM    [[:alnum:]ºª]
SYMNUM      [\.:,_/\=\*\+\-ºª&$€£¢¥#%²]
SYM         [ºª&$€£¢¥#%²]
OTHERS      .
</Macros>
##
## Tokenization rules. They are applied in the order of definition.
## The first matching the *beginning* of the line is applied
## and a token built. The process is repeated until the line
## has been completely processed.
##    -The first field in the rule is the rule name. If it starts
##     with a "*", the RegExp will only produce a token if the
##     match is found in abbreviation list <Abbreviations> below.
##    -The second field in the rule is the substring to form the token/s with
##     It may be 0 (the match of the whole expression) or any number
##     from 1 to the number of substrings (up to 9). A token will be
##     created for each substring from 1 to the specified value.
##    -The third field is the regexp to match against the line
##
<RegExps>
INDEX_SEQUENCE  0  (\.{4,}|-{2,}|\*{2,}|_{2,}|/{2,})
INITIALS1       1  ([A-Z](\.[A-Z])+)(\.\.\.)
INITIALS2               0  ([A-Z]\.)+
NRO                     0  (n|N)(\.)?(º|ª)
POTENCIA                0  [\-\+]?[0-9]+([\.],][0-9]+)?((<sup>[\-\+]?[0-9]+</sup>)|²|³)
NOTCIENTIFICA           0  [\-\+]?[0-9]+([\.],][0-9]+)?x10<sup>[\-\+]?[0-9]+</sup>
GRADOS_COORD    0  [\-\+]?([0-9]+([\.],][0-9]+)?(º|°)([F|C])?([\-\+]?[0-9]+([\.],][0-9]+)?['|'|']?([\-\+]?[0-9]+([\.],][0-9]+)?["|"|"|"])?(S|s|N|n|E|e|O|o|w|W)?)
MINUTOS                         0  [\-\+]?([0-9]+([\.],][0-9]+)?['|']
SEGUNDOS                0  [\-\+]?([0-9]+([\.],][0-9]+)?["|"|"])
HORAS                   0  (([0-1]?[0-9])|2[0-3]):[0-5]?[0-9](:[0-5]?[0-9])
TIMES           0  (([01]?[0-9]|2[0-4]):[0-5][0-9])
MEDIDAS                         0  ({ALPHA}|[0-9])+/({ALPHA}|[0-9])+
NAMES_CODES     0  ({ALPHA}|{SYMNUM})*[0-9]({ALPHA}|[0-9]|{SYMNUM}+{ALPHANUM})*
THREE_DOTS      0  (\.\.\.)
APOSTROFES      0  {ALPHA}+['´]{ALPHA}+
QUOTES          0  (``|<<|>>|'')
APOSTR_CAT      1  ([dD]')({ALPHA})
MAILS           0  {ALPHANUM}+([\._]{ALPHANUM}+)*@{ALPHANUM}+([\._]{ALPHANUM}+)*
URLS1           0  ((mailto:|(news|http|https|ftp|ftps)://)[\w\.\-]+|^(www(\.[\w\-]+)+))
URLS2           1  ([\w\.\-]+\.(com|org|net))[\s]
KEEP_COMPOUNDS  0  {ALPHA}+([_\-\+]{ALPHA}+)+
*ABREVIATIONS1  0  (({ALPHA}+\.)+)(?!\.\.)
*ABREVIATIONS2  0  ({ALPHA}+\.)(?!\.\.)
NAMES_CODES2    0  {ALPHANUM}+{SYM}
WORD            0  {ALPHANUM}+[\+]*
ENDLINE         0  <S>
OTHERS_C        0  {OTHERS}
</RegExps>

## Abbreviations.  The dot is not tokenized separately
## in the cases listed below.
<Abbreviations>
aa.rr.
abr.
abrev.
a.c.
……
</Abbreviations>
```

Figura 3.1: Contenido del archivo tokenizer.dat

Las reglas son expresiones regulares que se aplican siguiendo el orden de definición, es decir, se usa la primera regla que genere una coincidencia con una subcadena del comienzo del texto de entrada. Si esto ocurre se ignora el resto de las reglas y se repite el proceso hasta llegar al final del texto. El formato de cada regla es:

1. El primer elemento de una regla es el nombre, si comienza con un *, la expresión regular generará una palabra si la misma se encuentra en la lista de abreviaturas (sección <Abbreviations>).

2. El segundo elemento es un número que define cómo se crean las palabras a partir de la cadena que coincide con la expresión regular: 0 se crea una palabra que coincide con toda la expresión regular, un número entre 1 y 9 define que se creará una palabra para cada subexpresión (de la expresión regular) desde 1 hasta el valor indicado.

   Ejemplo: INITIALS1 1 ([A-Z](.[A-Z])+)(...)

   Con el texto “ J.F.K...”, *Tokenizer* produce una palabra para la primera subexpresión entre paréntesis, ([A-Z](.[A-Z])+), y deja la segunda parte. En este caso se crea la

palabra “J.F.K” y “...” queda para la siguiente iteración.

Si el valor fuese 0, se crearía la palabra con la cadena completa que coincide con la expresión regular: “J.F.K....”.

Si el valor fuese 2, se producirían dos palabras, una para cada subexpresión: “J.F.K” y “...”.

3. El tercer elemento es la expresión regular, escrita en sintaxis POSIX.

La sección <Abbreviations> define las abreviaturas más frecuentes, indica que la palabra no debe ser separada del “.” que le sigue. Ejemplos: etc., Sr., Sra.

En la Figura 3.1 se pueden ver las reglas que se aplican para separar las palabras. Por ejemplo, la macro ALPHA define los símbolos alfabéticos, letras, excluyendo los símbolos enumerados en la expresión regular. Se usa $\wedge$ para definir la exclusión, \s para denotar espacios y \d para dígitos (0-9).

Las reglas definidas que trae la distribución de *FreeLing* resultaron adecuadas pero también fue necesario agregar algunas. En la figura 3.1 aparecen en amarillo las reglas agregadas para este trabajo. Por ejemplo, la regla “NRO” se definió para obtener en una única palabra expresiones como: “N^{o}”, “N.^{o}”, “n^{o}”. Las regla POTENCIA se utiliza para obtener en una única palabra expresiones que el módulo 1 detecta como potencias, por ejemplo “1,66 x 10^{-23}”, el módulo 1 la convierte en el texto: “1,66x10⁻²³” y gracias a esta regla *Tokenizer* asigna toda esta expresión en una única palabra, esto facilita la tarea posterior de normalización. De manera similar las reglas NOTCIENTIFICA define números en formato de notación científica, GRADOS_COORD define expresiones numéricas para ángulos, temperaturas o coordenadas MINUTOS se definió para expresiones como 14’, SEGUNDOS para expresiones como 34”, HORAS para expresiones como “12:23:23”, MEDIDAS para expresiones como “cm^2, m^3”, APOSTROFES para expresiones como “I’m”. Por último ENDLINE identifica en una única palabra el marcador de fin de oración, dado que se usa *Tokenizer* para separar palabras en los módulos siguientes.

También se agregaron casos al listado de abreviaturas para enriquecerlo con los casos más frecuentes que se encontraron en *Wikipedia*. Con estas reglas se distinguen las apariciones de puntos en expresiones numéricas: regla NAMES_CODES (ej.: 20.000,23), abreviaturas: reglas ABREVIATIONS (ej.:“ Y.P.F. subió la producción”) y otras como direcciones de correo electrónico y URLs: reglas MAILS y URLS (ej.:“alejandro@dc.uba.ar”, “www.dc.uba.ar”) de las que actúan como signos de puntuación: regla OTHERS (ej.: “Ayer, fuimos al cine.”). En este último caso los signos de puntuación quedan formando una palabra de un sólo caracter (ej.: “,” , “.”). Esto facilita la tarea siguiente, que es delimitar el final de las oraciones.

3.4. Splitter

El módulo splitter recibe un listado de palabras y genera un listado de oraciones.

La Figura 3.2 muestra el contenido del archivo Splitter.dat que define algunos parámetros para el funcionamiento del módulo.

El archivo está dividido en cuatro secciones: <General>, <Markers>, <SentenceEnd>, <SentenceStart>.

La sección <General> contiene opciones generales, una de ellas es *AllowBetweenMarkers* que determina si se permite separar una oración aunque el separador quede dentro

de delimitadores (paréntesis, llaves, comillas, etc.). Para nuestro trabajo preferimos tener esta opción habilitada (*AllowBetweenMarkers*=1) para evitar problemas de memoria en situaciones donde falte el delimitador de cierre. La opción *MaxLines* define cuántas palabras como máximo puede tener una oración dentro de los delimitadores, este valor tiene sentido cuando se activa *AllowBetweenMarkers*, para este trabajo se definió *MaxLines* = 150, también para evitar problemas por uso excesivo de memoria.

La sección <Markers> define los delimitadores: “”, (), , /**/.

La sección <SentenceEnd> define qué caracteres son considerados como potenciales delimitadores de oraciones. Cada caracter se define con un valor (0/1) que indica si es ambiguo/no ambiguo. Si se define como no ambiguo (valor 1), cualquier aparición de ese caracter define un corte de oración. Los caracteres ambiguos producen un corte de oración sólo si a continuación existe una palabra que comience en mayúscula o un caracter de comienzo de oración. Para nuestro trabajo nos quedamos con las opciones por defecto: “. 0”, “? 0”, “! 0”.

La última sección del archivo es <SentenceStart>, que define qué caracteres aparecen sólo al comienzo de oraciones: signos de interrogación y exclamación.

```
<General>
AllowBetweenMarkers 1
MaxLines 150
</General>
<Markers>
" "
( )
{ }
/* */
</Markers>
<SentenceEnd>
. 0
? 0
! 0
</SentenceEnd>
<SentenceStart>
¿
¡
</SentenceStart>
```

Figura 3.2: Contenido del archivo Splitter.dat

3.5. Resultados y Trabajo Futuro

Para tener una idea concreta del desempeño de este módulo del sistema, se seleccionaron 10 artículos de *Wikipedia* de diferentes temas y se procesaron con los dos primeros módulos para obtener un archivo de texto con las oraciones delimitadas con el marcador <S>, en total los 10 artículos suman un volumen de 90.186 palabras. Luego, un etiquetador humano analizó la salida producida para encontrar los errores en la segmentación de oraciones. Se distinguieron dos tipos de errores:

1. Por omisión: estos casos suceden cuando no se detecta el final de la oración. Ejemplo:

".... de energía con gas natural , biomasa , etc. En algunos de estos aspectos ..." En este ejemplo, no se detecta el final de oración posterior a "etc.", es una omisión de fin de oración.

2. Por error: comprenden casos donde se marca erróneamente el final de una oración. Ejemplo: "... hacia Europa , la Argentina y EE . <S> UU . . <S> ". En este caso el primer marcador <S> no debería existir.

	Oraciones				
Artículo	**Marcadas**	**Incorrectas**	**Omitidas**	**Total Oraciones**	**%error**
Argentina	842	12	11	841	2,73%
Átomo	181	0	4	185	2,16%
Clima	208	0	1	209	0,48%
Julio Cortázar	206	0	2	208	0,96%
UBA	445	0	2	447	0,45%
Uruguay	791	5	4	790	1,14%
Astronomía	283	0	1	284	0,35%
Sol	196	0	1	197	0,51%
Comunismo	184	2	2	184	2,17%
Messi	351	0	0	351	0,00%
Total	**3.687**	**19**	**28**	**3.696**	**1,27%**

Figura 3.3: Evaluación de errores del módulo de segmentación de oraciones

La Tabla 3.3 muestra los resultados obtenidos en la evaluación del módulo. El error promedio de los 10 artículos analizados es 1,27 %, el artículo con mayor porcentaje de errores es "Argentina", con 2,73 %.

La tasa de error en las pruebas es comparable a las que encontramos en la literatura para el español y otros idiomas [15, 10]. Estos resultados, junto con el hecho de que no se detectaron errores en el funcionamiento, nos permite concluir que el desempeño del módulo es aceptable.

Como trabajo a futuro, se identificaron los siguientes casos a resolver:

- Cuando se usa el punto de una abreviatura también como punto final de la oración. Ejemplo: "...se erradicó el paludismo, etc. A través de la Fundación Eva Perón,..."
- Cuando luego de un punto, la oración siguiente no comienza con una letra (comienza con comillas, números, paréntesis, corchetes, etc.). En estos casos no se marca el final de la oración anterior. Ejemplo: "...pero no fue totalmente feliz. « Mucha servidumbre , excesiva sensibilidad..."
- Un caso más dificil de abordar es cuando el escritor omite el punto al final de la oración. Ejemplo: "...condiciones económicas y sociales desfavorables El Censo de 2001 estableció..."
- Por último, estimamos que se puede mejorar el desempeño enriqueciendo aún más el listado de siglas y abreviaturas.

Capítulo 4

Creación del Corpus de Datos

Antes de avanzar con la descripción de los siguientes módulos del sistema, se presenta el corpus de datos que se construyó para detectar los patrones más importantes y así determinar las transformaciones que se realizarán durante el proceso de normalización. También se utilizó el cuerpo de datos para estimar el porcentaje de casos cubiertos y el nivel de error.

4.1. Origen de los datos

Wikipedia ofrece descargar archivos con el contenido de su base de datos de artículos. Se puede acceder a ellos a través de la URL: http://dumps.wikimedia.org/eswiki/. El texto de los artículos en español se encuentra en el archivo: eswiki-latest-pages-articles.xml.bz2, es un archivo comprimido que contiene el archivo XML eswiki-latest-pages-articles.xml Para esta tesis se trabajó con la versión del 2 de marzo de 2012 de 6.0GB.

4.2. Procesamiento

El archivo eswiki-latest-pages-articles.xml.bz2 fue procesado con el script en python WikiExtractor (versión 1.5) [1]. Este script fue creado por Giuseppe Attardi (attardi@di.unipi.it) y Antonio Fuschetto (fuschett@di.unipi.it) de la universidad de Pisa y se distribuye bajo la Licencia Pública General (GPL) de GNU[2] de software libre. El script convierte el archivo XML original en cientos de archivos de texto más pequeños.

<doc id="2153" url="http://it.wikipedia.org/wiki/Pol%C3%ADtica_del_Per%C3%BA">
Política del Perú.
La población del Perú, cuya denominación oficial es <a href="Rep
%C3%BAblica_del_Per%C3%BA">República del Perú, está organizada bajo un Estado,
conformado con base en la Constitución política
del Perú aprobada en 1993 mediante referéndum,
promulgada a finales de ese mismo año y vigente desde el 1 de enero de <a
href="1994">1994.
Las directrices dictadas por la Constitución permiten un amplio espectro de posibilidades y posturas políticas. Si bien
el artículo 58° de la Constitución señala que el Perú se rige bajo una <a href="Econom
%C3%ADa_social_de_mercado">economía social de mercado, donde la iniciativa privada es libre y el Estado
asume un rol regulador, las prácticas políticas dependen de la iniciativa del gobierno de turno.

Figura 4.1: Fragmento de ejemplo de la salida del proceso WikiExtractor.py

[1] http://medialab.di.unipi.it/Project/SemaWiki/Tools/WikiExtractor.py
[2] http://www.gnu.org/licenses/

El resultado de este proceso no deja archivos de texto puros, sino que hay tags que quedan sin procesar. En la Figura 4.1 se puede ver un fragmento de ejemplo. Para obtener archivos de texto puros, sin tags, fue necesario crear un proceso adaptando el módulo 1, que convierte archivos HTML a texto. En la Figura 4.2 se muestra el resultado del fragmento de ejemplo.

Política del Perú.

La población del Perú, cuya denominación oficial es República del Perú, está organizada bajo un Estado, conformado con base en la Constitución política del Perú aprobada en 1993 mediante referéndum, promulgada a finales de ese mismo año y vigente desde el 1 de enero de 1994.

Las directrices dictadas por la Constitución permiten un amplio espectro de posibilidades y posturas políticas. Si bien el artículo 58° de la Constitución señala que el Perú se rige bajo una economía social de mercado, donde la iniciativa privada es libre y el Estado asume un rol regulador, las prácticas políticas dependen de la iniciativa del gobierno de turno.

Figura 4.2: Fragmento de ejemplo de la salida del proceso HTML a texto

Como resultado de los dos procesamientos se obtuvieron 9.796 archivos de texto plano con el contenido de los artículos de *Wikipedia* en español.

4.3. Resultados

Como resultado, se obtuvo una base de datos con el listado de todas las palabras distintas y su frecuencia de aparición. La tabla 4.1 muestra algunas estadísticas. Se distinguieron 2.3 millones de palabras distintas y 339.5 millones de apariciones en total. La definición de "palabra" es la que surge de aplicar las reglas de tokenización descriptas en el capítulo 3.

Cnt Apariciones	#Palabras	%tot palabras	# apariciones	% tot. Apariciones
>=1000	18.040	0,76%	307.892.156	90,67%
>= 500	29.225	1,24%	315.777.558	92,99%
>= 100	82.516	3,50%	327.455.113	96,43%
>= 50	127.084	5,39%	330.568.619	97,35%
1	1.182.581	50,15%	1.182.581	0,35%
Total	**2.358.211**	**100,00%**	**339.564.233**	**100,00%**

Tabla 4.1: Estadísticas sobre la cantidad de palabras de Wikipedia en español

También se generó un proceso que recorre los 9.796 achivos de texto de Wikipedia en busca de palabras que cumplieran con un patrón determinado, por ejemplo palabras formadas por las letras: I, V, X, L, C, D, M para analizar números romanos. Cuando se encontraba una palabra que cumplía esas condiciones se tomaba el contexto donde aparecía: las 5 palabras anteriores y posteriores. De esta manera se construyó una base de datos que permite analizar las situaciones más importante en que se utiliza una expresión

con el fin de crear reglas de normalización. Esto se verá con mayor detalle en el capítulo 6. La Figura 4.3 muestra un fragmento de ejemplo de la base de datos construida para analizar el contexto en que aparcen los números romanos.

Anterior5	Anterior4	Anterior3	Anterior2	Anterior1	Palabra	Posterior1	Posterior2	Posterior3	Posterior4	Posterior5
por	Su	Santidad	Juan	Pablo	**II**	.	Está	incardinado	en	la
por	el	papa	Juan	Pablo	**II**	Obispo	auxiliar	de	Oviedo	y
nombrado	por	el	papa	Benedicto	**XVI**	obispo	de	Ciudad	Rodrigo	
los	tiempos	del	Papa	Pío	**XII**	.	Obras	.	Hablemos	
tienen	una	longitud	de	5	**cm**	con	largos	pelos	rojos	
la	agalla	es	de	5	**mm**	de	longitud	;	la	cabeza
vez	a	finales	del	siglo	**XVII**	como	la	propiedad	privada	

Figura 4.3: Fragmento de la base de datos construida para analizar números romanos

Capítulo 5

Palabras en Otros Idiomas

Si bien se asume que el contenido a sintetizar está escrito en idioma español, suelen aparecer en el texto palabras o frases de otros idiomas que no se traducen al español. En general se trata de nombres propios, marcas, frases literales o simplemente palabras que con el uso se incorporaron al lenguaje español. Por ejemplo: Washington, Hollywood y hockey son palabras del idioma inglés que podemos encontrar en textos en español y para que el sintetizador las pronuncie en forma correcta hace falta transformarlas. Siguiendo el ejemplo, Washington podría ser *uashinton*, Hollywood: *joligud* y green: *grin*.

En este capítulo se describe el funcionamiento del módulo que detecta palabras de idiomas distintos al español y las transformaciones que realiza para que su pronunciación en español sea lo más natural posible.

5.1. Reconocimiento de palabras del idioma Inglés

El primer intento para determinar si una palabra pertenece a otro idioma consistió en buscarla en un diccionario, inglés en este caso, y si se encontraba entonces se dictaminaba que pertenecía a ese idioma.

Para evaluar cómo funciona esta estrategia, se utilizó la base de datos con las 2.358.211 palabras distintas extraídas de *Wikipedia* con su frecuencia de aparición (ver capítulo 4). Además se utilizó un diccionario inglés con 121.761 palabras.

Se cruzaron lo dos listados de palabras y el resultado fue que 92.688 palabras únicas de *Wikipedia* se encontraron en el diccionario inglés. Estas representan 161.961.166 apariciones en artículos de *Wikipedia* en español, el 47 %. Es decir, si para detectar si una palabra de un artículo de *Wikipedia* en español es del idioma inglés se utiliza simplemente la estrategia de buscarla en el diccionario, casi 1 de cada 2 apariciones de palabras se etiquetarían como del idioma inglés. Estos resultados claramente indican que esta estrategia no es lo suficientemente buena ya que hay muchas palabras en español que están en el diccionario inglés.

Para mejorar los resultados, se analizó el listado de las 92.688 palabras, y se construyeron reglas en forma de expresiones regulares que ayudan a discriminar las palabras del idioma inglés de las del español.

Expr. Regular	Palabras	Apariciones	Ejemplos	Observaciones
(w)	6.575	928.899	new, web, world	
(sh)	2.231	259.588	washington, show, marshall	
(th)	2.324	530.960	the,thomas, smith, with	
(ck)	2.361	262.328	rock, black, hockey, track	
(wn)	276	56.057	brown, down, town	
(oo)	1.500	162.442	school, hollywood, good	
(ph)	714	86.134	stephen, murphy, memphis	
(my)	161	44.433	my, academy, enemy	
(ou)	2.130	288.422	tour, you, house	
(ee)	1.545	192.067	street, creek, green	excluir: cree, creen, lee
^sc	1.039	73.090	science, scout, scott	
tion$	907	65.540	association, playstation	
Total	**19.347**	**2.648.132**		

Figura 5.1: Reglas para discriminar palabras en inglés

En la Tabla 5.1 se muestran las reglas que se definieron para determinar si una palabra es del idioma inglés. Estas reglas mejoran el uso del diccionario pero no lo sustituyen. Es decir, se determina que una palabra es del idioma inglés si se encuentra en el diccionario y además cumple con alguna de las reglas. Por ejemplo, cualquier palabra que se encuentre en el diccionario inglés y que contenga una "w" (regla 1), una "sh" (regla 2), que comience con "sc" (regla 11) o que termine con "tion" (regla 12) se trata como una palabra del idioma inglés.

Las reglas cubren 18.453 palabras de las 92.688 que se encuentran en el diccionario inglés, el 20 % pero tan sólo representan 2.585.019 apariciones de las 161.961.166 en total, el 1.60 %, ver figura 5.2. El porcentaje de apariciones es bajo porque efectivamente se trata de palabras del idioma inglés, que tienen una frecuencia baja en textos en español, mucho menor a la frecuencia de palabras del listado que también pertencen al idioma español.

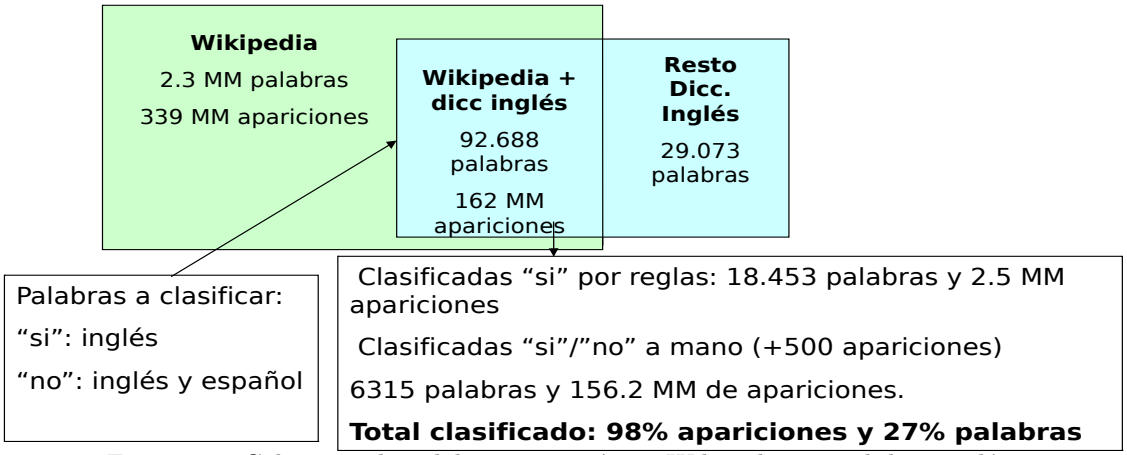

Figura 5.2: Cobertura de palabras que están en Wikipedia y en el dicc. inglés

Para ampliar la cobertura de casos se analizó el listado de las 74.235 palabras que no quedaron comprendidas por ninguna regla, ordenado por la cantidad de apariciones en forma descendente. Se recorrió a mano el listado de palabras con más de 500 apariciones: 6315 palabras que acumulan 156.213.746 apariciones en artículos de *Wikipedia* en español. Se trata del 96.45 % del las apariciones que se deben clasificar. Se marcaron cuáles eran del idioma inglés y precisaban alguna transformación previa antes de sintetizarlas en es-

pañol. El resultado es un listado de 850 palabras del idioma inglés. La Tabla 5.3 muestra las primeras palabras del listado. Las restantes 5.465 palabras eran palabras del idioma español, por lo que puden ignorarse.

Además, se analizó el listado de 5000 palabras más frecuentes del idioma inglés [1], se excluyeron a mano las palabras que también se usan en español y las que ya estaban en el listado de 850 palabras. Quedaron 3751 palabras de uso frecuente en el idioma inglés que se agregaron al listado de 850 palabras más frecuentes del inglés en *Wikipedia* en español, de esta forma se llegó a un listado de 4601 palabras del idioma inglés.

Palabra	Apariciones	Palabra	Apariciones	Palabra	Apariciones	Palabra	Apariciones
york	55.075	jones	9.184	johann	5.256	sport	3.904
john	56.051	harry	9.060	jacques	4.947	oklahoma	3.862
to	26.412	lake	9.006	beatles	4.921	utah	3.782
george	21.254	mark	9.023	bay	4.901	dead	3.607
paul	20.902	punk	8.805	lady	4.856	jefferson	3.586
michael	19.852	hot	8.672	heavy	4.779	renault	3.579
single	17.859	one	8.405	road	4.753	baby	3.555
charles	16.620	joe	8.264	home	4.753	jerry	3.540
love	14.504	joan	7.371	amateur	4.748	maryland	3.468
jazz	13.650	blue	7.085	jan	4.705	great	3.447
st	12.653	jersey	6.885	rose	4.568	cleveland	3.372
henry	12.434	blues	6.830	apple	4.560	seattle	3.360
college	11.507	roger	6.729	hard	4.419	orange	3.337
jean	11.334	rugby	6.395	ryan	4.186	heart	3.294
royal	10.975	taylor	6.210	cup	4.179	dream	3.136
national	10.838	beach	6.136	jeff	4.144	graham	3.086
league	10.561	by	5.920	channel	4.131	grove	3.047
billboard	9.903	day	5.899	jason	4.047	plate	3.038
ohio	9.753	up	5.825	harvard	3.968	kate	3.015
saint	9.744	team	5.757	russel	3.967	rangers	3.009

Figura 5.3: Palabras en inglés que necesitan alguna transformación

El esquema completo para detectar si una palabra es del idioma inglés se muestra en la Figura 5.4. Primero se busca la palabra en el listado de 4601 palabras si se encuentra se la considera una palabra en inglés y se la transforma. Si no se encuentra en el listado, se busca en el diccionario inglés; si se encuentra y cumple alguna de las reglas (Tabla 5.1) se la considera una palabra en inglés y se modifica para mejorar su pronunciación en español. De acuerdo a este esquema, la transformación definida en el listado tiene prioridad por sobre la definición del diccionario.

Para poder contemplar algunas conjugaciones de verbos, si la palabra a analizar termina en “ed” (pasado simple) o “ing” (presente continuo) se analiza también la palabra sin la terminación y en caso de ser inglesa, a la pronunciación resultante se le concatena la pronunciación de la terminación (“id” o “ing”). Por ejemplo, al analizar “walked” o “talking”, se analizan también “walk” y “talk”. En el primer caso, la pronunciación resultante es “uók” + “id” = “uókid”, en el segundo es “tók” + “ing” = “tóking”. En la sección siguiente se describe cómo se genera pronunciación en español de las palabras en inglés.

Se evaluó hacer algo similar con las palabras en plural, las que terminan en “s” o “es”. Sin embargo se generan errores como, por ejemplo, la palabra “grandes” al quitarle la terminación “es” y analizar la palabra “grand” se dictaminaría que es inglesa y se pronunciaría en inglés cuando es española.

[1] http://www.wordfrequency.info

Un caso particular e importante es la palabra "I" ("yo" en inglés). En textos en español, la palabra "I" tiene diversos usos: como número romano (Juan I), como variable (I + 2) o como palabra en inglés (I love you), entre otros. Se tomó la decisión de tratarla como palabra inglesa sólo si la palabra inmediata anterior o posterior se dictamina inglesa de acuerdo al esquema de la Figura 5.4. Por ejemplo, en la frase "How can I do that?", la palabra "I" se transforma a la pronunciación "ai" si la palabra "can" o "do" se dictaminan que son palabras inglesas.

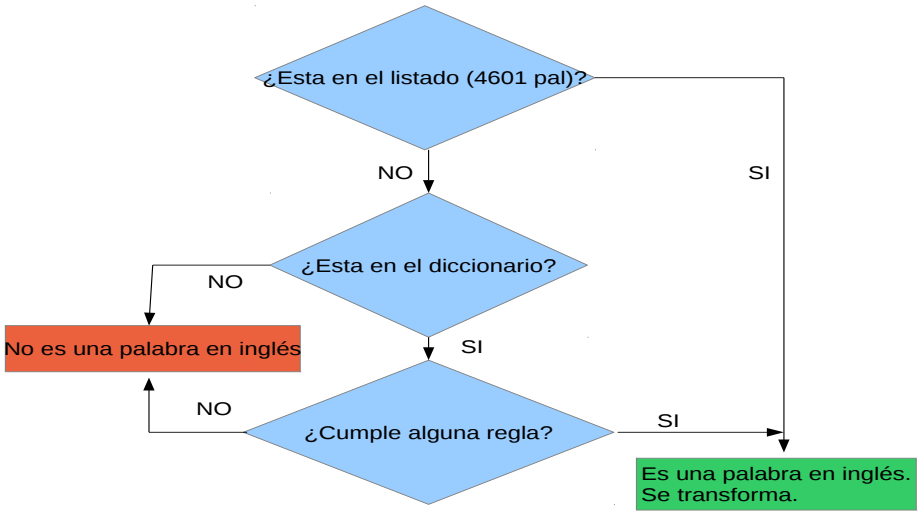

Figura 5.4: Esquema de detección de palabras en inglés

La figura 5.5 muestra un resumen de las estadísticas. En total se clasificaron 24.768 palabras, el 26,72 % de las 92.688 palabras distintas que se encontraron en el diccionario inglés. Debido a que la clasificación se focalizó en las palabras más frecuentes, se logró una buena cobertura si se tiene en cuenta las apariciones: las palabras clasificadas aparecen 158.798.765 veces en *Wikipedia*, el 98,05 % de las apariciones totales de las 92.688 palabras.

	Palabras	%Palabras	Apariciones	% Apariciones
En diccionario Inglés	92.688	100,00%	161.961.168	100,00%
Total cubierto	24.768	26,72%	158.798.765	98,05%
Reglas	18.453	19,91%	2.585.019	1,60%
Palabras con >500 apariciones	6.315	6,81%	156.213.746	96,45%

Figura 5.5: Detección de palabras en inglés - Resultados

5.2. Transformación de palabras en inglés

Una vez que se detecta una palabra que pertenece al idioma inglés, el siguiente paso es transformarla de manera tal que su pronunciación en español sea lo mejor posible. Algunos ejemplos ya mencionados de lo que se pretende son: Washington: *uayinton*, Hollywood: *joligud*, green: *grin*.

Para esta tarea se utilizó el diccionario fonético inglés de la Universidad Carnegie

	Ejemplo		
Fonema	Palabras	Fonemas	Español
AA	odd	AA D	a
AE	at	AE T	a
AH	hut	HH AH T	a
AO	ought	AO T	o
AW	cow	K AW	au
AY	hide	HH AY D	ai
B	be	B IY	b
CH	cheese	CH IY Z	ch
D	dee	D IY	d
DH	thee	DH IY	d
EH	Ed	EH D	e
ER	hurt	HH ER T	er
EY	ate	EY T	ei
F	fee	F IY	f
G	green	G R IY N	g
HH	he	HH IY	j
IH	it	IH T	i
IY	eat	IY T	i
JH	gee	JH IY	sh
K	key	K IY	k

	Ejemplo		
Fonema	Palabras	Fonemas	Español
L	lee	L IY	l
M	me	M IY	m
N	knee	N IY	n
NG	ping	P IH NG	n
OW	oat	OW T	ou
OY	toy	T OY	oi
P	pee	P IY	p
R	read	R IY D	r
S	sea	S IY	s
SH	she	SH IY	sh
T	tea	T IY	t
TH	theta	TH EY T AH	t
UH	hood	HH UH D	u
UW	two	T UW	u
V	vee	V IY	v
W	we	W IY	u
Y	yield	Y IY L D	i
Z	zee	Z IY	z
ZH	seizure	S IY ZH ER	sh

Figura 5.6: Asignación de letras (español) a fonemas (inglés)

Mellon[2]. Contiene 133.261 palabras, para algunas de ellas existe más de una definición fonética así, que se optó por la primera definición, que es la más frecuente.

El procedimiento consistió en asignar a cada fonema del diccionario una expresión en letras que represente, en español, el sonido más cercano a la pronunciación del fonema inglés. La Tabla 5.6 muestra todos los fonemas con las letras en español asociadas.

Para transformar una palabra en inglés, se busca su definición fonética en el diccionario y se reemplaza cada fonema por las letras en español asociadas. Por ejemplo, la palabra *green* tiene la definición fonética G R IY1 N que se transforma a letras del español como "grin". De esta manera, a cada palabra del diccionario inglés se le asignó una pronunciación en español, que se utiliza en el momento de reproducir su sonido.

Los fonemas del diccionario están representados con los símbolos *ARPAbet* [3]. Las vocales tienen asociado un código (0, 1 o 2) que indica el tipo de acento que llevan: 0- sin acento, 1- acento primario, 2- acento secundario.

Para la transformación a letras en español se tiene en consideración el acento primario; en ese caso, se acentúa la letra vocal correspondiente. Por ejemplo, la palabra *billboard* cuyos fonemas son B IH1 L B AO2 R D se transforma a *bílbord*, la primera vocal tiene un acento primario por lo tanto lleva tilde cuando se transforma a letras del español.

5.3. Excepciones a la regla

El asignar una letra o conjunto de letras a un fonema no funciona bien en todos los casos. El fonema representado con el símbolo *ARPAbet* AH y que se denomina *Schwa* [4]

[2]http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[3]http://en.wikipedia.org/wiki/Arpabet
[4]http://en.wikipedia.org/wiki/Schwa

puede corresponder en español a sonidos de diferentes vocales de acuerdo a la palabra que se trate. Algunos ejemplos son:

- *about* AH0 B AW1 T : el fonema AH corresponde a la letra 'a' y en español también suena como 'a': "abaut"
- *taken* T EY1 K AH0 N : el fonema AH corresponde a la letra 'e' y en español también suena como 'e': "teiken"
- *eloquent* EH1 L AH0 K W AH0 N T: el primer fonema AH corresponde a la letra 'o' y en español también suena como 'o': "elokuent"

Al observar estos ejemplos se puede advertir que para el caso del fonema *Schwa*, AH, habría que tener en cuenta también la letra asociada al fonema, para determinar el sonido que mejor se aproxima en español.

Una situación similar ocurre con el fonema AA. En algunos casos corresponde a una 'a' en español y en otros a una 'o'. Algunos ejemplos son:

- *odd* AA1 D: el fonema AA corresponde a la letra 'o' y en español también suena como 'o': "od"
- *observation* AA2 B Z ER0 V EY1 SH AH0 N : el fonema AA corresponde a la letra 'o' y en español también suena como 'o': "obzerveishan"
- *harmonic* HH AA0 R M AA1 N: IH0 K: el primer fonema AA corresponde a la letra 'a' y en español también suena como 'a': "armonik"
- *mark* M AA1 R K: el fonema AA corresponde a la letra 'a' y en español también suena como 'a': "mark"

Como ilustran estos ejemplos, estos dos fonemas presentan ambiguedad al momento de convertirlos en letras que representen su sonido en español.

El fonema *Schwa*, AH, es muy frecuente. Está presente en el 43 % de las palabras del diccionario, mientras el fonema AA figura en el 16 % de las mismas. Como se trata de fonemas muy usados, fue necesario flexibilizar el esquema único de asignación de letras a fonemas representado en la Tabla 5.6. De otro modo, la tasa de tasa de errores en la transformación sería muy alta. Para lograrlo, el primer problema a resolver fue: dada una palabra del diccionario y sus fonemas, determinar qué letra/s corresponde a cada fonema. Por ejemplo:

- *about* AH0 B AW1 T : a-AH0, b-B, ou-AW1, t-T
- *taken* T EY1 K AH0 N : t-T, a-EY1, k-K, e-AH0, n-N
- *odd* AA1 D: o-AA1, dd-D
- *mark* M AA1 R K: m-M, AA1-a, r-R, k-K
- *defense* D IH0 F EH1 N S: d-D, e-IH0, f-F, e-EH1, n-N, s-S, e-

Este problema no es sencillo de resolver ya que la correspondencia no es uno a uno, no hay reglas definidas y en algunos casos hay letras que no se asocian a ningún fonema (por ejemplo la última 'e' en *defense*).

Como los fonema AH y AA corresponden a vocales, se redujo el problema a mapear vocales con fonemas de vocales. Aún así el problema es complejo. Se optó por resolver el caso más general, en el cual coincide la cantidad de letras vocales con la cantidad de fonemas de vocales y se consideraron las vocales contiguas como una única vocal. Para esos casos, se asumió que la letra vocal número n corresponde al fonema de vocal número n. Los siguientes ejemplos aclaran esta definición:

- *about* AH0 B AW1 T : 2 vocales (a, ou) y 2 fonemas de vocales (AH,AW). Se asocia la primera vocal 'a' con el primer fonema de vocal: AH y la segunda vocal 'ou' con el segundo fonema de vocal AW.
- *defense* D IH0 F EH1 N S: d-D, e-IH0, f-F, e-EH1, n-N, s-S, e- : 3 vocales (e,e,e) y 2 fonemas de vocales (IH, EH). La cantidad de vocales no coincide con la cantidad de fonemas de vocales, es un caso no cubierto.

De las 53.919 palabras del diccionario que contienen el fonema AH, el 73 % está cubierto con el caso general, mientras que para el fonema AA, el 79 %. Finalmente, la transformación de los fonemas AH y AA se realizó de acuerdo a la Tabla 5.7. Esta tabla se definió analizando varios casos del diccionario.

	Fonema	
Vocal	**AH**	**AA**
a	a	a
e	e	a
i	a	a
o	o	o
u	a	a
ea	i	a
otras:au, ei	vocal	a
Sin dato	a	a

Figura 5.7: Tabla para transformar los fonemas AH y AA

5.4. Resultados

La Tabla 5.8 muestra algunas de las palabras en inglés con mayor frecuencia de apariciones en *Wikipedia* en español, y su pronunciación generada con el procedimiento descrito.

Palabra	Fonemas	Letras
i	AY1	ai
john	JH AA1 N	sh ó n
york	Y AO1 R K	i ó r k
to	T UW1	t ú
george	JH AO1 R JH	sh ó r sh
paul	P AO1 L	p ó l
michael	M AY1 K AH0 L	m ai k a l
single	S IH1 NG G AH0 L	s í n g e l
charles	CH AA1 R L Z	ch á r l z
love	L AH1 V	l á v
jazz	JH AE1 Z	sh á z
st	S T R IY1 T	s t r í t
henry	HH EH1 N R IY0	j é n r i
college	K AA1 L IH0 JH	k á l i sh

Palabra	Fonemas	Letras
jean	JH IY1 N	sh í n
royal	R OY1 AH0 L	r oi a l
national	N AE1 SH AH0 N AH0 L	n á sh a n a l
league	L IY1 G	l í g
billboard	B IH1 L B AO2 R D	b í l b o r d
ohio	OW0 HH AY1 OW0	ou j ai ou
saint	S EY1 N T	s ei n t
all	AO1 L	ó l
jones	JH OW1 N Z	sh ou n z
harry	HH EH1 R IY0	j é r i
mark	M AA1 R K	m á r k
lake	L EY1 K	l ei k
punk	P AH1 NG K	p á n k
hot	HH AA1 T	j ó t

Figura 5.8: Ejemplos de transformación de palabras

Para estimar el desempeño de estas reglas, se analizaron las 200 palabras del idioma inglés de mayor frecuencia en *Wikipedia* en español y se clasificó la pronunciación obtenida en dos categorías: correcta o incorrecta, de acuerdo al criterio del autor de este trabajo. La Tabla 5.9 muestra los resultados obtenidos, al 90.5 % de las palabras se les asignó una pronunciación correcta y si se pondera por la frecuencia de aparición, este porcentaje asciende a 94.4 %.

Pronunciación en español	#Palabras	% Palabras	#Apariciones	% Apariciones
Correcta	181	90,50%	979.184	94,44%
Incorrecta	19	9,50%	57.659	5,56%
Total	**200**		**1.036.843**	

Figura 5.9: Estimación de desempeño en la transformación de palabras

La Tabla 5.10 muestra algunos casos de palabras con pronunciaciones correctas e incorrectas.

Palabra	Fonemas	Pronunciación	Evaluación
paul	P AO1 L	p ó l	Correcta
day	D EY1	d ei	Correcta
johann	Y OW1 HH AA0 N	i ou j a n	Correcta
beatles	B IY1 T AH0 L Z	b í t e l z	Correcta
road	R OW1 D	r ou d	Correcta
home	HH OW1 M	j ou m	Correcta
apple	AE1 P AH0 L	á p e l	Correcta
channel	CH AE1 N AH0 L	ch á n e l	Correcta
manhattan	M AE0 N HH AE1 T AH0 N	m a n j á t a n	Correcta
graham	G R EY1 AH0 M	g r ei a m	Incorrecta
harry	HH EH1 R IY0	j é r i	Incorrecta
taylor	T EY1 L ER0	t ei l er	Incorrecta
harvard	HH AA1 R V ER0 D	j á r v er d	Incorrecta
utah	Y UW1 T AO2	i ú t o	Incorrecta
maryland	M EH1 R AH0 L AH0 N D	m é r a l a n d	Incorrecta
seattle	S IY0 AE1 T AH0 L	s i á t a l	Incorrecta
charlotte	SH AA1 R L AH0 T	sh á r l a t	Incorrecta

Figura 5.10: Ejemplos de evaluación de la pronunciación obtenida

5.5. Trabajo a Futuro

Como trabajo a futuro lo más importante sería ampliar el alcance a palabras de otros idiomas además del inglés, por ejemplo: alemán, francés, portugués, etc.

También se podría continuar el trabajo realizado con el idioma inglés, generando nuevas reglas que permitan aumentar la precisión del clasificador o intentar construir un clasificador nuevo utilizado técnicas de aprendizaje automático.

En cuanto a la transformación de palabras, quedan algunos casos en que se podría mejorar la pronunciación en español, en particular para palabras que contienen el fonema *Schwa*, *ARPAbet* AH. Los casos con cantidad de vocales distinto a cantidad de fonemas de vocales se podría intentar resolver emparejando las consonantes con los fonemas de consonantes

Por ejemplo *defense* D IH0 F EH1 N S: d-D, e-IH0, f-F, e-EH1, n-N, s-S, e- : 3 vocales (e,e,e) y 2 fonemas de vocales (IH, EH). La cantidad de vocales no coincide con la cantidad de fonemas de vocales, es un caso no cubierto.

Capítulo 6

Normalización

Es frecuente encontrar dentro de cualquier fragmento de texto, palabras como números, abreviaturas, siglas o códigos que están escritos en forma diferente a su pronunciación oral. A estas palabras se las conoce en la literatura como NSWs *non-standard words*. En general, las NSWs no se hallan en diccionarios y tampoco se puede construir su pronunciación aplicando reglas que traducen letras a fonemas como en las palabras ordinarias. Además, las NSWs tienen una propensión mayor que las palabras ordinarias a presentar ambigüedades respecto a su interpretación o pronunciación[16]. Antes de enviar el texto al sintetizador, es necesario detectar las NSWs y reemplazarlas por otro texto que represente su pronunciación habitual, a esta última acción se la denomina "expansión". Por ejemplo, la sigla YPF se debe expandir al texto "i pe efe", el número 1998 se debe expandir a "mil novecientos noventa y ocho" y la expresión UTF-8 a "u te efe ocho". Al proceso general de detectar y reemplazar NSWs del texto por otras que representen su pronunciacción, se lo denomina "Normalización". En la literatura se pueden encontrar algunos trabajos que abordan este problema, entre otros: [16] [18].

6.1. Introducción

6.1.1. Detección

En el desarrollo del módulo de normalización, el primer paso es definir qué palabras del texto requieren ser expandidas. Para abordar este problema se analizó el Corpus de Datos descrito en el capítulo 4, con el fin de descubrir los patrones más importantes que describen estas palabras. Estos patrones se definieron mediante expresiones regulares. Por ejemplo, dada la oración: *"La población de Argentina es de 40 millones de habitantes."* y la siguiente expresión regular:

```
^[0-9]+$ : expresión regular para números naturales
```

el proceso de detección consiste en evaluar cada palabra de la oración con la expresión regular. En el ejemplo la única palabra que coincide es "40", que deberá ser reemplazada por su versión en letras. Para este trabajo se definieron 47 reglas de detección además de un procedimiento específico para tratar números romanos, que siguen el procedimiento descrito. Se definió un orden de evaluación de las reglas que va de casos particulares a generales. En las siguientes secciones se enumeran y describen las reglas definidas. En la

implementación del módulo de normalización se utilizó el componente *Regexp* de la biblioteca *Boost*[1] versión 1.48.0-3 y se optó por la sintaxis *Perl*[2] para el manejo de expresiones regulares en *C++*.

6.1.2. Expansión

Cuando se detecta una palabra que se debe normalizar, el siguiente paso es la expansión, es decir, la construcción del texto que represente la pronunciación de esa palabra para su reemplazo. Este problema se resuelve principalmente aplicando el trabajo de Tesis de Licenciatura de **Verónica Pechersky** bajo la dirección de **Dr. Agustín Gravano**[14]. En particular, se utilizaron los FSTs (traductores de estados finitos, en inglés *finite-state transducers*) que traducen una palabra a su versión expandida.

Los FSTs que se utilizan son:

- Números Naturales
- Números Reales
- Números Ordinales
- Fechas
- Letras

Los FSTs definidos para esta tarea fueron desarrollados por Verónica Pechersky[14] utilizando la biblioteca de software libre y código abierto *OpenFST*[3] para Windows. Para poder utilizarlos en el desarrollo del módulo de normalización, fue necesario realizar adaptaciones al código fuente para compilarlos con la versión de *OpenFST* para Linux.

6.1.3. Alcance

El módulo de normalización construido intenta resolver la normalización de las siguientes expresiones:

- Expresiones Numéricas
    - Magnitudes
    - Monedas
    - Fechas y Horas
    - Números Ordinales, Temperaturas, Coordenadas y Angulos
    - Números Enteros, Reales, Potencias y Notación Científica
    - Otras expresiones con dígitos numéricos
- Siglas y Abreviaturas
- Números Romanos

A continuación se describe el análisis y la resolución de cada uno de estos items.

[1] http://www.boost.org
[2] http://www.boost.org/doc/libs/1_34_1/libs/regex/doc/syntax_perl.html
[3] www.openfst.org

6.2. Expresiones numéricas

Uno de los principales problemas a resolver en la etapa de normalización es la detección y expansión de expresiones numéricas. Para este trabajo, el concepto "expresiones numéricas" abarca todo el espectro de palabras que contenga dígitos numéricos en forma total o parcial. En los artículos de *Wikipedia* en español se detectaron 8.615.475 palabras con esta definición, el 2,53 % del total de palabras. Este universo de palabras es el que se toma como base para los porcentajes que se muestran en esta sección como total a cubrir. Se trabajó en construir reglas que cubran la mayor cantidad de expresiones numéricas contenidas en la base de datos. Las reglas están definidas mediante expresiones regulares que evalúan la palabra a analizar y, en caso de ser necesario, también se analiza el contexto (5 palabras anteriores y posteriores) para resolver ambigüedades. Si una palabra cumple con alguna de las reglas, la expansión se resuelve utilizando alguno de los FSTs mencionados en la introducción del capítulo, y en algunos casos también es necesario realizar transformaciones adicionales previas o posteriores a la aplicación de los FSTs.

A continuación se describen las reglas creadas siguiendo el órden de evaluación definido en la implementación de este módulo. Para cada regla se muestra el orden de evaluación, el nombre de la regla, las expresiones regulares que se usan para detectar el caso, la cantidad de casos que se cubren con la regla, el porcentaje que representa, algunas expresiones de ejemplo, el FST que se utiliza y comentarios adicionales.

Orden	Regla	Exp. Regular	#Casos	%Casos	Ejemplos	FST
1	Ms (símbolo la misma palabra)	^[[.plus-sign.][.hyphen.]]?[0-9]+(([.period.]|,)[0-9]+)*SIMBOLO(\\.)?$	88.280	1,02%	25cm / 450hab/km2 / 22,3mts	Reales
2	Medidas (símbolo y nro en palabras distintas)	Palabra Actual ---> ^[[.plus-sign.][.hyphen.]]?[0-9]+(([.period.]|,)[0-9]+)*$ Palabra Posterior---> ^SIMBOLO(\\.)?$	529.775	6,15%	25 cm / 450 hab/km2 / 22,3 mts	Reales
3	Medidas (nro en notación científica)	Palabra Actual---> ^[[.plus-sign.][.hyphen.]]?[0-9]+([[.period.]|,][0-9]+)?x10^{[[.plus-sign.][.hyphen.]]?[0-9]+}$ Palabra Posterior ---> ^SIMBOLO(\\.)?$	S/D	S/D	1,66x10⁻²³ kg	Reales
		Total	**618.055**	**7,18%**		

SIMBOLO =	(ac|a\\.c\\.|dc|d\\.c\\.|km|mts|m|M|cm|cms|mm|UA|mph|mp/h|kmh|km/h|m/s|mts/s|m/seg|mts/seg|m/segs|mts/segs|h|hs|min|mins|'|'|s|seg|segs|"|"|"|"|\"|msnm|ft|km2|km²|mi2|mi²|ha|has|m2|m²|mts2|mts²|cm2|cm²cms2|cms²|mm2|mm²|m3|m³|mts3|mts³|cc|ccc|cm³|cm3|cms³|cms3|mm³|mm3|l|lts|lt|ml|m3/s|m³/s|tn|kg|kgs|grs|gr|mg|lb|lbs|hab|hab/km2|hab/km²|hab/mi2|hab/mi²|GPa|MPa|kPa|hPa|Pa|bar|mbar|PSI|mmHg|atm|Ghz|Mhz|khz|hz|FM|AM|RPM|hp|W|kW|MW|GW|Wh|W/h|kWh|kW/h|MWh|MW/h|GWh|GW/h|J|kJ|MJ|GJ|N|kN|MN|GN|kcal|cal|BTU|kDa|Da|A|mA|dB|²|³|Kb|Mb|Gb|Tb|Kbps|Mbps|Gbps|Mbit|pts)

Tabla 6.1: Reglas para detectar y expandir Magnitudes

6.2.1. Magnitudes

Como primer paso se evalúa si una expresión numérica corresponde a una expresión de magnitud, es decir, un número con una sigla que representa una medida de peso, longitud, temperatura, velocidad, etc.. En la tabla 6.1 se pueden observar las reglas creadas para detectar las magnitudes más frecuentes en *Wikipedia* en español. La primera regla detecta los casos en los cuales el número y el símbolo de medida no tienen un espacio de separación y, por lo tanto, el tokenizador genera una única palabra con ambos (número y símbolo). La segunda regla detecta el caso más frecuente, donde hay un espacio que separa el número del símbolo de medida y quedan en palabras distintas. El tercer caso abarca magnitudes

donde el número esta escrito en notación científica. Para los tres casos la resolución es similar: se separa el número y se lo expande aplicando las reglas de expresiones numéricas que se describen a lo largo de esta sección. Al resultado se le concatena la expansión de la magnitud de acuerdo a la tabla 6.2, esta tabla describe las magnitudes consideradas y si la comparación es *case sensitive* o no. Para la regla 3, números en notación científica, no se cuentan con casos en la base de datos de expresiones numéricas ya que para detectarlos es necesario que el módulo 1, que convierte el HTML a texto, genere los tags necesarios para identificarlos y la construcción de la base de datos fue anterior a la detección de esta necesidad. Por ejemplo, dada la expresión: "1,66 x 10^{-23} kg", el módulo 1 la convierte en el texto: "1,66x10⁻²³kg", es decir, se usa el tag para indicar el superíndice en el texto.

Con las reglas de Magnitudes definidas se cubren 618.055 casos, el 7.18 % de las expresiones numéricas de la base de datos.

Categoría	Expr.Regular	Case sensitive?	Expansión
Años	ac|a\\.c\\.	no	antes de cristo
	dc|d\\.c\\.	no	después de cristo
Longitud	km	no	kilómetros
	mts	no	metros
	m	si	metros
	cm|cms	no	centímetros
	mm	si	milímetros
	UA	si	unidades astronómicas
Velocidad	mph|mp/h	no	millas por hora
	kmh|km/h	no	kilómetros por hora
	m/s|mts/s|m/seg|mts/seg|m/segs|mts/segs	no	metros por segundo
Tiempo	h|hs	si	horas
	min|mins|'|	no	minutos
	s|seg|segs	si	segundos
Altura	msnm	no	metros sobre el nivel del mar
	ft	si	pies
Superficie	km2|km²	no	kilómetros cuadrados
	mi2|mi²	no	millas cuadradas
	ha|has	no	hectáreas
	m2|m²|mts2|mts²	no	metros cuadrados
	cm2|cm²|cms2|cms²	no	centímetros cuadrados
	mm2|mm²	no	milímetros cuadrados
Volúmen	m3|m³|mts3|mts³	no	metros cúbicos
	cc|ccc|cm³|cm3|cms³|cms3	no	centímetros cúbicos
	mm³|mm3	no	milímetros cúbicos
	l|lts|lt	no	litros
	ml	si	mililitros
Caudal	m3/s|m³/s	no	metros cúbicos por segundo
Peso	tn	si	toneladas
	kg|kgs	no	kilogramos
	grs|gr	no	gramos
	mg	si	miligramos
	lb|lbs	no	libras
Presión	GPa	si	yigapascales
	MPa	si	megapascales
	kPa	si	kilopascales
	hPa	si	hectopascales
	Pa	si	pascales
	bar	si	bares
	mbar	si	milibares
	PSI	si	pe ese i
	mmHg	si	milímetros de mercurio
	atm	si	atmósferas

Categoría	Expr.Regular	Case sensitive?	Expansión
Frecuencia	Ghz	no	yigajertz
	Mhz	no	megajertz
	khz	no	kilojertz
	hz	no	jertz
	FM	no	efe eme
	AM	no	a eme
	RPM	no	revoluciones por minuto
Potencia	hp	no	caballos de fuerza
	kW	no	kilovatios
	MW	no	megavatios
	GW	no	yigavatios
Energía	Wh|W/h	no	vatios hora
	kWh|kW/h	no	kilovatios hora
	MWh|MW/h	no	megavatios hora
	GWh|GW/h	no	yigavatios hora
	J	si	julios
	kJ	si	kilojulios
	MJ	si	megajulios
	GJ	si	yigajulio
Fuerza	kN	si	kiloñutons
	MN	si	megañutons
	GN	si	yigañutons
Energía Térmica	kcal	si	kilocalorías
	cal	si	calorías
	BTU	si	be te us
Masa atómica	kDa	si	kilodaltons
	Da	si	daltons
Intesidad eléctrica	A	si	amperes
	mA	si	miliamperes
Sonido	dB	si	decibeles
Potencia matemática	²	si	al cuadrado
	³	si	al cubo
Datos	Kb	no	kilobaits
	Mb	no	megabaits
	Gb	no	yigabaits
	Tb	no	terabaits
	Kbps	no	kilobits por segundo
	Mbps	no	megabits por segundo
	Gbps	no	yigabits por segundo
	Mbit	no	megabits
Población	hab	no	habitantes
	hab/km2|hab/km²	no	habitantes por kilómetro cuadrado
	hab/mi2|hab/m²	no	habitantes por milla cuadrada
Otros	pts	no	puntos

Tabla 6.2: Expansión de Símbolos de Magnitudess

6.2.2. Monedas

El segundo paso en el procesamiento de expresiones numéricas es evaluar si la misma expresa un importe de dinero, es decir, si el número esta acompañado de un símbolo de

moneda. La tabla 6.3 muestra las reglas definidas para detectar estos casos. El número y el símbolo de moneda pueden estar en la misma palabra o en dos palabras distintas y el símbolo de moneda puede estar delante (más frecuente) o detrás del número, las reglas definidas abarcan las cuatro combinaciones. Para la expansión, se separa el número, se lo expande aplicando las reglas para números reales y se antepone la expansión del símbolo de moneda de acuerdo a la tabla 6.4.

Hay casos para los que la decisión de anteponer la expansión del símbolo de moneda genera frases correctas pero poco naturales, por ejemplo: "$2" se expande a "pesos dos". Si se agrega detrás de la expansión del número, se producen expansiones incorrectas como en el siguiente ejemplo: "$ 12 millones" se expandiría a "12 pesos millones". Para evitar estos errores, se prefiere anteponer la expansión del símbolo, el ejemplo anterior se expande a "pesos doce millones".

Con estas reglas se cubren 97.248 casos, el 1.13 % de las expresiones numéricas de la base de datos.

Orden	Regla	Exp. Regular	#Casos	%Casos	Ejemplos	FST
4	Monedas (símbolo adelante y en la misma palabra)	^SIMBOLO([0-9]|[.period]|,)+(([.period.]|,)[0-9]+)?	83.442	0,97%	$2.223 / €56,345.89 / ARS34.900	Reales
5	Monedas (símbolo detrás y en la misma palabra)	^([0-9]|[.period]|,)+(([.period.]|,)[0-9]+)? SIMBOLO$	43	0,00%	2.223$ / 56,345.89€ / 34.900ARS	Reales
6	Monedas (símbolo adelante y en distintas palabras)	Palabra Actual---> ^SIMBOLO$ Palabra Posterior ---> ^([0-9]|[.period]|,)+ (([.period.]|,)[0-9]+)?$	4.854	0,06%	$ 2.223 / € 56,345.89 / ARS 34.900	Reales
7	Monedas (símbolo detrás y en distintas palabras)	Palabra Actual ---> ^([0-9]|[.period]|,)+ (([.period.]|,)[0-9]+)?$ Palabra Posterior---> ^SIMBOLO$	8.909	0,10%	2.223 $ / 56,345.89 € / 34.900 ARS	Reales
		Total	**97.248**	**1,13%**		

SIMBOLO =	(\\$|€|£|¢|¥|U\\$S|US\\$|U\\$D|USD|AR\\$|ARS| AUD|CAD|BRL|CLP|EUR|JPY|MXN|GBP|CHF)

Tabla 6.3: Reglas para detectar y expandir Monedas

Expr.Regular	Case sensitive?	Expansión
\\$|AR\\$|ARS	no	pesos
U\\$S|US\\$|U\\$D|USD	no	dólares
€|EUR	no	euros
£|GBP	no	libras
CHF	no	francos suizos
¥|JPY	no	yenes
CAD	no	dólares canadienses
AUD	no	dólares australianos
BRL	no	reales
MXN	no	pesos mejicanos
CLP	no	pesos chilenos
¢	no	centavos

Tabla 6.4: Expansión de Símbolos de Monedas

6.2.3. Fechas y Horas

El tercer paso en el procesamiento de expresiones numéricas es evaluar si se trata de fechas u horas. La tablas 6.3 y 6.6 muestran las reglas definidas para detectar fechas y horas respectivamente. Para fechas se contempla el formato DD-MM-AAAA, DD/MM/AAAA y DD.MM.AAAA, no se trabajó en definir reglas para otros formatos posibles de fechas dado que la frecuencia de aparición es baja. Lo mismo sucede con expresiones de hora, sólo se contempla el formato hh:mm(:ss). Para expandir las expresiones de horas no se pudo hacer funcionar el FST de horas, creemos que el problema surge del set de caracteres usados para la construcción y el utilizado por este trabajo, por lo tanto se resolvió utilizando el FST naturales.

Con las reglas de Fechas y Horas se cubren 41.036 casos, el 0.48 %.

Orden	Regla	Exp. Regular	#Casos	%Casos	Ejemplos	FST
8	Fechas (DD/MM/AAAA)	^((0?[1-9])|([1-2][0-9])|(3[0-1]))/((0?[1-9])|(1[0-2]))/(([0-9]{2})|([0-9]{4}))$	2.845	0,03%	23/01/1987 – 2/5/02	Fechas
9	Fechas (DD-MM-AAAA)	^((0?[1-9])|([1-2][0-9])|(3[0-1]))-((0?[1-9])|(1[0-2]))-(([0-9]{2})|([0-9]{4}))$	1.416	0,02%	23-01-1987 – 2-5-02	Fechas
10	Fechas (DD.MM.AAAA)	^((0?[1-9])|([1-2][0-9])|(3[0-1]))[.period.]((0?[1-9])|(1[0-2]))[.period.](([0-9]{2})|([0-9]{4}))$	1.395	0,02%	23.01.1987 – 2.5.02	Fechas
		Total	**5.656**	**0,07%**		

Tabla 6.5: Reglas para detectar y expandir Fechas

Orden	Regla	Exp. Regular	#Casos	%Casos	Ejemplos	FST
11	Horas (hh:mm(:ss))	^(([0-1]?[0-9])|2[0-3]):[0-5]?[0-9](:[0-5]?[0-9])?$	35.380	0,41%	23:45:10 / 10:00 / 23:30	Naturales

Tabla 6.6: Reglas para detectar y expandir Horas

6.2.4. Números Ordinales, Temperaturas, Coordenadas y Angulos

En esta sección se describen las reglas definidas para expresiones numéricas que contienen el símbolo º. La tabla 6.7 enumera las reglas que se aplican en la detección y expansión de este tipo de expresiones. Como se puede observar, es variado el tipo de expresiones para el cual se utiliza el símbolo º. Se lo puede encontrar en números ordinales: 25º, en temperaturas: 23ºC , en coordenas/ángulos: 24º14'34"N, 360º o en la abreviatura de "número": Nº, nº. Se intentó cubrir los casos más frecuentes encontrados en la base de datos de expresiones numéricas, evaluando el contexto para resolver las ambigüedades. Las dos últimas reglas son por defecto, se definió la expansión evaluando una muestra de casos y analizando cuál era la expansión más frecuente.

Con este conjunto de reglas se cubren en total 75.753 casos, el 0.88 %.

Orden	Regla	Exp. Regular	#Casos	%Casos	Ejemplos	FST	Comentarios
12	Ordinales	^(([0-9]+(\\.)?º)|([0-9]+[.period.]°))$	2.534	0,03%	1.° / 123.º	Ordinales femenino y neutro	Si la palabra anterior es "la" se considera ordinal femenino. Para el resto de los casos se considera ordinal neutro.
13	Temperaturas	^[[.plus-sign.][.hyphen.]]?[0-9]+([[.period.],][0-9]+)?(°|º)(C|F)$	4.250	0,05%	34°C / -45,5°F	Reales	Se extrae el número y se expande. Se agrega "grados celcius" o "grados farenjeit" al resultado.
14	Temp/ángulos/Coord (con punto decimal)	^[[.plus-sign.][.hyphen.]]?[0-9]+([[.period.],][0-9]+)(°|º)$	2.054	0,02%	34,4° / 20.3º	Reales	Se extrae el número y se expande. Se agrega "grados" al resultado.
15	N°##	^N(°|º)[0-9]+$	2.534	0,03%	N°23 / Nº1	Naturales	Se extrae el número y se lo expande. Se agrega "número" al inicio.
16	Coordenadas	^[[.plus-sign.][.hyphen.]]?[0-9]+([[.period.]|,][0-9]+)?(°|º)[[.plus-sign.][.hyphen.]]?[0-9]+([[.period.]|,][0-9]+)?['|'']([[.plus-sign.][.hyphen.]]?[0-9]+([[.period.]|,][0-9]+)?(["|"|"|"|[.quotation-mark.]]))?(S|N|E|O|W)?$	4.428	0,05%	39°20'S / 48°40'34,5"S	Reales	Se extraen los componentes y se expanden por separado, luego se concatenan los resultados agregando al final "norte" "sur" "este" "oeste" si corresponde.
17	Coordenadas (sólo grados con punto cardinal)	^[[.plus-sign.][.hyphen.]]?[0-9]+([[.period.]|,][0-9]+)?(°|º)(S|N|E|O|W)$	1.045	0,01%	39°S / 48°N	Reales	Se extrae el número y se expande, luego se agrega al final "norte" "sur" "este" "oeste" según corresponda.
18	Ordinal (Palabra Ant. Del/el/la)	PalabraActual ---> ^[0-9]+(°|º)$ PalabraAnterior--->^(el|del|la)$	17.361	0,20%	"La 2° ..." / "el 3° ..."	Ordinales femenino y masculino	Si la palabra anterior es "el" o "del" se expande como neutro, si es "la" como femenino.
19	Ordinal (Palabra Posterior)	PalabraActual ---> ^[0-9]+(°|º)$ PalabraPosterior--->^(regimiento|ejercito|batallon|lugar|aniversario|escuadron|puesto|distrito|grado|dan|comando|festival|grupo|departamento|piso|dpto|dto|escuadra|division|categoria)$	3.349	0,04%	"12° regimiento" "3° escuadra"	Ordinales femenino y neutro	Se expande como ordinal femenino o neutro de acuerdo a la palabra posterior.
20	Temp.Coord. (palabra posterior)	PalabraActual ---> ^[0-9]+(°|º)$ PalabraPosterior--->^(norte|sur|este|oeste|west|n[n[.period.]]s|s[.period.]]e|e[.period.]]o|o[.period.]]w|w[.period.]]c|f|c[.period.]]f[.period.])$	2.691	0,03%	"80° norte" "23° f" "45° c"	Naturales	Se agrega "grados" al final.
21	Temp.Coord. (nro posterior)	PalabraActual ---> ^[0-9]+(°|º)$ PalabraPosterior--->[0-9]+	7.329	0,09%	"80° 14' "	Naturales	Se agrega "grados" al final.
22	Temp.Coord. (contexto)	PalabraActual ---> ^[0-9]+(°|º)$ Alguna palabra contexto ---> (palarelos|paralelo|latitud|ecuador|gira|girar|longitud|meridiano|lat|coordenadas|sur|norte|este|oeste|angulo|inclinacion|temperatura|temp|temp[.period.])	2.483	0,03%	"30° latitud sur" / "24° de temperatura"	Naturales	Se agrega "grados" al final.
23	Ordinales neutro (palabras ant. / post.)	PalabraActual ---> ^[0-9]+(°|º)$ Alguna ant. o post. ---> ^(articulo|teniente|regimientos|regimiento|cumpleaños|tomo|humberto|oficial|capitulo|Art[.period.]|cuartel|capitan|subteniente|soldado|batallon)$	790	0,01%	"teniente 2°" "25° cumpleaños"	Ordinales neutro	Ninguno
24	Angulos comunes	^(180|90|360|45|30|120|270)(°|º)$	1.955	0,02%	360° / 180° / 45°	Naturales	Se agrega "grados" al final.
25	Default ordinales	PalabraActual ---> ^[0-9]+(°|º)$ y <= 20°	18.088	0,21%	1° / 12º / 19°	Ordinales neutro	Ninguno
26	Default grados	PalabraActual ---> ^[0-9]+(°|º)$ y >= 20°	4.862	0,06%	38° / 50° / 123°	Naturales	Se agrega "grados" al final.
		Total	**75.753**	**0,88%**			

Tabla 6.7: Reglas para detectar y expandir Temperaturas, Números Ordinales, Coordenadas y Angulos

6.2.5. Números Enteros, Reales, Potencias y Notación Científica

La escritura de los números naturales y reales presenta ambigüedades en la utilización de la coma y punto como separador decimal y de miles. Su utilización varía de acuerdo a las costumbres de cada país. En general, los textos en español utilizan la coma como separador decimal y el punto como separador de miles, sin embargo en algunos artículos de *Wikipedia* en español se usan en forma inversa, tal como sucede en la escritura en inglés. También es frecuente en artículos de *Wikipedia* el uso de un espacio (código HTML: 160; - *nonbreaking space*), como separador de miles. El módulo 1 del sistema, el que convierte la página HTML a texto plano, detecta estos casos y quita los espacios cuando se utilizan como separador de miles. En la tabla 6.8 se enumeran las reglas definidas para detectar y expandir números enteros y reales con los formatos más importantes que se encontraron en la base de datos de expresiones numéricas. También se crearon reglas para detectar y expandir potencias y números en notación científica. Para estos casos, el módulo 1 debe detectar estas expresiones y generar el formato especificado en las reglas 33 y 34 para que el módulo de normalización las expanda correctamente. No se cuentan con casos en la base

de datos de expresiones numéricas, porque la construcción de la base de datos fue anterior a que se detecte la necesidad de formatear estas expresiones en la conversión de HTML a texto.

Con estas reglas se cubren 7.031.853 casos, el 81.63 % de las expresiones numéricas de la base de datos.

Orden	Regla	Exp. Regular	#Casos	%Casos	Ejemplos	FST	Comentarios
27	Enteros	^[[.plus-sign.][.hyphen.]]?[0-9]+$	6.263.534	72,72%	15 / 45 / 1984 / -23 / +38	Naturales masculino	Se quita el signo (+ -) y se agrega "menos" al resultado si corresponde.
28	Enteros (Sep. Miles = .)	^[[.plus-sign.][.hyphen.]]?([1-9]){1}([0-9]){0,2}([.period.]([0-9]){3})+$	216.520	2,51%	5.000.000 / 1.567	Naturales masculino	Se quitan los puntos, el signo (+/-) y se agrega "menos" al resultado si corresponde.
29	Enteros (Sep. Miles = ,)	^[[.plus-sign.][.hyphen.]]?([1-9]){1}([0-9]){0,2}(,([0-9]){3})+$	30.154	0,35%	100,000 / 5,000	Naturales masculino	Se quitan las comas y el signo y se agrega "menos" al resultado si corresponde.
30	Reales (Sep. Miles =. Decimal = ,)	^[[.plus-sign.][.hyphen.]]?([1-9]){1}([0-9]){0,2}([.period.]([0-9]){3})+,[0-9]+$	3.228	0,04%	1.026,99 / -4.903,34	Reales	Se quitan los puntos que se usan como separador de miles.
31	Reales (Sep. Miles =, Decimal = .)	^[[.plus-sign.][.hyphen.]]?([1-9]){1}([0-9]){0,2}(,([0-9]){3})+[.period.][0-9]+$	1.490	0,02%	1,234.98 / -7,889.93	Reales	Se quitan las comas que se usan como separador de miles.
32	Reales (sin Sep. Miles Decimal = . / ,)	^[[.plus-sign.][.hyphen.]]?[0-9]+([.period.]|,)[0-9]+$	516.927	6,00%	1.98 / -0.12 / +4,5 / -0,56	Reales	Ninguna.
33	Notación Científica	^[[.plus-sign.][.hyphen.]]?[0-9]+([[.period.]|,][0-9]+)?x10^{[[.plus-sign.][.hyphen.]]?[0-9]+}$	S/D	S/D	3,4x10⁻²	Reales	Se separa el coeficiente y el exponente para expandirlos por separado y se concatenan los resultados.
34	Potencias	^[[.plus-sign.][.hyphen.]]?[0-9]+([[.period.]|,][0-9]+)?^{[[.plus-sign.][.hyphen.]]?[0-9]+}$	S/D	S/D	4²	Reales, Ordinales femenino.	Se separa la base y el exponente para expandirlos por separado. Exponente entre 1 y 10 se expande como ordinal femenino, el resto como real. Exponente 2 y 3. se expanden como "al cuadrado" y "al cubo" respectivamente.
		Total	**7.031.853**	**81,63%**			

Tabla 6.8: Reglas para detectar y expandir Números

6.2.6. Otras expresiones con dígitos numéricos

En la tabla 6.9 se muestra un conjunto de reglas que se definieron para cubrir otras expresiones numéricas que se encontraron en la base de datos construida. En este conjunto, se destacan los números ordinales escritos sin el símbolo º, por ejemplo: 4to, 1er, 8va, también hay perídos de años: 1989/1990, 2001-2005, expresiones alfanuméricas en diferentes formatos, por ejemplo: UTF8, 3X, 11-S y abreviaturas que tienen algún dígito en su composición: cm3, m2. Por último, se definió una regla por defecto, número de orden 47, que procesa las expresiones que contienen algún dígito y que no son cubiertas por ninguna regla. Para estos casos se deletrea la expresión utilizando el FST de letras. Sólo el 1.4 % de las expresiones numéricas de la base de datos se resuelve con esta regla, es decir, el 98.6 % se resuelve con alguna de las 46 reglas anteriores.

Orden	Regla	Exp. Regular	#Casos	%Casos	Ejemplos	FST	Comentarios
35	Perído de años (AAAA-AAAA)	^[0-9]{4}[-][0-9]{4}$	102.295	1,19%	1987-1990 / 2001-2005	Naturales masculino	Se expande cada año por separado y se concatenan los resultados, se inserta " a " entre ambos.
36	Perído de años (AAAA/AAAA)	^[0-9]{4}[/][0-9]{4}$	5.547	0,06%	1903/1904 2002/2003	Naturales masculino	Se expande cada año por separado y se concatenan los resultados.
37	Perído de años (AAAA(/ o-) AA)	^[0-9]{4}(/|\\-)[0-9]{2}$	36.776	0,43%	1985/86 1998/99	Naturales masculino	Se expande cada año por separado y se concatenan los resultados.
38	Ordinales	^[0-9]+(er|ero|era|do|da|to|ta|vo|va|no|na|mo|ma[.period.]?ª)[.period.]?$	31.414	0,36%	23.ª / 4to / 7mo / 1er	Ordinales femenino, masculino y neutro	Se extrae el número y se determina en base a la terminación qué FST usar.
39	Marcador (#-#)	^[0-9][-][0-9]$	66.542	0,77%	1-2 / 3-5 / 9-7	Naturales masculino	Se expande cada número y se concatenan los resultados dejando un espacio entre medio.
40	Nros. separados por / o -	^[0-9]+((\\.|,)[0-9]+)*(/|\\-)[0-9]+((\\.|,)[0-9]+)*$	61.828	0,72%	95-88 2,5-3,5 1/9 50/1997	Naturales masculino y reales	Se expande cada número por separado aplicando las reglas anteriores. Se concatenan los resultados agregando la palabra "guión" o "barra".
41	#	^\\#[0-9]+$	11.145	0,13%	#1 / #12 / #102	Naturales masculino	Se expande el número y se agrega "número" al inicio.
42	Abreviaturas	^(KM2|KM²|M2|M²|MTS2|MTS²|M3|M³|MTS3|MTS³|CM2|CM²|CMS2|CMS²|MM2|MM²|MM3|MM³|m3/s|m³/s|HAB/KM2|HAB/KM²|HAB/MI2|HAB/M²)$	8.628	0,10%	km2 /cm3 / mts2	Ninguno	Se expande de acuerdo al listado de abreviaturas.
43	Códigos (LetrasDígitos)	^[a-z]+[0-9]+$	113.961	1,32%	AS400 / UTF8/ A4	Naturales masculino y letras	Se separa las parte alfabética de la numérica. La parte alfabética se deletrea si corresponde (ver siglas y abrev.) y el nro se expande como entero o dígito a dígito si comienza con cero.
44	Códigos (Letras(-|_)Dígitos)	^[a-z]+(-|_|\\.)[0-9]+$	139.441	1,62%	Sub-17 / B-18 / julio-30 / formula_1	Naturales masculino y letras	Se separa las parte alfabética de la numérica. La parte alfabética se deletrea si corresponde (ver siglas y abrev.) y el nro se expande como entero o dígito a dígito si comienza con cero.
45	Códigos (DígitosLetras)	^[0-9]+[a-z]+$	34.651	0,40%	2D / 16px / 3x	Naturales masculino y letras	Se separa las parte alfabética de la numérica. La parte alfabética se deletrea si corresponde (ver siglas y abrev.) y el nro se expande como entero o dígito a dígito si comienza con cero.
46	Códigos (Dígitos(-|_)Letras)	^[0-9]+(-|_|\\.)[a-z]+$	14.653	0,17%	2009-presente / 7-bit / 11-S	Naturales masculino y letras	Se separa las parte alfabética de la numérica. La parte alfabética se deletrea si corresponde (ver siglas y abrev.) y el nro se expande como entero o dígito a dígito si comienza con cero.
47	Default		122.976	1,43%	H2O / 1.R6-F3	Letras	Se deletrean las letras y los dígitos.
		Total	**749.857**	**8,71%**			

Tabla 6.9: Reglas para detectar y expandir Otras Expresiones Numéricas

6.3. Siglas y Abreviaturas

6.3.1. Abreviaturas

Definición: Una abreviatura es un tipo de abreviación, una convención ortográfica que acorta la escritura de cierto término o expresión, y consiste en la representación escrita de una palabra o grupo de palabras con solo una o varias de sus letras. Para crearla, se emplea la letra inicial, mayúscula o minúscula, por sí sola o acompañada de otras letras, ya sean del medio o del fin de dicha palabra, y uno o varios puntos para indicar que la palabra está incompleta[4]. Algunos ejemplos de abreviaturas comunes son: "etc.", "aprox.", "art.".

La expansión de abreviaturas consiste en reemplazar su aparición en el texto por la palabra completa, o por el texto que represente la pronunciación de la abreviatura. Por ejemplo, la expansión de la abreviatura "etc." es "etcétera" y la de "aprox." es "aproximadamente". Existen abreviaturas ambiguas, es decir, la misma abreviatura puede ser usada para abreviar palabras distintas. Por ejemplo, "dto." puede ser la abreviatura de

[4]http://lema.rae.es/drae/?val=abreviatura

"departamento" o de "descuento", "pto." puede ser la abreviatura de "puerto" o "punto" o "presupuesto". El lector interpreta el significado correcto de acuerdo al contexto.

Para este trabajo, se detectan las abreviaturas buscando cada palabra en un listado de abreviaturas (*abreviaturas.txt*), en caso de encontrarse en el listado se reemplaza con la expansión que figura en el listado, para abreviaturas ambiguas se eligió la que se estimó que era la más frecuente. El listado fue construido a partir de las abreviaturas que incluye el archivo *tokenizer.dat* de *Freeling* (ver capítulo 3) más algunas abreviaturas que se consideraron frecuentes, a todas se le asoció la expansión correspondiente. En algunos casos se busco la definición de las abreviatura en la *web*. La tabla 6.10 muestra algunas abreviaturas incluidas en el archivo que en total contiene 209 abreviaturas. Para enriquecer el listado, además de agregar las abreviaturas en el archivo *abreviaturas.txt*, es necesario incluirlas también en el archivo *tokenizer.dat* para que el tokenizador delimite correctamente la palabra considerando el punto como parte de la misma y no como delimitador de oración (ver capítulo 3).

Abreviatura	Expansión	Abreviatura	Expansión	Abreviatura	Expansión	Abreviatura	Expansión	Abreviatura	Expansión
aa.rr.	altezas reales	ag.	a je	atte.	atentamente	cia.	compañía	ntra.	nuestra
abr.	abreviatura	ago.	agosto	av.	avenida	cía.	compañía	ntro.	nuestro
abrev.	abreviatura	am.	a eme	avda.	avenida	ltd.	limitada	núm.	número
a.c.	antes de cristo	ap.	a pe	aprox.	aproximadamente	ltda.	limitada	ob.	obispo
adj.	adjetivo	apdo.	apartado	bros.	bros	mar.	marzo	obpo.	obispo
adm.	administrador	art.	artículo	bv.	bulevar	máx.	máximo	oct.	octubre
admón.	administración	arts.	artículos	cap.	capítulo	may.	mayo	op.	op
afma.	afectísima	arz.	arzobispo	caps.	capítulos	mín.	mínimo	pág.	página
afmas.	afectísimas	arzbpo.	arzobispo	cg.	ce je	mons.	monseñor	pag.	página
afmo.	afectísimo	assn.	asociación	cgo.	cargo	mr.	mister	ppal.	principal

Tabla 6.10: Muestra del listado de Abrevituras

6.3.2. Siglas

Definición: Una sigla es el resultado de un proceso de creación de una palabra a partir de cada grafema (letra) inicial de los términos principales de una expresión compleja[5]. Por ejemplo, OEA: Organización de Estados Americanos, YPF: Yacimientos Petrolíferos Fiscales, PYME: Pequeña y Mediana Empresa.

Las siglas tienen dos modos de pronunciación: deletreo y silábica. El deletreo es simplemente pronunciar el nombre de cada letra (o grafema): ONG (o-ene-ge). La pronunciación silábica es la que lee la sigla como una palabra: JASP (jasp). Esta última surge automáticamente siempre que el hablante sea capaz de pronunciarla según la fonética de su lengua.

El módulo de normalización debe tratar de identificar las siglas y decidir si se pronuncian o deletrean. En el caso que se deletreen se debe reemplazar la sigla por su pronunciación, por ejemplo ONG: o ene ge, antes de enviar el texto al sintetizador. En la literatura existen algunos trabajos que abordaron este problema usando técnicas de aprendizaje automático [17].

Para discriminar las siglas que se pronuncian de las que se deletrean, se analizó un conjunto de siglas y se propusieron las siguientes reglas:

1. Una sigla es pronunciable si para cada consonante existe una vocal antes o después de la misma, caso contrario se deletrea.

[5] http://lema.rae.es/drae/?val=sigla

Ejemplos: DGI: se deletrea, la 'D' no tiene una vocal antes ni después. AFIP: se pronuncia. Tiene vocales antes o después de cada consonante.

2. Si la cantidad de letras es 2, se deletra. Ejemplos: UP, JP, AA, RA, BA.

3. Si la sigla está formada por sólo vocales, se deletrea para marcar una separación entre letras aunque es prácticamente lo mismo deletrear o pronunciar en estos casos. Ejemplos: OEA, UIA.

4. Si tiene muchas letras (más de 4) no se suele deletrear porque quedaría muy larga. Ejemplos: FREPASO, NAPALM.

Se testeó este conjunto de reglas con un listado de 201 siglas recolectados a mano y el resultado fue que el 98 % de los casos fue clasificado correctamente.

Se decidió extender el análisis de estas reglas a un conjunto más grande de casos, para eso de *Wikipedia* en español se extrajeron las palabras cuyas letras figuraban en mayúsculas, se identificaron manualmente las siglas y se clasificaron las primeras 819 ordenadas por mayor frecuencia de aparición. Estas siglas se etiquetaron a mano de acuerdo a si se pronuncian o deletrean. Luego se aplicaron las reglas de clasificación, el resultado fue que el 92.4 % de las siglas se clasificó correctamente, 95.1 % si se pondera por la frecuencia de aparición. Algunos casos de error fueron:

- URSS, FARC, PRI, AS, ASCII: las reglas determinan que se deletreen y son siglas que se pronuncian.
- VIH, SOS, GUI, OIT: las reglas determinan que se pronuncien y son siglas que se deletrean.

En las tablas 6.11 y 6.12 se muestra la clasificación lograda por las reglas versus las etiquetas generadas a mano, en cantidad de palabras y en cantidad de apariciones. Dado que la tasa de error estimada en estos análisis es relativamente baja, se decidió adoptar estas reglas para la normalización de siglas.

		Reglas		
		Pronuncian	**Deletrean**	**Total**
Etiquetas	**Pronuncian**	196	36	**232**
	Deletrean	26	561	**587**
	Total	**222**	**597**	**819**

Tabla 6.11: Resultado de la clasificación de siglas en cantidad de palabras

		Reglas		
		Pronuncian	**Deletrean**	**Total**
Etiquetas	**Pronuncian**	132.583	13.495	**146.078**
	Deletrean	19.046	504.005	**523.051**
	Total	**151.629**	**517.500**	**669.129**

Tabla 6.12: Resultado de la clasificación de siglas en frecuencia de apariciones

En la implementación del módulo, se procesa cada palabra buscándola en un listado de siglas (*siglas.txt*)que contiene la expansión correcta. Este listado contiene las 819 siglas más

frecuentes de *Wikipedia* en español más algunas otras que en total suman 1003 siglas. En la tabla 6.13 se muestran algunas siglas que contiene el archivo, también se incluyen otras expresiones que no son estrictamente siglas, como magnitudes y símbolos que necesitan ser expandidos utilizando el mismo procedimiento que para las siglas. Si la palabra a analizar no se encuentra en el listado y es una palabra con todas sus letras en mayúscula, se aplican las reglas de siglas. Si las reglas indican que se debe deletrear entonces se transforma utilizando el FST Letras, en caso contrario se deja la sigla tal como aparece en el texto.

Sigla	Expansión	Sigla	Expansión	Sigla	Expansión	Sigla	Expansión	Sigla	Expansión
FC	efe ce	SD	ese de	BBVA	be be u ve a	Nº	número	KG	kilogramos
CD	ce de	AFC	a efe ce	ARIA	aria	N.°	número	KGS	kilogramos
TV	te ve	UTC	u te ce	UCV	u ce ve	Nº	número	GRS	gramos
DVD	de ve de	BMG	be eme ge	MS	eme ese	N.º	número	MSNM	metros sobre el nivel del mar
NBA	ene be a	VHS	ve ache ese	CSI	ce ese i	NRO	número	M²	metros cuadrados
EE	e e	AT	a te	AOC	aoc	%	porciento	HAB/MI2	habitantes por milla cuadrada
ADN	a de ene	MB	eme be	WRC	dobleve erre ce	&	y	HAB/M²	habitantes por milla cuadrada
EP	ep	IRA	ira	AOL	aol	A	a	°	grados
MTV	em ti vi	RCD	erre ce de	UEFA	uefa	B	be	°C	grados centígrados
UU	u u	CA	ce a	FIA	fia	C	ce	°F	grados farenjeit

Tabla 6.13: Muestra del listado de Siglas

6.3.3. Procesamiento de Abreviaturas y Siglas

La figura 6.1 muestra el esquema de procesamiento de abreviaturas y siglas implementado en el módulo de normalización de acuerdo a los descripto en las secciones anteriores.

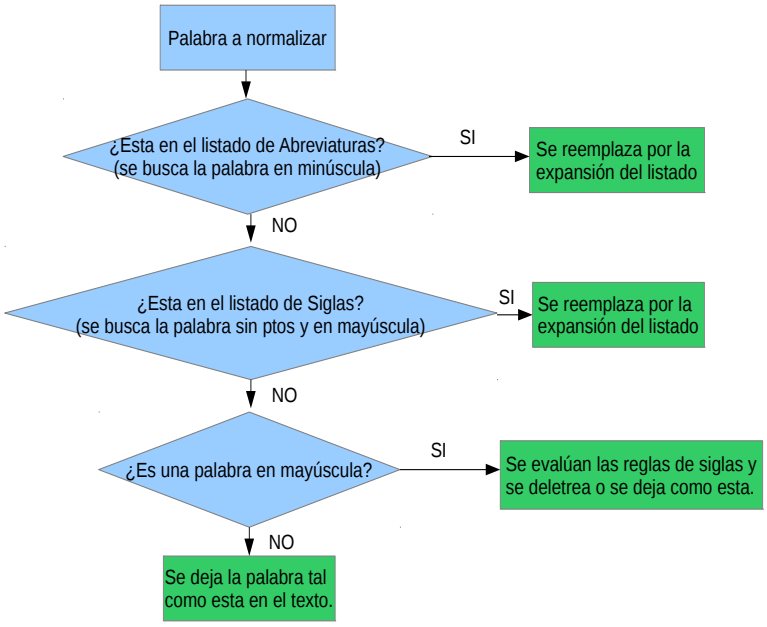

Figura 6.1: Esquema de procesamiento de Siglas y Abreviaturas

6.4. Números Romanos

Para expandir los números romanos primero hay que poder detectarlos correctamente. En la figura 6.14 se puede observar los símbolos que forman los números romanos.

Romano	Decimal	Nota
I	1	*Unus*
V	5	*Quinque*. V es la mitad superior de X; en etrusco Λ.
X	10	*Decem*
L	50	*Quinquaginta*
C	100	Letra inicial de *Centum*.
D	500	*Quingenti*. D, es la mitad de la Phi Φ.
M	1000	*Mille*. Originalmente era la letra Phi.

Tabla 6.14: Símbolos del sistema de números romanos. Fuente: Wikipedia en español.

No cualquier combinación de símbolos del sistema de los números romanos forma un número romano válido. Las siguientes reglas permiten determinar si un cadena de caracteres representa un número romano válido[6]:

1. Formado por caracteres I, V, X, L, C, D, M
2. Ninguna letra principal se escribe seguidamente más de tres veces (I, X, C, M) . Ej. I = 1, II = 2, III = 3, X = 10, XX = 20, XXX = 30, C = 100, CC = 200, CCC = 300, M = 1000, MM = 2000, MMM = 3000. Las letras secundarias (V, L, D) no se escriben seguidamente. Ej. VV, LL no son válidos.
3. La I, X, y la C (unidad decena y centena) precediendo a las dos letras que le siguen en valor, restan su valor. Es decir la I resta su valor cuando precede a la V y a la X, pero no a la L, a la C, a la D o a la M. La X resta su valor cuando precede a la L o a la C. La C resta su valor cuando precede a la D o la M Ej. DM, no es válido
4. No tiene sentido cadenas donde se suma y se resta un mismo valor. Ejemplo: CCD (+100 -100 +500) o CDC (-100 +500 +100). No se consideran válidos.

Siguiendo estas reglas se puede detectar qué palabras del texto representan números romanos válidos pero existen numerosas ambigüedades, por ejemplo: el caracter 'I' puede representar el número romano 1 como en "siglo I" o "página I" pero también puede aparecer en frases en inglés como "I believe in love...".

Además, una vez detectado un número romano se debe decidir si se lo considera como ordinal o cardinal. Por ejemplo: "Juan Pablo II" debe expandirse a "Juan Pablo segundo" mientras que "página ii" debe expandirse a "página dos".

Para analizar los diferentes casos que existen en *Wikipedia* en español, se detectaron todas las palabras que cumplían con el patrón de ser números romanos y a cada ocurrencia se le agregaron las 5 palabras anteriores y posteriores para obtener el contexto. Estos datos fueron almacenados en una base de datos para analizarlos y detectar los patrones más

[6] http://es.wikipedia.org/wiki/Numeración_romana

Anterior5	Anterior4	Anterior3	Anterior2	Anterior1	Palabra	Posterior1	Posterior2	Posterior3	Posterior4	Posterior5
por	Su	Santidad	Juan	Pablo	**II**	.	Está	incardinado	en	la
por	el	papa	Juan	Pablo	**II**	Obispo	auxiliar	de	Oviedo	y
nombrado	por	el	papa	Benedicto	**XVI**	obispo	de	Ciudad	Rodrigo	
los	tiempos	del	Papa	Pío	**XII**	.	Obras	.	Hablemos	
tienen	una	longitud	de	5	**cm**	con	largos	pelos	rojos	
la	agalla	es	de	5	**mm**	de	longitud	;	la	cabeza
vez	a	finales	del	siglo	**XVII**	como	la	propiedad	privada	

Tabla 6.15: Posibles Números Romanos con 5 palabras anteriores y posteriores

importantes. De esta forma se extrajeron 955.241 casos de posibles números romanos, en la tabla 6.15 se pueden observar algunos ejemplos.

A continuación se describe el análisis realizado sobre esta base de datos de casos con el fin de cubrir con reglas la mayor cantidad de ellos. A partir de este análisis se implementó el esquema de detección y expansión de números romanos esquematizado en la figura 6.2.

6.4.1. Números Romanos válidos

Cuando se detecta que una palabra cumple el patrón de ser número romano (palabras formadas por las letras: I, V, X, L, C, D, M), a continuación se evalúa si es un número romano válido de acuerdo a las reglas mencionadas en la sección anterior. Además, se excluyen algunos casos que no suelen aparecer como números romanos. Por ejemplo:

- CM: es el número romano 900, pero también es la abreviatura de centímetros. No se encontraron casos en los que se emplee como número romano, por lo tanto se tomó la decisión de no considerarlo como número romano.
- MM: es el número 2000, pero también es la abreviatura de milímetros. Idem caso anterior.

La decisión de no considerar estos casos como números romanos válidos es arbitraria y de compromiso, queda como trabajo futuro desambiguarlas en base al contexto.

De los 955.241 casos de análisis de posibles números romanos, con estos filtros se excluyen 229.317, quedando el universo a analizar en 725.924 casos.

6.4.2. Números Romanos para referenciar siglos

Los números romanos frecuentemente se utilizan para referenciar siglos, por ejemplo: “siglo XX”. Es un caso muy importante en el universo analizado. Se encontraron 174.588 casos, el 24 %, donde la palabra precedente es “siglo” o “siglos”. El módulo de normalización evalúa cada palabra y si encuentra una que cumple con el patrón del los números romanos válidos y la palabra que la precede es “siglo” o “siglos” entonces asume que es un número romano y lo expande utilizando el FST de números romanos. Se tomó la decisión de expandir los casos del I al X (1 al 10) como ordinales y a partir de XI (11) como cardinales, por ejemplo “Siglo III” se expande a “Siglo tercero”, mientras “siglo XVII” se expande a “siglo diecisiete”.

6.4.3. Números Romanos en nombres propios

Otro de los usos frecuentes de los números romanos es en nombres propios. Por ejemplo: "Fernando VII, Carlos V". Para detectar estos casos se armó un listado de los nombres propios asociados a números romanos más frecuentes en el universo analizado. Este caso está limitado a números romanos válidos formados por I, V y X; es decir números romanos bajos. Con la ayuda de este listado de nombres se cubren 146.113 casos de nombres masculinos y 5.287 casos de nombres femeninos, en total representa el 20.86 % del universo analizado. Si el número romano es del I al X (1 al 10) se expande como ordinal, y en ese caso lleva el género del nombre propio que le precede. Si es mayor a X (10) se expande como cardinal. Por ejemplo: "Luis XV" se expande a "Luis quince", "Fernando VII" se expande a "Fernando séptimo" y "Juana I" se expande a "Juana primera".

6.4.4. Números Romanos en nombres propios detectados por títulos de nobleza

Una forma complementaria de detectar el uso de números romanos asociados a nombres propios es por la proximidad, en el texto, de palabras que denoten títulos de nobleza. Por ejemplo, en fragmentos como: "... el rey Teodoro I ..." aunque el nombre propio "Teodoro" no figure en el listado, la cercanía de la palabra "rey" indica que probablemente "I" se utiliza en el texto como número romano y en consecuencia hay que expandirlo como tal. A partir de esta idea se identificaron manualmente las palabras que denoten títulos de nobleza, se seleccionaron las siguientes:

- Títulos de nobleza masculinos: rey, conde, duque, marqués, emperador, zar, sultán, emir, barón, príncipe.
- Títulos de nobleza femeninos: reina, condesa, duquesa, marquesa, emperatriz, zarina, baronesa, princesa.

Si se encuentra una palabra formada por los símbolos I, V o X, es un número romano válido y alguna de las 5 palabras anteriores o posteriores es alguna de las enumeradas, entonces se considera que la palabra en cuestión es un número romano. Se la expande siguiendo las mismas pautas mencionadas en la sección anterior. De esta forma se cubren 11.415 casos masculinos y 1.547 femeninos, adicionales a los ya cubiertos y que representan el 1.78 % de los casos totales.

6.4.5. Números Romanos en periodos, rangos y enumeraciones

Existen casos como: "I y II", "I al VII", "I, II, V, VIII" que pueden tratarse de perídos de tiempo, rangos o enumeraciones. Para cubrir estos casos se evalúa si la palabra actual es un número romano válido y la posterior es alguna de las siguientes: "y", "a", "al", "hasta", "-", "," y la palabra siguiente debe ser también un número romano válido. De la misma forma se evalúa en sentido contrario comparando las 2 palabras anteriores. Con este patrón se cubren 31.056 casos adicionales, el 4.28 % del total.

6.4.6. Números Romanos asociados a palabras clave

Hay palabras que frecuentemente aparecen antes de números romanos, se detectaron las siguientes: Acto, capítulo, División, Clase, tomo, Vol., Vol, Volumen, fase, grado, Región,

Episodio, Categoría, Apéndice, zona, padrino, pag., página, edición. El número romano que figure a continuación de algunas de estas palabras se lo expande como cardinal. Estos casos suman 4.779 casos más y representan el 0.66 % del total.

6.4.7. Numeros Romanos - Casos particulares

Los casos descritos hasta ahora cubren el 52 % de los casos a analizar, quedan 351.139 casos no alcanzados por estos patrones. Algunos de ellos son apariciones en el texto de números romanos y otros no.

A continuación se comenta el análisis que se realizó con los casos más importantes.

El número romano M

La letra "M" (mil en números romanos) es muy infrecuente que aparezca en el texto como número romano, al menos en el universo analizado. En todos los casos observados se utilizaba en el texto como letra y no como número romano. Si la palabra "M" no aparece en un contexto como los descritos en las casos anteriores no se la considera como número romano. Estos casos son 61.414, el 8.5 % de los casos por analizar.

El número romano I

A diferencia del caso anterior, el número romano I (1) es muy frecuente y además es una letra que se suele utilizar en expresiones donde no representa un número romano. Se trata de 54.004 casos a analizar, el 7.4 % de los casos por analizar.

A continuación se enumeran algunos patrones detectados:

- **Palabras en Inglés**: La letra I es muy utilizada en el idioma inglés (traducción: "yo"). Es conveniente identificar cuándo 'I' se utiliza en el contexto de una frase en inglés, para cortar el proceso de evaluación de la misma como número romano. Se identificaron las palabras en inglés más frecuentes halladas en cercanía de la palabra "I", son las siguientes: if, all, when, what, how, as, that, here, can, am, but, before, do, everything, say, wish, where, things, now, will, for, who, because, have, should, something, love, want, don't, wanna, cant, know, feel, did, got, was, need, believe, like, just, could, get, gotta, think, hate, saw, dream, wont, see, die, never, kissed, still, call, miss, go, found, ain't, quit, would, heard, guess, thought, belong, knew, had, due, turn, hadn't , you, and.

  El funcionamiento es el siguiente: cuando se encuentra la palabra "I", se evalúan las 5 palabras anteriores y posteriores, si alguna coincide con cualquiera de las palabras enumeradas, entonces se considera que no se trata de un número romano. La cantidad de casos cubiertos con esta condición es 13.305 el 1.83 %.

  Otra forma complementaria de detectar este tipo de uso es evaluando si la palabra anterior es una comilla doble, se encontraron muchos fragmentos de texto como por ejemplo: "I will go with you". Con esta condición se cubren 2.736 casos más, el 0.38 %.

- **"I" + sustantivo masculino**: se detectó un uso frecuente en el texto analizado del número romano 'I' precediendo determinados sustantivos donde 'I' se usa como una abreviatura de "primer". Los sustantivos con más alta frecuencia encontrados

son: distrito, congreso, campeonato, cuerpo, premio, festival, ejército, marqués, encuentro, milenio, duque, concurso, milenio, conde, señor, grupo, salón, centenario, certamen, torneo, concilio, frente, sínodo, consejo, curso, barón, batallón, foro, conteo, plan, gobierno, imperio, regimiento,mermorial, taller, gran, ciclo, trofeo, siglo, coloquio, seminario, comandante, sucesor, mundial, maratón.

Se trata de 1.509 casos, el 0.21 %.

- **"I" + sustantivo femenino**: es un caso similar al anterior donde 'I' se utiliza como una abreviatura de "primera". Los sustantivos con más alta frecuencia encontrados son: guerra, legislatura, república, edición, división, región, bienal, copa, asamblea, conferencia, edad, exposición, cumbre, dinastía, internacional, brigada, liga, bandera, feria, muestra, escuadra, fuerza, olimpíada, jornada, marquesa, convención, duquesa, semana, circunnavegación, condesa, escuela, universidad, batalla, cruzada, circunscripción, vuelta, fase, promoción, parte, media, carrera, alternativa.

  Se trata de 2.155 casos, el 0.30 %.

- **"I" en ecuaciones**: "I" o "i" suelen usarse como variables en ecuaciones, por ejemplo "i = i + 1". Para capturar esto casos, se evalúa la palabra inmediata anterior y posterior a "I", si es alguna de los siguientes símbolos: -", ", "x", "; -"se asume que "I" no aparece representando un número romano. Se detectaron 507 casos, el 0.07 %.

- **Regla por defecto**: si no se da ninguna de las situaciones anteriores, las acción por defecto es no considerar la palabra "I" como número romano y dejarla como está en el texto. A esta instancia llegan 33.792 casos, el 4.66 %.

El número romano II

El número romano II (2) también es muy frecuente en el texto analizado, se encontraron 30.864 casos, el 4.3 % de los casos por analizar.

A continuación se enumeran algunos patrones detectados:

- **"II" + sustantivo masculino**: Es igual al caso descrito para el número "I" y se expande como "segundo". Se detectaron 1.915 casos, el 0.26 %. También se utiliza la palabra inmediata anterior, si es alguna de las siguientes: 'el', 'del', 'vaticano' también se expande como "segundo", se encontraron 911 casos, el 0.13 %.

- **"II" + sustantivo femenino**: Es igual al caso descrito para el número "I" y se expande como "segunda". Se detectaron 3.226 casos, el 0.44 %. También se utiliza la palabra inmediata anterior, si es el artículo 'la' se expande como "segunda", se trata de 169 casos, el 0.02 %.

- **Regla por defecto**: en este caso la regla por defecto consiste en considerar la expresión "II" como número romano y se expande como ordinal masculino: "segundo". A esta instancia llegan 24.643 casos, el 3.39 %.

El número romano III

Para el número romano III (3) se siguió el mismo análisis que para el caso II, se trata de 14.145 casos, el 1.9 % de los casos por analizar.

- **"III" + sustantivo masculino**: igual al caso descrito para los casos "I" y "II" y se expande como "tercer". Se detectaron 1.075 casos, el 0.15 %.
- **"III" + sustantivo femenino**: Es igual al caso descrito para los casos "I" y "II", se expande como "tercera". Se detectaron 709 casos, el 0.10 %.
- **Regla por defecto**: la regla por defecto consiste en considerar la expresión "III" como número romano y se expande como ordinal masculino: "tercero". A esta instancia llegan 12.361 casos, el 1.70 %.

Otros casos importantes

- **X**: Son 26.555 casos, el 3.66 %. Estos casos que quedan son en la gran mayoría casos en que se usa como letra equis, ejemplo: "rayos X". Las situaciones en que representa un número romano se cubrieron casi totalmente con las reglas anteriores.
- **C**: Son 26.081 casos, el 3.59 %. Es similar al anterior, no se considera como número romano.
- **D**: Son 17.822 casos, el 2.46 %. Es similar al anterior, no se considera como número romano.
- **L**: Son 11.731 casos, el 1.62 %. Es similar al anterior, no se considera como número romano.

Caso por defecto

Si la palabra que cumple con el patrón de los números romanos válidos no se encuadra en ninguna de las situaciones descritas anteriormente, es necesario tomar alguna decisión: considerarla como número romano y expandirla como tal o descartar el caso de que se trate de un número romano.

Para ello se recorrió manualmente un conjunto de 100 casos al azar de cada uno: "I", "II", "V", "X"....etc. y se identificó la acción más frecuente: considerarlo número romano o no y en caso de considerarlo número romano qué tipo de expansión era la que prevalecía: ordinal masculino/femenino o cardinal.

Como resultado de este análisis de casos se determinó la siguiente acción: si la palabra es alguna de las siguientes: mi, V, I, CC, MC, Xi, Mix, MD, DX, dl, XL, MV, Dix, mX, CX, CCC, Dv, LV, DCC, MMC, CMI, MDC, Mii, MCD, Div, MCC, mmm, LX, CCI, MCM, MCI, DVI, DCI, cmd, mdl, Viv, MMX, LXX, CMM, CLI, CMV, XLI, LLL, LIX, civ, CDL, XC, DCL, CDDL, LiI, DDL, LVI, DDC, vix, CLV, CDV, MDI, DCM, CLX, MMI, LLI, mll, Cii, mdd, DLX, MCV, MVV, MDX, DDI, IxC, CCL, DXi, CLIX, CCV, cvi, Cdx, lxv, Mixx, CML, VVV, DDD, cdcd, CLL, DCD, CVV, CxI, cxx, CIII, CDD, CXL, DCV, DLI, MLI, dxx, miv, Lixx, CCCI, Mixc, Cixx, por defecto no se la considera número romano, caso contrario se la considera número romano y se expande como ordinal si es menor o igual a 10 o cardinal si es mayor. Aproximadamente en el 50 % de estos casos se expande como número romano y el resto no se considera número romano.

6.4.8. Resultados

El esquema completo de detección de los números romanos se puede observar en la figura 6.2.

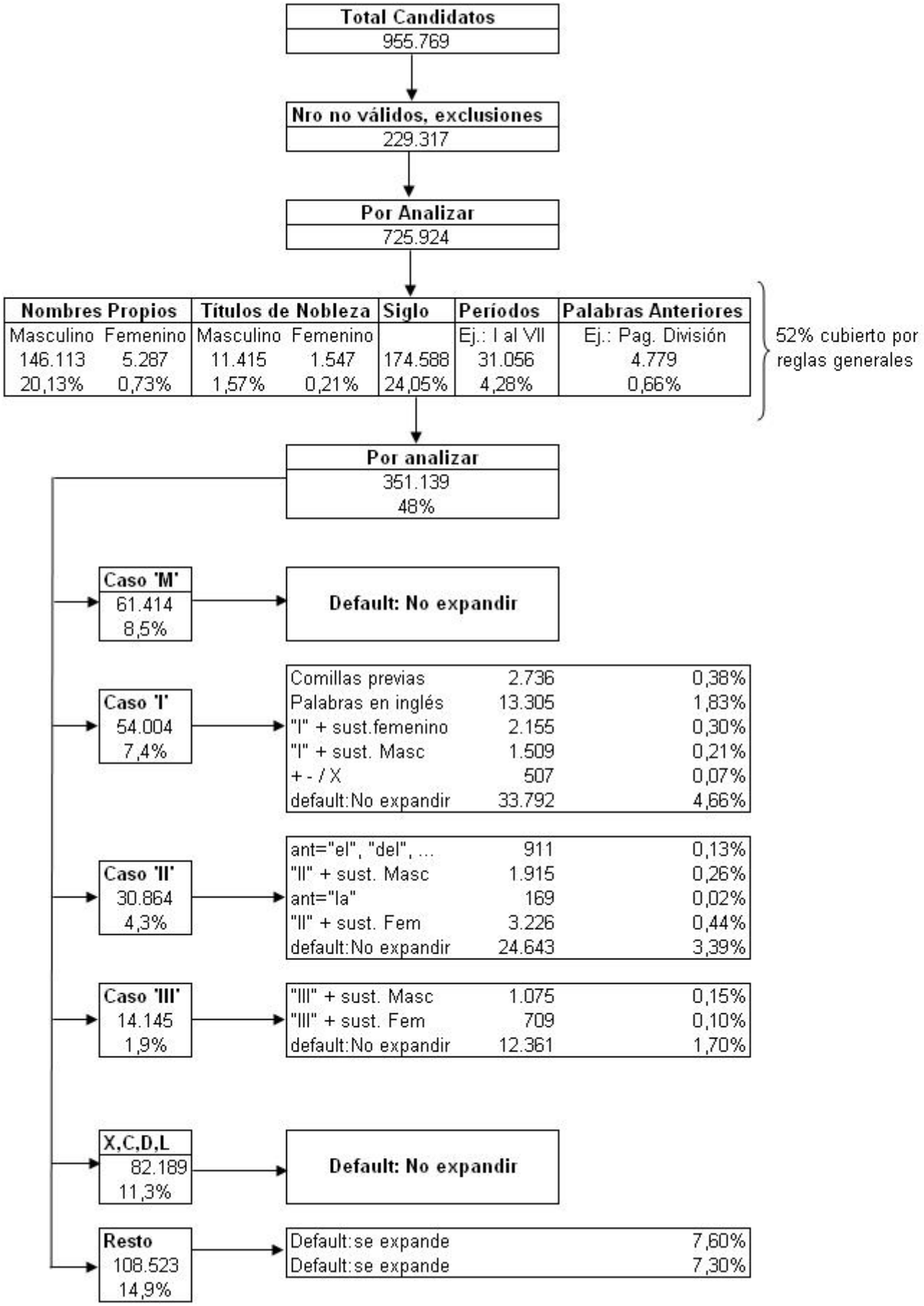

Figura 6.2: Esquema de detección de números romanos

Para estimar la tasa de error en la detección y expansión se generó una muestra de 400 casos al azar sobre los 955.241 casos extraídos de artículos de *Wikipedia*. Se analizó manualmente caso por caso determinando si la acción decidida por las reglas era correcta, es decir, si se consideraba la expresión como número romano o no. Para el primer caso también se analizó el resultado de la expansión.

La tabla 6.16 muestra los resultados de la detección. De los 400 casos evaluados, en 5 casos se detectó un error, las reglas omitieron considerar una expresión que aparecía en el texto como número romano. De esta manera, la estimación de la tasa de error en la detección según esta muestra es del 1.25 %. A continuación se listan los casos de error:

- "... consejo comunal Andrés Eloy Blanco **I** y 224 familias y 160 ..." : No se detectó **I** como número romano.
- "... 1422 por parte de Friedrich **V** de Núrember. Desde 1810 ...": No se detectó **V** como número romano.
- "... Historia. 3 vols. Volum **I**: 270 pagine 0 Volum ....": No se detectó **I** como número romano.
- "... segunda mitad del s. **XI**) .Escritor poeta ...": No se detectó **XI** como número romano.
- "... que se actualiza el anexo **V** clasificación de las variedades ...": No se detectó **V** como número romano.

		Reglas		
		Es nro. Romano	**NO es nro. romano**	**Total**
Clasificación correcta	**Es nro. romano**	219	**5**	224
	NO es nro. romano	**0**	173	173
	Total	222	178	400

Tabla 6.16: Resultados de detección de números romanos sobre una muestra

La tabla 6.17 muestra los resultados de la expansión de números romanos. Los errores de expansión detectados son:

- "... varios vizcondes (Ramón Trencavel **I**, **II**) así como ...": **I** se detecta como número romano por la regla de enumeraciones y se expande como "uno" cuando debe ser "primero".
- "... Ceremonia de apertura de los **XXX** Juegos Olímpicos se planificó para ...": **XXX** se detecta como número romano por la regla por defecto y se expande como "treinta" la forma habitual en este caso sería usar el ordinal: "trigésimos".
- "... con la disputa del **XVII** Rally Shalymar en Madrid ...": **XVII** se detecta como número romano por la regla por defecto y se expande como "diecisiete" la forma habitual en este caso sería usar el ordinal: "decimo séptimo".

- "... con acetato de mercurio (**II**) en tetrahidrofurano (THF ...": **II** se detecta como número romano por la regla por defecto y se expande como "segundo", entendemos que debe ser "dos".
- "... historia y religión. SECCIÓN **III**. BELLEZA DEL CUERPO ...": **II** se detecta como número romano por la regla por defecto y se expande como "tercero", en este caso debe ser "tres" o "tercera".

Expansión	#Casos	%
Correcta	213	97,26%
Incorrecta	6	2,74%
Total	**219**	

Tabla 6.17: Resultados de expansión de números romanos sobre una muestra

Finalmente, en la tabla 6.18 se puede observar un resumen de los resultados obtenidos con el análisis de la muestra. Sobre 400 casos analizados se clasificaron sin ningún tipo de error el 97.25 % de los casos.

Errores	#Casos	%
Detección	5	1,25%
Expansión	6	1,50%
Sin Errores	389	97,25%
Total	**400**	

Tabla 6.18: Tasa de error observada sobre una muestra

6.5. Evaluación del Módulo de Normalización

En la figura 6.3 se muestra un fragmento de texto de un artículo y el resultado generado por el módulo de normalización. En amarillo se resaltan las palabras que deben ser normalizadas en el texto original y las expansiones en el texto normalizado.

Texto Original

Entre los puertos Rosario y San Lorenzo son puntos de partida para la exportación de la producción de Santa Fe y muchas otras provincias, a través de ellos dejar el 65% de los cereales argentinos y 55 % de las exportaciones del país. En 2004, las exportaciones de Santa Fe (USD 7.170 millones) representaron el 21% del total nacional. Entre 2001 y 2004 se incrementaron 65,2%. Derivados de la soja, aceites vegetales y harinas compuesto por más de USD 2 mil millones y más de 7,6 millones de toneladas.En 2005 los puertos del sur de Santa Fe envió el 60% de los granos, el 93% La agricultura y subproductos del 85% de los aceites vegetales exportados por Argentina.

Texto Normalizado

Entre los puertos Rosario y san Lorenzo son puntos de partida para la exportación de la producción de Santa fe y muchas otras provincias, a través de ellos dejar el sesenta y cinco por ciento de los cereales argentinos y cincuenta y cinco por ciento de las exportaciones del país . <S>
En dos mil cuatro, las exportaciones de Santa fe (dólares siete mil ciento setenta millones) representaron el veintiuno por ciento del total nacional . <S>
Entre dos mil uno y dos mil cuatro se incrementaron sesenta y cinco coma dos por ciento . <S>
Derivados de la soja , aceites vegetales y harinas compuesto por más de dólares dos mil millones y más de siete coma seis millones de toneladas . <S>
En dos mil cinco los puertos del sur de Santa fe envió el sesenta por ciento de los granos , el noventa y tres por ciento La agricultura y subproductos del ochenta y cinco por ciento de los aceites vegetales exportados por Argentina . <S>

Figura 6.3: Fragmento de Texto Original y Normalizado

	Total	Enteros	Reales	Potencias	Ordinales	Porcentajes	Siglas	Abrev.	Nros. Romanos	Coord. / Grados	Rango Años	Unidades Medida	Monedas	Códigos
Artículo: Santa Fe (argentina) – 8.300 palabras														
#C	**319**	209	11		5	19	15	5	3	6	6	34	4	2
#E	**8**	5			1		2							
Pr	**97,49%**													
Errores: Articulo 1°. ---> Articulo primer, FPCyS ---> FPCyS , ITS ---> its, 1 millón ---> uno millón, 1 instituto superior ---> uno instituto superior , 200 especies -->doscientos especies, 2900 ha --->dos mil novecientos hectáreas, 1431--> mil cuatrocientos treinta y uno hectáreas,														
Artículo: Aconcagua – 1.506 palabras														
#C	**49**	25				1	1	3		2		16		1
#E	**2**									1				1
Pr	**95,92%**													
Errores: S32 39.11 W70.00. 42 ---> ese treinta y dos treinta y nueve punto once W70.00 . cuarenta y dos , IV+ ---> IV+														
Artículo: Océano Pacífico – 3.132 palabras														
#C	**58**	31			2		6		8	2	2	6		1
#E	**3**						3							
Pr	**94,83%**													
Errores: OHI ---> OHI (3 casos)														
Artículo: Vía Láctea – 2.421 palabras														
#C	**26**	14	2	1		1	6	1				1		
#E	**3**	1					1					1		
Pr	**88,46%**													
Errores: HII ---> HII, 1 trillón ---> uno trillón, 1.500 km/s ---> mil quinientos km/s														
Artículo: Manuel Belgrano – 12.292 palabras														
#C	**144**	130						3	10			1		
#E	**0**													
Pr	**100,00%**													
Total – 27.651 palabras														
#C	**596**	409	13	1	7	21	28	12	21	10	8	58	4	4
#E	**16**	6	0	0	1	0	6	0	0	1	0	1	0	1
Pr	**97,32%**													

Tabla 6.19: Resultados del Módulo de Normalización sobre una muestra de artículos (#C:Cantidad de casos, #E: Cantidad de errores, Pr: Precisión)

Para evaluar el desempeño del módulo de normalización en forma integral, se analizó el resultado de la normalización de 5 artículos de *Wikipedia* en español que no se habían testeado anteriormente y tratan diferentes temas: ciudades, geografía, astronomía y biografías.

Se recorrió cada artículo contabilizando las palabras que requieren normalización y evaluando si se resolvieron correctamente. La tabla 6.19 muestra los resultados obtenidos para cada artículo y el resultado general de los 5 artículos. En 5 artículos que suman 27.651 palabras se detectaron 596 palabras que requieren algún tipo de normalización, el 2.1 %. De las 596 palabras se expandieron correctamente 580 palabras, 97.3 %, sólo se verificaron errores en 16 palabras, el 2.7 % de los casos.

6.6. Trabajo a Futuro

A continuación se enumeran algunos aspectos a mejorar del módulo de normalización:

- **Monedas**: Hay casos para los que la decisión de anteponer la expansión del símbolo de moneda genera frases correctas pero poco naturales, por ejemplo: “$2” se expande a “pesos dos”. Se puede trabajar en detectar patrones que determinen cuándo la

expansión del símbolo de moneda debe ir antes o después del número.

- **Magnitudes y Monedas**: Se podría enriquecer el listado de símbolos de magnitudes y monedas.
- **Fechas y Horas**: Se podría trabajar en contemplar otros formatos de fechas, por ejemplo el formato "AAAA-MM-DD" o formatos con el mes en letras como "3-mayo-2014" considerando varios separadores distintos: guiones, barras o puntos. Para las Horas, se puede agregar el formato con puntos como separador por ejemplo: "20.30.20hs".
- **Números Enteros**: Para la expansión de los números enteros se utiliza el género masculino en todos los casos, en general la expansión con genero masculino coincide con la expansión femenina o neutra. En situaciones donde el número entero precede un sustantivo femenino y donde la expansión del número no sea igual para las 3 formas (masculino, femenino y neutro), puede quedar mal la frase resultante. Algunos ejemplos son: "1 millón" se expande a "uno millón" y debería ser "un millón", "200 especies" se expande a "doscientos especies" y debería ser "doscientas especies". Para resolver estos casos, se podría abordar con un análisis sintáctico que indique que la palabra posterior al número es un sustantivo de género femenino y entonces expandir el número con el género femenino por ejemplo. *Freeling* incluye funciones de análisis sintáctico que podrían emplearse para esta tarea. La resolución de este problema no es trivial, por ejemplo, dada la frase: "Cuándo le preguntaron cuántas bananas había comprado ese fin de semana en el mercado, respondió 500.", no hay ningún sustantivo posterior al número y tampoco el sustantivo anterior es el corresponde.
- **Números Ordinales/Temperaturas/Angulos/Coordenadas**: Existen casos con el símbolo de grado, por ejemplo: 20, 12 que pueden ser temperaturas, ángulos, coordenadas o un número ordinal. El error se produce cuando la expresión no cumple ninguna de las reglas definidas con lo cual se toma una acción por defecto que puede ser incorrecta. Se podría profundizar el análisis realizado con este tipo de expresiones y generar más reglas que permitan aumentar la precisión.
- **Otras Expresiones Numéricas**: Existen expresiones numéricas que no se llegaron a cubrir con las reglas definidas, son 122.976 casos que se tratan con la regla por defecto que consiste en deletrear la expresión. Este conjunto puede ser analizado nuevamente para definir nuevas reglas que mejoren expansión, algunos ejemplos son: "W70.000", 'E3-2010", "Super-8/Doble".
- **Abreviaturas**: Quedó pendiente realizar un análisis del Corpus de Datos construído con los artículos de *Wikipedia* para obtener un listado de abreviaturas más extenso. Se podría intentar detectar las abreviaturas detectando las palabras que contienen puntos, en ese caso no se debe utilizar el tokenizador de textitFreeling ya que separa el último punto de la palabra si la misma no está en el listado de abreviaturas. También se puede abordar las abreviaturas ambiguas, es decir, mantener más de una expansión para cada abreviatura y definir cuál utilizar en función del contexto. Se podría utilizar alguna técnica de aprendizaje automático para desarrollar un modelo de clasificación.

- **Siglas**: Se puede mejorar las reglas que definen si se pronuncia o deletrea una sigla. Las siglas donde alguna de las consonantes tiene un sonido especial como la "H" (muda) o la "Y" que actúa que en muchos casos como una vocal, produce errores en la clasificación. Por ejemplo, "OHI" se debería deletrear pero las reglas determinan que se pronuncie, "PYME" se debería deletrear si se considera que la "Y" actúa como vocal pero las reglas definidas no lo tienen en cuenta y determinan que se deletree. Además, la regla definida para detectar siglas considera que todas las letras deben ser mayúscula, sin embargo, hay siglas en donde la "y" se suele escribir en minúscula, como por ejemplo "FPCyS" y "PyME". Queda pendiente trabajar en analizar estos casos para contemplarlos.

- **Números Romanos**: Para los números romanos se realizó un análisis muy profundo, sin embargo, hay reglas por defecto cuyos casos se pueden seguir analizando para definir patrones que mejoren la precisión actual. También se puede trabajar en generalizar algunas reglas que utilizan listas de artículos, adjetivos o sustantivos para reemplazarlas con condiciones sobre atributos extraídos del análisis sintáctico de la oración. Ver caso " .... Carlos V." el tokenizador identifica la V. como inicial y crea una palabra con el punto. Esto hace que se expanda como inicial "ve" y no se trate como número romano.

- **URLs**: No se alcanzó a cubrir expresiones de URLs, por ejemplo: "http: www.dc.uba.ar", queda como trabajo a futuro detectar y expandir correctamente estas palabras.

- **Otras Expresiones (letras y símbolos)**: Hay expresiones que están formadas por letras y símbolos, por ejemplo: "C++", "IV+", "@Fernandez", "#Gol" que no se cubrieron. No son muy frecuentes y su resolución es relativamente simple.

Capítulo 7

Integración con el sistema back-end

En este capítulo se describe el último componente del sistema, que recibe el texto normalizado e interactúa con el sistema back-end suministrándole las oraciones a sintetizar y recibiendo los archivos de audio resultantes, para finalmente reproducirlos. Este componente fue desarrollado para interactuar con dos sistemas back-end distintos: Festival y MARY TTS y soportar cuatro voces distintas, dos de cada sistema. A continuación se describe los sistemas y el funcionamiento del módulo desarrollado.

7.1. Festival

Festival[1] es un sistema de síntesis de voz de propósito general para múltiples lenguajes, desarrollado originalmente por la Universidad de Edinburgo y la Universidad Carnegie Mellon, así como otros centros de enseñanza que han realizado contribuciones substanciales al proyecto. Festival y las herramientas de síntesis de voz se distribuyen bajo licencia tipo MIT-X11 permitiendo uso comercial y no comercial sin restricción. El proyecto está escrito en lenguaje C++ y está implementado como un intérprete de comandos el cual puede conectarse con diversos módulos y aplicaciones. El proyecto Festival es multilingüe (actualmente soporta inglés británico y americano, y castellano) aunque el inglés es el más avanzado. En este trabajo se utiliza la versión la 2.1 release Noviembre 2010.

Para el sistema TTS que se desarrolla en este trabajo, se utilizan dos voces de *Festival*:

- *uba_spanish_arg_secyt_cg*: voz en español argentino basada en HMM y desarrollada dentro del trabajo de Tesis de Licenciatura de **Luisina Violante** bajo la dirección del **Dr. Agustín Gravano**[19, 11]. Es la voz por defecto del sistema, para explicitar su selección se debe ejecutar el sistema con la opción "Festival_UBA".
- *el_diphone*: voz creada por concatenación de difonos en español ibérico incluida en la distribución de Festival. Para utilizarla, se debe ejecutar el sistema con la opción "Festival_el_diphone".

No se utilizan las voces basadas en selección de unidades construidas en el trabajo de Luisiana Violante dado que faltan algunas unidades, esto genera que se aborte el proceso de síntesis cuando se procesa una oración que requiere alguna de esas unidades.

[1] http://www.cstr.ed.ac.uk/projects/festival/

7.2. MARY TTS

MARY (Modular Architecture for Research on Speech Synthesis) es una plataforma escrita en Java, multilingüe y de código abierto, para sistemas TTS. Originalmente fue desarrollada como un proyecto colaborativo entre el laboratorio de Tecnologías del Lenguaje del Centro Alemán de Investigación de Inteligencia Artificial (DFKI) y el Instituto de Fonética en la Universidad de Saarland, en la actualidad está siendo mantenido por el DFKI.

Actualmente MARY TTS soporta los idiomas alemán, inglés británico y americano, telugú, turco y ruso; y nuevos idiomas están siendo preparados. MARY TTS ofrece herramientas para agregar soporte para nuevos lenguajes de manera rápida, y para construir voces basadas en sistemas de selección de unidades y HMM. En este trabajo se utiliza la versión 4.3.0 de MARY TTS.

Para el sistema TTS desarrollado están disponibles dos voces desarrolladas por **Luisina Violante** [19] para el sistema MARY TTS:

- *mary_dif_mod*: Voz creada por síntesis concatenativa con selección de unidades. Para utilizar esta voz, se debe ejecutar el sistema con la opción "MaryTTS_UBA1".
- *uba_secyt*: Es otra versión de voz creada por síntesis concatenativa con selección de unidades. Para utilizar esta voz, se debe ejecutar el sistema con la opción "MaryTTS_UBA2".

MARY TTS funciona bajo un esquema cliente-servidor, para poder ejecutar el sistema con alguna de las voces: MaryTTS_UBA1 o MaryTTS_UBA2, es necesario que el servidor de MARY TTS esté activo.

7.3. Funcionamiento del Módulo

El módulo de integración con los sistemas Back-end tiene un proceso principal que lee cada oración del texto y la envía Festival o MARY TTS para que se genere un archivo de audio (wav) con el resultado de la síntesis con la voz elegida. Existe un segundo hilo de ejecución (*thread*) que espera por los archivos de audio y, en paralelo con el proceso principal, reproduce los archivos de audio resultantes. La figura 7.1 ilustra el funcionamiento descrito.

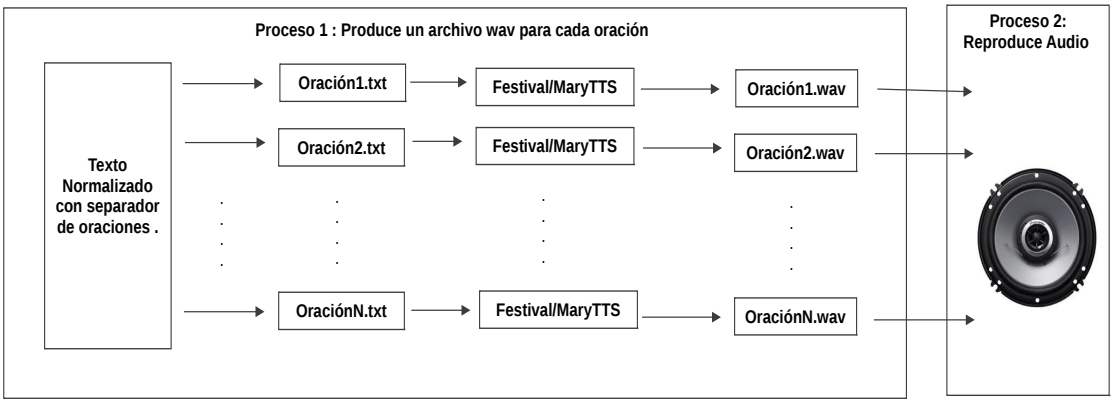

Figura 7.1: Funcionamiento del módulo de integración con los Back-ends

7.3.1. Integración con Festival

Transformaciones del Texto para Festival

Antes de enviar el texto a sintetizar a Festival, es necesario realizar las siguientes transformaciones:

- Se quitan guiones, paréntesis, corchetes, barras y comillas. En caso de dejarlos en el texto, se mencionarían en el audio. Por ejemplo "(ayer)" se sintetizaría como "abrir paréntesis ayer cerrar paréntesis" que no es lo que se pretende en este trabajo.
- Se reemplazan vocales sin pronunciación definida en español por vocal común o acentuada. Ejemplos: À por Á, Â por A, Ã por A, Ä por A, Å por A.
- Se quitan los espacios que existen antes de los signos de puntuación. Ejemplo: "¿ Qué hora es ?" se transforma en: "¿Qué hora es?". Si no se realiza esta transformación Festival no trata correctamente los símbolos de puntuación.
- Se reemplazan las vocales acentuadas por la codificación que usa Festival: á por 'a , é por 'e , í por 'i , ó por 'o , ú por 'u.
- Se reemplaza la ñ por la codificación que usa Festival: ñ por ~n.

Interacción con Festival

En la distribución de Festival existe una aplicación llamada *text2wave* que sintetiza un archivo de texto y produce un archivo de audio wav con el resultado. El módulo desarrollado ejecuta *text2wave* para cada archivo de oración generado. Ejemplo: text2wave oracion1.txt -o oración1.wav. Luego, cuando se sintetizaron las primeras 3 oraciones, el segundo proceso comienza la reproducción de los archivos de audio mientras se sintetizan el resto de las oraciones del texto hasta completar el total de oraciones del artículo.

7.3.2. Integración con MARY TTS

MARY TTS es un sistema cliente-servidor, es necesario establecer una conexión socket con el servidor (puerto 59125) y enviarle la oración a sintetizar indicando la voz elegida. El servidor devuelve el audio en formato wav que se guarda en un archivo para su reproducción. En la carpeta de ejemplos de la distribución de MARY TTS se incluye un programa cliente de ejemplo que se adaptó para realizar la conexión con el servidor (archivos MaryClient.cc, MaryClient.h y MaryDemo.CC). El servidor funciona con conexiones socket o HTTP, la conexión HTTP no funcionó con la aplicación cliente adaptada para este trabajo.

7.4. Resultados y Trabajo a Futuro

El sistema completo fue testeado con 5 artículos, la tabla 7.1 muestra el tiempo de respuesta observado con la voz *uba_spanish_arg_secyt_cg* de Festival. La columna "Total (segs)" muestra el tiempo en segundos que requiere el sistema hasta que se reproduce el sonido de la primera oración del texto, a este tiempo lo denominamos Tiempo de Respuesta del Sistema. En la tabla también se muestra el tiempo de los módulos del sistema, se

Artículo	#Palabras	Tpo. Rta. (segs.)	WGET + HTML->TXT + Sep. Oraciones + Traductor (segs.)	Normalizador (segs.)	BackEnd (segs.)
Santa Fe (argentina)	8.300	10	2	7	1
Aconcagua	1.506	13	2	5	6
Océano Pacífico	3.132	17	2	10	5
Vía Láctea	2.421	11	2	7	2
Manuel Belgrano	12.292	23	3	17	3

Tabla 7.1: Evaluación de tiempos

agruparon los primeros 3 módulos y el tiempo de obtención de la página HMTL ya que los tiempos individuales no son importantes.

El módulo de Normalización es el que requiere más tiempo de ejecución. Los datos de la tabla 7.1 confirma que el tiempo de ejecución del módulo de normalización tiene una relación lineal a la cantidad de palabras del artículo.

El módulo de integración con el Back-end es otro módulo que consume varios segundos, en este caso el módulo debe procesar las 3 primeras oraciones antes de iniciar la reproducción de la primera oración. Cuanto más cortas sean las primeras 3 oraciones, menor será el Tiempo de Respuesta del Sistema.

La figura 7.2 muestra el esquema del funcionamiento del sistema implementado, se puede observar que en cada etapa se procesa en forma completa todo el artículo. Una propuesta para reducir el Tiempo de Respuesta del Sistema es particionar el texto del artículo y procesar en forma paralela cada partición. La primera partición sería la más importante ya que determina el Tiempo de Respuesta del Sistema. Habría que analizar el tamaño apropiado de las particiones para minimizar el Tiempo de Respuesta y asegurar la fluidez del audio final. La figura 7.3 muestra el esquema del funcionamiento propuesto para mejorar el Tiempo de Respuesta en futuras versiones del sistema. El esquema propuesto reduciría la incidencia del tiempo de normalización en el Tiempo de Respuesta del Sistema, pero no mejorará el tiempo del módulo de integración con el sistema Back-end. Para reducir este último tiempo, se puede considerar la longitud de las oraciones y si se tratara de oraciones lo suficientemente largas, se podría comenzar a reproducir el audio antes, es decir, tal vez no sería necesario esperar a tener sintetizadas las primeras 3 oraciones, podría alcanzar con 1 o 2 oraciones para asegurar la fluidez del audio.

Otra opción alternativa puede ser continuar con el esquema actual de la figura 7.2 pero realizar la separación de oraciones antes de la normalización, de manera de normalizar de a una oración por vez.

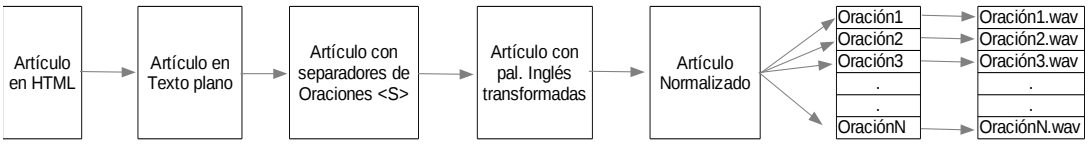

Figura 7.2: Esquema del Funcionamiento del Sistema Implementado

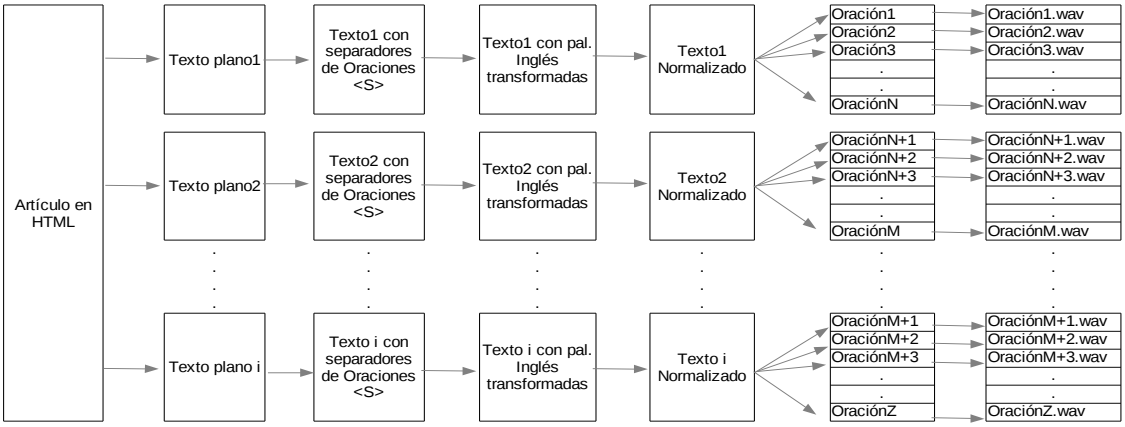

Figura 7.3: Esquema Propuesto para Mejorar el Tiempo de Respuesta del Sistema

A continuación se mencionan otras mejoras que se pueden desarrollar en el futuro:

- Controlar la reproducción del audio: se podría trabajar en controlar la reproducción del audio incorporando funciones para pausa, retroceso y avance.
- *Streaming*: el sistema desarrollado permite escuchar el audio de los artículos de *Wikipedia* sintetizados desde la computadora que aloja el sistema, para poder ser usado como proveedor de un servicio de transformación de texto a voz para muchos usuarios a través de internet, es necesario agregar una funcionalidad que trasmita el audio resultante a la PC cliente mediante *streaming*.

Capítulo 8

Conclusiones

Se logró el objetivo de construir el front-end de un sistema Texto-a-Habla y conectarlo con sistemas back-end existentes, de manera de obtener un sistema completo para artículos de *Wikipedia* en español. Se construyeron 6 módulos para resolver las tareas del front-end y se logró articular en ellos varias herramientas, entre ellas: *Freeling*[12, 13, 4, 3], *OpenFST*[1], *Festival*[8], *Mary TTS*[6] y el resultado de los trabajos de Tesis de Licenciatura de Verónica Pechersky[14] y Luisina Violante[19, 11].

En el módulo 1 del sistema se recorre el árbol de parsing de la página HTML construido con *htmlcxx*[5] y se extrae el texto de los tags que lo contienen. Hay filtros y condiciones particulares para el formato de los artículos de *Wikipedia* pero el esquema general debería funcionar para poder abarcar otros contenidos (ej.: páginas de diarios), introduciendo modificaciones menores.

En el módulo 2 se utilizó *Freeling* para separar las oraciones del texto, se evaluó el desempeño con una muestra de 10 artículos y se obtuvo una tasa de error del 1.27 %, comparable a las que encontramos en la literatura para el español y otros idiomas [15, 10].

Para construir el módulo 3, que procesa las palabras de otros idiomas, se abordó el caso de palabras en inglés. Se generó la pronunciación en español para un diccionario inglés de 121.761 palabras utilizando su definición fonética y reglas ad-hoc. También se definió un procedimiento para detectar las palabras en inglés del texto en español. Queda como trabajo futuro ampliar el alcance a palabras de otros idiomas además del inglés.

Como punto de partida en la construcción del módulo 4, normalización, se construyó una base de datos de artículos de *Wikipedia* a partir del procesamiento de 9.796 archivos con 339.5 millones de palabras. Se explotaron esos datos para generar reglas que definen patrones de texto que requieren ser normalizados. Esos datos también contribuyeron a utilizar el contexto de una palabra cuando se presenta una ambigüedad. Se utilizaron los FSTs desarrollados por Verónica Pechersky[14] para expandir las expresiones numéricas, previamente fue necesario adaptar el código fuente para que funcione en Linux. Para el tratamiento de siglas, se definieron reglas que discriminan las que se pronuncian de las que se deletrean. Para la evaluación del módulo, se tomó una muestra de 5 artículos y se verficó a mano la salida producida obteniendo una tasa de aciertos del 97.3 %. Queda como trabajo a futuro enriquecer las reglas para mejorar la precisión y ampliar el espectro a otros contenidos.

Finalmente, los módulos 5 y 6 del sistema permitieron integrar el front-end construido con *Festival* y *Mary TTS*. Para ambos sistemas se utilizaron voces desarrolladas por Luisina Violante[19, 11]. Como trabajo a futuro queda mejorar el tiempo de respuesta

del sistema procesando oración por oración o conjunto de oraciones, agregar funciones para controlar la reproducción del audio y adaptar el sistema para que pueda brindar el servicio de streaming.

Apéndice A

Instalación y Uso del Sistema

A.1. Requerimientos del Sistema

El sistema corre sobre el sistema operativo Linux, fue desarrollado y probado en Ubuntu 12.04.3 LTS (32 bits).

Antes de instalar los módulos desarrollados en esta tesis, se requiere tener instaladas las siguientes librerías y herramientas:

- **HTMLCXX**[5]: Esta librería se utiliza en el módulo 1 del sistema (descrito en el capítulo 2), que extrae el texto de la página HTML.

  Se instala con el archivo **htmlcxx-0.84.tar.gz** que se puede descargar de `http://sf.net/projects/htmlcxx`.

- **Boost c++ Libraries**[7]: Se requieren los siguientes componentes que se utilizan en varios de los módulos desarrollados:

  - libboost-regex1.48.0 (versión 1.48.0-3)
  - libboost-regex1.48-dev (versión 1.48.0-3)
  - libboost-filesystem1.48.0 (versión 1.48.0-3)
  - libboost-filesystem1.48-dev (versión 1.48.0-3)
  - libboost-thread1.48.0 (versión 1.48.0-3)
  - libboost-thread1.48-dev (versión 1.48.0-3)

  Se pueden descargar de `http://www.boost.org`.

- **FreeLing**[12, 13, 4, 3]: Se utiliza la versión 3.0 en varios de los módulos desarrollados. Se puede descargar desde `http://devel.cpl.upc.edu/freeling/downloads?order=time&desc=1`.

- **Open FST**[1]: Se utiliza la versión 1.3.4 en el módulo de normalización. Se puede descargar desde `http://www.openfst.org/twiki/bin/view/FST/FstDownload`

- **Festival**[8]: En la distribución se incluye el archivo Festival.zip con los archivos necesarios para instalar *Festival*. A continuación se describen los pasos a seguir para instalar *Festival* 2.1 y la voz construida en la tesis de Luisina Violante[19, 11] que se usa en el sistema.

```
### Instalación de Festival 2.1 y Speech Tools 2.1 ###

1) Bajar los archivos de: http://festvox.org/packed/festival/2.1/

2) speech_tools-2.1-release.tar.gz : se descomprime en /speech_tools y
luego instalar con: ./configure & sudo make & sudo make install

3) Descomprimir todos estos archivos para que queden en /festival

  festival-2.1-release.tar.gz

  festlex_CMU.tar.gz

  festlex_OALD.tar.gz

  festlex_POSLEX.tar.gz

  festvox_cmu_us_awb_cg.tar.gz

  festvox_cmu_us_rms_cg.tar.gz

  festvox_cmu_us_slt_arctic_hts.tar.gz

  festvox_kallpc16k.tar.gz

  festvox_rablpc16k.tar.gz

Instalar con: ./configure & sudo make & sudo make install

### Instalación de la voz construida por Luisina Violante ###

Voz: uba_spanish_arg_secyt_cg

- Copiar el contenido de /FEST_HMM_MOD/uba_spanish_arg_secyt/ en:

  /festival/lib/voices/spanish/uba_spanish_arg_secyt_cg/

- Copiar el archivo spanlex.scm en: festival/lib/dicts/
```

- **MARY TTS**[6]: En la distribución se incluye el archivo MARYTTS.zip con los archivos necesarios para instalar *MARY TTS*. A continuación se describen los pasos

a seguir para instalar *MARY TTS* 4.3.0 y las voces construidas en la tesis de Luisina Violante[19, 11] que se usan en el sistema.

```
Descomprimir el archivo MARYTTS.zip en una carpeta auxiliar.

### Instalación de Mary TTS versión 4.3.0 ###

  a) Descargar el archivo: openmary-standalone-install-4.3.0.jar

  b) Ejecutarlo: java -jar openmary-standalone-install-4.3.0.jar

  c) Elegir la carpeta y seguir el proceso de instalación.

### Instalación de las voces de Luisina Violante ###

  1- Copiar los archivos .zip de las voces en la carpeta \lib\voices
  (crear la carpeta voices si no existe)

  Descomprimir los archivos .zip, debe quedar una carpeta con el
  nombre de cada voz:

  mary_dif_mod
  uba_secyt

  2- Copiar en la carpeta \installed (crearla si no existe) el archivo:

  mary-uba_secyt-4.3.0-component.xml

  3- Copiar en la carpeta "conf" los archivos:

  es.config
  es-mary_dif_mod.config
  es-uba_secyt.config

  4- Copiar la carpeta "es" (archivos para el idioma español) en:
  "\lib\modules"
```

- **SoX - Sound eXchange**: Es una aplicación de audio que ofrece varias funcionalidades, en el sistema se utiliza para reproducir los archivos de sonido .wav. Se puede descargar desde `http://sox.sourceforge.net/`.

A.2. Instalación de los Módulos del Sistema

A.2.1. Archivos de Distribución

La distribución del sistema incluye los siguientes archivos:

- HTML2TXT-0.1.tar.gz: archivo de distribución del módulo 1.

- SeparaOraciones-0.1.tar.gz: archivo de distribución del módulo 2.
- Traductor-0.1.tar.gz: archivo de distribución del módulo 3.
- Normalizador-0.1.tar.gz: archivo de distribución del módulo 4.
- BackEndTransf-0.1.tar.gz: archivo de distribución del módulo 5.
- BackEnd-0.1.tar.gz: archivo de distribución del módulo 6.
- TTS: script para la ejecución del sistema.
- siteinit_uba.scm: archivo para definir la voz por defecto de *Festival*[8] a *uba_spanish_arg_secyt_cg*.
- siteinit_el_diphone.scm: archivo para definir la voz por defecto de *Festival*[8] a *el_diphone*.
- MARYTTS.zip: archivos para instalar *MARY TTS* 4.3.0[6] y las voces que se usan en el sistema.
- Festival.zip: archivos para instalar *Festival* 2.1[8]y la voz que se usa en el sistema.

Para instalar el sistema, se debe copiar estos archivos a una carpeta que será la carpeta de instalación del sistema. Por ejemplo:/home/usuario/SistemaTTS.

A continuación se describe como instalar y probar cada módulo individualmente, finalmente se describe como ejecutar el sistema completo.

A.2.2. Módulo 1: HTML2TXT

Es el módulo del sistema descrito en el capítulo 2.Para distribuir el módulo 1, se generó el archivo **HTML2TXT-0.1.tar.gz** con las herramientas GNU build system (Autotools)[1]

Contenido:

- main.cpp: código fuente con la implementación del módulo.
- README: archivo de texto con instrucciones para la instalación.

Para instalarlo, se debe descomprimir y luego ejecutar los comandos: **./configure && make**

La sintaxis para ejecutar el módulo es:

```
./HTML2TXT [-i ArchivoEntrada] [-o ArchivoSalida] [-f CarpetaInstalación]
```

La opción **-i** especifica el archivo de entrada y la opción **-o** el archivo de salida, si se omite el argumento de archivo de entrada y/o salida se usa *stdin* y/o *stdout* respectivamente. La opción **-f** especifica la ruta de la carpeta de instalación del módulo, que se debe especificar si se ejecuta el módulo desde otra ubicación.

Luego de la instalación se puede probar el módulo individualmente con esta sentencia:

[1] `http://es.wikipedia.org/wiki/GNU_build_system`

```
wget -qO- "http://es.wikipedia.org/wiki/Universidad_de_Buenos_Aires"
|./HTML2TXT -o "htmltotxt.txt"
```

En la sentencia de ejemplo se obtiene el arctículo de Wikipedia y se lo convierte a texto dejando el resultado en el archivo *htmltotxt.txt*.

A.2.3. Módulo 2: Segmentación de Oraciones

Es el módulo del sistema descrito en el capítulo 3.Para distribuir el módulo 2, se generó el archivo **SeparaOraciones-0.1.tar.gz**.

Contenido:

- main.cpp: código fuente con la implementación del módulo.
- README: archivo de texto con instrucciones para la instalación.
- tokenizer.dat: archivo de configuración para el módulo *Tokenizer* de *FreeLing*[12, 13, 4, 3].
- splitter.dat: archivo de configuración para el módulo *Splitter* de *FreeLing*[12, 13, 4, 3].
- EjemploIn.txt: archivo de texto de ejemplo para probar el módulo individualmente.

Para instalarlo, se debe descomprimir y luego ejecutar los comandos: **./configure && make**

La sintaxis para ejecutar el módulo es:

```
./SeparaOraciones [-i ArchivoEntrada] [-o ArchivoSalida] [-f CarpetaInstalación]
```

La opción **-i** especifica el archivo de entrada y la opción **-o** el archivo de salida, si se omite el argumento de archivo de entrada y/o salida se usa *stdin* y/o *stdout* respectivamente. La opción **-f** especifica la ruta de la carpeta de instalación del módulo, que se debe especificar si se ejecuta el módulo desde otra ubicación.

Luego de la instalación se puede probar el módulo individualmente con esta sentencia:

```
./SeparaOraciones -i HtmlToTxt.txt -o ArchivoSeparado.txt
```

En la sentencia de ejemplo se genera el archivo *ArchivoSeparado.txt* que contiene el texto del archivo *EjemploIn.txt* con las oraciones separadas.

A.2.4. Módulo 3: Palabras de Otros Idiomas

Es el módulo del sistema descrito en el capítulo 5.Para distribuir el módulo 3, se generó el archivo **Traductor-0.1.tar.gz**.

Contenido:

- main.cpp: código fuente principal con la implementación del módulo.

- Avl.h: encabezado del árbol binario de búsqueda de la librería *tree.hh*[2].
- Avl2.h: implementación del árbol binario de búsqueda de la librería *tree.hh*[2].
- Comparable.h: es la clase que encapsula el dato que va en las hojas del árbol binario de búsqueda, forma parte de la librería *tree.hh*[2].
- diccitem.h: encabezado de la clase que almacena cada item del diccionario.
- diccitem.cpp: implementación de la clase que almacena cada item del diccionario.
- README: archivo de texto con instrucciones para la instalación.
- tokenizer.dat: archivo de configuración para el módulo *Tokenizer* de *FreeLing*[12, 13, 4, 3].
- splitter.dat: archivo de configuración para el módulo *Splitter* de *FreeLing*[12, 13, 4, 3].
- EjemploIn.txt: archivo de texto de ejemplo para probar el módulo individualmente.

Para instalarlo, se debe descomprimir y luego ejecutar los comandos: **./configure && make**

La sintaxis para ejecutar el módulo es:

```
./Traductor [-i ArchivoEntrada] [-o ArchivoSalida] [-f CarpetaInstalación]
```

La opción **-i** especifica el archivo de entrada y la opción **-o** el archivo de salida, si se omite el argumento de archivo de entrada y/o salida se usa *stdin* y/o *stdout* respectivamente. La opción **-f** especifica la ruta de la carpeta de instalación del módulo, que se debe especificar si se ejecuta el módulo desde otra ubicación.

Luego de la instalación se puede probar el módulo individualmente con esta sentencia:

```
./Traductor -i EjemploIn.txt -o ArchivoTraducido.txt
```

En la sentencia de ejemplo se genera el archivo *ArchivoTraducido.txt* que contiene el texto del archivo *EjemploIn.txt* con las palabras en inglés reemplazadas por su pronunciación en español.

A.2.5. Módulo 4: Normalizador

Es el módulo del sistema descrito en el capítulo 6.Para distribuir el módulo 4, se generó el archivo **Normalizador-0.1.tar.gz**.

Contenido:

- main.cpp: código fuente principal con la implementación del módulo.
- funciones.h: encabezado del archivo que contiene las funciones que implementan las reglas de normalización.

- funciones.cpp: código fuente que contiene las funciones que implementan las reglas de normalización.
- utilfst.h: encabezado que contiene definiciones de enumeraciones y structs utilizadas por los FSTs
- fst_string.cpp: código fuente que contiene funciones de soporte para convertir FSTs en strings.
- misc.cpp: código fuente que contiene funciones de soporte para generación de FSTs.
- utility.cpp: código fuente que contiene funciones para conversión de texto UTF-8.
- Avl.h: encabezado del árbol binario de búsqueda de la librería *tree.hh*[2].
- Avl2.h: implementación del árbol binario de búsqueda de la librería *tree.hh*[2].
- Comparable.h: es la clase que encapsula el dato que va en las hojas del árbol binario de búsqueda, forma parte de la librería *tree.hh*[2].
- diccitem.h: encabezado de la clase que almacena cada item del diccionario.
- diccitem.cpp: implementación de la clase que almacena cada item del diccionario.
- README: archivo de texto con instrucciones para la instalación.
- tokenizer.dat: archivo de configuración para el módulo *Tokenizer* de *FreeLing*[12, 13, 4, 3].
- splitter.dat: archivo de configuración para el módulo *Splitter* de *FreeLing*[12, 13, 4, 3].
- Abreviaturas.txt: listado de abreviaturas.
- siglas.txt: listado de siglas.
- EjemploIn.txt: archivo de texto de ejemplo para probar el módulo individualmente.

Para instalarlo, se debe descomprimir y luego ejecutar los comandos: **./configure && make**

La sintaxis para ejecutar el módulo es:

```
./Normalizador [-i ArchivoEntrada] [-o ArchivoSalida] [-f CarpetaInstalación]
```

La opción **-i** especifica el archivo de entrada y la opción **-o** el archivo de salida, si se omite el argumento de archivo de entrada y/o salida se usa *stdin* y/o *stdout* respectivamente. La opción **-f** especifica la ruta de la carpeta de instalación del módulo, que se debe especificar si se ejecuta el módulo desde otra ubicación.

Luego de la instalación se puede probar el módulo individualmente con esta sentencia:

```
./Normalizador -i EjemploIn.txt -o ArchivoNormalizado.txt
```

En la sentencia de ejemplo se genera el archivo *ArchivoNormalizado.txt* que contiene el texto normalizado del archivo *EjemploIn.txt*.

A.2.6. Módulo 5: Transformación del texto para el sistema Back-end

Este módulo realiza las transformaciones de texto requeridas por *Festival*[8] descritas en el capítulo 7. Para distribuir el módulo 5, se generó el archivo **BackEndTransf-0.1.tar.gz**.

Contenido:

- main.cpp: código fuente principal con la implementación del módulo.
- README: archivo de texto con instrucciones para la instalación.
- EjemploIn.txt: archivo de texto de ejemplo para probar el módulo individualmente.

Para instalarlo, se debe descomprimir y luego ejecutar los comandos: **./configure && make**

La sintaxis para ejecutar el módulo es:

```
./BackEndTransf -i ArchivoEntrada -o ArchivoSalida
```

La opción **-i** especifica el archivo de entrada y la opción **-o** el archivo de salida, si se omite el argumento de archivo de entrada y/o salida se usa *stdin* y/o *stdout* respectivamente.
Luego de la instalación se puede probar el módulo individualmente con esta sentencia:

```
./BackEndTransf [-i EjemploIn.txt] [-o SalidaBackEndTransf.txt]
```

En la sentencia de ejemplo se genera el archivo *SalidaBackEndTransf.txt* que contiene el texto con las transformaciones realizadas sobre el archivo *EjemploIn.txt*.

A.2.7. Módulo 6: Integración con el sistema Back-end

Este módulo interactúa con el sistema back-end y reproduce el audio resultante de acuerdo a lo descrito en el capítulo 7. Para distribuir el módulo 6, se generó el archivo **BackEnd-0.1.tar.gz**.

Contenido:

- main.cpp: código fuente principal con la implementación del módulo, se usa parte del código del archivo MaryDemo.cc que se incluye en la distribución de MARY TTS[6].
- MaryClient.h: encabezado para la clase MaryClient que se usa para conectar con el servidor de MARY TTS. Este archivo viene en la distribución de MARY TTS.
- MaryClient.cpp: implementación de la clase MaryClient que se usa para conectar con el servidor de MARY TTS. Este archivo viene en la distribución de MARY TTS.
- BackEndConfig.txt: archivo de texto con opciones de configuración.
- README: archivo de texto con instrucciones para la instalación.

- EjemploIn.txt: archivo de texto de ejemplo para probar el módulo individualmente.

Para instalarlo, se debe descomprimir y luego ejecutar los comandos: **./configure && make**

Modificar el archivo de configuración *BackEndConfig.txt*, se muestra el contenido a continuación:

```
# IP del Servidor de MARYTTS, se usa conexión SOCKET
MARYTTS_server=localhost

# Puerto del Servidor de MARYTTS
MARYTTS_port=59125

# Localización del archivo text2wave de Festival
Festival=/home/ezequiel/Festival/festival/bin/text2wave

# Reproductor usado para los archivos .wav
Reproductor=padsp play
```

Lo que seguramente sea necesario modificar es la ruta al archivo *text2wave* de *Festival*.

La sintaxis para ejecutar el módulo es:

```
./BackEnd [-i ArchivoEntrada] -v NombreVoz [-f CarpetaInstalación]
```

La opción **-i** especifica el archivo de entrada, la opción **-v** la voz y el sistema back-end deseado y la opción **-f** la ruta de la carpeta de instalación del módulo, que se debe especificar si se ejecuta el módulo desde otra ubicación. El argumento *NombreVoz* puede tomar alguno de los siguientes valores:

Voces de Festival:

- uba_spanish_arg_secyt_cg
- el_diphone

Voces de MARY TTS:

- uba_secyt
- mary_dif_mod

Para utilizar las voces de MARY TTS el servidor debe estar activo (/bin/maryserver) y configurado para funcionar con la opción socket y con igual puerto de conexión que el definido en el archivo de configuración del módulo. Abrir el archivo de configuración de MARY TTS[6] (MARYTTS/conf/marybase.config) y editarlo de ser necesario, a continuación se muestra el fragmento que se hace referencia:

```
##########################################################################
####################### Global settings  #################################
##########################################################################

# Type of server? (socket/http/commandline)
server = socket
server.http.parallelthreads = 6

# server socket port:
socket.port = 59125
```

Luego de la instalación se puede probar el módulo individualmente con esta sentencia:

```
./BackEnd -i EjemploIn.txt -v uba_spanish_arg_secyt_cg
```

En la sentencia de ejemplo se reproduce el archivo *EjemploIn.txt* con la voz *uba_spanish_arg_secyt_cg* de *Festival*[8].

A.3. Ejecución del Sistema

El sistema completo se ejecuta concatenando los procesos mediante *pipes* de manera que la salida de un proceso sea la entrada del siguiente, esto esta definido en el script **TTS** que se incluye en la distribución del sistema.

El siguiente fragmento del script **TTS** define las rutas a las carpetas de instalación de los módulos:

```
# Ruta de los módulos
HTML2TXT_DIR=./HTML2TXT-0.1/
SepOraciones_DIR=./SeparaOraciones-0.1/
Traductor_DIR=./Traductor-0.1/
Normalizador_DIR=./Normalizador-0.1/
BackEndTransf_DIR=./BackEndTransf-0.1/
BackEnd_DIR=./BackEnd-0.1/
```

No hay necesidad de modificar estas rutas si se ejecuta el sistema desde la carpeta de instalación del sistema.

En la distribución del sistema se incluyen 2 archivos que se usan para definir la voz por defecto de *Festival*[8], son los siguientes:

- siteinit_uba.scm: este script define la voz por defecto a *uba_spanish_arg_secyt_cg*.
- siteinit_el_diphone.scm: este script define la voz por defecto a *el_diphone*.

Es necesario cambiar en el archivo script **TTS** la ruta a la carpeta lib de *Festival*[8]:

```
# Ruta al archivo siteinit de Festival
Festival_siteinit=/home/ezequiel/Festival/festival/lib/siteinit.scm
```

Esta sentencia reemplaza el archivo *siteinit* de *Festival*[8] por el que define la voz por defecto seleccionada:

```
# Reemplaza el archivo siteinit.scm con un archivo que define
# por defecto la voz uba_spanish_arg_secyt_cg.
cp siteinit_uba.scm $Festival_siteinit
```

La ejecución completa del sistema concatenando todos los módulos se realiza con esta sentencia:

```
wget -qO- $1 | \
$HTML2TXT_DIR./HTML2TXT -f $HTML2TXT_DIR | \
$SepOraciones_DIR./SeparaOraciones -f $SepOraciones_DIR | \
$Traductor_DIR./Traductor -f $Traductor_DIR | \
$Normalizador_DIR./Normalizador -f $Normalizador_DIR | \
$BackEndTransf_DIR./BackEndTransf -f $BackEndTransf_DIR | \
$BackEnd_DIR./BackEnd -v $Voz -f $BackEnd_DIR
```

Para cada módulo es necesario especificar la ruta de instalación con la opción **-f** ya que se ejecuta cada módulo desde fuera de su carpeta de instalación.

La sintaxis para ejecutar el sistema completo es:

```
./TTS URL [voz]
```

Donde **URL** es la dirección del artículo de *Wikipedia* y **voz** define la voz y el sistema back-end que puede alguna de las siguientes opciones:

- Festival_UBA: voz *uba_spanish_arg_secyt_cg* de *Festival*, es la opción por defecto.
- Festival_el_diphone: voz *el_diphone* de *Festival*.
- MaryTTS_UBA1: voz *mary_dif_mod* de *MARY TTS*, es necesario que el servidor de *MARY TTS* este activo.
- MaryTTS_UBA2: voz *uba_secyt* de *MARY TTS*, es necesario que el servidor de *MARY TTS* este activo.

Ejemplo:

```
./TTS http://es.wikipedia.org/wiki/UBA Festival_UBA
```

En el ejemplo, se procesa el artículo de *Wikipedia* de la Universidad de Buenos Aires con la voz *uba_spanish_arg_secyt_cg* de *Festival*[8].

Bibliografía

[1] OpenFst Library: es una librería para construir, combinar, optimizar y realizar búsquedas de traductores de estados finitos (FSTs).

[2] tree.hh: an STL-like C++ tree class. Autor: Kasper Peeters. http://tree.phi-sci.com/.

[3] Jordi Atserias, Bernardino Casas, Elisabet Comelles, Meritxell González, Lluís Padró, and Muntsa Padró. Freeling 1.3: Syntactic and semantic services in an open-source nlp library. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, May 2006. ELRA.

[4] Xavier Carreras, Isaac Chao, Lluís Padró, and Muntsa Padró. Freeling: An open-source suite of language analyzers. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, 2004.

[5] Robson Braga Araújo Davi de Castro Reis. 2005. HTMLCXX: Non-validating HTML parser library for C++.

[6] http://mary.dfki.de/. MARY Text-to-Speech System.

[7] http://www.boost.org/. Boost c++ libraries: librerias para C++ de uso general para múltiples aplicaciones.

[8] http://www.cstr.ed.ac.uk/projects/festival/. The Festival Speech Synthesis System.

[9] Daniel Jurafsky James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Second Edition*. Prentice Hall, 2000.

[10] Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.

[11] A. Gravano L. Violante, P. Rodríguez Zivic. Improving speech synthesis quality by reducing pitch peaks in the source recordings. In *Proc. of NAACL-HLT 2013 (short paper), pp. 502-506*, June 2013.

[12] Lluís Padró. Analizadores multilingües en freeling. *Linguamatica*, 3(2):13–20, December 2011.

[13] Lluís Padró and Evgeny Stanilovsky. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey, May 2012. ELRA.

[14] Verónica Pechersky. Normalización del texto de entrada para un sistema de síntesis del habla. Tesis de Licenciatura. Director: Agustín Gravano. Departamento de Computación, FCEyN, Universidad de Buenos Aires. 2012.

[15] Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC '97, pages 16–19, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

[16] Stanley Chen Shankar Kumar Mari Ostendorfk-Christopher Richards Richard Sproat, Alan W. Black. Normalization of non-standard words. 2001.

[17] Ruben San-Segundo, Juan M. Montero, Veronica Lopez-Luden, and Simon King. *Detecting Acronyms from Capital Letter Sequences in Spanish.* 2012.

[18] J. M. Montero R. Barra-Chicote J. Lorenzo V. López-Ludeña, R. San-Segundo. Architecture for text normalization using statistical. Universidad Politécnica de Madrid. Spain. Speech Technology Group. E.T.S.I. Telecomunicación.

[19] Luisina Violante. Construcción y evaluación del back-end de un sistema de síntesis de habla en español argentino. Tesis de Licenciatura. Director: Agustín Gravano. Departamento de Computación, FCEyN, Universidad de Buenos Aires. 2012.