UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

MARCACIÓN LÉXICA DE CORPUS: EL CASO DE EXPRESIONES TEMPORALES

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

RODRIGO POBLANNO BALP

FACULTAD DE CIENCIAS
UNAM

DIRECTORA DE TESIS: DRA. SOFÍA NATALIA GALICIA HARO

2 0 0 7

Hoja de datos del jurado

1. Datos del alumno

Poblanno
Balp
Rodrigo
56 35 18 09
Universidad Nacional Autónoma de México
Facultad de Ciencias
Ciencias de la Computación
40009367-7

2. Datos del tutor

Dra.
Sofía Natalia
Galicia
Haro

3. Datos del sinodal 1

Dra.
Amparo
López
Gaona

4. Datos del sinodal 2

M. en C.
María Guadalupe Elena
Ibargüengoitia
González

5. Datos del sinodal 3

M. en C.
José Antonio
Neme
Castillo

6. Datos del sinodal 4

Dra.
Hanna
Oktaba

7. Datos del trabajo escrito

Marcación Léxica De Corpus: El Caso De Expresiones Temporales
81 p
2007

A Naara, mi hermana. Mi compañera fiel de toda la vida. Por haber existido, por las risas, los llantos, los momentos. Agradezco todo lo que has hecho por mí.

A mi madre, Paulina "Pawa", por su apoyo, por su confianza en mí. Por haberme educado. Por haberme dado todo incondicionalmente. Por su suavidad y dulzura.

A Laura, por estar a mi lado y darme su amor. Por enseñarme a trabajar más y mejor. Por los exámenes juntos y las pintas a la cafetería.

A mi padre, Miguel Ángel, también por su apoyo y por haberme enseñado a nunca darme por vencido y a que uno sólo baja la cabeza para pensar o recoger algo.

A mi abuelita "pipis". Gracias por todo lo que dejaste en mi corazón.

A mis amigos, Abo, Brian, Mauro y Paola. Por todo lo que hemos vivido juntos. Por su apoyo, por escucharme, por pedir mi consejo, por provocar estos excelentes recuerdos.

A mi tío Enrique "Pareja" por ser excelente persona, por su apoyo y su cariño.

A Óscar "Gordo", Rodrigo "Rock", Paola "Pola", Gabriel "Peque", Gustavo "Koala", Patricia, Emmanuel, Yazmín, Didier.

A Sofía Natalia, mi directora de tesis. Por haberme tenido tanta paciencia, por entenderme, corregirme y enseñarme. A todos los profesores que tuve a lo largo de mi carrera.

A todos los familiares, tíos, tías, primos, primas que me ayudaron a lo largo de mi vida, desde aprender a caminar.

A Pícaro, por haber sido mi primer mejor amigo. A Chewbacca, por ser mi actual mejor amiga.

A los Beatles. A Mozart.

A todos los que deberían estar aquí mencionados.

A mí.

Tabla de contenido

Tabla de contenido i
Aclaraciones sobre la tipografía utilizada iii
Más información sobre este trabajo iv
1 Introducción 1
Definición del problema 2
Objetivo general 4
Estructura del documento 4
1.1 El lenguaje natural 6
1.2 Lingüística computacional 9
1.3 El tiempo en el lenguaje 11
1.4 Expresiones Temporales 13
1.5 Trabajos en el tema de expresiones temporales 15
2 Formalización del lenguaje natural 18
2.1 ¿Qué se necesita para entender el lenguaje natural? 19
3 Lingüística de corpus 24
3.1 Características del corpus utilizado 27
3.2 Marcado del corpus 28
4 Procesamiento del lenguaje natural 30
4.1 ¿Qué hace la computadora para entender el lenguaje natural? 31
4.1.1 Análisis morfológico 31
4.1.2 Análisis sintáctico 32
4.1.3 Análisis semántico 34
5 EXTE 36
5.1 Introducción 36
5.2 Planteamiento del sistema 37
5.3 Análisis del corpus 39
5.3.1 Clasificación de expresiones temporales 40
5.4 Estadísticas 43
5.5 Gramáticas 46
6 Desarrollo 52
6.1.1 Planteamiento y requerimientos 52
6.1.2 Implementación 52
6.1.3 Integración 61
6.1.4 Pruebas 62
6.2 Complicaciones 63
6.3 Cómo funciona 66
7 Resultados 70
Conclusiones 75

Aportaciones 76
Bibliografía 77
Apéndices 78
A.1 Gramática de TIEMPO 78
A.2 Gramática de Saquete 80

Aclaraciones sobre la tipografía utilizada

Existen dos versiones de este documento. Una electrónica y otra impresa. Para ambas versiones se describe a continuación la tipografía utilizada.

Las palabras escritas en *itálicas* indican que se quiere dar cierto énfasis a dicha palabra o que esa palabra no pertenece al idioma español. Por ejemplo: "La medida *precision* es mayor en este trabajo." También se utiliza para términos coloquiales.

El texto escrito en `monospace` indica que se trata de un ejemplo o de algo referente a un ejemplo: "`Los primeros tres días de enero.`"

Cuando a un ejemplo le anteceda un asterisco (*), se trata de un ejemplo *incorrecto* o que tiene una falla de alguna índole. Por ejemplo: "`* La cartas que se escribirán.`"

El texto escrito con esta tipografía quiere decir que se tiene una referencia dentro de este documento. Por ejemplo: "En la Tabla 5.2 se muestra dicha relación." En la versión electrónica, esto representa una liga que al presionar sobre ella, redireccionará al lector a la ubicación a la que hace referencia.

El texto escrito en **negritas** o subrayado indica que se trata de algo de mayor importancia. Por ejemplo: "Se intenta responder la pregunta **cuándo**."

El texto escrito con ***esta tipografía*** indica que es una referencia bibliográfica contenida al final del documento. Por ejemplo: "(***Saquete, 2005***)". En la versión electrónica, esto representa una liga.

Más información sobre este trabajo

La documentación, el código fuente y manuales referentes a este trabajo se encuentran hospedados en la dirección electrónica http://sourceforge.net/projects/exte. El lector puede descargar la versión correspondiente a su sistema operativo y agregar preguntas a los foros disponibles. Además, cualquier duda puede enviarse al correo electrónico balpo@gmx.net y con gusto será respondida.

1 Introducción

El lenguaje natural (o lenguaje humano) puede parecernos fácil, ya que desde niños aprendemos a utilizarlo, hablarlo e incluso a construir frases que nadie nos *enseñó*. Con el rápido desarrollo de las computadoras, los científicos creyeron que hacer que una computadora entendiera el lenguaje natural sería una tarea igualmente fácil; a fin de cuentas, una computadora puede procesar palabras, números y datos en general mucho más rápido que un ser humano. Poco tiempo después de que los primeros científicos, pioneros en la lingüística computacional, comenzaran la ejecución de esta nueva meta para las computadoras, se dieron cuenta de que no era una tarea fácil. Esta complejidad real en el lenguaje humano crearía una de las ramas de la inteligencia artificial: la lingüística computacional. La lingüística computacional es la rama encargada de formalizar, estudiar y analizar el lenguaje natural para implantarlo en las computadoras.

La generación y la comprensión del lenguaje natural, constituye el estudio y trabajo de muchas personas en todo el planeta, para todos los lenguajes. No es hasta que se profundiza en un tema, que nos damos cuenta de lo complicado –o simple— que éste es. Podemos enseñarle a un niño de tres años lo que 'hoy' quiere decir, quizá, sin muchas dificultades; pero ¿es igualmente fácil "enseñárselo" a una computadora? ¿Podemos, pues, escindir nuestro lenguaje y categorizar estas divisiones de su forma para su estudio? La transmisión de conocimientos a la computadora sin duda ha representado un gran reto y el desarrollo de la Lingüística Computacional y la Inteligencia Artificial[1], ya que luego de varias décadas de estudio en el tema, la Lingüística Computacional tiene un largo camino por recorrer. No debería parecernos raro: ser inteligente requiere millones de años de evolución. Tras darnos cuenta de lo complicado que pueda ser que una computadora entienda el lenguaje natural, sabremos por qué la lingüística computacional estudia el lenguaje natural de forma fragmentada; esto podría ser, por ejemplo, analizar la construcción de frases en español utilizando los verbos *ser* y *estar* en presente del indicativo o estudiar cómo se hace referencia al tiempo en el español.

Entender una frase en español cuando se trata de nuestra lengua materna, no es complicado. Incluso si la gramática del español no se ha estudiado a profundidad, se sabe cuándo una frase está bien o mal formada. Además de que la frase misma, y el contexto de lo que ya se habló o escribió, pueden darnos la información completa. Sin embargo, para la computadora no es así. Podría decirse que la

[1] En la década de los 50, científicos estadounidenses crearon programas para traducir textos automáticamente del ruso al inglés, iniciándose así la Lingüística Computacional. Actualmente se considera a la Lingüística Computacional como una rama de la Inteligencia Artificial.

lengua materna de una computadora es una serie de instrucciones precisas y limitadas, lo que difiere mucho de un lenguaje común. Actualmente las computadoras son muy utilizadas para procesar textos, sin implicar que la computadora los entienda.

Si leemos la siguiente frase

```
El buzo arrojó el tanque al agua, contento.
```

Ejemplo 1.1

Podemos saber, tras hacer un pequeño análisis, qué sucedió; sabemos que los buzos pueden utilizar tanques de oxígeno para bucear, que siempre que se bucea, se hace en agua. También sabemos que quien está contento es el buzo y no el tanque.

La computadora puede tomar estas palabras por separado fácilmente; es sólo cuestión de indicarle que las palabras estarán separadas por un espacio en blanco o por un signo de puntuación. Lo que no ha sido fácil, es indicarle a la computadora los hechos que nosotros como seres humanos sabemos: que se trata de un tanque de oxígeno, y no, por ejemplo, un tanque militar; que quien contento está, es el buzo y no el tanque, etc. Transmitir esos conocimientos a la computadora representa un reto. Para ello, en la lingüística computacional se han considerado las mismas fases de análisis de la lingüística general: fonología, morfología, sintaxis, semántica, pragmática, etc. En todas estas fases se llevan a cabo investigaciones que permitan, en un futuro, lograr que un sistema computacional entienda el lenguaje natural.

Esto abarca una infinitesimal parte del área de la Lingüística Computacional, aunque afortunadamente, juntando el trabajo de muchas otras personas, podemos decir que se avanza a paso seguro.

Definición del problema

El trabajo que aquí se presenta consiste en el marcado de expresiones temporales en textos en español en su variante mexicana, mediante la creación de una gramática y el desarrollo de un sistema computacional llamado **EXTE**.

El marcado de textos, sirve para eliminar la ambigüedad de un texto o para reconocer de forma más rápida algún componente. Así, se puede saber de qué tipo de componente gramatical se trata luego de haber realizado un análisis de una porción del texto.

Macedonio necesita ayuda $_{s}$.

Ejemplo 1.2

¿Ayuda $_{v}$ Macedonio?

Ejemplo 1.3

En este caso, la palabra *ayuda* toma dos significados según la oración. En la primera (Ejemplo 1.2), es un sustantivo, de ahí el subíndice **s**; en la segunda (Ejemplo 1.3) toma la forma de verbo, y por eso el subíndice **v**. Una vez marcada la palabra *ayuda* en el texto, puede obtenerse mayor información del contexto, además de saber qué tipo de componente es. El marcado se realiza en cada uno de los niveles morfológico, sintáctico y semántico, como se verá más adelante. Las colecciones de textos con marcas son herramientas útiles principalmente para los investigadores, como ejemplos del lenguaje, ejemplos para métodos de aprendizaje, o recursos para aplicaciones de procesamiento del lenguaje natural.

El marcado (o etiquetado[2]) de textos requiere de varias técnicas, lo cual se verá a lo largo de los siguientes capítulos. En este documento se muestran algunas de esas técnicas, abarcando una pequeña parte de la gran gama de elementos a marcar dentro de un texto: las expresiones temporales.

Y ahora surgirá la pregunta ¿por qué marcar expresiones temporales? Las expresiones temporales son referencias al tiempo que podemos encontrar en cualquier texto o incluso en cualquier conversación. Marcar las expresiones temporales ayuda en diversas tareas: entrenamiento de métodos de aprendizaje, extracción de información, responder preguntas, etc. En un sistema de pregunta-respuesta podría preguntarse:

¿Cuándo se inició la independencia de México?

Ejemplo 1.4

Además de relacionar *cuándo* con una respuesta de tiempo, es necesario saber que:

---

[2] En este documento se emplea más el término *marcado*. En el ámbito de la lingüística computacional es más utilizado el término *etiquetado*.

En 1911

Ejemplo 1.5

Corresponde a una fecha. Otro tipo de fecha que podría encontrarse sería:

Hace casi doscientos años

Ejemplo 1.6

Objetivo general

El fin de este trabajo es analizar cómo se expresan las frases temporales en textos en español, crear una gramática para ellas y desarrollar un programa que marque las palabras que conforman las expresiones temporales en textos en español en su variante mexicana.

Estructura del documento

Este documento se organiza de la siguiente forma: en lo que resta de este capítulo se da una idea global del problema a resolver (1.1 Computadoras y el lenguaje natural /1.2 Lingüística Computacional / 1.3 El Tiempo En El Lenguaje /1.4 Expresiones Temporales) y una descripción general del *Estado del Arte* (1.5 Trabajos en el tema de expresiones temporales). En el siguiente capítulo, 2 Formalización del Lenguaje Natural, se describe la complejidad del lenguaje natural, la forma en la que el ser humano lo entiende y los problemas que se presentan si se desea que la computadora lo *manipule*. También se indican las técnicas que utiliza la lingüística computacional para lograr esta manipulación. En el capítulo 3, Lingüística de Corpus, se describen las características de los corpus como herramientas de aprendizaje del lenguaje natural, en este trabajo se emplean como recurso inicial y como destinatario del resultado que será la anotación de expresiones temporales. En el capítulo cuarto, Procesamiento del Lenguaje Natural, se detallan los pasos de análisis del lenguaje natural para su procesamiento y estudio considerados en este trabajo con respecto a las etapas mencionadas en el capítulo 2. A lo largo del capítulo quinto, EXTE, se describen los pasos para la creación del sistema y en qué consiste cada uno. En el sexto capítulo, Desarrollo, se muestra a detalle cómo se creó el sistema y las complicaciones que se encontraron, además de señalar las herramientas utilizadas. El séptimo capítulo contiene los resultados generales en términos de eficiencia y su comparación con un sistema similar; también contiene las conclusiones sobre el trabajo. Los apéndices (A.1 Gramática

de TIEMPO) y (A.2 Gramática de Saquete) muestran las gramáticas utilizadas por este sistema y por el sistema considerado como referencia, respectivamente.

1.1 El lenguaje natural

La computadora es actualmente una de las herramientas más usadas en todo el mundo. Mientras que hace aproximadamente 25 años se tenían unas cuantas decenas de computadoras en todo el mundo, actualmente se tiene la posibilidad de tener una en casa. Es bien sabido que la evolución de las computadoras se da extremadamente rápido, pero esta evolución es únicamente técnica; es decir, a lo largo de las últimas cinco décadas, se ha reducido el tamaño de una computadora y aumentado su eficiencia notablemente. Antes, las computadoras ocupaban espacios muy grandes, debían estar en lugares especialmente designados y ser manejadas por gente entrenada; hoy en día, podemos tomarnos un café fuera de casa mientras trabaja la computadora para nosotros buscando información en Internet, por ejemplo.

Por supuesto que esta evolución ha sido benéfica para la sociedad, pero hace falta una evolución más *completa*, en la que se pueda decir que las computadoras no sólo son más pequeñas y rápidas, sino que también nos pueden ayudar a hacer más tareas más difíciles. Las ciencias de la computación, la física y la ingeniería han trabajado conjuntamente para hacer esto posible, no obstante, se está aún muy lejos de poder afirmar que las computadoras son inteligentes. No es la intención decir que esto sea imposible, sólo que es muy difícil y llevará mucho tiempo lograrlo. Precisamente uno de los campos en los que se refleja la verdadera evolución de la computación, es en la lingüística computacional, y más ampliamente, en la inteligencia artificial.

Ahora bien, el *lenguaje natural* es el que se refiere al lenguaje del ser humano (español, ruso, japonés, etc.). Este término es opuesto al *lenguaje formal* o estructurado, que (aunque también creado por el humano) tiene un fin específico y único (un lenguaje de programación).

Existen diferencias entre estos tipos de lenguajes. En el lenguaje formal, el significado de una frase o cadena, está definido únicamente por su forma[1], es decir, por los caracteres que conforman cada cadena; mientras que en el lenguaje natural interviene la semántica y el significado según el contexto. Estas sutiles pero grandes diferencias hacen del lenguaje natural un complejísimo universo de posibilidades. El lenguaje natural es capaz de expresar —por su propia naturaleza— cualquier significado (aunque pudieran requerirse un gran número de palabras); mientras que el lenguaje estructurado

[1] Formalmente, pues los lenguajes formales también adquieren diferentes significados semánticos según los tipos.

está abruptamente limitado por reglas estrictas y construcciones no ambiguas. Se ilustrará esto con el siguiente ejemplo.

En un lenguaje de programación (lenguaje formal), podemos escribir la instrucción

```
Comida miComida = new Comida();
```

Ejemplo 1.7

Y significaría que se desea crear una nueva entidad lógica de tipo `Comida` con nombre `miComida`. Lamentablemente, si por error escribiéramos

```
* Comida miComida = new Comida()
```

Ejemplo 1.8

Obtendríamos un error, ya que hace falta el punto y coma `';'` al final de la instrucción y no se obtendría algún resultado.

En cambio en el lenguaje natural podemos decirle algo a María de diferentes formas, y María (sin importarnos cómo) desempeñaría distintas labores según la instrucción, sin necesidad de la presencia o la ausencia de signos y formas fijas.

Si le dijéramos, por ejemplo

```
María, ¿podrías contestar el telegrama?
```

Ejemplo 1.9

María de igual forma lo contestaría si la pregunta (escrita) fuera:

```
* Maria podrias contestar el telegrama?
```

Ejemplo 1.10

O bien,

```
* Podrías María contestar el telegrama?
```

Ejemplo 1.11

Sin importar los tres errores del Ejemplo 1.10, o la variante del Ejemplo 1.11, María sabría que se quiere que conteste el telegrama.

En muchos lenguajes de programación, el signo de puntuación punto-y-coma es utilizado como *fin de instrucción*, y es indispensable, mientras que en el lenguaje natural no lo es, ya que los humanos entendemos oraciones incluso sin que sean gramaticales.

Las diferencias entre el lenguaje natural y el lenguaje formal crean un enfoque muy distinto sobre cómo se manejará cada uno dentro de la computadora. De ningún modo se intenta hacer pensar que el lenguaje formal sea fácil de manejar, es solamente que comparándolo con el lenguaje natural debería parecernos más fácil.

Los científicos que en la década de los cincuenta querían traducir textos de un lenguaje natural a otro, se dieron cuenta de que no era tarea fácil. En un lenguaje formal, digamos uno de programación, era sencillo decir "error, esto no lo reconozco" y detener la ejecución; en cambio, en un lenguaje natural no existen esas reglas limitadas y precisas por no cumplir la sintaxis estricta. Tal es el caso del Ejemplo 1.7 y del Ejemplo 1.8: podemos tener diferencias en sólo un signo, y tener significados completamente distintos o iguales, sin la posibilidad de contestar "error, esto no lo reconozco" puesto que ambas frases deben ser reconocidas por ser completamente válidas.

Puede ser la computadora una herramienta ideal para facilitar el análisis del lenguaje natural, pero actualmente sólo es eficiente al utilizar lenguajes formales que no describen completamente un lenguaje natural.

1.2 Lingüística computacional

La lingüística computacional es un campo interdisciplinario encargado de modelar el lenguaje natural desde una perspectiva computacional. En un principio lo que se deseaba era traducir automáticamente textos de un idioma a otro. Los científicos, entusiasmados por los resultados exitosos que las computadoras habían mostrado para realizar cálculos aritméticos, se dedicaron a hacer programas automatizados para realizar traducciones. Poco tiempo después se dieron cuenta de que el problema era mucho más complejo de lo que aparentaba, ya que pasar de una oración en un lenguaje tan reducido como: 4 + 4 = 8, a una oración en uno tan vasto como:

El resultado de sumar dos veces cuatro es igual a ocho

Ejemplo 1.12

O tal vez

Ocho es el resultado de sumar el número cuatro un par de veces

Ejemplo 1.13

Era un paso ambicioso y muy grande. Concretamente, estos tres ejemplos denotan lo mismo, y quizá podríamos encontrar muchas otras maneras de decirlo. La principal diferencia entre el primer lenguaje, un lenguaje estructurado, y el segundo, un lenguaje natural, es el poder para expresar ideas o conceptos.

La lingüística computacional analiza en los mismos niveles de la lingüística general el lenguaje natural para intentar formalizarlo y representarlo. El principal obstáculo es la ambigüedad en todos los niveles de análisis. Ciertas frases pueden significar *cosas distintas* según el contexto. Por ejemplo,

Macedonio *salió disparado* a comprar la medicina.

Ejemplo 1.14

Y

Al momento del choque Macedonio *salió disparado* por el parabrisas.

Ejemplo 1.15

Otro ejemplo de ambigüedad (en el tiempo) sería:

En la lista de los cincuenta Macedonio aparecía como el mejor.

Ejemplo 1.16

La lingüística computacional estudia métodos para reducir dicha ambigüedad, por ejemplo haciendo una selección de lo que es correcto según el contexto, esta selección es una toma de decisión que debe realizar la computadora por sí sola mediante algunos métodos. Es necesario aclarar que la reducción o eliminación de la ambigüedad no es de lo único que se encarga la lingüística computacional.

Existen diferentes áreas de estudio para analizar y describir el lenguaje natural dentro de la lingüística computacional. Entre esas áreas se encuentra la lingüística de corpus, y un caso concreto es el marcado de corpus.

La lingüística de corpus se encarga de estudiar el lenguaje natural por medio de muestras de textos llamados *Corpus*. Básicamente intenta generar un conjunto de reglas abstractas que describan la estructura general del lenguaje que se esté estudiando, para este caso el español. Como se verá más adelante, el uso de corpus[1], y en general la lingüística de corpus, han sido muy criticados desde sus inicios; aunque con el avance de las computadoras y su facilidad para procesar, ha ganado terreno entre los estudiosos de la lingüística computacional.

[1] El singular y el plural para *Corpus* se escriben igual.

1.3 El tiempo en el lenguaje

El lenguaje y el tiempo, ambos por sí solos, encierran conceptos amplios y no fáciles de explicar. No se intenta profundizar en cada uno. Aprender un lenguaje desde niños no nos parece complicado. Lo arduo realmente es estudiarlo, entenderlo. Es hasta la primaria que vemos el *gerundio*[1], pero muchos años antes de estudiarlo, lo utilizábamos sin preguntarnos nada acerca de él. El tiempo —dentro y fuera de la lingüística— es algo todavía más complejo, y realmente es difícil estudiarlo a profundidad, no obstante lo utilizamos incondicionalmente siempre (solamente en este párrafo hay cinco expresiones de tiempo).

Es increíble el número de incidencias de referencias al tiempo que pueden encontrarse en unas cuantas líneas de texto (o en unas cuantas frases pronunciadas). Para este trabajo se usaron textos de periódicos mexicanos, sin embargo, es posible afirmar que léxicamente las apariciones de expresiones temporales en textos del mismo género, que no sean periódicos, serán –si no iguales— muy similares.

Cabe mencionar que el tiempo puede encontrarse no sólo en una fecha, sino, por ejemplo, en un verbo conjugado. El español es una lengua complicada, aunque no nos parezca. Las voces verbales y los tiempos en los cuales un verbo puede ser conjugado, la libertad para concatenar elementos gramaticales, etc. lo hacen difícil. Cada verbo conjugado hace referencia al tiempo.

Consideremos el siguiente ejemplo:

La iglesia quedó destruida porque los aviones bombardearon sin compasión sobre el pueblo.

Ejemplo 1.17

Podemos comprender que la iglesia ya fue destruida. Eso porque, antes en el tiempo (tal vez ayer o hace 50 años), aviones bombardearon sin compasión sobre el pueblo, pero éstos ya no lo están haciendo.

[1] El gerundio es una forma invariable no personal del verbo, posee un matiz de adverbio, aunque también es usado en perífrasis verbales, como adjetivo, y con valores causal, condicional, concesivo y copulativo.

Este ejemplo no nos indica si la iglesia sigue existiendo, si ya fue reconstruida o si fue hace unos minutos el bombardeo. Dependiendo del contexto en el que se lea esto, supondremos que la iglesia sigue existiendo o no. No obstante, puede haber expresiones temporales mucho más detalladas y explícitas sobre cuándo ocurrió el hecho en cuestión.

Aquel viernes 5 de enero de 1945 la iglesia quedó destruida tras el bombardeo realizado por los alemanes desde el lunes anterior. El gobierno francés ha venido reconstruyendo la iglesia hasta la fecha. Sin embargo, dado que el costo de la restauración ha sido muy elevado, se está planeando su demolición, la cual podría llevarse a cabo el próximo año o hasta finales de 1947.

Ejemplo 1.18

En este otro ejemplo ya sabemos cuándo fue bombardeada la iglesia. También sabemos desde cuándo se hizo el bombardeo; que hasta la fecha en la que se escribió el texto, seguía existiendo la iglesia. Nos dice que actualmente –en aquel presente— se planea su demolición para un futuro próximo a la fecha en la que ese texto fue escrito originalmente. Es probable que la iglesia ya no exista, o que nunca se haya destruido y siga en pie, pero ya tenemos más información para responder una posible pregunta que incluyera **cuándo**. Existen muchas frases que nos pueden informar sobre el tiempo en el que un evento sucedió, por ejemplo, el martes 18 de noviembre, el próximo año, hasta finales de 1949, etc. Y son este tipo de frases las que se desean reconocer, entender.

Cada frase hablada o escrita en lenguaje natural tiene como intención comunicar algo. Si carece de sentido o de significado, no tiene caso decirla o escribirla. Hay enunciados que informan la manera en la que se realizó una acción, otros que informan sobre la acción misma, y los que nos interesan para este trabajo: aquéllos que comunican cuándo pasó dicha acción. Sin importarnos de qué acción se está hablando o escribiendo, siempre podremos hacernos la pregunta *cuándo* o *desde cuándo*. En esto consiste este trabajo, en realizar métodos computacionales que determinen tiempos. La aplicación directa es poder responder a esa pregunta en textos en español.

1.4 Expresiones Temporales

Es muy común que en un texto encontremos expresiones temporales. Como se mencionó brevemente en la introducción, una expresión temporal es un enunciado, o una parte de uno, que hace referencia al tiempo. Esta referencia al tiempo puede ser implícita o explícita; puede ser una fecha, una hora o una duración. Una expresión temporal informará:

- *Cuándo* sucede algo
- *Cuánto* dura
- *Cuán* a menudo sucede

Las expresiones temporales explícitas son aquellas que por sí solas nos informan sobre ese punto en el tiempo:

Ayer votaron los mexicanos.

Ejemplo 1.19

Nos indica cuándo fue que los mexicanos votaron, y aunque no tenemos una hora o fecha puntual, sabemos que un día antes de la escritura de ese texto los mexicanos realizaron su voto.

El domingo 2 de julio votaron los mexicanos

Ejemplo 1.20

Indica la fecha en la que sucedió el evento; no obstante, a pesar de ser más explícita, no nos da *toda* la información.

El domingo 2 de julio de 2006, entre las ocho de la mañana y las seis de la tarde, votaron los mexicanos.

Ejemplo 1.21

En este último ejemplo tenemos toda la información temporal sobre el evento sin tener que crear una referencia a la fecha en la que se escribió el texto.

En este trabajo se contempló el marcado de expresiones temporales sólo de forma léxica; y queda para un trabajo futuro el marcado sintáctico y semántico, de tal modo que:

El domingo 2 de julio de 2046, entre las seis de la tarde y las ocho de la mañana votaron los mexicanos.

Ejemplo 1.22

no tendría sentido semánticamente, pues el año 2046 no ha llegado y el verbo votar está conjugado en pasado simple del indicativo; además de que no es usual votar entre las seis de la tarde y las ocho de la mañana, sería como retroceder en el tiempo. Sin embargo sintácticamente la frase no está mal construida.

Técnicas avanzadas de marcado de expresiones temporales crean una referencia a partir de un punto en el tiempo. Si el Ejemplo 1.18 hubiera estado escrito en un periódico, se tendría la fecha de publicación como fecha base o punto de referencia en el tiempo. De este modo, la información extraída de la expresión temporal como en el Ejemplo 1.19 podría ser más completa.

Las expresiones temporales implícitas no indican con precisión un punto o rango en el tiempo. Generalmente las encontraremos en la conjugación del verbo. En el ejemplo siguiente,

¡Rompiste el plato!

Ejemplo 1.23

Lo único que podemos saber es que el plato está roto. La expresión temporal en este último ejemplo es implícita, pues es el verbo quien la contiene al estar conjugado en pasado simple.

Con tanta frecuencia encontramos expresiones temporales en textos, que muchas veces las pasamos por alto. Sería extraño leer un texto que no las contuviera. Es por eso que se considera importante su marcado, bien que las expresiones temporales representan una muy pequeña parte de lo que cualquier texto pudiera contener, su estudio y manejo ayudan en parte a la comprensión del lenguaje natural en la lingüística computacional.

1.5 Trabajos en el tema de expresiones temporales

Las distintas formas en las que el tiempo se expresa en los lenguajes naturales plantea desafíos interesantes para su representación como conocimiento y para el razonamiento en general. Pero antes de intentar trabajar en esta problemática es necesario resolver el problema inicial de reconocimiento automático de referencias del tiempo en lenguaje natural.

En las Conferencias MUC (*Message Understanding Conference*), iniciadas para desarrollar métodos de extracción de información y realizadas entre los años 1987 y 1997, se incorporó la tarea denominada *Reconocimiento de entidades con nombre* (NER: *Named Entity Recognition*) en la sexta y séptima conferencias. Esta tarea consistió a su vez de tres subtareas de reconocimiento: nombres de entidades, expresiones temporales, y expresiones de número ***(Chinchor, 1997)***. Las expresiones temporales en esta subtarea se refieren solamente a los tipos *absoluto* y *relativo*. En el tipo absoluto la expresión debe indicar un segmento específico del tiempo, por ejemplo: "las doce del mediodía". En el tipo relativo las frases temporales indican una fecha concerniente a la fecha del documento o una porción de una unidad temporal concerniente a la unidad temporal dada, por ejemplo: "ayer", "en la mañana". El marcado de estas frases no indica a qué tipo pertenecen, solamente se indica una categoría TIEMPO que define una unidad temporal más corta que un día completo (por ejemplo: las 16:00 horas) y otra categoría FECHA que es una unidad temporal de un día completo o más (por ejemplo: 5 de mayo).

Diversos sistemas fueron evaluados como parte de la conferencia MUC6[1] en 1995. La lengua objetivo era el inglés. Los sistemas que participaron funcionaron bien en general, aunque muchos de ellos utilizaron recursos específicos de la lengua para realizar la tarea de reconocimiento y por lo tanto se desconocía si podrían funcionar en otra lengua diferente al inglés. En la conferencia MUC7[2] se contemplaron diferentes lenguas: inglés, chino, japonés y español. Sin embargo las frases temporales solamente correspondieron entre el 20% y el 25% del total de las frases consideradas para la prueba completa de entidades. Estas conferencias trataron el problema de una manera limitada que se ha ampliado recientemente.

[1] http://www.cs.nyu.edu/cs/faculty/grishman/muc6.html
[2] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html

En las conferencias CoNLL (*Conference on Computational Natural Language Learning*) en el año 2002 se realizó el reconocimiento de entidades con nombre de forma independiente al lenguaje. En la tarea compartida, doce diversos sistemas basados en aprendizaje fueron aplicados a los datos en español y holandés. La tarea se limitó a las entidades con nombre, a diferencia de las Conferencias MUC, se eliminaron las expresiones temporales y las expresiones de número.

Otros trabajos que van más adelante del reconocimiento automático consideran temas como esquemas de anotación, límites de las expresiones temporales, granularidad en la clasificación de expresiones temporales, interpretación de expresiones temporales, etc. En contraste con los esquemas temporales de anotación, por ejemplo (MUC-7, 1998) existen trabajos como el de (Mani & Wilson, 2000) para el reconocimiento, así como la resolución de referencias temporales, empleando esquemas de granularidad en expresiones temporales. Las expresiones temporales reflejan dos aspectos bien conocidos de granularidad. El primer aspecto tiene que ver con los límites vagos, por ejemplo la mañana del sábado, ¿incluye las 2 a.m. del sábado? Un segundo aspecto de granularidad es el hecho de que puede haber diversos niveles de enfoque. El enfoque implica una cierta noción de aproximación, es decir, movimiento en una escala de *más exacto* (o concreto o específico) a *menos preciso* (abstracto o general), o viceversa. Por ejemplo, un día se puede dividir en 24 horas, una hora en sesenta minutos, etc. Asimismo, los intervalos pueden separarse en instantes.

Otros trabajos importantes se han desarrollado en el campo de los sistemas de pregunta-respuesta. Por ejemplo, TimeML (Pustejovsky, et al., 2003) es un lenguaje de especificación para eventos y expresiones temporales en textos en lenguaje natural, desarrollado en el contexto del programa AQUAINT para sistemas de pregunta-respuesta. Diferente a la mayoría de los trabajos previos sobre anotación de eventos, TimeML captura tres fenómenos distintos en marcado temporal: (1) ancla sistemáticamente predicados de eventos a una amplia gama de expresiones denotando temporalidad; (2) ordena expresiones de evento en texto, concerniente tanto a oraciones como a discurso; y (3) permite una interpretación retrasada (sin especificar) de expresiones temporales parcialmente resueltas.

Pero los trabajos relativos a granularidad de marcado, eventos e interpretación son temas avanzados, actualmente considerados en tesis doctorales. Retomando la anotación o marcación de expresiones temporales, para el español, el trabajo más amplio y con mayor documentación es el trabajo de E. Saquete et al. ***(Saquete & Martínez-Barco, 2000) (Saquete, Martínez-Barco, & Muñoz, 2002) (Saquete,***

Martínez-Barco, & Muñoz, 2002b), razón por la cual se hace referencia a él en varias secciones. El trabajo realizado en esta tesis también está dedicado al español pero considerando materiales en la variante mexicana. La justificación de obtener nuevamente una gramática es precisamente comparar las diferencias, puesto que existen frases en español peninsular como `las diez menos cinco`, `a fines del viernes`, `más tarde el viernes`, que no se utilizan frecuentemente en México. Obtener una gramática para expresiones temporales considerando periódicos mexicanos de circulación nacional permitirá marcar frases en textos mexicanos y su uso en sistemas de procesamiento de lenguaje natural que consideren este tipo de textos.

2 Formalización del lenguaje natural

Como ya se mencionó en la introducción (1.1), el lenguaje natural es el lenguaje de comunicación entre seres humanos: español, francés, alemán, etc. El lenguaje natural lleva consigo una evolución y todo un árbol genealógico que siguen familias de lenguas antiguas[1]. El español está clasificado en la familia de las lenguas romances, las cuales provienen del latín que a su vez está situado en la familia indoeuropea.

Es entonces la evolución el elemento que hace del lenguaje natural algo tan vasto, ya que tiene como ascendencia otras lenguas. El lenguaje formal o estructurado implica un desarrollo mecánico, llevado y pensado por el hombre con intenciones claras, bien delimitadas y específicas y con un objetivo en particular; el lenguaje natural no. No se puede decir quién "inventó" el gerundio, pero sí se puede decir cuál fue el primer lenguaje de programación que utilizó la sintaxis if-then-else.

En las lenguas podemos encontrar elementos en común (sujetos, verbos, adjetivos, etc.) que van definiendo su estructura para su composición o estudio. También es importante marcar los propósitos de un lenguaje natural y uno formal. El primero evoluciona con la intención de comunicar personas; el segundo nace como vía para comunicar a la computadora lo que debe hacer.

El estudio del lenguaje natural dentro de la computación lleva aproximadamente sesenta años en curso. En la década de los años cincuenta, Avram Noam Chomsky ***(Chomsky, 1956)*** propuso incorporar los lenguajes naturales al tipo de lenguajes que pueden ser estudiados por sistemas formales. Propuso que la gramática de un lenguaje se viera como la teoría de la estructura de ese lenguaje. Dado que toda teoría científica está basada en un número finito de observaciones (estableciendo leyes en términos de ciertas construcciones hipotéticas) se intenta mostrar que dichas observaciones están relacionadas y son capaces de predecir un número indefinido de fenómenos[2]. Una gramática está basada en un número finito de enunciados observados (corpus) y proyecta este conjunto en un número infinito de oraciones gramaticales tras establecer leyes generales (reglas gramaticales) enmarcadas en dichas construcciones hipotéticas del lenguaje analizado. De tal forma que una gramática correctamente formulada debería distinguir sin ambigüedad el conjunto de oraciones gramaticales.

---

[1] Obtenido de la dirección electrónica http://es.wikipedia.org/wiki/Lenguas_indoeuropeas

[2] En este caso, los fenómenos son descritos como oraciones o frases que pueden construirse a partir de la gramática que defina un lenguaje.

2.1 ¿Qué se necesita para entender el lenguaje natural?

En pocas palabras, lo que realmente se necesita para **entender** el lenguaje natural es muchos años de evolución. Los niños, incluso antes de aprender a hablar, saben el significado de *mamá*, *hambre* o *sueño*, porque son agentes —externos o internos— que interactúan con ellos: los ven, los sienten, los sufren. Aproximadamente entre los 16 meses y los seis años de edad, los niños adquieren sus primeras palabras, su cerebro crece y crea conexiones neuronales, fomentando así el aprendizaje y el uso del habla, y desarrollan la capacidad de crear frases nuevas ***(Davidoff, 1989)***; es decir, asocian a la persona que está todo el tiempo con ellos con la palabra *mamá* y no al revés. Durante mucho tiempo antes han vivido en carne propia *lo que es* mamá, lo único que les faltaba era decirlo, asociarlo a la palabra y **no** a la letra *'m'* y la letra *'a'* y de nuevo la letra *'m'* y por último la *'á'*. Como podemos ver, la manera en la que el ser humano aborda el aprendizaje es mediante el uso de los cinco sentidos. ¿Cómo va a entender la computadora el lenguaje natural si no tiene sentidos? Las computadoras no interactúan con el medio por sí solas, es necesario "meter" información usando un dispositivo de entrada, como el teclado.

El entendimiento verdadero del lenguaje involucra muchos niveles de conocimiento y cada uno de estos niveles ***(Wintner, 2002)*** es una etapa en la lingüística general ***(Hausser, 1999)***:

Fonología y fonética: La fonología es la ciencia que estudia los sonidos del lenguaje. La fonética se encarga de estudiar los sonidos y la voz humanos, la manera en la que se articula al pronunciar, la acústica y el proceso auditivo del habla. La fonología estudia el sistema de sonidos de un lenguaje específico. Ambos son necesarios para que un ser humano adopte el habla y su comprensión. Los sonidos básicos de un lenguaje son conocidos como *fonemas*, que son las unidades simples del nivel fónico del lenguaje humano. Las palabras que sólo difieren en un sonido y tienen significados diferentes, son denominadas *pares mínimos*.

Casa

Ejemplo 2.1

Tasa

Ejemplo 2.2

En el español de México tenemos un ejemplo claro (y muy bonito) de cómo una letra, en este caso la x, puede tener distintos sonidos como en las palabras `México`, `Xochimilco`, `axioma` y `mixiote`.

Morfología: Estudia la estructura de las palabras, clasifica las palabras de un lenguaje con respecto al habla. Define sufijos, prefijos e infijos para dar forma a la palabra como unidad básica. Como seres humanos, sabemos que `perro`, `perrito` y `perros` se refieren a *lo mismo*, pero con diferentes formas que contienen características morfosintácticas como número y género. El número y el género juegan un papel importante para el reconocimiento de expresiones dentro del lenguaje natural, ya que también definen si la expresión es válida o no.

Sintaxis: Es el estudio de las reglas con las que se rige un lenguaje para crear frases. Así como la morfología, la sintaxis divide la frase en unidades más básicas: sustantivo, verbo, etc. Y la relación entre ellas se encarga de dar estructura a las frases en términos de reglas gramaticales.

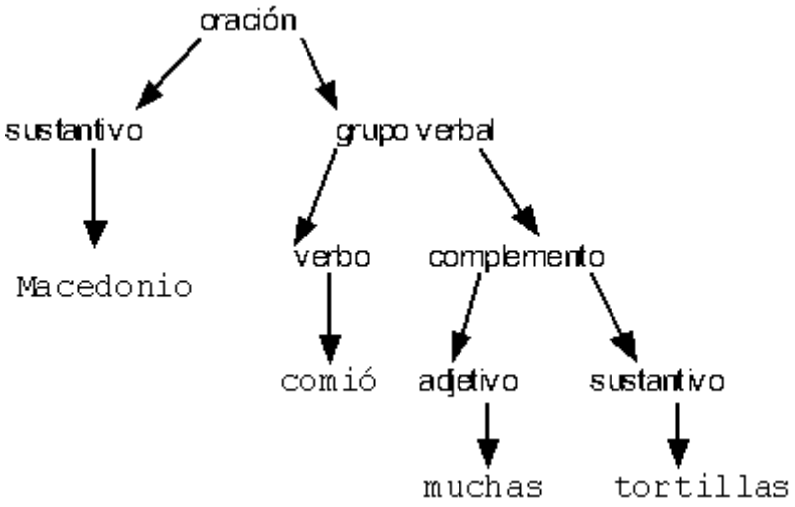

Figura 2.1—Árbol sintáctico.

La gramática crea una relación, entre el conjunto de oraciones observadas (corpus) y el conjunto de oraciones gramaticales (frases en lenguaje natural, ya sea generadas o leídas). Utilizando esta relación se pueden reconocer o generar elementos en lenguaje natural.

`Macedonio salió con María.`

Ejemplo 2.3

```
* María con salió Macedonio.
```

Ejemplo 2.4

De este modo, tras seguir las leyes que gobiernan nuestro lenguaje, veríamos que la frase en el Ejemplo 2.3 debería ser generada o reconocida –no sólo sin ambigüedad sino correctamente—; en cambio aquélla del Ejemplo 2.4 no debería ser reconocida como correcta.

No obstante, al menos para el español, la teoría gramatical tiene que incluir la flexibilidad que el lenguaje natural tiene para generar o reconocer una frase no-ambigua, diferente y correcta.

```
Con María salió Macedonio.
```

Ejemplo 2.5

Esta última frase no es incorrecta[1], sin embargo pudo haberse generado con una regla distinta a la frase del Ejemplo 2.3 y seguir significando casi exactamente lo mismo.

La flexibilidad que da el lenguaje natural para la formación de frases nuevas es enorme. Ayudándose de signos de puntuación, el lenguaje natural puede generar innumerables frases correctas, e incluso varias frases distintas sintácticamente e idénticas semánticamente.

Dado que la comunicación es la principal intención que tiene cualquier lenguaje natural, éstos han evolucionado para adquirir formas, vocablos, expresiones, etc. nuevos o modificar los existentes. Al estudiarlos y entenderlos podemos ver que también tienen una estructura, reglas –a veces poco estrictas— para formar frases válidas; esta estructura interna se conoce como gramática. Por lo regular cualquier lenguaje, natural o formal, tiene una gramática asociada (reglas). En ambos tipos de lenguajes se puede hacer una categorización de los símbolos utilizables y el significado que adquieren según el contexto. Los lenguajes formales, a diferencia de los naturales, tienen menos reglas, menos símbolos y sobre todo menos significados.

Semántica: Trata los significados de las palabras en el lenguaje. Una representación o frase semántica debe ser precisa y no-ambigua. Dentro del lenguaje natural podemos encontrar mucha ambigüedad,

[1] Aunque probablemente no se utilice muy frecuentemente.

en algunas ocasiones el contexto y, otras la fonética y la morfología, eliminan dicha ambigüedad, Ejemplo 1.14 y Ejemplo 1.15.

Pragmática: Existe una diferencia entre lo que se dice y lo que se entiende y de estudiar esto se encarga la pragmática. Quiere decir que el contexto influye en la interpretación del significado. La pragmática toma en consideración los factores extra-lingüísticos, esto es, todos aquellos factores a los que no se hace referencia en el estudio lingüístico.

¿Puedes abrir la puerta?

Ejemplo 2.6

En este ejemplo, quien pregunta puede tener dos intenciones: (a) saber si se tiene la *capacidad* de abrir la puerta o (b) pedir como favor que se abra la puerta.

El lenguaje natural, para su estudio, está dividido en partículas, y el orden de aparición de cada partícula dentro de una frase, modifica su significado. Para este trabajo, en el que se marcan expresiones temporales, son de gran importancia las preposiciones y las palabras con contenido temporal. Muchas de las expresiones temporales que aparecen en los corpus analizados, comienzan con una preposición.

Desde el martes.

Ejemplo 2.7

Y con las preposiciones podemos clasificar las expresiones temporales. En el caso del Ejemplo 2.7, la preposición desde clasificaría a toda la expresión temporal dentro de la categoría con nombre DESDE. También existen expresiones temporales dentro del lenguaje natural que no utilizan preposiciones. Una fecha numérica, por ejemplo, no las contiene.

Jueves 17

Ejemplo 2.8

O expresiones temporales que hacen referencia a una fecha implícita

Mañana

Ejemplo 2.9

Dado que las expresiones temporales aparecen "libremente" en el lenguaje natural, será necesario clasificarlas no sólo por la preposición que contienen, sino por su estructura general. Como se verá en 5.5 Gramáticas, las expresiones temporales que **EXTE** reconoce se marcan y clasifican según su estructura.

3 Lingüística de corpus

La palabra *corpus* deriva del latín, y quiere decir "cuerpo"; dentro del estudio de la lingüística, un corpus se define como: un conjunto de textos, cuya extensión por lo general es vasta, escritos o hablados, que contiene ejemplos reales del uso del lenguaje natural. Actualmente, dentro del campo de la lingüística computacional, la definición adquiere rasgos más refinados, como que cualquier corpus debe ser legible por computadoras. La lingüística de corpus, es la rama de la lingüística que se encarga de analizar y estudiar el lenguaje por medio de colecciones de textos heterogéneos (periódicos, revistas, artículos, etc.) que contienen usos del lenguaje natural. En la lingüística moderna, se tiene una definición aún más detallada ***(McEnery & Wilson, 2006)*** de lo que es un corpus:

Un corpus es una colección de más de un texto que debe cumplir con las siguientes características:

Ser extraíble: comúnmente cuando se analiza un texto, no estamos interesados en uno individualmente o en un solo autor; más bien, en una variedad entera del lenguaje. Para poder extraer una variedad del lenguaje, se puede analizar cada una de las expresiones que existan en la variedad del corpus. Este enfoque se utiliza cuando el lenguaje en cuestión es una lengua muerta y por lo tanto el conjunto de textos es mucho más pequeño. Otro enfoque más eficaz es extraer una porción del total y analizarla. Ése es nuestro caso. No se tomaron en cuenta todos los periódicos del español para generar un analizador que marcara expresiones temporales; ni siquiera se utilizaron todos los periódicos de México.

Chomsky criticó mucho durante los años cincuenta el uso de corpus dentro de la lingüística ***(McEnery & Wilson, 2006)***, diciendo que el lenguaje es infinito, por lo tanto cualquier corpus sería una representación menguada del lenguaje en sí. Esto porque algunas de las expresiones que aparecieran en el corpus serían excluidas por ser raras (en cuestión de aparición), otras más comunes serían omitidas y por último, expresiones extremadamente raras podrían ser incluidas muchas veces. Aunque actualmente con el uso de las computadoras el tratamiento de corpus es muy distinto a aquél de tiempos de Chomsky, se debe considerar que es necesario construir corpus **representativos** y no *prejuiciosos*[1]

[1] En este caso el prejuicio sería cometer exclusiones de partes —reales— del lenguaje porque ciertas expresiones aparecen más que otras. La expresión muchas veces puede despreciarse según el número de incidencias dentro de un texto, entonces el análisis del corpus no sería completo, ya que hay al menos esas expresiones que no están siendo tomadas en cuenta.

que incorporen una parte real del lenguaje natural. Por eso es importante la creación de corpus que sean verdaderamente representativos del lenguaje analizado, que maximicen la precisión de esa visión del lenguaje natural.

Tener representatividad: las críticas de Chomsky sobre el lenguaje natural afectaron no sólo la lingüística, sino a la investigación científica basada en métodos empíricos. En un principio esto preocupó mucho a los estudiosos de corpus, pero se tomaron medidas que garantizaban maximizar la representatividad de los corpus. Es importante recalcar que cuando Chomsky hizo esas aseveraciones, la manipulación de corpus era hecha *a mano*, de tal modo que analizarlos se convertía en una tarea larga y tediosa. Tomando en cuenta esto, puede decirse que, al menos en el tiempo en que no se contaba con computadoras, Chomsky estaba en lo correcto. Afortunadamente hoy día se tienen computadoras extremadamente poderosas, capaces de analizar millones de palabras en cuestión de minutos: el tamaño del corpus ya no importa.

Se ha sugerido también definir los límites del conjunto que se va a estudiar *antes* de definir los procedimientos para representar un texto.

Ser de tamaño finito: el término corpus también implica un tamaño finito, es decir, aunque se tengan diez billones de palabras —y esto represente mucho— es finito. Esta limitante del tamaño no es universal, ya que actualmente existen corpus tipo *monitor*. Este término se refiere a que esa colección de textos está creciendo constantemente, pues se agregan nuevos textos siempre, por ende no son estáticos. Los corpus monitor tienen la ventaja de tener una representatividad muy extensa, pero tienen la desventaja de no ser tan confiables en cuestión de ser cuantitativos, es decir, que no puede definirse la cantidad de textos que contiene.

Ser legible por computadoras: actualmente decir corpus es casi decir legible por computadoras. Será raro encontrar un corpus impreso en libros; claro que antes los corpus eran grandes impresiones de textos y por eso no podían ser leídos por las computadoras.

Ser una referencia estándar: esta característica es más bien una táctica, y consiste en hacer que un corpus (o varios, como es el caso real) sean una referencia estándar para la variedad del lenguaje que representan. Esto presupone que estará disponible para otros investigadores o lingüistas, y aña-

de una "medición" para futuras investigaciones. Estableciendo una metodología clara, nuevos resultados pueden compararse (y publicarse) directamente sin necesidad de reanalizarse.

El análisis de corpus puede categorizarse en dos tipos: cualitativos y cuantitativos. El enfoque del análisis **cuantitativo** es clasificar características, agruparlas y contarlas, e incluso construir modelos estadísticos que intentan explicar los resultados que se obtienen. Esos resultados encontrados pueden generalizarse a una población más grande o compararse con los resultados obtenidos del análisis en otro corpus. En consecuencia, el análisis cuantitativo nos permite descubrir cuáles eventos se dan con más frecuencia y son, entonces, un reflejo del comportamiento del lenguaje, y cuáles son mera coincidencia o extremadamente raros. El análisis **cualitativo** se propone examinar el lenguaje de manera completa y detallada. No intenta asignar frecuencias a las características lingüísticas identificadas en el texto, de tal modo que los eventos raros recibirán la misma atención o importancia que los más frecuentes; las ambigüedades que todo lenguaje natural tiene, pueden reconocerse claramente.

El enfoque que este trabajo sigue es cuantitativo, como se verá en los siguientes capítulos. El corpus utilizado fue una colección de periódicos (El Universal, Excélsior y La Jornada) en formato digital. El tipo de los artículos que se analizó fue muy variado, desde temas políticos hasta temas deportivos y culturales. La variedad de tipos de artículos afectó en las distintas expresiones temporales que se encontraron, aunque también variaban de periódico en periódico, e incluso de año en año; por ejemplo, en Excélsior se utilizan más las referencias a fechas como `en mayo del 95` a diferencia de La Jornada que utiliza fechas como `en mayo de 1995`.

3.1 Características del corpus utilizado

Para la creación de **EXTE**, se utilizó una colección de periódicos mexicanos con las siguientes características:

Periódico	Rango de años	Número de artículos	Tamaño
El Universal	1997 - 1999	952	438 MB
Excélsior	1997 - 2000	1434	609 MB
La Jornada	1997 - 2002	2086	1090 MB
		4472	**2.137 GB**

Tabla 3.1—Detalles sobre el corpus usado.

Los artículos de Excélsior y La Jornada fueron utilizados para el análisis y la creación de la gramática; los artículos de El Universal se utilizaron para realizar las pruebas. Más adelante, en la sección 5.3 Análisis del corpus, se describirá con más detalle la manera en la que se analizó el corpus y en la sección 5.5 Gramáticas, cómo se construyó la gramática que utiliza **EXTE**.

3.2 Marcado del corpus

Para la lingüística computacional, serían más útiles los corpus marcados en las diversas etapas de análisis: morfológico, sintáctico, etc. Sin embargo, la colección de textos utilizada no contiene las marcas de expresiones temporales en ninguno de los niveles de análisis. Marcar las expresiones temporales es el objetivo de este trabajo, crear un programa capaz de determinar las expresiones temporales en los textos del corpus. El marcado de corpus es una tarea que inserta la información dentro de textos. Por ejemplo, un marcado de corpus que considerara identificar los verbos y sus características morfosintácticas, incluiría información extra. Teniendo un corpus que contuviera el texto:

Venid acá, peces, vosotros, los de la margen derecha, que estáis en el río Douro, y vosotros, los de la margen izquierda, que estáis en el río Duero, venid acá todos y decidme cuál es la lengua en que habláis cuando ahí abajo cruzáis las acuáticas aduanas, y si también ahí tenéis pasaportes y sellos para entrar y salir.

Ejemplo 3.1— Fragmento de "Viaje a Portugal", José Saramago, Alfaguara, 1995.

El marcado (morfológico) del corpus se encargaría de que el texto anterior fuera como:

{v:venir:2:p:imp} acá, peces, vosotros, los de la margen derecha, que {v:estar:2:p:ind[pres]} en el río Douro, y vosotros, los de la margen izquierda, que {v:estar:2:p:ind[pres]} en el río Duero, {v:venir:2:p:imp} acá todos y {v:decir:2:p:imp:ref[1:s]} cuál es la lengua en que {v:hablar:2:p:ind[pres]} cuando ahí abajo {v:cruzar:2:p:ind[pres]} las acuáticas aduanas, y si también ahí {v:tener:2:p:ind[pres]} pasaportes y sellos para {v:entrar} y {v:salir}.

Ejemplo 3.2

De tal modo que el corpus mismo contiene la información sobre los verbos, sus tiempos y sus voces. En la primera palabra del Ejemplo 3.1, venid, se hace referencia al verbo venir, en imperativo y en segunda persona del plural, de ahí la notación. El signo ':' sirve solamente como separador. La 'v' indica que es la marca de un verbo, a continuación se escribe el verbo, luego la persona y si es del singular o plural, y por último el tiempo del verbo. Para el verbo conjugado decidme, la notación es un poco más extensa ya que debe contener la información sobre el objeto indirecto como pronom-

bre enclítico. De nuevo la v indica que se trata de un verbo, después que se trata del verbo decir que está conjugado en segunda persona del plural en la voz imperativa y que se tiene un complemento indirecto a la primera persona del singular: *me*.

EXTE realiza un marcado similar de forma sintáctica. Dado un fragmento del corpus que contiene artículos de periódico, marca[1] las expresiones temporales encontradas. Por ejemplo:

Nunca habremos hablado de esto, la imaginación nos reúne hoy tan vagamente como entonces la realidad.

Ejemplo 3.3— Fragmento de "Las caras de la medalla" de Cuentos Completos 2 de Julio Cortázar. Alfaguara, 2004.

El marcado de las expresiones temporales quedaría:

{Nunca:et} habremos hablado de esto, la imaginación nos reúne {hoy:et} tan vagamente como entonces la realidad.

Ejemplo 3.4— Marcado de expresiones temporales.

A continuación se presentan algunas de las expresiones temporales que **EXTE** puede marcar:

Tabla 3.2—Tabla con ejemplos de las expresiones temporales que **EXTE** puede marcar.

Expresión temporal	Descripción
En <año>	<año> contiene la referencia a un año
En el mes de <mes>	<mes> contiene algún mes escrito
En el último <sustantivo>	<sustantivo> contiene alguna partícula como mes, año, sexenio, etc.
Hasta <año>	<año> contiene la referencia a un año
A partir de <año>	<año> contiene la referencia a un año

[1] La sintaxis utilizada en el Ejemplo 3.2 fue inventada. No conozco algún corpus marcado que la utilice; **EXTE** no utiliza este marcado. Dicha sintaxis fue sólo con fines ilustrativos.

4 Procesamiento del lenguaje natural

El procesamiento del lenguaje natural está enfocado al procesamiento del lenguaje humano mediante computadoras. Intenta emular ciertas características del aprendizaje cognitivo humano. Sus aplicaciones actualmente son muy variadas:

- Traducción por computadora
- Interfaces de usuario que utilicen el lenguaje natural
- Reconocimiento de voz
- Generación de texto a partir de la voz
- Síntesis automática de textos
- Manejo de spam en Internet
- Motores inteligentes de búsqueda

Entender el lenguaje natural es complicado, como se mostrará en las siguientes secciones, e involucra un conocimiento enorme[1] y casi todas las etapas cognitivas del lenguaje pueden tener ambigüedad. Incluso si los científicos de la década de los cincuenta hubieran hecho un sistema mucho más complejo, los resultados habrían sido pobres.

[1] Aunque no nos damos cuenta nosotros cuando lo aprendemos desde niños, cada palabra nueva que adquirimos lleva consigo más que solamente la pronunciación.

4.1 ¿Qué hace la computadora para entender el lenguaje natural?

Entender el lenguaje es difícil para la computadora; la forma en la que el humano se capacita para utilizarlo, no se va dando de igual manera que para su estudio, y sobre todo si involucra una interacción con el medio. Es por esto que para la comprensión del lenguaje natural utilizando la computadora, se ha utilizado la misma división en etapas que hace la lingüística general y de las cuales se presentó una idea general en la sección 2.1 ¿Qué se necesita para entender el lenguaje natural?

Más adelante en la sección 6 Desarrollo se verá que los pasos utilizados para la creación de **EXTE** fueron básicamente el análisis léxico y el sintáctico y muy poco del semántico. En las siguientes secciones se muestran las etapas utilizadas para el manejo del lenguaje natural en textos, indicándose los procesos que se llevaron a cabo para su creación.

4.1.1 Análisis morfológico

Esta etapa del análisis es la primera y en ella se realiza una separación de las oraciones en palabras. Una de las tareas principales de este análisis es identificar las palabras, números y signos que son posibles dentro del lenguaje y cuáles son sus componentes. Para los fines de **EXTE**, los números son muy importantes, porque son los que definen las fechas y horas numéricas a ser marcadas. Pero, en este caso, los signos de exclamación y puntuación no nos interesan, ya que el marcado de una fecha será igual para

```
¡El pasado martes 12!
```

Ejemplo 4.1

Que para

```
El pasado martes 12.
```

Ejemplo 4.2

Esto porque en general los signos de puntuación no definen, ni afectan, la fecha. Es importante hacer ver que los signos de puntuación sí afectan el significado de la oración. En esta etapa del análisis

también se hace la comprobación de las palabras y sus derivaciones. Así, en esta fase, podrían reconocerse las palabras

```
El perro tiene frío.
```

Ejemplo 4.3

Y se deberían también reconocer las palabras del texto

```
El perrito tiene frío.
```

Ejemplo 4.4

En ambos casos se hace referencia a un perro, pero en la referencia del Ejemplo 4.4 está en diminutivo. El reconocimiento del género y número, también se realiza en esta etapa.

Una expresión temporal como

```
El lunes temprano
```

Ejemplo 4.5

Debería marcarse, pero también una como

```
Los lunes temprano
```

Ejemplo 4.6

Tiene que marcarse, pues sólo cambia el rango de lunes, a todos los lunes. El reconocimiento del género también se hace exclusivamente para las reglas consideradas.

4.1.2 Análisis sintáctico

El analizador sintáctico es el encargado de reportar errores en la estructura del lenguaje generado por la gramática. Los componentes léxicos que lee el analizador en la etapa previa se estructuran dentro de lo que se conoce como *árbol sintáctico*. El árbol sintáctico es una representación de los símbolos de acuerdo a la gramática.

De esta forma una expresión temporal como:

La semana pasada

Ejemplo 4.7

Podría tomarse en cuenta. Sin embargo, según las reglas

* Semana pasada la

Ejemplo 4.8

No debería tomarse en cuenta, ya que no sigue las reglas para ser correcta.

Durante esta etapa, el analizador se encarga de asignar la estructura. Apegado a dicha estructura, se crean los árboles sintácticos como el mostrado en la Figura 4.2 para la frase Llegarán en cinco días. La representación también podría verse como:

```
(O
  (FV
    (V     Llegarán))
  (FP
    (PREP en)
    (FS
       (NUM      cinco)
       (S        días))))
```

Figura 4.1—Estructura para la frase "Llegarán en cinco días".

Que puede leerse de la siguiente manera. La oración (O) consta de una frase del verbo (FV) que define al verbo (**V** Llegarán). Le sigue una frase del predicado que contiene una preposición (**PREP** en) y una frase del sustantivo (FS) y ésta abarca un número (**NUM** cinco) y un sustantivo (**S** días). La siguiente figura muestra el árbol sintáctico para la misma oración:

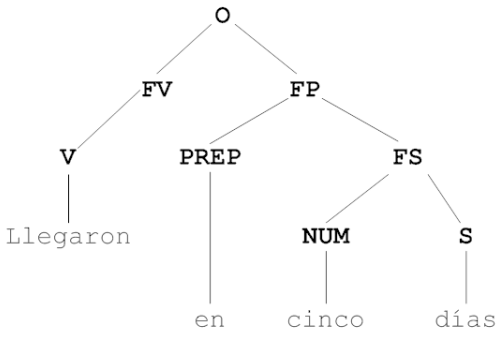

Figura 4.2— Árbol sintáctico para la frase "Llegaron en cinco días".

Y para construir este tipo de frases, debe tenerse un conjunto de reglas dadas por una gramática que indique cuáles estructuras son aceptables. Para esto se muestra la gramática que definiría la frase mostrada en la Figura 4.2.

O	→	FV	FP	NUM	→	cinco
FV	→	V		S	→	días
FP	→	PREP	FS	V	→	llegarán
FS	→	NUM	S	PREP	→	en

4.1.3 Análisis semántico

Para esta etapa, ya se tienen todos los componentes léxicos en el árbol de análisis sintáctico y se sabe que la estructura es correcta. Para el lenguaje natural, el nivel semántico es mucho más profundo que para un lenguaje formal, pues no sólo se tienen que verificar los tipos de datos, sino el significado de cada componente, tanto dentro de la oración como por sí mismo. Si en un texto se encontrara la oración

Macedonio comenzó su carrera hace diez años.

Ejemplo 4.9

En esta última oración la construcción quiere decir que Macedonio comenzó su carrera diez años atrás de la fecha en la que se escribe.

```
Macedonio comenzó su carrera diez años delante de Pascal.
```

Ejemplo 4.10

En la oración del Ejemplo 4.10 la construcción quiere decir que Macedonio inició su carrera diez años después de que Pascal comenzó la suya.

Si consideramos las frases: `alrededor del 4 de mayo`, `poco después del 4 de mayo` notamos que el contexto determina el tiempo expresado, aunque no lo precisa si marca una diferencia clara. Trabajo posterior en este tema incluirá el marcado de texto circundante para una mejor comprensión de la expresión temporal.

5 EXTE

5.1 Introducción

EXTE es un sistema que reconoce y marca **ex**presiones **te**mporales (1.4 Expresiones Temporales) utilizando un analizador léxico generado por una herramienta llamada JFlex ***(JFlex, 2005)***. A este tipo de herramientas en particular, se les conoce como *generadores de analizadores léxicos*. El analizador léxico y las reglas definidas para crearlo se encargan de la parte léxica para el reconocimiento de las expresiones temporales; los estados del analizador léxico comprenden la forma sintáctica de cada expresión temporal. Como se vio en el capítulo 4 Procesamiento del lenguaje natural, para procesar el lenguaje natural es necesario dividir su estudio en diferentes niveles interconectados. **EXTE** marca las expresiones temporales de un texto de forma sintáctica.

Como se mencionó en 1.5 Trabajos en el tema de expresiones temporales, la comunidad de la lingüística computacional está trabajando actualmente en marcadores de expresiones temporales que realizan etiquetados semánticos, creando una relación entre cada expresión temporal y una fecha o un rango conocidos (***Saquete & Martínez-Barco, 2000***).

5.2 Planteamiento del sistema

Extraer información de un texto es una tarea que se encuentra en desarrollo actualmente. El marcado de expresiones temporales en textos ayuda precisamente a extraer información, en este caso información acerca del tiempo.

La mayoría de los editores de texto como Word® o Emacs® tienen integrados buscadores de texto, algunos incluso con la posibilidad de buscar texto utilizando expresiones regulares[1]. Esta función de búsqueda podría ayudarnos a encontrar todas las horas con formato estándar como `13:35` contenidas en el texto, usando una expresión regular como:

```
\d?\d:\d\d
```
[2]

Ejemplo 5.1

Pero esta expresión regular no encontraría las horas del tipo `13 horas 35 minutos`. De igual modo, si la expresión regular fuera construida como:

```
\d?\d “horas” \d?\d “minutos”
```

Ejemplo 5.2

No estarían incluidas las horas con el formato del Ejemplo 5.1.

Considerar estos dos formatos no representa un problema, pero para buscar todas las horas escritas en un texto con la función de búsqueda que incluyen los procesadores de texto, tendríamos que escribir —además de dominar— expresiones regulares muy complejas. Para generalizar la tarea de búsqueda es necesario un programa como **EXTE**, que, tras haber analizado un corpus que contenga expresiones temporales, pueda reconocer la mayoría de las formas en que aparecen en textos comunes. Como se verá en la sección 5.3 Análisis del corpus, algunas de las expresiones temporales que fueron encontradas en el análisis del corpus fueron descartadas dado que el número de incidencias no excedía el umbral fijado.

---

[1] Una expresión regular es una cadena con reglas de sintaxis que describe, mediante un patrón, un conjunto de cadenas.

[2] En este ejemplo `\d` denota un dígito y `?` que lo que le antecede (en este caso el dígito) no es obligatorio.

El planteamiento general e inicial de **EXTE** propuso analizar un corpus que contuviera artículos de periódico en español mexicano, extraer las expresiones temporales ahí contenidas y crear una gramática que fuera capaz de reconocer, mediante un analizador léxico-sintáctico (en este caso JFlex), las expresiones temporales. Además, la gramática debía ser escalable, es decir, si algún usuario analizara textos que contienen una expresión temporal no reconocible por **EXTE**, debía existir algún modo para "enseñarle" esa nueva expresión temporal a ser reconocida en futuros usos.

Tal como se plantea en el trabajo de Saquete et al. (***Saquete & Martínez-Barco, 2000***) , el reconocimiento de las expresiones temporales no solamente abarca aquellas del tipo numérico (`25/12/2007`), también expresiones que no son numéricas (`dos días antes`). Saquete y Martínez-Barco además proponen hacer un marcado semántico utilizando referencias a fechas ya conocidas, y almacenar dichas marcas en un archivo XML.

La creación del programa involucra etapas relacionadas, como se verá en las siguientes secciones. Cada una de estas fases genera una parte de **EXTE**, que una vez completada, se conecta a la siguiente.

5.3 Análisis del corpus

El corpus utilizado para la creación de **EXTE** consta de periódicos mexicanos: La Jornada, El Universal y Excélsior, almacenados en formato HTML. Éstos abarcan todos los días del año dentro de un rango de años, de 1997 a 2002 para La Jornada, de 1997 a 1999 en El Universal y de 1997 a 2000 para Excélsior. Los periódicos La Jornada y Excélsior se utilizaron para el análisis del corpus y la creación de la gramática, y la parte de El Universal para realizar las pruebas.

El análisis del corpus consiste en identificar manualmente la mayor cantidad de candidatos[1] a ser expresiones temporales, llevar un conteo del número de veces que cada expresión temporal apareció y dónde apareció. Una vez terminado este análisis, se estableció cuál sería el mínimo de incidencias que debía tener una expresión temporal para ser agregada, es decir tomada en cuenta para ser reconocida por la gramática. Algunas de las expresiones temporales superaban las centenas de millar, mientras que otras aparecían una sola vez. Por ejemplo, la expresión temporal

En <año>

Ejemplo 5.3

que está conformada de la preposición en seguida de algún año, sea numérico o sea escrito, se encontró en cada uno de los textos analizados; a diferencia de

En <mes> y finales de <mes>

Ejemplo 5.4

Que se llegó a encontrar en algunos de los textos analizados, de forma dispersa y sin rebasar el umbral para que se tomara en cuenta. Más adelante se darán detalles de cómo se eligió y se modificó dicho umbral.

[1] En esta etapa cualquier referencia al tiempo, por simple o rara que pareciera o apareciera, se incluía.

5.3.1 Clasificación de expresiones temporales

Las expresiones temporales se separaron por categorías de acuerdo a la preposición con la que comenzaban, de tal modo que `a partir de mañana` formaría parte de la categoría A. Categorizar de este modo las expresiones temporales facilitaría la creación de la gramática, creando reglas distintas según el tipo de expresión encontrada; no se sigue con esto un estándar de cómo deberían clasificarse, pero facilita la manipulación de la gramática. También, el hecho de haber clasificado (algunas de) las expresiones temporales basándose en la preposición con la que comenzaban, no implica que no serían reconocibles las expresiones temporales que no comenzaran con preposición; es decir, `hoy` (que no contiene preposición) también se marca.

Dado que la cantidad de artículos que se tenía era muy grande, la clasificación de expresiones temporales resultó ser una tarea de mucho tiempo. Para dar un acercamiento global del número de expresiones temporales que se encontraron en los textos leídos, es necesario primero mencionar cuántos textos se leyeron para el análisis, cuántas oraciones contenían aproximadamente, y en promedio cuántas expresiones temporales se hallaban por oración; para una parte del análisis obtuve ayuda de un familiar que también leyó textos y marcó expresiones temporales.

Textos leídos	54
Oraciones por texto (aprox.)	5149
Expresiones temporales por oración (aprox.)	3
Expresiones temporales obtenidas por	2 personas

Tabla 5.1—Relación numérica entre los textos leídos, las oraciones por texto y las expresiones temporales por oración.

A continuación se describirá la manera en la que se categorizaron las expresiones temporales tras analizar el una parte del corpus (Excélsior y La Jornada); las categorías para las expresiones temporales encontradas dependían de su estructura general, como se vio en ejemplos anteriores. La categoría `Preposición` incluye todas las expresiones temporales que comenzaran con una preposición (a, de, desde, para, etc.), seguidas de algún otro tipo de expresión temporal. `Verbo` y `Adverbio` son categorías que sólo incluyen un tipo de estructura: `Verbo` las expresiones temporales que comienzan con `hace` y `Adverbio` las que comienzan con `antes`. La categoría de `Adverbio` realmente podría contener todos los adverbios temporales, pero como se verá más adelante, se incluyeron en otra categoría. `Adjetivo` es una de las categorías con más formas, incluye `este`, `esta`, `estos`, `estas`, `ese`, `esa`, `esos`,

esas, aquel, aquella, aquellos y aquellas. Artículo, aunque contiene pocas palabras en su estructura inicial, es quizá la categoría más importante, pues muchas de las expresiones temporales que comienzan con otra categoría, continúan con ésta. Luego tenemos las categorías de Fecha y Hora, que contienen cualquier fecha u hora con formato numérico y fechas simples como hoy. Por último, la categoría Aislada contiene cualquier otra expresión temporal, que por su número de ocurrencias (diez en el caso de este corpus) no alcanzó a crear una categoría nueva.

En la siguiente tabla se muestran algunos ejemplos de expresiones temporales según su categoría, algunos muy sencillos y otros más complejos.

Categoría	Ejemplo categoría	Ejemplo de expresión temporal
Preposición	En	En 1980, En el último mes del 2005
Verbo	Hace	Hace dos meses, hace no más de quince días
Adverbio	Antes	Antes, Antes del 12 de abril
Adjetivo	Este	Este 4 de septiembre, aquellos días
Artículo	El	El martes, El 18 de julio del presente
Fecha		15/10/2005, hoy pasada la media noche
Hora		13:35 p.m., minutos después de las dos de la tarde
Numérica		5 meses antes, quince días después
Aislada		Día de la bandera, casi dos años

Tabla 5.2—Ejemplos de expresiones temporales según su clasificación.

Esta categorización era necesaria para que **EXTE**, la gramática y los componentes en general fueran fáciles de manejar. Sin esta separación lógica de las expresiones temporales, la gramática seguiría reglas difíciles de manipular, porque a JFlex se le debe notificar cada una de las expresiones que reconocerá. En realidad para JFlex es lo mismo reconocer todas las expresiones temporales sin categoría que con categoría; sin embargo, al desarrollador y al usuario del programa, no les resultaría tan fácil determinar cuál regla de la gramática reconoce cierta expresión temporal. Es por esto que la categorización de las expresiones temporales ayuda a entender y leer la gramática haciendo que sea menos complicado detectar y corregir errores.

Se explicará con más detalle tres categorías que podrían no haberse definido claramente en la Tabla 5.2; éstas son Hora, Numérica y Aislada. Hora contiene todas las expresiones temporales que hagan referencia al tiempo dentro de un día. Es posible, claro, que la expresión temporal contenga aún

más información sobre el tiempo en el que se lleva a cabo. Podríamos encontrar una expresión temporal de tipo Hora como

12:30 p.m.

Ejemplo 5.5

Que sólo nos daría información sobre una hora puntual; pero también existen expresiones temporales que detallan mucho más, como

El sábado quince de enero del 2007 a las 12:30 de la tarde

Ejemplo 5.6

En la expresión temporal de tipo Hora del Ejemplo 5.5 sólo se tiene referencia a una hora "simple"; en cambio en aquélla del Ejemplo 5.6 no sólo tenemos la hora simple, sino toda una fecha que define exactamente en qué momento ocurrió el evento en cuestión. Lo que es importante recalcar, es que particularmente la expresión temporal como la del Ejemplo 5.6 no es de tipo Hora sino Artículo, pues comienza con el artículo el, luego le sigue una Fecha, continúa una expresión temporal de tipo Preposición que contiene otra de tipo Hora. Es por la ubicación general de las expresiones temporales de tipo Hora que es raro encontrar expresiones temporales *verdadera* o *exclusivamente* del tipo Hora. Por lo general este tipo de expresión temporal se encontrará dentro de otra, más comúnmente en las de tipo Fecha. Esto no implica que no existan las expresiones temporales únicamente de tipo Hora, simplemente que son menos comunes. De igual modo pasa para las expresiones temporales de tipo Fecha, se encuentran más como subexpresiones temporales de alguna otra más general. Para toda otra expresión temporal que no encaje dentro de las primeras ocho que se mencionaron en la Tabla 5.2 ni sea de tipo Hora o Fecha, se creó la categoría Aislada.

5.4 Estadísticas

En esta etapa de la creación de **EXTE** se analizó a fondo el número de apariciones que tenía cada expresión temporal dentro del corpus, según lo encontrado en la sección 5.3 Análisis del corpus, utilizando la herramienta de Linux `grep`, que sirve para encontrar incidencias de expresiones regulares o texto simple en un archivo de texto. Esta herramienta se maneja de manera similar a la función de búsqueda de texto que integran la mayoría de los procesadores de texto, con la ventaja de poder buscar una o varias expresiones regulares en uno o varios archivos a la vez.

El número de diferentes expresiones temporales encontradas es 478. Aproximadamente 15 de esas expresiones temporales se descartaron, pues su número de apariciones era menor a cuatro. La elección de este umbral varió. Inicialmente el umbral era de dos. Más tarde en el desarrollo, al tener demasiados estados en el AFN (se definirá AFN en la siguiente sección), el umbral subió a tres; incluso en tres el umbral, el AFN seguía siendo demasiado grande para generarse y se decidió aumentar el umbral para que quedara en cuatro. Es decir, en la sección 5.3 Análisis del corpus, se encontró la expresión temporal

`Apenas pasadas las primeras dos horas del sábado`

Ejemplo 5.7

Solamente una vez en todo el corpus[1]; dado que el número de incidencias es uno (y por lo tanto no excede el umbral cuatro), esta expresión temporal, aunque válida, fue descartada. Sin embargo, esta expresión temporal contiene otra que sí sería reconocida por **EXTE**:

`Las primeras dos horas del sábado`

Ejemplo 5.8

[1] Se realizaron dos pruebas utilizando la herramienta `grep`.

A continuación se muestran algunos ejemplos de las expresiones temporales y sus incidencias.

Entrada	Ocurrencias
Cuando resten <num> (días|semanas|años)	1
Al borde del año <año>	2
En <año> y hasta <año>	14
En <mes> del año <año>	185
En <año> y en <año>	191
Vísperas	8380
Desde <año>	18342
Hasta <año>	22621
En <mes>	65761
En <año>	163897

Tabla 5.3—Expresiones temporales y sus incidencias: las menos comunes y las más comunes.

Teniendo ya las expresiones temporales que serían tomadas en cuenta, lo que seguía era clasificarlas según su estructura sintáctica para continuar con la creación de la gramática.

La siguiente tabla muestra algunos ejemplos de las expresiones temporales que fueron descartadas.

Entrada	Ocurrencias	Ejemplo
cuando resten <num> (días|semanas|años)	1	Cuando resten 4 días
al borde del año <año>	2	Al borde del año 1984
En la era	1	En la era de Bush
<cardinal> septenio	1	Segundo septenio
<cardinal> bienio	1	Primer bienio
Números escritos con letra mayores a mil	Más de cien	Cuatro mil

Tabla 5.4—Expresiones temporales que no excedieron el umbral 4.

Las incidencias de números escritos con letra que representaban números mayores a mil, rebasaron claramente el umbral, sin embargo se descartaron, pues impedían que la gramática se generara, ya que contaba con un número de estados mayor a 250 mil. Al tener aproximadamente 186 mil estados el AFN, JFlex se quedaba sin memoria y no completaba su creación. El gran número de estados y su crecimiento al modificar algunas reglas de la gramática, depende de ciertas variables. Para el re-

conocimiento de los componentes léxicos no se realiza distinción entre mayúsculas y minúsculas; eso da al AFN el doble de posibilidades, de tal modo que `septiembre` se reconoce de igual forma que `SePtiEmBRE`. El agotamiento de la memoria se da por la forma en la que JFlex está implementado. Los generadores de analizadores léxicos están principalmente diseñados para lenguajes estructurados y no para el lenguaje natural.

5.5 Gramáticas

Una vez conocidas todas las expresiones temporales que aparecen en el corpus (5.3 Análisis del corpus) y su número de ocurrencias (5.4 Estadísticas) se analizó la estructura de cada una de ellas. Conocer su estructura sirve para construir la gramática que generará un analizador léxico construido por JFlex. JFlex es un generador de analizadores léxicos; es —a fin de cuentas— un programa que recibe como entrada una gramática y genera como salida el código fuente de otro programa que será capaz de "leer" lo descrito en la gramática. Es a groso modo un Autómata Finito No Determinista (AFN) que finalmente se transforma a un Autómata Finito Determinista (AFD). Un AFN ***(Aho, Ravi, & Jeffrey, 1990)*** es un modelo matemático formado por un conjunto de estados *S*, un conjunto de símbolos de entrada *Σ*, una función de transición *mueve* que transforma pares estado-símbolo en conjuntos de estado, un estado inicial y un conjunto de estados *F* considerados estados finales. Los AFN reconocen lenguajes representados por gramáticas; se representan generalmente mediante *gráficas dirigidas*, en las que cada nodo representa un estado y cada arista etiquetada la función de transición.

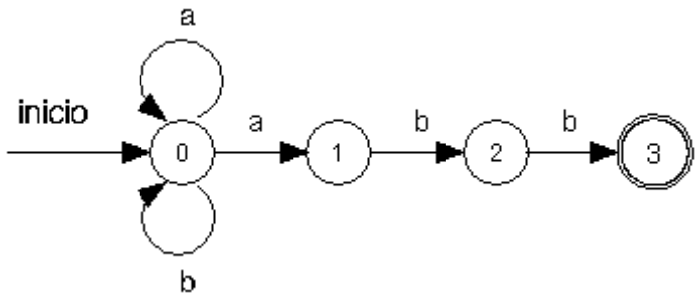

Figura 5.1— AFN que reconoce el lenguaje (a|b)*abb.

Un AFN *acepta* una cadena de entrada *x* si, y sólo si, existe un camino en la gráfica desde el estado inicial hasta alguno final[1], de tal modo que las etiquetas de las aristas intermedias *deletreen* la cadena *x*. Así, el AFN de la Figura 5.1 reconoce la cadena ababb, y se representa el camino utilizando los movimientos siguientes:

[1] En esta gráfica dirigida el estado final está marcado como una circunferencia con línea doble.

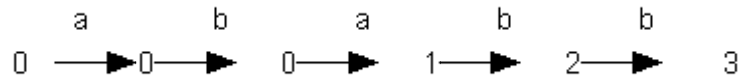

Figura 5.2— Movimientos para el AFN de la Figura 5.1 para reconocer la cadena ababb.

Un AFD ***(Aho, Ravi, & Jeffrey, 1990)*** es un caso especial de un AFN, en el que ningún estado tiene una transición Ɛ[2] y en el que para cada estado *s* y cada símbolo de entrada *a*, hay a lo más una arista etiquetada *a* que sale de *s*.

JFlex crea primero un AFN que, por las definiciones anteriores, contiene más estados y aristas, que un AFD; finalmente el AFN se convierte en un AFD y éste es el analizador léxico que sirve para reconocer el lenguaje definido en la gramática.

El primer paso para la creación de la gramática de las expresiones temporales fue agruparlas según su estructura. Por ejemplo, se encontraron expresiones temporales como

El año **pasado**

Ejemplo 5.9

El año **siguiente**

Ejemplo 5.10

Y

El año **próximo**

Ejemplo 5.11

Aunque éstas son diferentes expresiones temporales, tienen en común iniciar con el artículo el seguido del sustantivo año y por último un adjetivo, ya sea pasado, siguiente, próximo o cualquier

[2] El símbolo Ɛ es conocido como *cadena vacía.*

otro adjetivo similar. Agrupando así las expresiones temporales, se obtuvo una expresión temporal más general:

```
El año <adjetivo>
```

Ejemplo 5.12

La cual define que luego de las cadenas `el` y `año`, en esa secuencia, puede venir cualquier adjetivo afín[3].

La forma en la que se definen las gramáticas para JFlex es muy similar a la notación BNF, con algunos detalles adicionales.

Ya clasificadas y reducidas a su forma más general (Ejemplo 5.12), las expresiones temporales se catalogaron con respecto a un grupo más amplio: su comienzo. En la Tabla 5.2 se muestra la clasificación de las expresiones temporales y algunos ejemplos. No todas comienzan con preposición, algunas comienzan con artículo, adjetivo, etc. Siguiendo la clasificación de Tabla 5.2 y teniendo las formas reducidas como en el Ejemplo 5.12, se tenía que definir lo que, por ejemplo, <adjetivo> significaría. La definición para <adjetivo> dentro de la gramática es la siguiente (los plurales también se consideraron, aunque no se muestra en la regla):

```
<adjetivo> = siguientes|concluid[oa]|"en curso"
           |pasad[oa]|anterior|actual|próxim[ao]|"de hoy"
           |"de ayer"|(que{Espacio}faltan?)|"que entra"| previ[ao]| entrante
           |venider[oa]| reciente| posterior| (que{Espacio}inici[aó])
           |(que{Espacio}termin[aó])|(que{Espacio}concluy[eó])|"recién pasada"
```

Así, podría también reconocerse la expresión temporal

```
El año entrante
```

[3] Sólo se consideraron adjetivos que pudieran utilizarse para este tipo de expresiones temporales. Adjetivos como "bonito" o "especial" no se tomaron en cuenta.

Ejemplo 5.13

Siguiendo la forma general `el año <adjetivo>`. Claro que según esta definición de `<adjetivo>`, podrían reconocerse más expresiones temporales, sin implicar que siempre sean expresiones temporales válidas. La siguiente expresión temporal incorrecta podría ser reconocida

```
* El año de hoy
```

Ejemplo 5.14

Pero la resolución de la interpretación semántica requiere un contexto mayor que no es objeto de este trabajo, es por esto y por su complejidad que **EXTE** no realiza una revisión semántica de lo que está marcando y sí podría llegar a marcar una expresión temporal bien formada pero semánticamente incorrecta, como aquélla del Ejemplo 5.14. La única revisión semántica de **EXTE** se realiza principalmente en las expresiones temporales de tipo *fecha*. Durante el análisis del corpus, en varios de los artículos, algunos tipos de expresiones hacían referencia a documentos oficiales que podían confundirse con fechas.

```
El documento oficial 15/XII-9689
```

Ejemplo 5.15

En este último ejemplo, la expresión temporal comenzaría en la cadena 15/, y según la definición de la gramática, XII puede ser un mes (diciembre) y 9689, por constar de 4 dígitos, un año; el carácter separador / o – es tomado indistintamente, de tal modo que, en efecto, podría ser una fecha numérica. La revisión semántica se hace para verificar que el año sea un año *posible* o *común*. Es poco probable que en un texto encontremos referencias a años *muy lejanos en el futuro*, por ejemplo 9689 y por lo tanto este tipo de números no se tomó en cuenta. Ese año existe y sería posible que algún texto hiciera alusión a uno tan lejano; desafortunadamente para este tipo de años, **EXTE** sí reconocería la estructura pero no tomaría en cuenta la expresión como expresión temporal. El año más lejano que se encontró en 5.3 Análisis del corpus fue *3000* con la certeza de que se hablaba de un año, pues le precedía la cadena año.

La etapa más complicada de la creación de **EXTE** fue en la que se creó la gramática. En primer lugar hay que aprender cómo describir las gramáticas para cada generador de analizadores léxicos, que aunque similares, tienen ciertos detalles que pueden afectar el desempeño. La gran mayoría de estos generadores se utilizan para lenguajes formales y usan algoritmos muy eficientes, pero que finalmente no fueron pensados para analizar el lenguaje natural, que como hemos visto, es mucho más difícil de manipular. Para esta etapa, las reglas de la gramática que se iban escribiendo, se probaban "sobre la marcha" y realmente no dentro de un texto *real*; es decir, se hacían pruebas específicas para las reglas que se iban creando, sin tomar en cuenta que dichas expresiones temporales estarían contenidas dentro de más texto e incluso dentro de otra expresión temporal. Más adelante, en el capítulo 6 Desarrollo, veremos por qué fue tan complicada la integración de todas las reglas.

Pero **EXTE** utiliza dos gramáticas, la más extensa y recién descrita `tiempo.flex` (en adelante, el analizador léxico generado por `tiempo.flex` será llamado `TIEMPO`) se encarga de generar el reconocedor de las expresiones temporales; la siguiente y mucho más pequeña `trimmer.flex` (a este analizador léxico se le llamará `TRIM`) tiene como objetivo *limpiar* los archivos que leerá `TIEMPO`; esto es: (i) si el archivo es de texto, se encarga de eliminar todos los espacios redundantes y (ii) si el archivo leído es un archivo de hipertexto (HTML), se encarga de eliminar todas las etiquetas y caracteres especiales[4] y transformarlos a texto puro. `TRIM` es auxiliar de `TIEMPO`; separando el trabajo que cada uno debe hacer, se logra que `TIEMPO` sea más efectivo y rápido, garantizando que el archivo leído esté *limpio*[5]. La gramática utilizada por Saquete et al. (Saquete & Martínez-Barco, 2000) podría verse como una sub-gramática de la que utiliza **EXTE**, ya que sintácticamente **EXTE** reconoce más expresiones temporales. La diferencia con el trabajo de Saquete y Martínez-Barco radica fundamentalmente en que **EXTE** no crea la referencia a fechas como `ayer`. De este modo la expresión temporal `ayer` es solamente la concatenación de dichas letras, mientras que Saquete et al. lo representan como la referencia `FechaP - 1`, es decir la fecha en cuestión menos un día. Así, suponiendo que la fecha base es `martes 12 de enero de 2007`

`El próximo martes`

Ejemplo 5.16

[4] En HTML, los caracteres como ñ o é se escriben en un código especial a ser interpretado por el navegador: `&ntilde;` y `&eacute;`
[5] Existen transformadores de HTML a texto muy eficaces. Lo ideal sería utilizar uno de estos transformadores (Bayer, 2005) y (HDSE-Software, 2003).

Se marca como tal (la concatenación de las tres palabras) en **EXTE**, y en el trabajo de Saquete y Martínez-Barco como

```
Martes 19 de enero de 2007
```

Ejemplo 5.17

Otra de las diferencias con la gramática de Saquete y Martínez-Barco es que léxicamente se tienen menos posibilidades de fecha. Por ejemplo, se tiene el tipo de referencia `"dentro" + "de" + num + ["días"|"meses"|"años"]`, pero no se tiene una como `"dentro" + "de" + "los" + "siguientes" + num + ["días"|"meses"|"años"]` y **EXTE** sí la tiene[6].

Esto es resultado del esfuerzo, pues en la revisión semántica se intercambiaron un mayor número de expresiones temporales detectables por expresiones temporales con referencia semántica.

Ambas gramáticas, `TIEMPO` y la de Saquete et al., se encuentran en los apéndices, A.1 Gramática de TIEMPO y A.2 Gramática de Saquete respectivamente.

[6] Según lo mostrado en dicho artículo. Es probable que la gramática que se muestra ahí esté incompleta.

6 Desarrollo

6.1.1 Planteamiento y requerimientos

EXTE tiene como fin ayudar a la extracción de información acerca del tiempo en corpus. El requerimiento inicial para **EXTE** era que fuera un programa que marcara la secuencia de palabras que componen las expresiones temporales, luego de haber analizado sus variedades en un corpus; además debía tener un módulo de *mantenimiento* con la capacidad de agregar nuevas reglas a la gramática; de esta forma, el conjunto de expresiones temporales reconocibles por **EXTE** no permanecería igual. También se requería poder almacenar el resultado del etiquetado para un futuro uso; saber qué camino siguió el analizador léxico para reconocer cierta expresión temporal; desplegar las estadísticas de las expresiones temporales encontradas; poder buscar expresiones temporales desde un archivo o de texto tecleado por el usuario. Por último, se pedía que las expresiones temporales encontradas fueran *distinguibles visualmente* entre la otra parte del texto. Más adelante, en 6.2 Complicaciones, veremos cómo se limitó **EXTE**, dada la naturaleza del problema.

6.1.2 Implementación

Al comenzar esta etapa, el corpus ya estaba analizado, se tenía la gramática que sería capaz de reconocer expresiones temporales; además ya se habían hecho pruebas unitarias para las reglas, una a una, de la gramática.

Muchos de los programas hechos para computadoras cuentan con una interfaz gráfica, la cual hace que el programa sea más amigable con el usuario, facilitando su uso. **EXTE** cuenta con una interfaz gráfica desarrollada en Java que muestra ventanas, botones, cuadros de diálogo, etc. al usuario. La implementación de la interfaz gráfica debe ser independiente de la parte lógica del programa, de manera que a partir del inicio de su creación, **EXTE** se separó en tres partes: la capa visual (la interfaz gráfica), la capa de análisis (donde residen `TIEMPO` y `TRIM`) y la capa lógica (la conexión entre la capa visual y la de análisis) (***Gamma, Helm, Johnson, & Vlissides, 1995***).

Para esta etapa, la capa de análisis, aunque no depurada[1], estaba completa. Ya se había creado la gramática que se utilizaría y los analizadores léxicos que servirían a dicha gramática para **reconocer** las expresiones temporales encontradas luego del análisis del corpus. Se resalta *reconocer* porque TIEMPO es lo único que *sabe* hacer: delimitar expresiones temporales. Queda, pues, pendiente manipular estos componentes léxicos reconocidos (las expresiones temporales), y es la capa lógica la encargada de *darles sentido*.

La capa lógica se encarga, como ya se dijo, de conectar a las otras dos capas; entonces antes de crearla, era necesario conocer la forma de la otra capa: la capa visual. Programar componentes visuales resulta una tarea relativamente sencilla en Java, cuando se requiere de una funcionalidad básica. Se tienen que definir los componentes que se utilizarán y su posición, es decir, qué verá el usuario para seguir los requerimientos.

[1] Sin haber realizado pruebas sobre las expresiones temporales.

6.1.2.1 Capa visual

Los lenguajes de programación contienen componentes ya programados para definir un botón o algún otro objeto visual. De forma general, lo que **EXTE** necesitaba eran botones y componentes de texto. Los botones simplifican las tareas que puede solicitar el usuario; se genera una función cuando el usuario hace clic sobre alguno de ellos. Los requerimientos para **EXTE** debían ser suficiente base para saber qué botones (funciones en la capa lógica) y qué componentes de texto (lugar donde se desplegarían los textos) necesitaría **EXTE**, y son los siguientes:

Abrir un documento: Botón. Abre un documento de texto o hipertexto para ser analizado.

Guardar un documento: Botón. Guarda un documento modificado.

Encontrar expresiones temporales: Botón. Analiza el archivo de texto para encontrar las expresiones temporales contenidas.

Ver reglas de la gramática: Botón. Despliega las reglas gramaticales por las que pasó `TIEMPO` para reconocer una expresión temporal.

Desplegar texto original: Componente de texto. Despliega el texto original de un archivo, tal como se leería en cualquier otro editor de texto. Tiene la capacidad de modificar el texto y guardarlo utilizando la función de guardado.

Desplegar texto marcado: Componente de texto. Despliega el texto original con las expresiones temporales marcadas, visualmente distinguibles unas de otras. El texto en este componente no es editable, pues depende completamente del texto original.

Desplegar y encontrar expresiones temporales en un texto tecleado por el usuario: Componente de texto con botón asociado. Recibe texto tecleado por el usuario y analiza dicho texto para encontrar y marcar expresiones temporales, con la opción de guardado.

Configurar: Botón. Configura las opciones para desplegar los textos original y marcado, directorios, fuentes, etc.

Rehacer y deshacer: Botones. En los componentes de texto editables, deshace la última acción tecleada o rehace la última acción deshecha.

Copiar, pegar y cortar: Botones. En un componente de texto editable, cortar elimina el texto seleccionado y lo almacena en el portapapeles del sistema; pegar, inserta el último texto del portapapeles en la posición del cursor; copiar, crea una copia del texto seleccionado en el portapapeles del sistema.

Buscar y reemplazar: Botones. En un componente de texto, encuentra, si existen, todas las ocurrencias del criterio de búsqueda; reemplazar, dado un criterio de búsqueda y un reemplazo, sustituye todas las incidencias del primero por el segundo.

Cada botón tiene una acción asociada y para cada acción hay un acceso rápido, para más información sobre los accesos rápidos deberá revisarse el manual de **EXTE**.

6.1.2.1.1 Imágenes

A continuación se muestran imágenes de los principales componentes de la capa visual y se da una descripción breve de su funcionalidad con respecto a la capa lógica (6.1.2.2 Capa lógica).

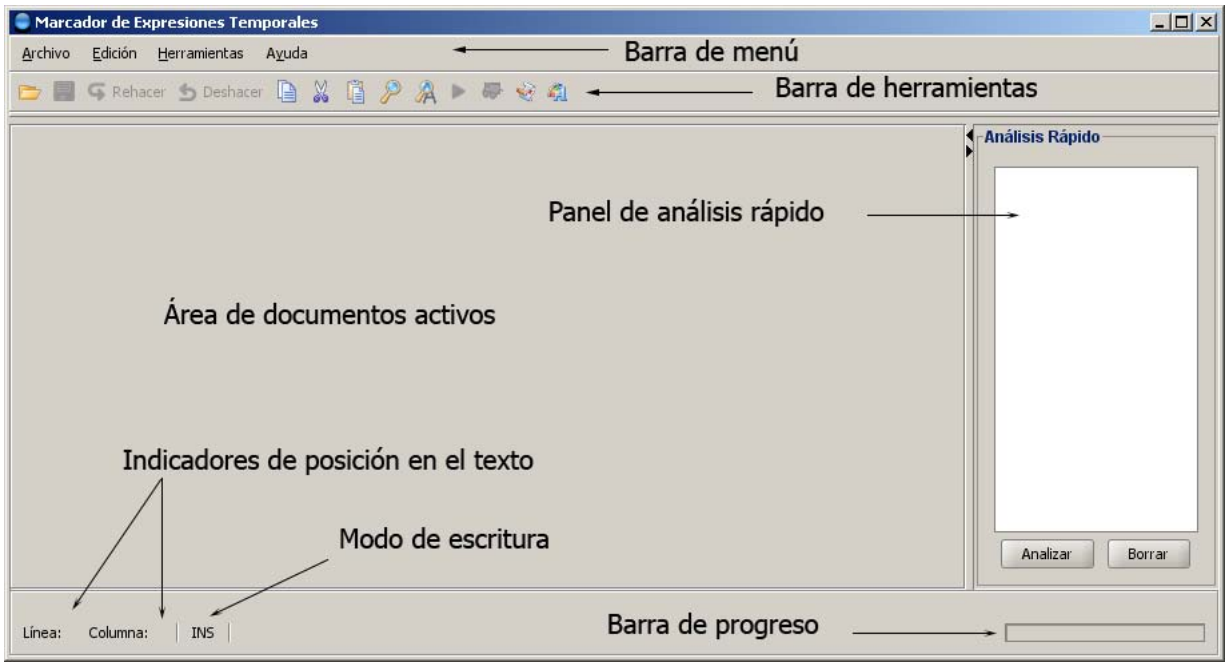

Figura 6.1—Ventana principal de **EXTE**.

La imagen que se muestra en la Figura 6.1 es la pantalla principal (lo primero que el usuario ve). En esta pantalla es visible casi todo lo que **EXTE** contiene: la barra de menú, la barra de herramientas,

el panel de análisis rápido, el área de documentos y los indicadores. Esta pantalla se muestra siempre.

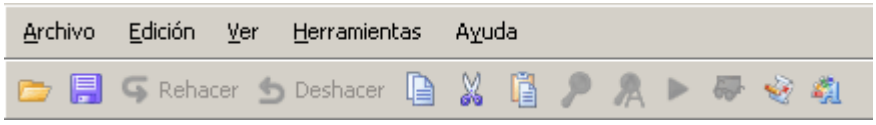

Figura 6.2—Área de las barras de herramientas y menú.

La pantalla de la Figura 6.2 muestra con más detalle el área de la barra de menú y la barra de herramientas. Cada una de estas barras tiene funcionalidad dependiendo del contexto en el que se esté trabajando. Ésta es un área de trabajo que siempre es visible.

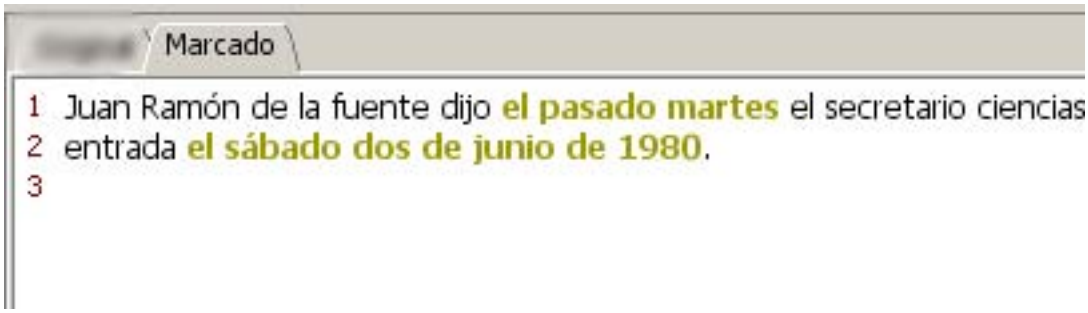

Figura 6.3—Editor estilizado.

En la pantalla anterior (Figura 6.3) se puede visualizar el editor estilizado descrito en Desplegar texto marcado. Este componente no siempre será visible, es necesario que **EXTE** haya marcado las expresiones temporales. Como puede apreciarse, lo que **EXTE** reconoció como expresión temporal se marca con otro color.

Figura 6.4—Componente que despliega la gramática de **EXTE**.

La Figura 6.4 muestra el componente que despliega la gramática. Este editor tampoco es visible durante toda la ejecución; es necesario que el usuario indique que desea ver la definición de la gramática (refiérase a Ver reglas de la gramática y a 5.5 Gramáticas). También se puede acceder a él, cuando se está analizando la trayectoria que siguió la capa de análisis para marcar una expresión temporal.

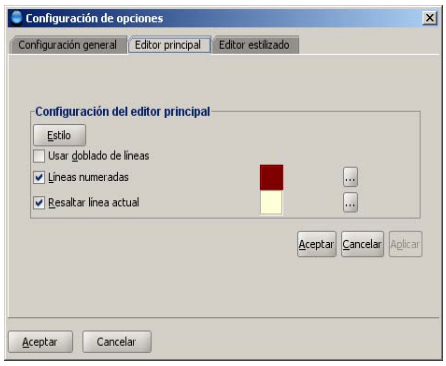

Figura 6.7—Pantalla de configuración del editor principal.

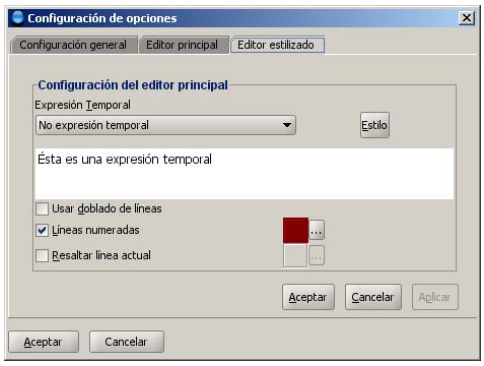

Figura 6.7—Pantalla de configuración del editor estilizado.

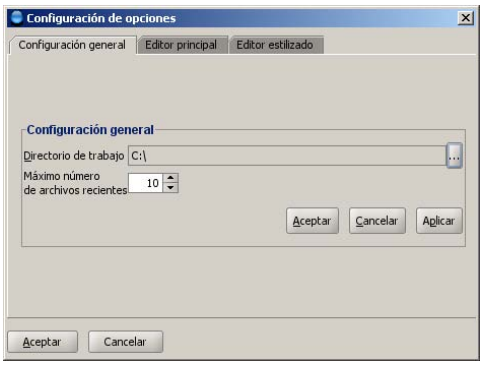

Figura 6.7—Pantalla de configuración de **EXTE**.

Las tres pantallas arriba mostradas (Figura 6.5, Figura 6.6 y Figura 6.7) forman parte de la configuración de **EXTE**. Se accede a ellas accionando el evento de configuración (Configurar).

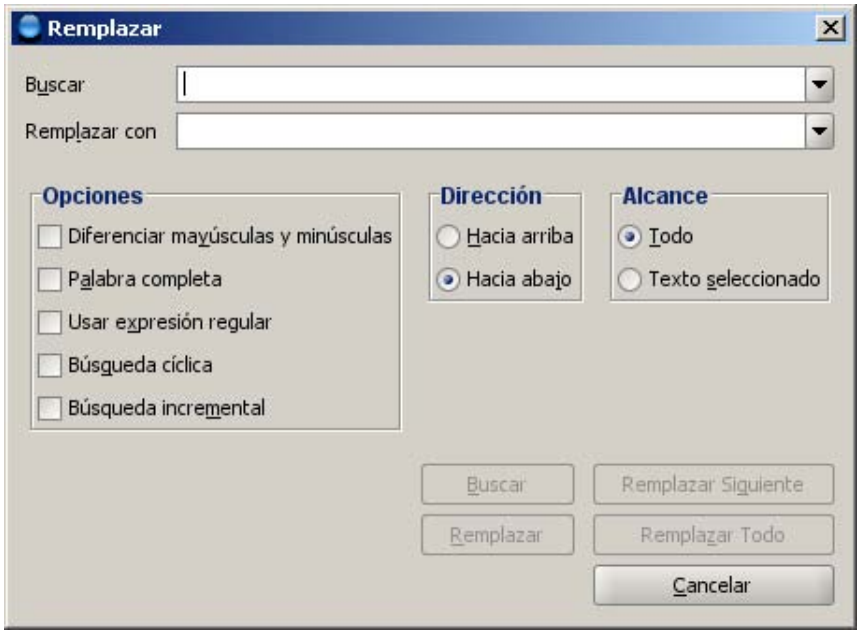

Figura 6.8—Pantalla de búsqueda y reemplazo de texto.

La Figura 6.8 muestra la ventana de búsqueda/reemplazo de texto. Se accede por medio del botón de búsqueda o reemplazo (Buscar y reemplazar).

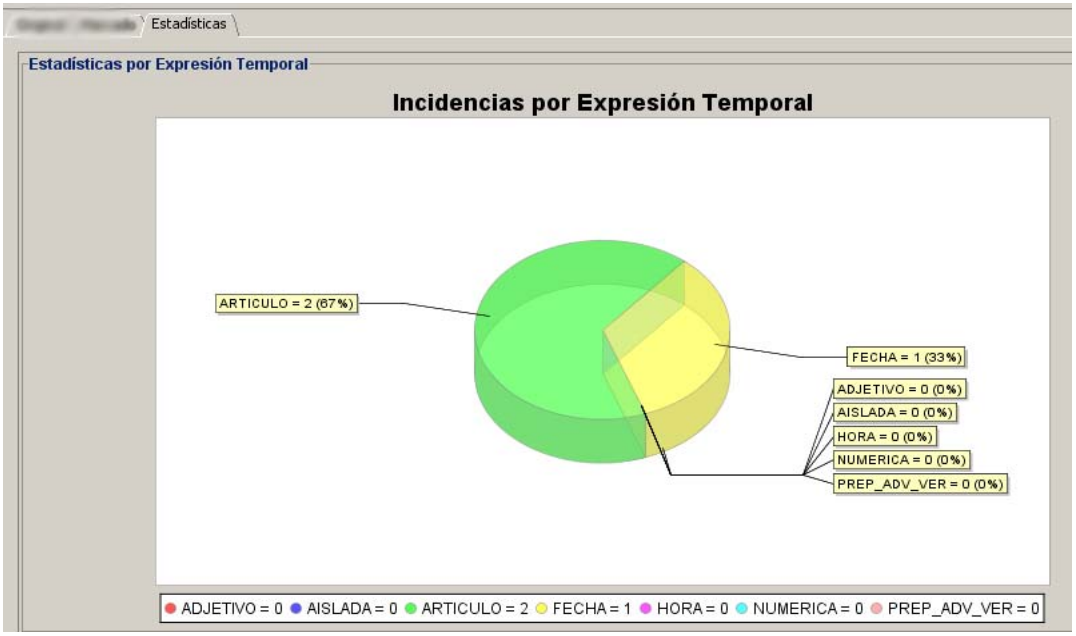

Figura 6.9—Pantalla de estadísticas.

Ésta es la pantalla de estadísticas (5.4 Estadísticas) que despliega gráficamente la relación de las expresiones temporales que se encontraron en el texto. Esta pantalla es visible solamente si el usuario accede a ella utilizando el evento de estadísticas.

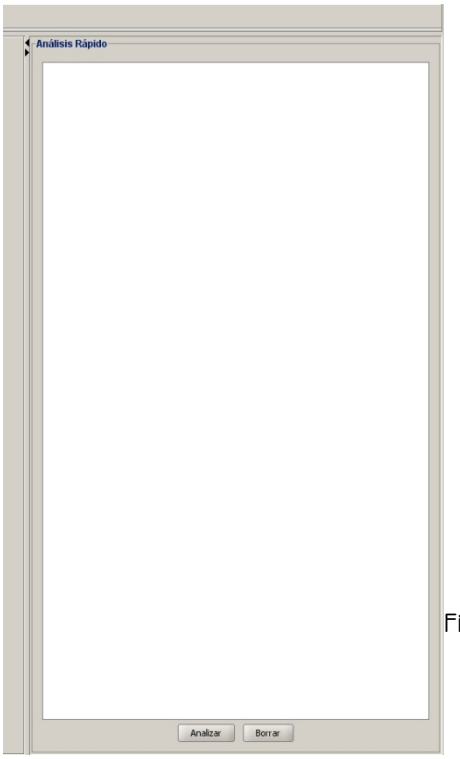

Figura 6.10—Panel de análisis rápido.

Por último, la Figura 6.10 muestra el panel de análisis rápido (Desplegar y encontrar expresiones temporales en un texto tecleado por el usuario). Este componente se muestra el inicio y luego de cinco segundos se oculta automáticamente.

6.1.2.2 Capa lógica

La capa lógica es la encargada de manejar lógicamente las acciones generadas por el usuario en la capa visual. Por cada componente que genere una acción en la capa visual, habrá una acción asociada en la capa lógica que generará una respuesta. La siguiente figura muestra la relación entre la capa visual y la capa lógica.

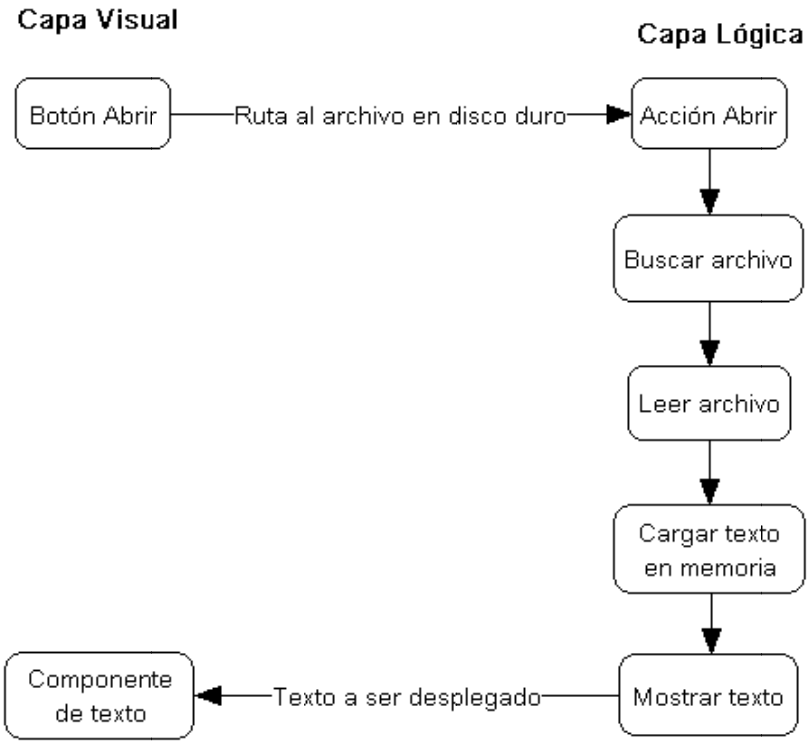

Figura 6.11— Relación entre la capa lógica y la capa visual para la acción *abrir*.

En la figura anterior se puede observar qué es lo que el usuario percibe y los pasos que ejecuta el programa entre la acción y la respuesta. De igual modo, para cada botón visible para el usuario, existen varios pasos dentro de la capa lógica, a los cuales sólo el programa tiene acceso.

El manual de **EXTE** contiene la descripción detallada de cada uno de los componentes visuales y las acciones tomadas.

6.1.3 Integración

Juntar las tres capas que forman parte de **EXTE** es la labor en la que más cuidado se debe tener. Aunque en la capa visual todo componente (obviamente) es visible para el usuario, no es igual en la capa lógica, y mucho menos en la capa de análisis. Como desarrollador, se crean las primeras dos, pero recordemos que la tercera (realmente) fue creada por JFlex a partir de la gramática. En la siguiente figura se muestra la forma en la que la función *analizar* de **EXTE** es interconectada con las tres capas.

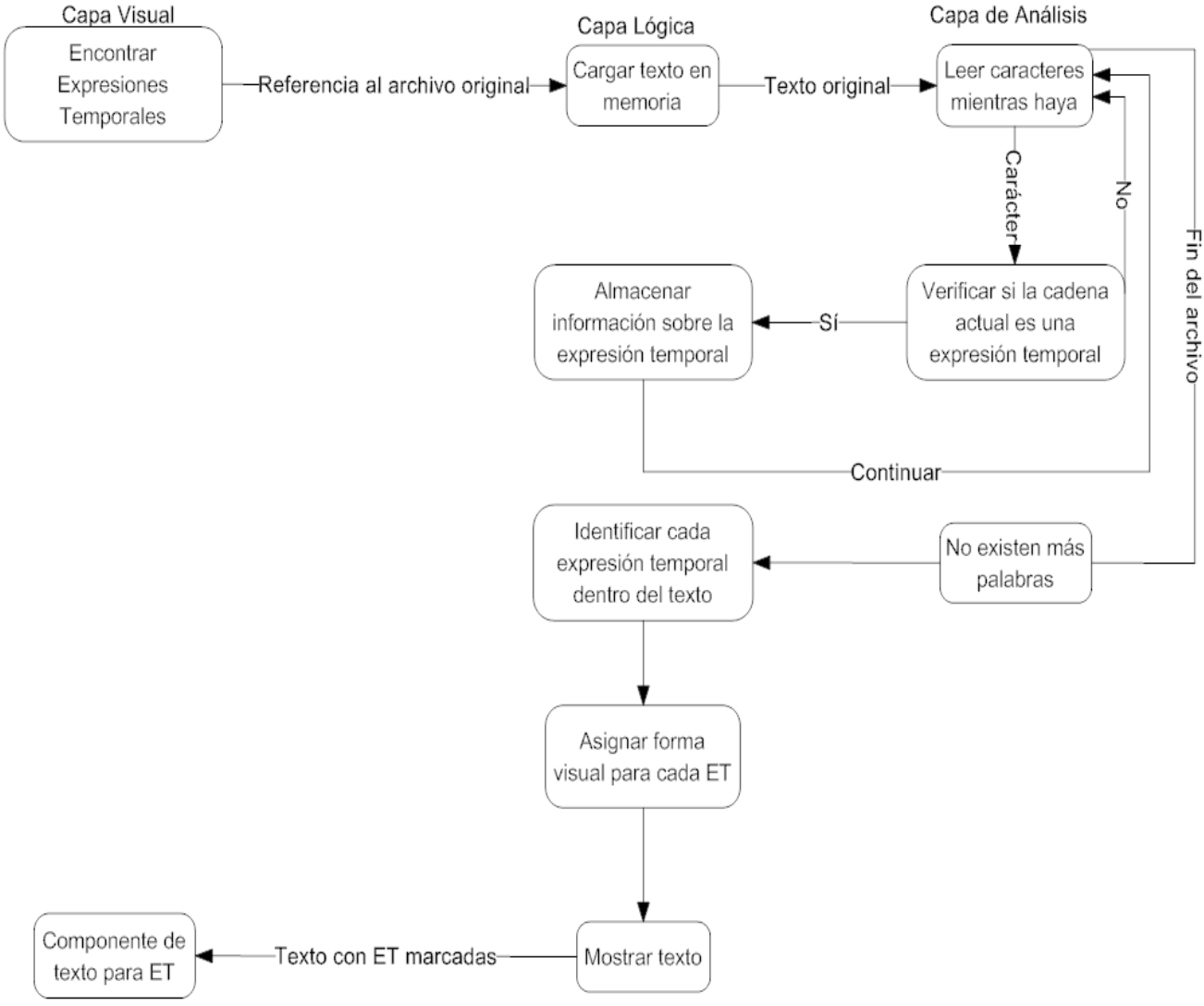

Figura 6.12— Relación entre las tres capas de **EXTE**.

Igual que para la función *analizar*, cada una de las otras funciones fue conectada lógicamente entre las tres capas.

6.1.4 Pruebas

Teniendo las tres capas relacionadas, lo que quedaba por hacer era realizar las pruebas del conjunto; es decir, de lo que "realmente" es **EXTE**. No capa por capa, regla por regla o acción por acción, sino como un *todo*.

La parte del corpus utilizado para las pruebas fue la de artículos de El Universal. Etapa en la que se tenía que identificar *a mano* cada una de las expresiones temporales marcables[1], para ser cotejadas por aquéllas que encontrare **EXTE** al ser ejecutado. La etapa de pruebas fue larga, pues se encontraron errores en las tres capas. Por cada error encontrado era necesario identificar la capa en la que residía, y ya en la capa, su lugar exacto en código fuente. También había que hacer una prueba de todas las funciones de **EXTE** por cada error reparado; uno debe ser cuidadoso, pues alterar una función, puede afectar directa o indirectamente otra(s).

[1] Que la gramática describía, y por ende todas aquellas que TIEMPO sería capaz de reconocer.

6.2 Complicaciones

La primera y mayor complicación que fue encontrada durante la creación de **EXTE**, fue JFlex y su manejo de memoria. Ya se mencionó antes, que JFlex (y en general los generadores de analizadores léxicos) es utilizado principalmente para lenguajes estructurados, implementando algoritmos muy eficientes, pero a fin de cuentas, no para el lenguaje natural. En secciones anteriores vimos que las combinaciones sintácticas para el lenguaje natural son muchas más que para el estructurado. Por esa razón, el número de estados del AFN generado por JFlex era muy grande para la gramática `tiempo.flex` entera. También, dado que las pruebas de cada una de las reglas gramaticales de `tiempo.flex` se hicieron por separado, el AFN consistía de entre mil y tres mil estados cada que se realizaba una prueba unitaria. Juntando todas las reglas de la gramática de `TIEMPO`, el AFN generado por JFlex llegó a ser de hasta 186 mil estados aproximadamente. Esta gran cantidad de estados agotaría eventualmente la memoria de la computadora[1] en la que se generaba `TIEMPO`. El primer paso para solucionar este problema fue eliminar cualquier tipo de redundancia que pudiera contener la gramática de `TIEMPO`; tarea ardua que dio resultados benéficos pero no suficientes: se redujo el número de estados del AFN a 112 mil, aproximadamente; sin embargo, la memoria no era suficiente todavía. La gramática ya no contenía redundancia (es decir, dos o más estados iguales) ¿tendrían que eliminarse reglas para lograr generar `TIEMPO`? En efecto, muchas se eliminaron. Entre ellas, los números desde 1000 hasta 9999 escritos con caracteres (es decir, `mil`, `doce mil catorce`, `nueve mil novecientos noventa y nueve`, etc.). Esta reducción de estados fue suficiente para generar `TIEMPO`, con la desventaja de que menos números escritos con palabras serían reconocibles por **EXTE**, además de estar dependiendo completamente del número de estados del AFN. Para el punto en el que `TIEMPO` podía ser generado sin presentar problemas de memoria, se tenían aproximadamente 83 mil estados en el AFN. A pesar de haber generado `TIEMPO` tantas veces para las pruebas generales del sistema, no es posible decir cuál es la cifra exacta sobre el máximo número de estados que puede tener su AFN. Esta cifra variará según la computadora, el sistema operativo, la versión del compilador, etc. que se utilicen para la generación del analizador léxico. Apoyando esta hipótesis, aquí se refiere al caso particular de la computadora que se utilizó para la implementación; `TIEMPO` no pudo ser generado en el

[1] Intel Pentium D 3.4GHz, 4GB RAM.

sistema operativo Windows® a pesar de que, claramente, la gramática era exactamente la misma[2]. Es por esto que el requerimiento del módulo que permitiría a un usuario agregar nuevas reglas a la gramática se descartó: la memoria requerida es demasiada, además de que necesita de un sistema operativo que haga un excelente manejo de ésta y mucha gente todavía utiliza Windows®.

Este problema realmente surgió porque JFlex fue pensado para lenguajes estructurados, y no para el lenguaje natural. Ese tipo de generadores de analizadores léxicos son verdaderamente muy eficaces y veloces, contienen algoritmos de órdenes muy bajos y son compactos, pero a fin de cuentas dichos algoritmos se idearon para lenguajes de programación, por ejemplo. Antes de optar por la eliminación de estados (aproximadamente teniendo 200 mil) se pensó en la opción de utilizar un generador de analizadores léxicos implementado en el lenguaje de programación C[3], pero surgió otra complicación rápidamente: la comunicación entre C y Java®. Dado que implementar esta comunicación habría requerido de más tiempo, se descartó dicha idea.

La segunda complicación fue la manera en la que Windows® y Linux® manejan los archivos. Cabe mencionar que las capas de análisis y lógica fueron desarrolladas en Linux® y la capa visual en Windows®. Cada archivo de la computadora tiene una codificación de caracteres, y esa codificación cambia de un sistema operativo a otro. También la forma en la que los sistemas operativos representan el salto de línea (es decir, pasar al siguiente renglón) es diferente para cada uno. Macintosh® utiliza el carácter CR, Linux LF y Windows® utiliza los dos: CRLF. Dado que una parte de la capa fue creada en Linux® y la otra en Windows®, existía incompatibilidad de codificación dentro de las capas. Esto se resolvió manejando todo salto de línea con el carácter \n de UTF-8 para cualquier tipo de salto de línea encontrado, unificando así el de cada sistema operativo.

En tercer lugar, se tuvo que eliminar toda la redundancia dentro de la gramática; esto también para eliminar estados. Sabiendo que la memoria era un recurso que requería de mucha atención, depurar y perfeccionar la gramática era imperativo. Como se menciona en la sección 5.3.1 Clasificación de expresiones temporales, la categoría Adverbio, por ejemplo, contenía no sólo las expresiones temporales que comenzaran con antes, sino hoy, mañana, etc. Pero esta estructura de la gramática y de

[2] No sé ciertamente a qué se deba esto, queda por investigarse. Es posible que el manejo de memoria en Windows sea extremadamente más deficiente que en Linux, ya que ambos sistemas operativos utilizaban la misma cantidad máxima de memoria y el mismo procesador.
[3] Por su bajo nivel es uno de los lenguajes de programación más eficientes.

sus reglas hacía que para expresiones temporales tan sencillas como hoy, se hiciera trabajo adicional innecesario. Por eso, dentro del perfeccionamiento de la gramática, muchas de las reglas redundantes (y que por ende hacían menos efectivo a TIEMPO) se redujeron a formas más básicas y aisladas. Expresiones temporales sencillas como hoy se pasaron a una subcategoría de Fecha llamada internamente FechaSimple.

Por último, la cuarta complicación fue la forma en la que se marcaría cada expresión temporal. Como se explica en la sección 5.4 Estadísticas, se encontraron 478 expresiones temporales distintas, agrupadas cada una por su comienzo, dando como resultado una clasificación de nueve categorías de expresiones temporales. Inicialmente, como los requerimientos lo exigían, las expresiones temporales reconocidas debían marcarse para que resaltaran visualmente, y marcarlas en **negritas** fue buena idea habiendo diez o veinte de ellas; no obstante, en los artículos del corpus se encontraban millares de ellas, haciendo que el componente de texto que desplegaba las expresiones temporales estuviera sobrecargado visualmente, al punto de no reconocer qué era expresión temporal y qué no lo era. Lo que siguió fue asignar un estilo para cada expresión temporal según su categoría. A la categoría PREPOSICIÓN, por ejemplo, se le asignaría un color distinto a cualquier otra categoría. La asignación de un estilo por cada categoría crearía un requerimiento *sobre la marcha*: la posibilidad de que el usuario configurara a su gusto el estilo de cada grupo de expresiones temporales.

6.3 Cómo funciona

Para explicar cómo funciona **EXTE** se incluye una ilustración que muestra los componentes involucrados y cierta noción del tiempo, es decir, del orden en el que los eventos de **EXTE** se van dando según lo que ejecute el usuario.

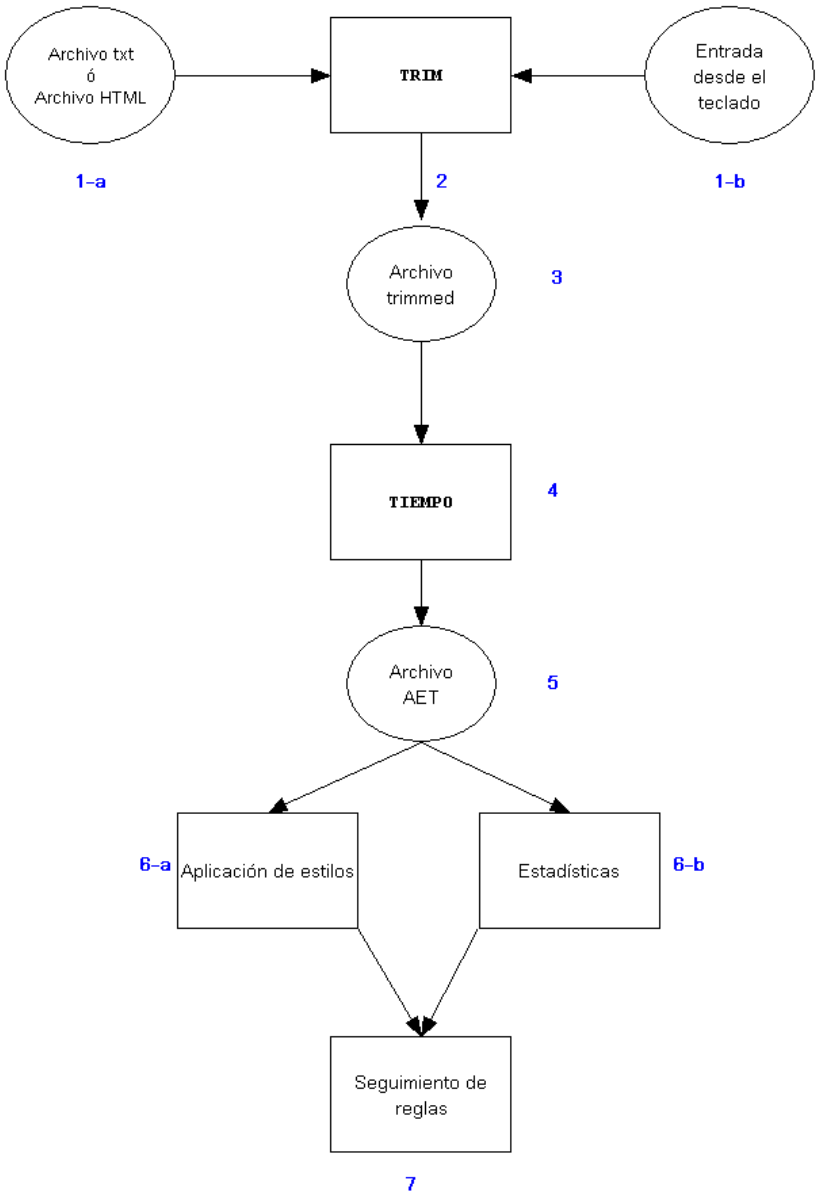

Figura 6.13— Esquema general del funcionamiento de **EXTE**.

A continuación se profundiza en cada paso ilustrado en la Figura 6.3.

1-a: Inicialmente se tiene un archivo de texto o de hipertexto, se carga el archivo en memoria y se utiliza TRIM para transformar las etiquetas HTML a texto simple, remover caracteres especiales de HTML y eliminar espacios redundantes dentro del archivo a analizar. TRIM es un analizador léxico muy pequeño, y por consiguiente su ejecución es muy rápida. El texto resultante (ejecutado en el paso 2) se almacena en otro archivo, llamado en este caso trimmed, por el analizador que lo manipuló inicialmente. Este archivo trimmed es el que manejarán los demás componentes. Para mejores resultados en la transformación de HTML a texto plano, se recomienda utilizar herramientas especializadas (ver ***(Bayer, 2005)*** y ***(HDSE-Software, 2003)***), ya que realizan conversión entre todos los caracteres codificados de HTML, además de proveer personalización sobre las etiquetas HTML transformadas y la longitud de las columnas.

1-b: El usuario puede no contar con un archivo de texto y podría desear ingresar el texto desde el teclado. A esta etapa se le llama *Análisis rápido* y se efectúan casi los mismos pasos que para la entrada de archivos. La diferencia, además del tipo de entrada, radica en que no se almacena el trimmed (en el paso 2); el trimmed se pasa directamente a TIEMPO para su manipulación. El usuario puede, no obstante, guardar el marcado generado en un nuevo archivo. Los resultados del análisis serán los mismos para cualquier tipo de entrada.

2: El texto que se desea analizar está ya en memoria y se transfiere a TRIM para que elimine los espacios.

3: El resultado de TRIM es texto sin espacios redundantes y sin etiquetas HTML; en caso de haberse ingresado desde un archivo, otro archivo (trimmed) se genera con este resultado; si la entrada fue desde el teclado, no se genera un nuevo archivo.

4: Para este punto, la ejecución no es automática. El usuario debe llamar a la función de análisis para continuar con el proceso. Si sucede esto, TIEMPO es llamado y recibe el archivo trimmed para buscar las expresiones temporales que contenga. Este análisis es bastante rápido (no tanto como TRIM), y al terminar, se sigue al paso 5.

5: Una vez que se terminó el análisis realizado por TIEMPO, se genera otro archivo, llamado aet, que contiene el mismo texto que trimmed pero con las expresiones temporales marcadas. El archivo aet

se genera sin importar el tipo de entrada que se tuvo, ya sea 1-a o sea 1-b. Habiendo escrito este archivo en memoria, se sigue con los pasos 6-a y 6-b.

6-a: Como el archivo `aet` ya contiene las marcas de las expresiones temporales, su ubicación y su tipo, es posible aplicar a cada una de ellas el estilo correspondiente. Este proceso tarda más que cualquier otro, ya que hay que analizar cada una de las expresiones temporales e identificar sus propiedades para asignarle un estilo. Una vez que cada una de las expresiones temporales tiene asignado un estilo, el componente de texto estilizado[1] debe desplegarlas y ése es un proceso costoso para la computadora.

6-b: Al mismo tiempo[2] que se realiza el paso 6-a, la capa lógica analiza las expresiones temporales para generar las estadísticas. Terminado este proceso, relativamente rápido, la capa lógica transfiere las estadísticas a la capa visual para que se desplieguen.

7: Este paso es opcional y depende de las acciones que tome el usuario. Dentro del componente de texto estilizado (el que despliega el archivo `aet`), para cada expresión temporal, se puede seguir la trayectoria dentro de la gramática `tiempo.flex`.

Una descripción más detallada sobre las acciones que puede ejecutar el usuario se puede encontrar en el manual de **EXTE**.

Supongamos que se tiene el siguiente texto en un periódico:

```
En este siglo, tras la Revolución de 1910, sólo la creación de un partido ómnibus
donde cupieran los generales revolucionarios y los cientos de partidillos locales,
nos pudo dar 71 años de muy precaria estabilidad, aunque poca o nula democracia.
```

Ejemplo 6.1

[1] El que despliega las expresiones temporales con distintos estilos.
[2] Concurrentemente. Las computadoras con un solo procesador realizan tareas de modo que parezca que se ejecutan simultáneamente. Los sistemas operativos son concurrentes, ya que podemos escuchar música mientras leemos nuestro correo, y aunque el procesador—dado un tiempo—sólo está haciendo una tarea, a nosotros nos parece que lo hace al mismo tiempo. En realidad para una computadora con un procesador, hacer dos tareas al mismo tiempo es imposible, pues sólo hay un "chip" que puede hacer el trabajo en un punto en el tiempo; sin embargo la concurrencia simula la simultaneidad y hace un mejor uso del tiempo muerto del procesador.

Tras seguir los pasos descritos, **EXTE** marcaría esa parte del texto así:

En este siglo, tras la Revolución **de 1910**, sólo la creación de un partido ómnibus donde cupieran los generales revolucionarios y los cientos de partidillos locales, nos pudo dar **71 años** de muy precaria estabilidad, aunque poca o nula democracia.

Ejemplo 6.2

Resaltando las expresiones temporales que se encontraron. En este último ejemplo hay tres expresiones temporales de diferente categoría y por lo tanto se marcan con distintos estilos para ser distinguibles.

7 Resultados

En esta sección se describirán los resultados obtenidos, la precisión del sistema **EXTE** y su comparación con el sistema creado por Saquete y Martínez-Barco (***Saquete & Martínez-Barco, 2000***).

Por las complicaciones que ya se mencionaron, **EXTE** no es capaz de marcar todas las expresiones temporales que un texto contenga, las cuales debían rebasar un umbral para que se tomaran en cuenta dentro de la gramática. El hecho de que **EXTE** no marque algunas de las expresiones temporales de un texto no implica que sean irrelevantes, sin embargo se garantiza que aquéllas no marcadas no aparecen con tanta frecuencia como las que sí.

Para describir el grado de exactitud de **EXTE**, se introducirán conceptos como *precision* y *recall*, el tiempo de ejecución para porciones del algún texto del corpus y se hará una comparación de estos términos con el sistema realizado por Saquete y Martínez-Barco (***Saquete & Martínez-Barco, 2000***).

Existen varias medidas que se usan para los sistemas de recuperación de información (*information retrieval systems*) que pueden utilizarse en este caso. Entre ellas se tienen *precision* y *recall* (***Baeza-Yates & Ribeiro-Neto, 1999***). *Precision* es la medida que describe la relación entre las expresiones temporales correctas con respecto a todas las expresiones temporales que el método detectó como correctas; se puede explicar esta relación para expresiones temporales utilizando la siguiente fórmula:

$$precision = \frac{\#\text{ de ET correctamente marcadas}}{\#\text{ de ET marcadas}}$$

Figura 7.1

De igual modo, *recall* es la relación de las expresiones temporales correctas con respecto a todas expresiones correctas marcadas manualmente; la siguiente fórmula lo muestra:

$$recall = \frac{\#\ de\ ET\ correctamente\ marcadas}{\#\text{ de ET manualmente marcadas}}$$

Figura 7.2

A continuación se darán ambas medidas como resultados del sistema **EXTE** para tres artículos[1]. La columna **ETCM** corresponde al valor *# de ET correctamente marcadas*, **ETM** al valor *# de ET marcadas* y **ETMM** al valor *# de ET manualmente marcadas*.

Artículo	Tamaño	ETCM	ETM	ETMM	Precision	Recall	Tiempo
Excélsior, 18/01/2000 (e000118.txt)	228k	695	843	776	0.824	0.896	4797 ms
La Jornada, 27/04/1997 (j970427.txt)	95kB	251	337	261	0.745	0.962	6236 ms
El Universal, 21/03/1998 (u21MAR98.txt)	156kB	616	783	761	0.787	0.809	5070 ms

Tabla 7.1— Medidas de *precisión*, *recall* y tiempo para tres artículos del corpus.

El proceso para obtener las medidas descritas en la tabla anterior se efectuó de la siguiente manera. Primero se tomó un artículo al azar de Excélsior, otro de La Jornada y otro de El Universal. Se marcaron manualmente cada una de las expresiones temporales ahí contenidas; esto es, realizado por un ser humano, no el programa. El número de expresiones temporales encontradas por el ser humano corresponde al valor de *# de ET manualmente marcadas* (ETMM). Después cada uno de los artículos elegidos se analizó utilizando **EXTE**, es decir, se detectaron las expresiones temporales por el programa. El número total de "expresiones temporales"[2] encontradas por **EXTE** corresponde al valor de *# de ET marcadas* (ETM). Por último, se compararon las expresiones marcadas manualmente con las expresiones marcadas por **EXTE**, definiendo que en caso de no ser iguales, sería una expresión temporal incorrecta. Se muestran algunos resultados del programa de Saquete y Martínez-Barco.

Artículo	Precision	Recall
Artículo #1	0.9615	0.806
Artículo #2	0.777	0.777
Artículo #3	1.0	0.842
Artículo #4	1.0	1.0

Tabla 7.2—Algunos resultados mostrados por Saquete.

[1] Dado que el número de expresiones temporales promedio por artículo oscila entre 900 y 1000, la revisión sólo se hizo en tres artículos elegidos aleatoriamente.

[2] "Expresiones temporales" porque no todo el texto que **EXTE** marcó *realmente* fue expresión temporal, es decir, algunas expresiones temporales no lo eran.

Los resultados mostrados en la Tabla 7.1 reflejan numéricamente la eficiencia de **EXTE**, y considerando el número de expresiones temporales que reconoce, son los números esperados. Más alto en el valor de *recall* y más bajo en el valor de *precision* con respecto a los resultados mostrados para el programa de Saquete y Martínez-Barco (Tabla 7.2). También debe considerarse que el desglose de los números de expresiones temporales (como ETM, ETMM y ETCM de la Tabla 7.1) encontradas en los artículos de Saquete y Martínez-Barco es desconocido. Otro punto muy importante es que no se puede realizar una comparación directa con el sistema de Saquete y Martínez-Barco puesto que los artículos son diferentes. Los valores que más fluctúan son los de *recall*; dicha fluctuación se debe a la manera en la que los artículos de cada periódico fueron escritos. Un ejemplo claro se dio durante el reconocimiento de expresiones temporales del periódico La Jornada, ya que el mismo nombre contiene parte una expresión temporal `la jornada`. Ésta no es una expresión temporal completa, pero contiene la sintaxis de una y **EXTE** la marca como tal. Por eso la medida obtenida de *precision* difiere (más que para los otros periódicos) de la de *recall*. Esto se da porque hay una diferencia mayor entre el valor de *# de ET manualmente marcadas* y el valor de *# de ET marcadas*, ya que **EXTE** marcó como dos expresiones temporales todos los encabezados de tipo `La Jornada 14 de marzo de 1999`, cuando realmente debió haber marcado sólo una.

Dentro de este análisis de resultados, se apreció que la verificación de género y número no afectó considerablemente la medida *precision*, esto debido a que son pocas las expresiones temporales que contienen errores de ese tipo:

* `Durante la mes de noviembre`

Ejemplo 7.1

Aunque dentro de las pruebas que se realizaron durante la integración de **EXTE** (6.1.4 Pruebas) se marcaron expresiones temporales mal formadas (por puntuación, faltas de ortografía, espacios innecesarios, etc.), puede observarse que no afectaron en gran medida los resultados. También es importante saber que en el corpus llegaron a encontrarse errores ortográficos, "errores de dedo", por llamarlos de alguna manera común; se infiere que la corrección de estilo del periódico que se publica en la red es menos rigurosa que la aplicada para la publicación impresa. Dado que el corpus se descargó de páginas web, es más fácil encontrar errores dentro de los textos. Esta última razón afecta

las medidas de eficiencia. Si se encontrara la expresión temporal como la siguiente, que introduce un guión (-) para separar la palabra en sílabas (por ejemplo, al final de una línea):

… dijo que el mar-
tes saldrían los resultados.

Ejemplo 7.2

EXTE no marca el martes como una expresión temporal.

Quizá garantizando que todas las palabras del corpus están bien formadas, podría aumentar la eficiencia de **EXTE**. Para un humano

* Jueves quincede febre-ro

Ejemplo 7.3

Es una expresión temporal, o al menos es posible reconocerla; mas para este sistema –y en general cualquier otro— lograr reconocer esa expresión temporal le llevaría, por lo menos, otras dos docenas de millares de líneas de código, es decir, aproximadamente unos cien mil estados más.

El tipo de expresiones temporales que **EXTE** no reconoce varía mucho. Recordemos que se eliminaron todos los números escritos mayores a mil (5.4 Estadísticas) y hay expresiones temporales que mencionan al año 2000 escrito con letras (año dos mil). Igualmente todas las expresiones temporales que por no rebasar el umbral fueron descartadas. Las expresiones temporales que tienen más recursión, por ejemplo:

En los años 56, 57, 59-62

Ejemplo 7.4

Y las expresiones temporales que contienen dos o más expresiones temporales –a veces cortadas— o que no siguen un orden común, por ejemplo:

A lo largo de los siglos cuarto y XIX

Ejemplo 7.5

Los ejemplos anteriores son expresiones temporales, pero hacer marcable ese tipo de expresiones temporales costaría a TIEMPO muchos estados más.

Conclusiones

Determinar cuáles son las expresiones de tiempo en un texto cualquiera es una tarea muy sencilla para los seres humanos; desarrollar un algoritmo para que un programa de computadora pueda hacerlo automáticamente no simple.

Por ejemplo

En los setenta la moda era tener cuerpo de esqueleto

Ejemplo 7.6

Donde los setenta corresponde a la década de los años 1970 a 1979. Y

El pintor colombiano presentó cincuenta cuadros nuevos; en los cincuenta se muestran las torturas a prisioneros.

Ejemplo 7.7

Donde los cincuenta corresponde a la cantidad de cuadros.

El trabajo aquí desarrollado intenta avanzar en esta tarea que ayudará en múltiples aplicaciones del procesamiento de lenguaje natural: determinar las reglas para reconocer en forma automática las expresiones temporales más comunes a nivel léxico y llevarlo a la práctica. Tanto en el lenguaje natural como en un lenguaje de programación, la verificación semántica es esencial; empero para este trabajo, la verificación semántica es mínima y gran parte de ella se realizaría en las etapas posteriores al análisis de la oración completa.

Aunque otros autores han trabajado en este sentido, han considerado el español peninsular y no su variante mexicana. Además se fijó el objetivo de realizarlo para textos de dominio no restringido, por lo que fueron analizados textos de diversos periódicos mexicanos y de distintos temas. Queda para un proyecto a futuro hacer un módulo o programa adicional que realice la verificación semántica de las expresiones temporales marcadas. Como **EXTE** ya detectó léxicamente cuáles son, queda solamente interpretarlas.

Aportaciones

Las principales acciones realizadas en este trabajo son:

- Análisis de textos mexicanos de tres periódicos distintos.
- Identificación de las expresiones temporales más comunes.
- Desarrollo de una gramática para el reconocimiento de las expresiones temporales.
- Desarrollo de un reconocedor de la gramática, basado en generadores de analizadores léxicos.

La realización de este trabajo ha permitido llegar a las siguientes conclusiones:

- El análisis realizado y la gramática resultante servirán de base para desarrollos futuros, ya que permiten obtener un valor de precision promedio de 78.3% y uno de recall promedio de casi 89% en las pruebas realizadas. Aunque no se puede hacer una comparación directa con otros sistemas, existen referencias de trabajos para el inglés entre 83% y 95% de precision y entre 80% y 96% de recall con métodos más complejos. Los valores obtenidos en este trabajo son más bajos por permitir la identificación de números como posibles fechas (`18.11.1980` ó `15/02-2005`), considerando el criterio descrito en el siguiente punto.
- Confirmamos que es necesario marcar todas las posibles expresiones temporales a nivel léxico aunque de antemano se sepa que algunas de las reglas no son exclusivas para su identificación, por la cantidad de su uso. En un análisis sintáctico posterior se desambiguarán estas frases.
- La gramática desarrollada para reconocer las expresiones temporales se ha reducido a su forma más simple, sin que se tengan revisiones de género y número rigurosas; a pesar de esto no parece haberse degradado su función. Sin embargo, podría considerarse en el futuro y evaluar su impacto en el valor de precision total.
- **EXTE** facilita al usuario la detección de expresiones temporales alejándolo de la necesidad de utilizar herramientas específicas de búsqueda (`grep`, por ejemplo), dándole un ambiente configurable, adaptable y fácil de manejar.

Bibliografía

- Aho, A. V., Ravi, S., & Jeffrey, U. D. (1990). *Compiladores: Principios, técnicas y herramientas.* México: Pearson.
- Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval.*
- Bayer, M. (2005). *html2text for Linux*. Obtenido de http://userpage.fu-berlin.de/~mbayer/tools/html2text.html
- Chinchor, N. (1997, September). MUC-7 Named Entity Task Definition Dry Run Version. p. Version 3.5 NE/training/guidelines/NE.task.def.3.5.ps.
- Chomsky, N. (1956). Three Models for the Description of Language. *IRE Transaction on Information Theory , II*, 113-124.
- Davidoff, L. L. (1989). *Introducción a la Piscología* (3ª ed.). México D.F.: Mc Graw Hill.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns : elements of reusable object-oriented software.* Addison Wesley.
- Hausser, R. (1999). *Foundations of Computational Linguistics* (1ª ed.). Berlín: Springer Verlag.
- HDSE-Software. (2003). *Convert Html to Text*. Obtenido de http://www.hdse.org/html2txt/
- JFlex, w. (2005). *The Fast Scanner Generator for Java*. Obtenido de http://www.jflex.de
- Mani, I., & Wilson, G. (2000). Robust Temporal Processing of News, Proceedings of the ACL'2000 Conference., (págs. 69-76). Hong Kong.
- McEnery, T., & Wilson, A. (marzo de 2006). *Corpus Linguistics.* Recuperado el marzo de 2006, de Corpus Linguistics: http://bowland-files.lancs.ac.uk/monkey/ihe/linguistics/contents.htm
- Pustejovsky, J., Castaño, J. M., Ingria, R., Sauri, R., Gaizauskas, R. J., Setzer, A., et al. (2003). TimeML: Robust Specification of Event and Temporal Expressions in Text. *In Proceedings of the IWCS-5 Fifth International Workshop on Computational Semantics New Directions in Question Answering.*, (pp. 28-34).
- Saquete, E., & Martínez-Barco, P. (2000, November). Grammar specification for the recognition of temporal expressions. *Proceedings of Machine Translation and Multilingual applications in the new millenium, MT2000.* , 21-27.
- Saquete, E., Martínez-Barco, P., & Muñoz, R. (2002). A grammar-based system to solve temporal expressions in Spanish Texts. *Lecture Notes in Computer Science. PorTal* (págs. 53-62). Faro, PORTUGAL. 2002a.
- Saquete, E., Martínez-Barco, P., & Muñoz, R. (2002b). Recognising and Tagging Temporal Expressions in Spanish. *3rd International Conference on Language Resources and Evaluation (LREC'2002).* Las Palmas, SPAIN.
- Wintner, S. (2002). Computational Linguistics and Modular Context-Free Grammars (1ª ed.).

Apéndices

A.1 Gramática de TIEMPO

A continuación se muestra la gramática de TIEMPO

```
Espacio={SaltoLínea}|[\b\t\f]
SaltoLínea =[\r|\n|\r\n]
NúmeroEscrito={NúmeroCardinalContinuación}|{NúmeroCardinalInicial}
Número=[0123456789]+
Año=[0-9]{2,4}
NúmeroTodos={NúmeroEscrito}|{Número}
NúmeroCardinalInicial="un"|"uno"|"una"|"dos"|"tres"|"cuatro"|"cinco"|"seis"|"siete"|"ocho"|"nueve"|"diez"
NúmeroCardinalContinuación="once"|"doce"|"trece"|"catorce"|"quince"|"dieciséis"|"diecisiete"|"dieciocho"
        |"diecinueve"|"veintiuno"|"veintián"|"veintidós"|"veintitrés"|"veinticuatro"|"veinticinco"
        |"veintiséis"|"veintisiete"|"veintiocho"|"veintinueve"|"treinta"
        |"treinta y " {NúmeroCardinalInicial}
        |"cuarenta"|"cuarenta y " {NúmeroCardinalInicial}|"cincuenta"
        |"cincuenta y " {NúmeroCardinalInicial}|"sesenta"
        |"sesenta y " {NúmeroCardinalInicial}|"setenta"
        |"setenta y " {NúmeroCardinalInicial}|"ochenta"
        |"ochenta y " {NúmeroCardinalInicial}|"noventa"
        |"noventa y " {NúmeroCardinalInicial}
        |"cien"|"ciento"|"doscientos"|"trescientos"|"cuatrocientos"|"quinientos"
        |"seiscientos"|"setecientos"|"ochocientos"|"novecientos"
NúmeroOrdinal = (1{Espacio}?[ºª°])|(primer[oa]?)
Mes="enero"|"febrero"|"marzo"|"abril"|"mayo"|"junio"|"julio"|"agosto"|"septiembre"|"octubre"|"noviembre"|"d
iciembre"
Día="lunes"|"martes"|"miércoles"|"jueves"|"viernes"|"sábado"|"domingo"
Estación="primavera"|"verano"|"otoño"|"invierno"
SeparadorRango=[/a-]
Romano = [iIxXvVmMcCdDlL]+
Artículo=el|la|los|las
Adjetivo = est[eao]s?|es[ea]|aquel|aquella|aquell[oa]s|es[ao]s
FechaDDdMMMdAAAA=({NúmeroOrdinal}|{NúmeroTodos}){Espacio}"de"{Espacio}({Mes})
FechaMDdA = {Mes}{Espacio}{NúmeroTodos}
Fecha={FechaSimple}|{FechaMDdA}|{FechaDDdMMMdAAAA}
FechaSimple=hoy|ayer|mañana|antier|anteayer|(pasado mañana)|vísperas?|ahora
Hora = [0-9]{1,2}([:.][0-
9]{1,2})?({Espacio}de{Espacio}la{Espacio}(mañana|tarde|noche))?({Espacio}horas)?({Espacio}?[ap][.]?m[.]?)?

Puntuación = \u003A|\u003B|\u002C|\uu002E
```

Expresión encontrada		Estado
"algún"|"alrededor"|"año con"|"cada"|"cuando"|"dentro"| "día de la bandera"|"día del trabajo"| "semana santa"|"luego de"	—»	<AISLADAS>
"antes"	—»	<ANTES_PREPOSICION>
"tras"	—»	<TRAS_PREPOSICION>
"hacia"	—»	<HACIA_PREPOSICION>
"hasta"	—»	<HASTA_PREPOSICION>
({Día}{Espacio})?{Fecha}	—»	<FECHA>
"durante"	—»	<DURANTE_PREPOSICION>
"desde"	—»	<DESDE_PREPOSICION>
"para"	—»	<PARA_PREPOSICION>
"por"	—»	<POR_PREPOSICION>
"hace"	—»	<HACE_PREPOSICION>
"entre"	—»	<ENTRE_PREPOSICION>
"en"	—»	<EN_PREPOSICION>
(("casi"|"cerca de"){Espacio})?{NúmeroTodos}	—»	<NUM>
"del"|"de las"|"de los"|"de la"	—»	<DE_PREPOSICION><ARTÍCULO>
"al"|"a los"|"a las"	—»	<A_PREPOSICION><ARTÍCULO>
"a"	—»	<A_PREPOSICION>
{Artículo}	—»	<ARTÍCULO>
{Adjetivo}	—»	<ADJETIVO>
"de"	—»	<DE_PREPOSICION>
{Espacio}+	—»	<NULO>
[<>_,;.\{\}\[\]\(\):\!i\?¿\"'-]+	—»	<NULO>
[:jletter:]+	—»	<NULO>

Tabla 7.3— Tabla de transiciones para las posibles cadenas que encuentra **EXTE**. Dada alguna de las cadenas en la columna "Expresión encontrada" se realiza una transición a un "Estado" (columna derecha). El estado es el encargado de ejecutar las reglas definidas en la gramática. Es posible que dentro de un estado se tenga una referencia a uno otro; para mostrar esta tabla de forma más compacta se omitieron dichas transiciones recursivas.

A.2 Gramática de Saquete

Ahora se muestra la gramática de Saquete y Martínez-Barco

Fecha → dd + " /" + mm + "/" + (aa)aa →
Fecha → dd + "-" + mes + " –"+ (aa)aa
Fecha → dd + "de" + mm + "de" + (aa)aa
Fecha → ("El") + diasemana+ dd + "de"+ mes + "de" + (aa)aa
Dd → ["01"|"02"|"03"|....|"31"]
Dia → ["uno"|"dos"|... |"treinta y uno"]
Mm → ["01"|"02"|"03"|...|"12"]
Mes → ["enero"|
"febrero"|"marzo"|"abril"|"mayo"|"junio"|"julio"|"agosto"|"septiembre"|"octubre"|"noviembre"|"diciembre"]
A → ["1"| "2" | "3"|..."9"| "0"]
Diasemana → ["lunes"|"martes"|"miércoles"|"jueves"|"viernes"|"sábado"|"domingo"]
referencia → "ayer"
referencia → "mañana"
referencia → "anteayer"
referencia → "anoche"
referencia → "el"+ "próximo" + ["día" | "mes" | "año"]
referencia → "un" + ["día" | "mes" | "año"] + "después"
referencia → num + ["días" | "meses" | "años"] + "después"
referencia → "un" + ["día" | "mes" | "año"] + "antes"
referencia → num + ["días" | "meses" | "años"] + "antes"
referencia → "dentro" + "de" + "un" + ["día" | "mes" | año"]
referencia → "dentro" + "de" + num +["días"|"meses"| "años"]
referencia → "el" + "pasado" + ["día" | "mes" | "año"]
referencia → "el" + ["día" | "mes" | "año"] + "siguiente"
referencia → "los" + num + ["días" | "meses" | "años"] + "siguientes"
referencia → "el" + ["día" | "mes" | "año"] + "pasado"
referencia → "los" + num + ["días" | "meses" | "años"] +"pasados"
num → ["dos" | "tres" | "cuatro" | "cinco" | ....]
"ayer" → Dia(FechaP) –1 / Mes(FechaP) / Año(FechaP)
"mañana" → Dia(FechaP) +1 / Mes(FechaP) / Año(FechaP)
"anteayer" → Dia(FechaP) –2 / Mes(FechaP) / Año(FechaP)
"anoche" → Dia(FechaP) –1 / Mes(FechaP) / Año(FechaP)
"el"+ "próximo"+"día" → Dia(FechaP)+1 / Mes(FechaP) / Año(FechaP)
"un"+"mes"+"después" → [DiaI/Mes(fechaAnterior)+1/Año(fechaAnterior)-- DiaF/Mes(fechaAnterior)
+1/Año(fechaAnterior)]
num+"años"+ "después" → [01/01/ Año(fechaAnterior) + num --31/12/ Año(fechaAnterior) + num]
"un" + "día" + "antes" → Dia(fechaAnterior)-1/Mes(fechaAnterior)/Año(fechaAnterior)
num+"meses"+ "antes" → [DiaI/Mes(fechaAnterior) -num / Año(fechaAnterior) – DiaF/ Mes(fechaAnterior)-num /
Año(fechaAnterior)]
"dentro"+"de"+"un"+ "año" → [01/01/ Año(fechaAnterior) +1 – 31/12/ Año(fechaAnterior) +1]
"dentro"+"de"+num+ "días" → Dia(fechaAnterior)+num / Mes(fechaAnterior) / Año(fechaAnterior)
"el" + "pasado" + "día" → Dia(fechaAnterior)-1/Mes(fechaAnterior) / Año(fechaAnterior)
"el"+"mes"+"siguiente" → [DiaI / Mes(fechaAnterior) +1 / Año(fechaAnterior) -- DiaF / Mes(fechaAnterior)+1 /
Año(fechaAnterior)]
"los"+num+"años"+ "siguientes" → [01/01/Año(fechaAnterior) -- 31/12 / Año(fechaAnterior) +num]
"el" + "día" + "pasado" → Dia(fechaAnterior)-1/Mes(fechaAnterior) / Año(fechaAnterior)

“los”+num+“meses”+ “pasados” → [DiaI/Mes(fechaAnterior) - num / Año(fechaAnterior) – DiaF/Mes(fechaAnterior)– 1 / Año(fechaAnterior)]