Reconocimiento automático de eventos en textos

Informe de Proyecto de Grado

Alan Descoins

Tutores:
MSc. Guillermo Moncecchi
Ing. Aiala Rosá

Facultad de Ingeniería
Universidad de la República

Proyecto de grado

2010

Resumen

En el presente trabajo se describe la concepción de un sistema de reconocimiento de eventos para textos en idioma español, basado en técnicas de aprendizaje automático. Se utilizó como base un esquema de anotación de texto denominado SIBILA, y un corpus de texto etiquetado para la tarea por dos estudiantes avanzados de lingüística.

Se estudió el problema de reconocimiento de eventos en un marco más general, como instancia particular de una clase de problemas llamados de clasificación secuencial. Se realizó un análisis de varias técnicas que se utilizan para resolver problemas de este tipo, no solo desde un punto de vista pragmático sino intentando comprender su funcionamiento con un nivel razonable de detalle. Para la tarea se eligió utilizar dos paradigmas de aprendizaje radicalmente diferentes, que han dado buenos resultados en numerosos trabajos del área: los *Conditional Random Fields* (CRF) y los *Support Vector Machines* (SVM).

A partir del corpus etiquetado se entrenaron clasificadores basados en estas técnicas, y también se realizó un análisis para ver los niveles de acuerdo de etiquetadores humanos. La evaluación de los clasificadores arrojó resultados muy similares a aquellos obtenidos por otros trabajos para el idioma inglés, logrando una medida F de $76,72\,\%$ para CRF y $80,34\,\%$ para SVM, aunque todavía se está lejos de alcanzar una performance de niveles humanos.

Palabras clave: eventos, esquema SIBILA, aprendizaje automático, clasificación secuencial, conditional random fields, support vector machines.

Índice general

Capítulo 1

Introducción

La explosión de la Internet en las últimas décadas ha revolucionado el tratamiento de diversos problemas del área de procesamiento del lenguaje natural (PLN). Grandes volúmenes de texto pasaron rápidamente a estar disponibles, lo que junto con el aumento del poder de cómputo de los ordenadores, determinó que técnicas que solo formaban parte del ámbito teórico pasaran a tener un papel preponderante en la práctica. (respecto a una medida de performance) Existen dos paradigmas predominantes para la construcción de sistemas que resuelven tareas de PLN: el uso de reglas y el aprendizaje automático. Hasta la década del 80, la mayoría de los sistemas estaban basados en reglas que codificaban el conocimiento de personas expertas en cierta disciplina. La tarea de creación estas reglas no es en general sencilla pues la cantidad de reglas que se requieren codificar para obtener un desempeño aceptable puede llegar a ser sumamente extensa. Casos particulares deben ser tratados como tales, con reglas especiales. Para empeorar la situación, el lenguaje natural es muy cambiante; palabras y expresiones surgen o cambian su uso a diario. Mantener un sistema basado en reglas escritas a mano para que se adapte a nuevos usos del lenguaje puede ser sumamente costoso. La única forma de mejorar el desempeño de un sistema basado en reglas es aumentando su complejidad, y llega un punto en el que estos sistemas tienden a volverse más y más inmanejables.

El aprendizaje automático es una rama de la inteligencia artificial cuyo cometido es desarrollar técnicas para que las computadoras puedan *aprender*. El hecho de aprender refiere a crear programas que sean capaces de mejorar su desempeño (respecto a una cierta medida) al realizar alguna tarea, según van adquiriendo experiencia. En el PLN,

los métodos de aprendizaje automático vienen ganando terreno ya que son capaces de inferir conocimiento a partir de grandes cantidades de datos, en general sin necesidad de ser alimentados con el *conocimiento experto* que sí necesitan los sistemas basados en reglas. El costo de construir los sistemas se reduce enormemente y a la vez se vuelven más manejables. Si el uso del lenguaje cambia, un sistema que utiliza aprendizaje automático es capaz de adaptarse sin sufrir modificaciones. El aprendizaje automático se puede usar también para tareas donde el mismo diseño de un sistema con reglas puede no ser trivial (por ejemplo, reconocimiento de voz). La principal desventaja que tienen estos métodos resulta de que para que funcionen se necesita contar con muchos ejemplos con los que alimentarlos, por ejemplo, grandes cantidades de texto etiquetado, lo cual no siempre es fácil de conseguir. [BM97]

1.1. Eventos en textos

Es difícil dar una definición concisa de evento o de lo que constituye un evento. Intuitivamente, se trata de cualquier tipo de **situación o acontecimiento** que ocurre, o elementos que describen un **estado o circunstancia** [PCI+03]. Los eventos pueden ser expresados por verbos conjugados o en infinitivo, predicados en general y frases preposicionales. Sobre un texto plano, pueden delimitarse entre las etiquetas `<evento>` y `</evento>`. Por ejemplo

1. *"El fugitivo* `<evento>`***estaba***`</evento>` `<evento>`***acorralado***`</evento>` *por la policía."*
2. *"Este lunes* `<evento>`***se registraron***`</evento>` `<evento>`***nevadas***`</evento>` *en Nueva Palmira."*
3. *"El testigo lo* `<evento>`***reconoció***`</evento>` *ante el juez."*
4. *"Tras* `<evento>`***haber sido reprimidos***`</evento>` *por la policía, los manifestantes* `<evento>`***se retiraron***`</evento>`*."*
5. *"La IMM* `<evento>`***hizo***`</evento>` `<evento>`***cerrar***`</evento>` *el local nocturno."*

En el contexto de este trabajo se cuenta con un esquema que establece reglas claras con guías de cómo se deben marcar los eventos, de manera de reducir las ambigüedades al mínimo.

1.2. Reconocimiento automático de eventos

El objetivo es construir un programa cuya entrada conste de texto plano en español, y su salida indique los eventos a los que se hace referencia en el texto. Por ejemplo, ante la entrada

"El precio del dólar comenzó a caer a causa de la crisis."

una salida correcta podría ser el texto

"El precio del dólar `<evento>`***comenzó***`</evento>` *a* `<evento>`***caer***`</evento>` *a causa de la* `<evento>`***crisis***`</evento>`*."*

La dificultad de la tarea radica en que una palabra puede a veces tener carácter eventivo y otras veces no. Por ejemplo en *"durante la* ***cena*** *se habló de varios asuntos"* la palabra *cena* hace referencia a un evento mientras que en *"la* ***cena*** *está servida"* la misma palabra hace referencia a la comida (física) y por lo tanto no es un evento. En el primer ejemplo la palabra *durante* da la pauta de que *cena* se trata de un evento. Desafortunadamente, este tipo de palabras no siempre están presentes; en una entrada como *"una* ***cena*** *con tres invitados"* ya no se cuenta con un indicador de este tipo, pero *cena* mantiene su carácter eventivo.

1.2.1. Abordaje

Para lograr construir un sistema de reconocimiento automático de eventos se propone utilizar métodos de *aprendizaje automático*. La técnica en particular será de aprendizaje **supervisado**. En general, el aprendizaje de tipo supervisado se utiliza para inferir el valor de una función a partir de datos de entrenamiento. Estos datos normalmente vienen en forma de un vector con una cierta entrada y otro con los resultados esperados, lo que posibilita a un algoritmo descubrir ciertas reglas o relaciones subyacentes al proceso que genera los resultados deseados a partir de la entrada. De esta forma este algoritmo es capaz de predecir el valor correspondiente a cualquier dato de entrada, incluso aquellos que nunca vio, efectivamente generalizando el comportamiento que observó en los ejemplos de entrenamiento.

En este caso, el uso de aprendizaje supervisado implica que el sistema se nutre de una entrada que consta de un conjunto de textos y sus correspondientes marcas de eventos.

Este conjunto de textos se denomina *corpus etiquetado o anotado*. Se cuenta con un corpus de textos en español especialmente construido para la tarea (aunque dentro de un proyecto más general), con los eventos marcados por parte de dos estudiantes avanzadas de lingüística. En su mayor parte, las anotaciones fueron realizadas por ambas personas de manera independiente.

Se verá que el problema de delimitar los eventos en oraciones puede verse como el de asignar ciertas etiquetas a secuencias de palabras o, más correctamente, unidades denominadas *tokens* (palabras, números y símbolos). De esta forma, el *corpus* se segmentará en oraciones donde cada *token* tendrá asignada una etiqueta, y este conjunto de secuencias etiquetadas será la entrada de un sistema de aprendizaje. Este sistema luego realizará cierta inferencia que le permitirá etiquetar cada uno de los *tokens* de nuevas secuencias. Un punto sumamente importante es que al realizar el etiquetado se tendrá en cuenta el contexto y los resultados no serán simplemente la asignación de una etiqueta a cada *token* por separado, sino que se intentará obtener la mejor *secuencia de etiquetas* (de manera global) para cada oración. Los problemas que pueden reducirse a etiquetar secuencias de esta manera se conocen como problemas de **clasificación secuencial**. En este proyecto se hará bastante hincapié en mostrar algunas técnicas para resolver problemas de clasificación secuencial.

1.2.2. Motivación

Se ha sugerido el problema de reconocimiento de eventos como importante en aplicaciones complejas de métodos de procesamiento del lenguaje natural. Los textos a menudo describen secuencias de eventos en una línea del tiempo. En el campo de la extracción automática de resúmenes, obtener los eventos puede ayudar a obtener mejores resúmenes cuando éstos tienen que poner foco en ciertos hechos concretos [DRA03]. De la misma forma, el tener información de este tipo puede ser crucial en sistemas de respuestas a preguntas (*question answering*) cuando se trata de responder preguntas relacionadas a eventos o situaciones en el mundo (estados, acciones, propiedades, etc.) [Pus02, SKVP05].

El reconocimiento de eventos puede situarse en el área de la extracción de información. Actualmente está siendo aplicado exitosamente en diferentes áreas como bioinformática y clasificación de textos. En el futuro también podría ser útil para otras

tareas como el entendimiento de la narrativa y la creación de bases de datos de hechos u acontecimientos a partir de textos [SKVP05].

1.3. Estructura de la documentación

La documentación se organiza de la siguiente forma: en el capítulo 2 se presenta un panorama general del trabajo y dónde se encuentra enmarcado. Se realiza además un estudio del estado del arte, viendo los resultados y las técnicas aplicadas por otros trabajos.

El capítulo 3 trata del acercamiento al problema, sin entrar en detalles de implementación. Se verá que el problema de reconocer eventos es uno de varios problemas conocidos como de clasificación secuencial, y se estudiarán las acciones que implica resolverlos. Se prestará atención a qué implica querer determinar qué tan bien se comporta un sistema y finalmente se plantearán métricas concretas para evaluar el desempeño.

En el capítulo 4 se describen las decisiones de implementación y cómo se terminó desarrollando el sistema. Se describen las herramientas externas que se utilizaron, tanto para el preprocesamiento del corpus como para implementar las técnicas de clasificación secuencial que se utilizaron.

El capítulo 5 presenta los resultados de la evaluación de los clasificadores. En principio se presentan resultados parciales que se obtuvieron al realizar diversas pruebas, para culminar con la evaluación final y una comparación de los resultados con los del estado del arte para el idioma inglés.

Se finaliza con las conclusiones e ideas para el trabajo futuro, en el capítulo 6.

Además de este documento se cuenta con otro, en el que se introducen con mayor grado de detalle los métodos de clasificación secuencial [Des11]. Dicho documento es un complemento del presente y fue la salida de la investigación realizada en lo que respecta a métodos de aprendizaje automático.

Capítulo 2

Marco de trabajo

En esta sección se verá un panorama general del trabajo. Se comenzará viendo que es parte de un proyecto de análisis temporal de textos más ambicioso. Se mostrarán los insumos con los que se cuenta para poder resolver el problema planteado y se culminará con una reseña de trabajos que se han hecho en el área, a modo de tener un panorama del estado del arte en materia de reconocimiento de eventos.

2.1. El proyecto Temantex

El grupo PLN del InCo está trabajando desde inicios de 2009 en el proyecto "Temantex: Análisis temporal de textos en español". En líneas generales el proyecto se propone identificar los eventos o sucesos mencionados en textos y determinar el momento en que éstos ocurrieron, si es que efectivamente ocurrieron. Lograr el reconocimiento automático de eventos es por tanto uno de los pilares del proyecto Temantex. También existen otros, como el reconocimiento de las expresiones temporales[1] en textos y el reconocimiento de relaciones temporales entre eventos (con el objetivo de poder ordenarlos en una línea del tiempo) y de eventos con intervalos o instantes denotados por expresiones temporales.

[1] Una expresión temporal es una secuencia de *tokens* que denota tiempo, es decir, expresa algún momento en el tiempo, una duración, o una frecuencia. Por ejemplo "12 de octubre", "5 minutos", "cada varias horas", "el viernes pasado", etc.

2.2. El esquema de anotación SIBILA

El esquema de anotación SIBILA fue creado en marco del proyecto Temantex y conforma la base para la tarea del presente trabajo. Está basado en otro esquema de anotación de eventos para el inglés, denominado TimeML [PCI+03]. Lo que aquí se expone no es más que un resumen de lo que se encuentra detallado en [WMR08], el documento que describe formalmente el esquema.

Se comenzará describiendo brevemente qué se consideran eventos según el esquema de anotación SIBILA, con más énfasis en ejemplos que en la formalidad.

2.2.1. Eventos según SIBILA

Según la definición del esquema, se considera un **evento** cualquier tipo de situación o acontecimiento denotado por un **predicado**. Los eventos pueden ser categorizados en tres tipos

Acciones Acontecimientos llevados a cabo voluntariamente por un sujeto agente.

- *"Los antropólogos forenses* `<evento>`***delimitaron***`</evento>` *el predio."*
- *"El criminal* `<evento>`***se fugó***`</evento>` *del establecimiento penitenciario."*

Procesos Acontecimientos desencadenados espontáneamente o causados por una fuerza externa al proceso.

- *"Los árboles* `<evento>`***están floreciendo***`</evento>` *prematuramente por las altas temperaturas."*
- *"Se prevé que los fuertes vientos* `<evento>`***derrumbarán***`</evento>` *varios techos."*

Estados Situaciones que se mantienen a lo largo de un período o son permanentes.

- *"El tránsito* `<evento>`***está detenido***`</evento>` *a causa de los cortes de ruta."*
- *"Los ejércitos dieron la* `<evento>`***guerra***`</evento>` *por comenzada."*

Si bien se puede intuir de los ejemplos que la mayoría de los verbos denotarán eventos, estos pueden también constar de palabras con otras categorías gramaticales como nombres y adjetivos.

Para reducir ambigüedades al mínimo, el esquema plantea reglas claras de anotación que determinan cómo se deben marcar los eventos e intentan abarcar la mayor cantidad de casos posibles. No se verán estas reglas aquí en detalle. Los ejemplos presentados en 1.1 sumados a algunos que se verán en esta sección pueden servir para hacerse una idea de lo que se espera ver en un *corpus* de texto etiquetado con este esquema.

2.2.2. Atributos

La etiqueta `<evento>` se complementa con varios atributos que aportan mayor cantidad de información. A continuación se listan.

id Atributo numérico, que funciona como identificador único del evento.

categoria Representa la categoría gramatical del evento, como es asignada por un analizador morfosintáctico. Toma valores en el conjunto {`VERBO, NOMBRE, ADJETIVO, PRONOMINAL, OTRO`}. En caso que el evento sea una frase preposicional, tomará el valor `OTRO`.

clase Marca la clase del evento. Por ejemplo, si se trata de una acción intencional o causal (*"el juez* ***ordenó*** *reabrir el expediente"*), de un estado (*"la multinacional se encuentra* ***interesada*** *en la oferta"*), entre otras. Toma valores en {`REPORTE, PERCEPCION, ASPECTO, ACCION_INT_CAUS, ESTADO_INT, ESTADO, EXISTENCIA, OCURRENCIA`}.

item_lex Es un atributo opcional que se usa para registrar eventos cuya mención se omite porque el predicado que los nombra puede ser recuperado recurriendo a otra mención en el texto. Por ejemplo *"En el norte del país* `<evento>llovió</evento>` *abundantemente el sábado y* `<evento item_lex=''llovió''></evento>` *el domingo."*.

determinacion Es obligatorio solo para los eventos referidos por un nombre y toma valores en {`DEFINIDO, INDEFINIDO, DESUSO`} de acuerdo con el especificador del grupo nominal del cuál es núcleo el evento.

forma_verbal Es obligatorio solo para eventos de categoría `VERBO`. Toma valores en {`INFINITIVO, GERUNDIO, PARTICIPIO, FORMA_FINITA`}, los que pueden ser determinados por un analizador morfosintáctico.

modo Al igual que en el caso anterior, es obligatorio para los eventos verbales. Toma valores en {`INDICATIVO, SUBJUNTIVO, CONDICIONAL, IMPERATIVO`}.

tiempo Solo es obligatorio para eventos verbales que tengan el valor `FORMA_FINITA` para el atributo **forma_verbal**. Toma valores en {`PASADO, PRESENTE, FUTURO`}, y representa el valor temporal que efectivamente tiene en el texto el evento referido (que puede ser distinto del derivado por un analizador morfosintáctico para la palabra que lo constituye).

polaridad Toma valores en {`NEGATIVA, POSITIVA`}, siendo el segundo el valor por defecto.

modalidad Es opcional según datos del entorno del evento que se interpreten como índices de modalidad. Por ejemplo *"Hay inquietud en el gobierno por el posible* `<evento modalidad=''POSIBLE''>`***paro***`</evento>` *de la semana próxima."*

factividad Representa el grado de certeza del enunciador respecto a la ocurrencia del evento referido. Toma valores en {`SI, NO, FUTURO_PROG, FUTURO_NEG, POSIBLE, INDEF`} según el evento se haya realizado o no, esté programado para el futuro o sea su no ocurrencia la que esté programada, la realización sea posible o no esté definida. Para más detalles se puede consultar [WMR09].

2.2.3. Vínculos entre eventos

El esquema de anotación permite representar **vínculos** entre eventos con sus correspondientes tipos, tanto en el caso de vínculos aspectuales (etiqueta `<vinculo_asp>`) como de subordinación (etiqueta `<vinculo_sub>`). Todo vínculo tiene al menos una referencia a los dos eventos relacionados así como un atributo **relacion** que toma diferentes valores dependiendo del tipo de relación que componga el vínculo. Por ejemplo

1. *"El precio del dólar* `<evento>`***comenzó***`</evento>` *a* `<evento>`***caer***`</evento>` *a causa de la* `<evento>`***crisis***`</evento>`*."*

2. *"Los testigos* <evento>**vieron**</evento> *al ladrón* <evento>**salir**</evento> *y* <evento>**entrar**</evento> *en un edificio."*

El atributo **relacion** de los vínculos aspectuales puede ser de alguno de los tipos {INICIO, REITERACION, TERMINACION, CONTINUACION, OTRO}. De esta manera, en el ejemplo 1 existe un vínculo aspectual (expresa una relación de CONTINUACION) entre los eventos ***comenzó*** y ***caer*** siendo el primero de ellos el evento aspectual.

En el caso de los vínculos de subordinación, el atributo **relacion** toma valores en el conjunto {MODAL, EVIDENCIAL, NEG_EVIDENCIAL, FACTICO, CONTRAFACTICO, CONCIDIONAL, EXISTENCIAL, CAUSAL}. En el ejemplo 2 existe un vínculo de subordinación (expresa una relación de tipo EVIDENCIAL) entre el evento subordinante ***vieron*** y los subordinados ***salir*** y ***entrar***.

2.2.4. Índices

Se anota con la etiqueta <indice> a cualquier elemento del texto, sin importar su categoría gramatical, que contribuye a determinar el valor del atributo **clase** de un evento. Los índices cuentan con un atributo **clase** que indica el tipo de señal que éstos representan. Este atributo puede tomar valores en {MODALIDAD, FACTIVIDAD, POLARIDAD, CONDICION, EVENTIVIDAD}.

Pueden llegar a ser de interés los índices de eventividad. Se anotarán con el valor EVENTIVIDAD aquellos índices que pongan de manifiesto la naturaleza eventiva de un elemento nominal. Por ejemplo

- *"*<indice clase=''EVENTIVIDAD''>**Durante**</indice> *la* <evento>**cena**</evento> *el presidente mantuvo varias conversaciones con sus pares."*
- *"El corte se levantó momentáneamente* <indice clase=''EVENTIVIDAD''>**después de**</indice> *la* <evento>**asamblea**</evento>*."*

2.3. El corpus anotado

Ya se hizo mención de que un corpus anotado (o un juego de datos etiquetados) es parte fundamental de cualquier proceso de aprendizaje automático supervisado. Es a partir de él que un algoritmo es capaz de realizar inferencia y lograr ser capaz de

clasificar datos nunca antes vistos, además de que se utiliza para evaluar el desempeño una vez finalizado el proceso de clasificación.

2.3.1. Origen y tarea de anotación

Una de las tareas del proyecto Temantex (ver 2.1) consistió en la anotación manual, por parte de estudiantes avanzados de Lingüística, de un corpus de textos en español. El corpus se anotó siguiendo el esquema de anotación SIBILA que se vio en 2.2. Para la tarea, las anotadoras utilizaron Knowtator [Ogr06], un plug-in de la herramienta Protégé [NFM00], en el cual el esquema de anotación es representado mediante una ontología.

En este caso, los textos que se utilizaron son artículos de prensa y textos históricos (la mayoría elaborados por el historiador José Pedro Barrán). Es importante el hecho de que el proceso de anotación fue realizado por ambas anotadoras de manera independiente. Si bien los primeros textos fueron anotados una sola vez, ya que las anotadoras recién se estaban familiarizando con el esquema SIBILA y era pertinente que se realizaran consultas mutuamente al dar los primeros pasos, para un número significativo de textos se cuenta con las anotaciones que realizaron ambas anotadoras. Esto es importante a efectos de estudiar algunas cosas como

- posibles deficiencias en el esquema de anotación que generen inconsistencias entre anotadores o dificulten considerablemente el proceso de anotación, de manera de corregirlos en una futura versión del esquema.

- qué tan de acuerdo están las anotadoras entre sí. Sirve para evaluar la dificultad de la tarea desde un punto de vista objetivo, lo que determina un tope para el nivel de performance que puede llegar a obtener un clasificador automático (ver 3.6.2).

Cada archivo anotado del corpus corresponde a un archivo **.txt** que contiene el texto plano, y un archivo **.xml** que contiene las anotaciones de los eventos en el formato que exporta la herramienta Knowtator. Un ejemplo de archivo XML se encuentra en la figura 2.1. Puede verse que el formato no es muy amigable para realizar un análisis manual ni como entrada a un clasificador. Como se verá en 4.2, combinándolo con el archivo de texto se convertirá a otro formato mucho más apropiado.

```
<?xml version="1.0" encoding="UTF-8"?>
<annotations textSource="caudillesco.txt">
  <annotation>
    <mention id="anotEventos2009_Instance_100004" />
    <span start="10453" end="10466" />
    <spannedText>fue asesinado</spannedText>
    <creationDate>Tue Jan 26 18:01:48 ACT 2010</creationDate>
  </annotation>
  <classMention id="anotEventos2009_Instance_100004">
    <mentionClass id="evento">evento</mentionClass>
    <hasSlotMention id="anotEventos2009_Instance_110006" />
    <hasSlotMention id="anotEventos2009_Instance_130008" />
    <hasSlotMention id="anotEventos2009_Instance_130012" />
    <hasSlotMention id="anotEventos2009_Instance_130013" />
    <hasSlotMention id="anotEventos2009_Instance_130014" />
    <hasSlotMention id="anotEventos2009_Instance_130015" />
    <hasSlotMention id="anotEventos2009_Instance_130016" />
  </classMention>
  <stringSlotMention id="anotEventos2009_Instance_110006">
    <mentionSlot id="modo" />
    <stringSlotMentionValue value="indicativo" />
  </stringSlotMention>
  <stringSlotMention id="anotEventos2009_Instance_130008">
    <mentionSlot id="tiempo" />
    <stringSlotMentionValue value="pasado" />
  </stringSlotMention>
  <stringSlotMention id="anotEventos2009_Instance_130012">
    <mentionSlot id="polaridad" />
    <stringSlotMentionValue value="pos" />
  </stringSlotMention>
  <stringSlotMention id="anotEventos2009_Instance_130013">
    <mentionSlot id="factividad" />
    <stringSlotMentionValue value="realizado" />
  </stringSlotMention>
  <stringSlotMention id="anotEventos2009_Instance_130014">
    <mentionSlot id="forma_verbal" />
    <stringSlotMentionValue value="forma_finita" />
  </stringSlotMention>
  <stringSlotMention id="anotEventos2009_Instance_130015">
    <mentionSlot id="clase_evento" />
    <stringSlotMentionValue value="ocurrencia" />
  </stringSlotMention>
  <stringSlotMention id="anotEventos2009_Instance_130016">
    <mentionSlot id="categoria" />
    <stringSlotMentionValue value="verbo" />
  </stringSlotMention>
</annotations>
```

Figura 2.1: Ejemplo de archivo XML con la anotación de un evento según esquema SIBILA, para el archivo de texto *caudillesco.txt*.

Cantidad de tokens	11986
Cantidad de oraciones	408
Tokens por oración (promedio)	29, 37
Cantidad de eventos	1677
Eventos por oración (promedio)	4, 11
Cantidad de tokens fuera de eventos	9943

Cuadro 2.1: Estadísticas del corpus anotado completo.

2.3.2. Algunas estadísticas

En el cuadro 2.1 se presentan algunos datos cuantitativos del corpus completo con el que se cuenta. Los tokens representan palabras, signos y números, que a este nivel pueden considerarse atómicos (indivisibles). El corpus se construyó exclusivamente para las tareas del proyecto Temantex y no existe otro corpus anotado con el esquema SIBILA. Debe notarse que los datos del cuadro 2.1 se extraen a partir de una versión del corpus que ha sido preprocesada donde, por ejemplo, se eliminan los títulos. Se retomará este punto más adelante.

2.3.3. Potenciales inconvenientes

Uno de los inconvenientes a notar apenas viendo los datos es que el tamaño del corpus es bastante reducido, en comparación con otros corpus que se han utilizado para otras tareas de aprendizaje en el PLN. El problema se conoce como dispersión (del inglés *sparseness*) del corpus, dado que al contar con pocos datos no se tiene suficiente información representativa como para utilizar en un sistema de aprendizaje.

Para poner algo de perspectiva, los corpus para la tarea (relativamente sencilla) de etiquetado gramatical típicamente tienen más de 1 millón de palabras. El conjunto de entrenamiento número 9 de la tarea de reconocimiento de entidades con nombre planteada en CoNLL-2003 [TKSDM03] tiene más de 200.000 palabras. El conjunto de entrenamiento para el inglés de TERN[1], tarea focalizada en la detección y normalización de expresiones temporales, consta de casi 800 documentos y más de 300.000 palabras. Hay algo más de 8000 ejemplos de expresiones temporales, y aún así se considera un conjunto algo pequeño. El corpus TimeBank, lo más parecido que existe al corpus con

[1] http://timex2.mitre.org/tern.html

el que se cuenta para nuestra tarea, tiene apenas 186 documentos y alrededor de 68.500 *tokens*[1] [BA05b]. Se han realizado trabajos para buscar métodos de solucionar este y otros inconvenientes en el corpus TimeBank [BA05a].

En contraste, el corpus con el que se cuenta consta de casi 12.000 *tokens* lo que lo hace casi 6 veces más pequeño que el corpus TimeBank. En cantidad de eventos, se cuenta con 1677 mientras TimeBank tiene 7935, casi 5 veces menos. Está claro que se padecerán todos los problemas que tienen estos corpus mencionados. La situación se agrava aún más teniendo en cuenta la complejidad de la tarea de reconocimiento de eventos[2]. Desafortunadamente no se tiene salida de este punto hasta tanto se realice un esfuerzo de construir un corpus más extenso.

2.4. Trabajos relacionados

El análisis de los eventos así como su clasificación según la forma que éstos adoptan no son problemas nuevos [Ven67]. Además de relacionarse con la lingüística también lo están con otras áreas como la filosofía, psicología, etc. A continuación se hará una reseña de algunos trabajos relacionados a la tarea, lo que dará un panorama general del estado del arte. Ha de notarse que el problema de reconocimiento automático de eventos casi nunca está aislado, sino que se enmarca dentro del análisis temporal de textos. Por esto es usual que en los trabajos del área se haga referencia a expresiones temporales, vínculos entre eventos su ordenamiento en una línea de tiempo.

2.4.1. Esquemas de anotación

Durante los 90, se hicieron grandes esfuerzos para desarrollar un corpus anotado de manera de usarlo de base para otras aplicaciones. El primer paso era definir un esquema de anotación que fijara un estándar. El esquema TIDES [Fer01] (*Translingual Information Detection, Extraction, and Summarization*) surgió con el propósito de anotar solamente las expresiones temporales en los textos, y contaba con el soporte de la organización DARPA. Una vez que culminara el proceso de anotación manual, la idea era utilizar el corpus en aplicaciones concretas como extracción automática de resúmenes, respuestas a preguntas, etc.

[1] En su versión 1.1.

[2] En el sentido de que si se tratara de una tarea más simple, quizá alcanzaría con menos cantidad de datos de entrenamiento.

El esquema STAG (*Sheffield Temporal Annotation Guidelines*) fue propuesto por Setzer y Gaizauskas y desarrollado en la universidad de Sheffield [SG00a, SG00b]. Está orientado a la identificación de eventos y sus relaciones en el tiempo, en particular en textos de noticias. Además, es capaz de relacionar expresiones temporales a eventos. En contraste TIDES solo anotaba expresiones temporales.

Más recientemente surge un nuevo esquema de anotación, el cuál fue bautizado TimeML [PCI+03]. TimeML es un esquema rico para anotar tanto eventos como expresiones temporales. Combina y extiende características de TIDES y STAG, lo que lo hace más poderoso que ambos. TimeML es hoy en día el esquema de anotación de eventos por antonomasia. De hecho, el esquema SIBILA que se vio en 2.2 está fuertemente basado en TimeML, si bien cuenta con algunas diferencias (detalladas en [WMR08]) como un mayor énfasis en determinar si los eventos referidos en un texto efectivamente ocurrieron. TimeML aborda cuatro problemas básicos relacionados con el análisis temporal y los eventos

- anclaje de los eventos en el tiempo (cuándo ocurrieron).
- orden cronológico de los eventos con respecto a los demás.
- razonamiento con algunas expresiones temporales poco específicas (ej. *la semana pasada*).
- razonamiento acerca de la persistencia de los eventos (cuánto dura el evento o sus consecuencias).

TimeML es un desarrollo significativo en el área de análisis temporal de textos, ya que se enfoca explícitamente en proveer un detalle suficiente de información que ha demostrado ser crucial para una amplia gama de tareas de inferencia y razonamiento.

2.4.2. Reconocimiento automático de eventos

Solo hay disponible un corpus anotado con el esquema TimeML: el corpus TimeBank [PHS+03]. Por eso, casi todos los trabajos posteriores a TimeML relacionados al reconocimiento de eventos se han evaluado utilizando este corpus. Además de las referencias en esta sección se presentan los resultados que se han obtenido en los trabajos, en función de las métricas de precisión, *recall* y medida F. Para entender lo que estas medidas significan es recomendable leer la sección 3.5.

En [JT03] (anterior a TimeML) se aplican Modelos Ocultos de Markov (*Hidden Markov Models* o HMM) para la tarea de extraer automáticamente eventos correlacionados de un texto. Para lograrlo, se agrupan cláusulas para formar eventos coherentes, manejando la variedad que se puede presentar en las descripciones (por ejemplo sinónimos y abreviaciones). Los HMM se utilizaron para capturar correlaciones de eventos significativos en los textos: los nodos corresponden a los eventos, y los arcos indican qué cláusulas ocurren juntas en el texto. El contexto y similitud textual se utilizan para decidir si agrupar cláusulas.

El sistema EVITA [SKVP05] reconoce eventos combinando técnicas estadísticas con reglas lingüísticas. Sus funcionalidades se pueden dividir en dos: la identificación de los eventos y el análisis de sus características gramaticales. A los efectos de la identificación de eventos se utiliza información como las etiquetas gramaticales de las palabras, lematización, análisis sintáctico superficial y búsqueda de las palabras en un lexicón. Para desambiguar eventos nominales, se utiliza información de WordNet[1] combinada con un clasificador Bayesiano. EVITA logra $74,03\,\%$ de precisión, $87,31\,\%$ *recall* para dar una medida F de $80,12\,\%$ sobre el corpus de TimeBank. En [BM06] se hace notar que estos resultados se ven aumentados por el hecho de que EVITA contiene un chequeo para determinar si una palabra ocurre o no como evento en el corpus TimeBank (de manera que fue evaluado con cierta información que ya tenía cuando se entrenó). Los autores desarrollaron una implementación de EVITA denominada Sim-Evita para poder obtener resultados comparables. Obtuvieron de este modo una medida F de $73\,\%$.

En [BA07] se ataca la tarea de reconocimiento de eventos TimeML como un problema de clasificación, utilizando un clasificador lineal que funciona con minimización del riesgo (error) denominado RRM (*Robust Risk Minimization*). El resultado de la validación fue de una medida F de $78,6\,\%$. Utilizando una técnica conocida como *word profiling* (ver [BA05a]), útil para obtener datos a partir de textos sin anotar (y así paliar el fenómeno de contar con un corpus de entrenamiento reducido) la medida F aumentó a un $80.3\,\%$. No se reportan valores para la precisión o el *recall*.

STEP [BM06] es otro sistema de identificación de eventos. Funciona con un algoritmo de aprendizaje basado en *Support Vector Machines* (SVM), y utiliza un rico

[1] Una base de datos léxica para el idioma inglés, disponible en http://wordnet.princeton.edu/.

conjunto de atributos morfológicos, dependencias e información de WordNet. Se obtuvo un $82,0\,\%$ de precisión, $70,6\,\%$ *recall* para dar una medida F de $75,9\,\%$.

En [MB08] también se construyen clasificadores en dos niveles. El primero determina si una oración contiene o no eventos, lo cuál se implementó con un clasificador bayesiano simple, árbol de decisión (algoritmo C4.5) y SVM. Los tres métodos se comportaron excepcionalmente bien (medida F de $99,6\,\%$ para el bayesiano y $100\,\%$ para los otros dos) para determinar si las oraciones contienen eventos aunque como se hace notar, el $82\,\%$ de las oraciones del corpus de entrenamiento contenían uno o más eventos y solo hace falta identificar correctamente uno para clasificar una oración. Luego de un primer paso que filtra oraciones que no contienen eventos, un segundo clasificador que funciona a nivel de las palabras o *tokens* identifica los eventos dentro de las oraciones que quedan. Los mejores resultados obtenidos llegaron a una medida F de $76,4\,\%$.

En [LSNC10] se analiza la contribución de los roles semánticos a la detección y clasificación de eventos TimeML. Se construye un sistema que utiliza *Conditional Random Fields* (CRF) para el aprendizaje, que se nutre de estos roles además de variada información morfosintáctica. Se alcanza una medida F de $81,4\,\%$ para la detección de los eventos.

Es pertinente aclarar que hay dos versiones del corpus TimeBank. EVITA [SKVP05] y [LSNC10] son las únicas referencias que utilizaron la última versión, 1.2, mientras los demás trabajos que se basan en TimeML usaron TimeBank 1.1.

Capítulo 3

La solución propuesta

En esta sección se presentará el acercamiento al problema. Se verá que, por como está planteado, el problema de reconocer los eventos de forma automática puede ser visto como uno más de la gama de problemas conocidos como de clasificación secuencial. Luego, se planteará el problema de construir un modelo de clasificador como el de hallar los atributos relevantes, y se verán algunas técnicas capaces de resolver problemas de clasificación secuencial. Se estudiará cómo utilizar el *corpus* con el que se cuenta para que un clasificador aprenda y pueda ser evaluado lo más objetivamente posible. A continuación se analizarán atributos para el reconocimiento de eventos que se han utilizado en otros trabajos, llegando a un conjunto de atributos candidatos que se utilizarán en este trabajo. Finalmente se prestará atención a lo que implica querer determinar qué tan bien se comporta un sistema, para culminar el capítulo planteando métricas concretas para evaluar el desempeño.

3.1. Alcance

Como se vio en la breve reseña del esquema SIBILA en la sección 2.2, los eventos tienen una serie de atributos. Los valores para algunos de ellos (**categoria**, **forma_verbal** y **modo**) pueden derivarse de un analizador morfosintáctico. Si bien es posible plantearse un sistema para aprender los restantes atributos utilizando las mismas técnicas y herramientas que se verán, la tarea de aprender cada atributo es independiente de la tarea de identificar cuáles son los eventos en el texto. Usualmente los trabajos que atacan el reconocimiento de eventos también intentan determinar el valor de alguno de

los atributos. Por ejemplo en [LSNC10] se plantea aprender el valor del atributo **clase** para el esquema TimeML.

En este caso, se plantea un sistema que será capaz de **delimitar** los eventos en el texto de entrada, pero sin dar ninguno de los valores para los atributos de éstos (exceptuando, claro está, los que pueden ser recuperados por un analizador morfosintáctico). No se tomarán en cuenta aquellos eventos cuya mención se omite (como se vio al hablar del atributo **item_lex**). En principio no se requiere que el sistema sea capaz también de delimitar los índices, si bien se harán comentarios al respecto más adelante. Tampoco se plantea identificar los vínculos entre eventos.

3.2. Clasificación secuencial

Previo a realizar la tarea de aprendizaje deben plantearse algunos lineamientos sobre cómo trabajará el sistema. Los eventos no ocurren en medio de las palabras, es decir, el evento más "pequeño" que se puede encontrar consta de una palabra entera[1], de manera que la tarea de reconocer eventos puede verse como la de **etiquetar** apropiadamente palabras o, más correctamente, unidades denominadas *tokens* (palabras, números y símbolos). Con este propósito es que se utiliza comúnmente un sistema de **etiquetas BIO** (por siglas en inglés). Su funcionamiento se expone a continuación

- Se dispone de tres clases de etiquetas para asignar: `B-EVENTO` ("begin"), `I-EVENTO` ("inside") y `O` ("outside").
- A una palabra le corresponderá `B-EVENTO` si delimita el inicio de un evento.
- Una palabra recibe `I-EVENTO` si forma parte del último evento que se etiquetó con `B-EVENTO`.
- Se le asigna la etiqueta `O` a las palabras que no forman parte de eventos.

Eventualmente se pueden agregar más etiquetas para representar otros tipos de entidades, como los índices. Por ejemplo, se podrían agregar `B-INDICE_X` e `I-INDICE_X` donde X es un tipo de índice (`EVENTIVIDAD`, `MODALIDAD`, etc.).

[1] Sin incluir los eventos cuya mención se omite, que como ya se dijo, no se consideran.

Con un sistema de etiquetas como el planteado el problema de reconocer los eventos en una oración pasa a ser el de asignar una de las etiquetas a cada uno de sus *tokens*. En el cuadro 3.1 se muestra un ejemplo etiquetado de esta manera.

El	*precio*	*fue*	*comenzando*	*a*	*caer*	*por*	*la*	*crisis*	.
O	O	B-EVENTO	I-EVENTO	O	B-EVENTO	O	O	B-EVENTO	O

Cuadro 3.1: Ejemplo de salida utilizando etiquetas BIO.

Al reducir el problema al de etiquetar una secuencia de *tokens*, puede verse que el problema de utilizar técnicas de aprendizaje automático para el reconocimiento de eventos es parte de un problema más general. La tarea de etiquetar unidades como *tokens* se conoce como problema de **clasificación**, ya que asignar una etiqueta a un *token* es equivalente a clasificarlo como perteneciente a una de C clases predefinidas (cada una de las etiquetas). Realizar aprendizaje implica alguna forma de, en base a secuencias de *tokens* etiquetadas, inferir las reglas subyacentes que gobiernan el etiquetado para poder lograr la capacidad de etiquetar secuencias de *tokens* que no se han visto. Afortunadamente, existen diversos métodos de aprendizaje que pueden utilizarse para llevar a cabo esta tarea, la cual no deja de ser un campo de investigación abierto.

Un clasificador (o etiquetador) secuencial es un modelo matemático cuya función consiste en asignar una etiqueta (o clase) a cada unidad dentro de una secuencia [JM08]. La teoría detrás de la clasificación secuencial se ha aplicado exitosamente en numerosos problemas del procesamiento del lenguaje natural, como ser reconocimiento de entidades con nombres [ZS02], etiquetado gramatical [Fis09], reconocimiento del habla [HAJ90, Rab89], traducción automática [VNT96], entre otros.

3.2.1. Construcción de un modelo: búsqueda de atributos relevantes

El primer paso para crear un clasificador es decidir qué atributos de la entrada resultarán relevantes para la tarea a realizar, y cómo codificar dichos atributos. Por ejemplo, en 2.2 se vieron varios ejemplos de eventos anotados, a partir de los cuales se puede intuir que si una palabra es un verbo tendrá muchas posibilidades de ser o estar dentro de un evento. Un atributo para el clasificador que reconozca eventos podría ser entonces un indicador de si la palabra es un verbo o no, o generalizando, un indicador de su categoría gramatical.

La decisión de qué atributos utilizar puede tener un enorme impacto en la habilidad del método de aprendizaje de inferir un buen modelo. Aunque en ocasiones se pueden lograr niveles de performance aceptables utilizando un conjunto simple de atributos razonablemente obvios, usualmente se logran grandes ganancias al incluir atributos cuidadosamente construidos en base a un mayor entendimiento de la tarea [BKL09].

La selección de atributos relevantes y la eliminación de los que no lo son es un problema central en el aprendizaje automático. Es un campo de investigación abierto que se conoce como ***feature selection*** [Lan94]. La selección de atributos tiene tres objetivos [Guy03]

1. Mejorar el desempeño predictivo de los clasificadores, al tratar con atributos que influyan en la salida en mayor grado.

2. Hacer los clasificadores más rápidos y menos costos en cuanto a recursos computacionales, al tener que manejar menor cantidad de atributos.

3. Proveer un mejor entendimiento del proceso subyacente que generó los datos de entrenamiento.

Una de las formas de selección de atributos sigue un proceso de ensayo y error, guiado por la intuición de qué información es relevante para el problema. Se suele pensar en todos los atributos que podrían ser relevantes y luego se sigue algún proceso para verificar si efectivamente lo son. También hay formas más costosas que corren algoritmos de búsqueda sobre un espacio de posibles atributos.

La mayoría de los métodos de clasificación requieren que los atributos estén codificados utilizando tipos simples como booleanos, números y strings. Sin embargo es pertinente notar que solo porque un atributo tiene un tipo simple no implica que ese atributo sea simple de expresar o calcular. De hecho, es posible utilizar atributos muy complejos e informativos, como por ejemplo la salida de otro clasificador [BKL09]. La forma que tendrán los atributos al fin dependerá del modelo de clasificador secuencial que se termine utilizando.

Se cierra la discusión mencionando un fenómeno que se puede dar habitualmente, y es uno de los enemigos del aprendizaje automático. Se podría pensar en agregar todos los atributos que se le ocurran a uno y luego dejar que el algoritmo de aprendizaje determine cuáles son relevantes y cuáles no. No obstante, hay límites para la cantidad de atributos

que se deberían usar en un algoritmo de aprendizaje dado. Si se usan demasiados atributos, será más probable que el algoritmo aprenda relaciones que se dan en los datos de entrenamiento pero que no generalizan bien a nuevos ejemplos. Este problema se conoce como **sobreajuste** (*overfitting*) y puede ser especialmente problemático al trabajar con conjuntos de entrenamiento reducidos, o cuando la cantidad de atributos es desmedidamente grande.

3.2.2. Técnicas para la clasificación secuencial

No se podría cerrar una sección de clasificación secuencial sin hacer mención a algunas de las técnicas que pueden brindar solución a este tipo de problemas. A continuación se presenta una breve reseña de las técnicas estudiadas. Para mayor detalle se debe consultar el documento de métodos de clasificación secuencial que acompaña a este trabajo [Des11].

El primer tratamiento del problema vino de la mano de los Modelos Ocultos de Markov (*Hidden Markov Model* o *HMM* por sus siglas en inglés). En vez de asignar etiquetas a las palabras una a una en cierta dirección hasta completar la secuencia, un HMM es capaz de determinar la secuencia de etiquetas más probable para una cierta entrada. Los HMM pueden verse en un grafo cuyos nodos o estados emiten las observaciones. Como en general se conoce la secuencia de observaciones pero no los estados (etiquetas) que les corresponden, puede decirse que estos últimos están ocultos y de ahi viene el nombre del modelo. Una contra de los HMM es que éstos solo manejan probabilidades de la forma P(etiqueta|etiqueta) y P(observación|etiqueta), por lo que incluir otro tipo de información al modelo en forma de atributos resulta imposible. Igualmente, resultan uno de los modelos más utilizados en el PLN.

Para lidiar con la dificultad que presentan los HMM surgen los Modelos de Markov de Máxima Entropía (*Maximum Entropy Markov Model* o MEMM). Este tipo de modelo es capaz de usar ciertos atributos de la entrada y combinarlos linealmente ponderándolos con pesos según su importancia relativa. Los atributos vienen dados en forma de funciones, generalmente booleanas, que expresan alguna propiedad de la entrada. Por ejemplo si en una aplicación de reconocimiento de entidades con nombre se quisiera tener en cuenta el uso de mayúsculas, se puede definir un atributo que indique si la palabra actual comienza con una mayúscula.

Los MEMM tienen un inconveniente denominado *label bias problem*, en el que ciertos estados del modelo tienen preferencia sobre otros. Lafferty et. al. introducen un nuevo modelo que lo corrige: los Campos Aleatorios Condicionales (*Conditional Random Fields* o CRF) [LMP01]. Los CRF al igual que los MEMM pueden utilizar una variedad de atributos en forma de funciones, que permiten capturar mucha información adicional de la entrada. Otra ventaja de los CRF sobre los HMM, que también poseen los MEMM, es que éstos no se ocupan de generar las observaciones sino que condicionan sobre ellas (de ahí su naturaleza condicional). Este hecho es importante pues se relajan muchas hipótesis acerca del proceso que las generó, dando mejores resultados. Los CRF se encuentran entre los modelos más exitosos de la última década, encontrando diversas aplicaciones en el procesamiento del lenguaje natural [TAK02, PFM04, Set05, SP03], bioinformática [SS05, LCWG05], visión por computadora [HZCPn04, KH03], entre otras áreas.

Otro método basado en un paradigma diferente a los anteriores son las Máquinas de Vectores de Soporte (*Support Vector Machines* o SVM) [Vap95]. No son de por si un método de clasificación secuencial, aunque es posible adaptarlos para la tarea. En el caso no secuencial, los SVM funcionan considerando las instancias a clasificar como puntos en un espacio con una cierta dimensión (posiblemente infinita) y construyendo un hiperplano que particiona el espacio y divide las instancias según la clase. Se ha demostrado que al utilizarlos para clasificación secuencial tienen performance comparable con la de los CRF [Kee09].

3.3. Entrenamiento y evaluación

Cuando se realizan tareas de aprendizaje supervisado sobre un corpus de texto anotado siempre es de interés poder obtener medidas para evaluar el desempeño del clasificador. Esto requiere de ejemplos etiquetados correctamente, de manera de poder comparar la etiqueta asignada a cada ejemplo por el clasificador con su etiqueta "esperada" y a partir de esto, sacar conclusiones. Es deseable que se evalúe la performance sobre ejemplos que sean razonablemente parecidos a los que se utilizaron para entrenar. Por ejemplo, si se entrena un clasificador para identificar eventos sobre textos periodísticos de deporte, éste no debería evaluarse sobre textos con noticias financieras.

Cada tipo de texto tiene sus particularidades, y elegir qué textos formarán parte del corpus es fundamental y va a depender de qué usos se le quieren dar al clasificador.

Un grave error sería evaluar la performance sobre el corpus que se utilizó para realizar el aprendizaje (denominado **corpus de entrenamiento**) ya que el clasificador pasó previamente por todos los ejemplos etiquetados y realizó inferencia a partir de ellos, por lo que es de esperar que dé muy buenos resultados si luego se le pide clasificar dichas instancias. De hecho, dependiendo del tipo de clasificador que se utilice, es muy posible que logre etiquetar todos estos ejemplos correctamente lo que induciría al pensamiento erróneo de que el clasificador no comete errores. La tarea de evaluación **debe** realizarse sobre ejemplos nuevos, que no se han visto durante el entrenamiento. De lo contrario, se estará sesgando la evaluación y ésta carecerá de sentido.

Lo que habitualmente se hace es separar el corpus etiquetado en dos partes, una de las cuales será el corpus de entrenamiento y se utilizará para que el clasificador realice el aprendizaje, y la otra el **corpus de evaluación o testeo** el cual servirá para obtener las medidas de desempeño. Naturalmente se destinará mayor porcentaje del corpus para el entrenamiento que para el testeo, aunque no existe una regla que indique qué porcentaje es mejor usar. Usualmente se toma 80 % para entrenamiento y 20 % para testeo, pero esto es más bien arbitrario. Este método se conoce como **validación *hold-out*** [Dzi09].

Existen otras formas de validación diferentes a la recién mencionada. Por ejemplo, en la **validación cruzada con k particiones** (conocida como *k-fold cross validation* en inglés) se divide el corpus original en k particiones obtenidas al azar. De estas k particiones, se toma una como corpus de validación y las $k - 1$ restantes se utilizan para entrenar el modelo. Se repite el proceso k veces de manera de usar cada una de las particiones como validación exactamente una vez. Para obtener como resultado una medida de performance global se puede tomar un promedio de los resultados que se obtuvieron evaluando en cada partición. La ventaja de este método es que todas las instancias se utilizan tanto para el entrenamiento como para la evaluación, y que cada instancia se usa para la evaluación una sola vez. El valor $k = 10$ se utiliza frecuentemente. Una desventaja es que puede tomar más tiempo de ejecutar, en el caso que el clasificador requiera un tiempo elevado de entrenamiento.

De más está decir que cuanto mayor sea el tamaño del corpus que se cuenta para entrenar (y por lo tanto para evaluar), más representativos serán los resultados que se obtengan. Desafortunadamente, lograr etiquetar manualmente grandes corpus de texto

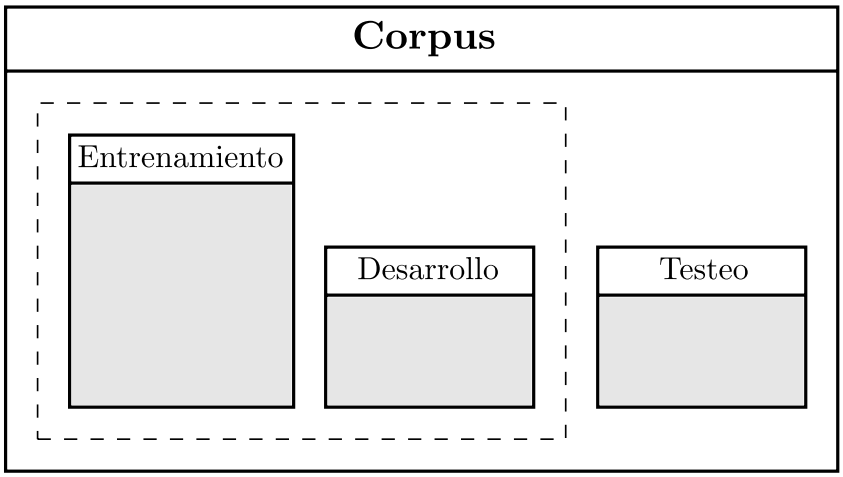

Figura 3.1: Separación del corpus. Lo que está dentro del rectángulo punteado es lo único a lo que se tiene acceso durante el desarrollo del clasificador. El corpus de testeo se consulta una vez el clasificador está finalizado.

es una tarea que necesita de mucho trabajo y recursos, por lo que en la práctica esta puede llegar a ser una limitación (ver 2.3.3).

Como se vio brevemente en 3.2.1 gran parte del proceso de aprendizaje consiste en elegir atributos relevantes para la tarea a llevar a cabo, probando distintas combinaciones para ver cuáles sirven y cuáles no. La única forma que se tiene de ver si la elección de un atributo fue buena es evaluar el resultado del clasificador sobre parte del corpus, por ejemplo usando el corpus de testeo. Así, se va probando con distintos atributos y sobreviven aquellos que hacen que una cierta medida de performance en el corpus de testeo aumente. Sin embargo, esto tiene un problema. Se puede estar eligiendo justo una combinación de atributos que funciona muy bien en el corpus de testeo, pero frente a otro texto (de la misma naturaleza) nunca visto, no tanto. Al estar eligiendo atributos que hagan que el clasificador funcione bien sobre el corpus de testeo, los resultados de la evaluación de performance ya no son representativos de lo que se esperaría en una aplicación práctica "real". Es por esto que a veces se toma también un **corpus de desarrollo**, además del de testeo. La tarea de ajustar atributos para obtener mayor performance se hace sobre el corpus de desarrollo **únicamente** [BKL09]. Resumiendo lo dicho, el corpus se particiona de la siguiente manera

Corpus de entrenamiento Representa la mayor proporción del corpus completo. El

clasificador trabaja realizando inferencia sobre este corpus, obteniendo así conocimiento que luego le permitirá clasificar nuevas instancias no vistas.

Corpus de desarrollo Se utiliza para evaluar el desempeño mientras se están probando diferentes combinaciones de atributos y parámetros. Se buscan atributos que aumenten una medida de performance sobre este corpus. Es pertinente realizar un análisis de los errores que comete el clasificador para buscar atributos que puedan ayudar a reducirlos, o atributos ya existentes que puedan estar causándolos.

Corpus de testeo Se utiliza para evaluar el desempeño una sola vez, cuando ya se terminó de construir el clasificador. Los resultados deberían acercarse a los esperados en una aplicación práctica de la vida real, cuando al clasificador se lo alimenta con texto nunca antes visto, pero de la misma naturaleza que el que se utilizó para entrenar.

Puede verse esta partición gráficamente en la figura 3.1.

Realizar algún tipo de análisis de error o mirar los resultados que se están obteniendo en el corpus de testeo durante el proceso de ajuste de atributos y parámetros es una mala práctica, pues además de dar una falsa impresión puede llegar a influir en cómo se eligen atributos con la consecuencia de que los resultados sobre el corpus de testeo aparenten ser mejores de lo que en realidad son. Tampoco hay reglas generales para determinar los porcentajes óptimos de desarrollo-testeo, lo cuál dependerá del tamaño del corpus con el que se cuente.

3.3.1. Estadísticas de las particiones del corpus

En base a todo el corpus con el que se cuenta, se realizó la partición asignando los porcentajes $70\,\% - 15\,\% - 15\,\%$ a entrenamiento, desarrollo y evaluación respectivamente. Para lograrlo, se toma el corpus original y se separa en oraciones utilizando un analizador morfosintáctico. De este conjunto de oraciones se extraen al azar las porciones que corresponden al corpus de desarrollo y testeo, dejando el resto para entrenamiento. Ninguno de los tres corpus comparte oraciones. La única posibilidad de que esto suceda es que exista una oración que se repita en el corpus original.

Al contar con aún menos ejemplos para entrenar, producto de la partición, se agrava el problema de dispersión que se vio en 2.3.3.

	Corpus		
	Entrenamiento	Desarrollo	Testeo
Cantidad de tokens	8501	1780	1705
Cantidad de oraciones	286	61	61
Tokens por oración (promedio)	29.72	29.18	29.95
Cantidad de eventos	1173	258	246
Eventos por oración (promedio)	4.10	4.23	4.03
Cantidad de tokens fuera de eventos	7069	1472	1402

Cuadro 3.2: Estadísticas de cada uno de los corpus.

3.4. Atributos relevantes para el reconocimiento de eventos

En la literatura, el conjunto de atributos que se ha usado para el reconocimiento de eventos es similar al usado para otras tareas de clasificación secuencial, como el reconocimiento de entidades con nombre. Los atributos refieren a propiedades del elemento actual de la secuencia así como los que lo rodean. Para introducir información del contexto se maneja el concepto de *ventana*, que refiere a mirar cierta cantidad de elementos hacia atrás y adelante. Por ejemplo con una ventana $[-1, 1]$ se estaría mirando información del elemento actual (0) así como el anterior (-1) y el que le sigue (1). Manejar ventanas de tamaño grande aporta mayor cantidad de información pero también puede introducir ruido, lo que hace que el clasificador se comporte peor. Cada atributo puede tener un tamaño de ventana óptimo diferente al de los demás atributos.

Es usual también usar atributos bigramas (o trigramas). Por ejemplo un atributo bigrama podría ser w_{-1}, w_0 y w_0, w_1 donde w_i son los *tokens* de la secuencia.

En la tarea de reconocimiento de eventos TimeML de [BA05b] se utilizan atributos como

- *token*, uso de mayúsculas, etiqueta gramatical en una ventana $[-1, 1]$.
- bigramas de las palabras adyacentes en una ventana $[-2, 2]$.
- palabras en el mismo componente (*chunk*) sintáctico.
- primeras palabras de cada componente para una ventana de $[-1, 1]$ de componentes sintácticos.

- trigramas de etiqueta gramatical, uso de mayúsculas y fin de las palabras.
- ...entre otros.

Para la misma tarea en [BM06] además de atributos similares a los anteriores se plantean otros, como

- atributos de afijos, primeros tres y cuatro caracteres de las palabras, últimos tres y cuatro caracteres.
- sufijos de nominalización, para determinar si la palabra termina con uno de éstos.
- atributos morfológicos, como el lema de las palabras y el verbo raíz para el caso de los verbos y sustantivos deverbales (ej. *investigación* es derivado de *investigar*).
- información de hiperónimos de WordNet, como por ejemplo a cuál sub-jerarquía de nombres y verbos pertenecen las palabras (se eligieron 10 categorías).

Otros autores utilizan un etiquetador de roles semánticos y muestran que utilizarlos como atributos mejora el desempeño del clasificador [LSNC10].

3.4.1. Atributos candidatos para el sistema

En este caso se plantea usar principalmente atributos morfosintácticos ya que muchos de los recursos que se han usado en la literatura no están disponibles para el español, o al menos no son tan completos. Por ejemplo, no se cuenta con un etiquetador de roles semánticos. Se plantea una serie de atributos candidatos a usar

- el *token* y el lema.
- la etiqueta gramatical completa.
- tiempo y modo (para los verbos).
- uso de mayúsculas.
- inicio y terminación de las palabras (4 letras).
- tipo para los nombres (propio, común, otro).
- bigrama con la etiqueta gramatical, únicamente la categoría gramatical, y con los *tokens*.

Estos atributos no corresponderán solo al elemento actual sino que se trabajará con una ventana para aprovechar el contexto. Los atributos que resultan útiles y los tamaños de las ventanas se determinarán a partir de pruebas realizadas con el corpus de desarrollo. El acercamiento será manual, probando algunas combinaciones y quedándose con los atributos que hagan que las medidas de desempeño no empeoren.

Se consideró la posibilidad de construir una lista de palabras que puedan actuar como índices de eventividad, para tener un atributo que indique si hay un índice de eventividad cierto número de posiciones hacia atrás del elemento actual. Sin embargo un análisis mostró que solamente hay 5 índices de eventividad etiquetados en el corpus completo de los cuales solamente 3 son distintos entre sí (la palabra *durante* se repite tres veces). Para que este acercamiento funcione se debería contar con una lista de posibles índices de eventividad, pero dado que existen tan pocos en el corpus se decidió no utilizar los índices.

En el caso de números y fechas, el lema se cambiará por una etiqueta `<NUMERAL>` o `<FECHA>`. Se evaluará también los beneficios de cambiar el *token* en sí por esta misma etiqueta. Además, se utilizará un analizador morfosintáctico que soporta reconocimiento de entidades con nombres, fechas, cantidades, etc. Todo esto constituye una forma de *feature generalization*, como se usa en [MB08] donde se muestra que al agrupar estos tipos de palabras reduce el espacio de atributos del clasificador sin pérdida de la habilidad de desambiguar.

3.5. Métricas para evaluación de desempeño

Hasta ahora siempre se ha asumido que se cuenta con una cierta medida que cuantifica el desempeño del clasificador, de manera de poder realizar comparaciones. Llega el momento de bajar este concepto a tierra y presentar las medidas más utilizadas.

3.5.1. Evaluando clasificadores en general

Al referirse al desempeño de un clasificador, es de interés determinar qué tan bien el modelo logra predecir (o separar) las clases. Una forma simple de poder ver el panorama completo de los errores (y aciertos) cometidos por el clasificador es con la llamada **matriz de confusión**. Ésta consiste de una matriz cuyas filas corresponden a la clase conocida de los datos, y las columnas a las predicciones hechas por el modelo. El

		Predicción		
		A	**B**	**C**
Clase correcta	**A**	38	16	2
	B	11	21	3
	C	0	1	10

Cuadro 3.3: Ejemplo de matriz de confusión para un problema con tres clases A, B, y C.

valor de una posición (i, j) en la matriz representa la cantidad de elementos de clase i clasificados con la clase j. De esta forma, la diagonal principal de la matriz representa la cantidad de clasificaciones correctas para cada clase, y los elementos fuera de la diagonal son los errores cometidos. Por ejemplo, si se tienen tres clases A, B, y C, una matriz de confusión puede verse como en 3.3. Estudiando esta matriz puede verse que el clasificador tiene bastantes problemas en diferenciar las clases A y B, no así el resto. Este dato resulta sumamente útil pues con este conocimiento se puede pensar en agregar más atributos al modelo que ayuden a diferenciar entre las mencionadas clases.

Para lo que sigue, se asume que se está ahora ante una clasificación binaria y se nombran las clases $+$ y $-$. En el caso de tener más de 2 clases, como en el ejemplo anterior, se convierte el problema en uno de clasificación binaria al posicionarse en cualquiera de las clases. Por ejemplo si un elemento tiene la clase A entonces pertenece a la clase $+$, y si no la tiene entonces pertenece a $-$. En este contexto se define

- **Verdaderos positivos** (*true positives*) elementos de clase $+$ correctamente clasificados como $+$.
- **Verdaderos negativos** (*true negatives*) elementos de clase $-$ correctamente clasificados como $-$.
- **Falsos positivos** (*false positives*) elementos de clase $-$ incorrectamente clasificados como $+$.
- **Falsos negativos** (*false negatives*) elementos de clase $+$ incorrectamente clasificados como $-$.

Equivalentemente puede plantearse la matriz de confusión 3.4. Se definen

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

		Predicción	
		+	-
Clase correcta	+	Verdadero positivo (TP)	Falso negativo (FN)
	-	Falso positivo (FP)	Verdadero negativo (TN)

Cuadro 3.4: Matriz de confusión para clasificación binaria.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Intuitivamente, la precisión mide la proporción de ejemplos clasificados como + que efectivamente pertenecen a dicha clase. El *recall* (o recuperación) mide la proporción de ejemplos etiquetados con + que el clasificador etiquetó correctamente. Estos valores están comprendidos entre 0 y 1 y también se les puede dar una interpretación probabilística

- *precision* es la probabilidad de que un ejemplo etiquetado con + (seleccionado al azar) efectivamente haya sido clasificado como + por el clasificador.
- *recall* es la probabilidad de que un ejemplo (seleccionado al azar) haya sido clasificado como +.

Un puntaje perfecto 1.0 de precisión significa que todos los ejemplos clasificados como + efectivamente pertenecen a esta clase (pero no dice nada sobre cuántos ejemplos de clase + fueron clasificados incorrectamente como −). Un puntaje perfecto 1.0 de *recall* significa que el clasificador fue capaz de encontrar todos los ejemplos etiquetados con + (pero no dice nada sobre cuántos ejemplos de clase − fueron clasificados incorrectamente como +).

En las tareas de clasificación de la práctica por lo general se presenta una relación inversa entre *precision* y *recall*, siendo posible aumentar una a costas de reducir la otra. Clasificar con + solo los ejemplos en los que se tiene casi absoluta certeza hará que aumente significativamente la precisión, pero al costo de bajar el *recall* por los ejemplos etiquetados + sobre los que el clasificador no tiene tanta certeza.

Una medida que se utiliza para combinar *precision* y *recall* es la **medida F**, que se define como la media armónica de ambos

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

que es un caso particular de la medida F_β (para un β real no negativo)

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

El parámetro β indica cuánto más peso se le da a la precisión frente al *recall*. En la medida F vale 1 lo que indica que ambos pesan igual. Además de esta, se utilizan comúnmente las medidas F_2 (*recall* pesa el doble que *precision*) y $F_{0.5}$ (*precision* pesa el doble que *recall*).

3.5.2. El caso de la detección de eventos

Esta sección trata de cómo adaptar las métricas de precisión y *recall* al caso de la detección de eventos. Si bien se podrían calcular de forma simple para cada una de las etiquetas `B_EVENTO`, `I_EVENTO` y `O` independientemente esto no tiene tanto sentido pues no refleja exactamente qué tan bien funciona el clasificador para detectar los eventos. Lo que se busca es adaptar estas medidas para que trabajen sobre eventos completos.

Para lograr esto se debe definir un criterio para calcular las medidas de verdaderos positivos, falsos positivos y falsos negativos. Se supone que el clasificador será lo suficientemente inteligente como para marcar siempre el inicio de los eventos con `B_EVENTO`, es decir, no asignará una etiqueta `I_EVENTO` al principio de la oración o luego de que asignó una etiqueta `O`[1]. Entonces estas medidas representarán

Verdaderos positivos eventos correctamente etiquetados.

Falsos positivos secuencias de *tokens* que no son eventos, etiquetadas como eventos.

Falsos negativos eventos clasificados como no eventos.

Se debe definir un criterio sobre cuándo se considera que un evento está bien reconocido. Si se llama *extensión* de un evento (en inglés *span*) a todas las palabras que se encuentran entre la primera y la última (inclusive), algunos criterios posibles son

- Un evento está bien reconocido si su comienzo y su extensión están perfectamente reconocidos.

[1] Los experimentos realizados demostraron que este es el caso con los clasificadores que se utilizaron en este trabajo. Igualmente, de no haber sido así, se podría haber implementado una pasada que realice una corrección.

Etiqueta clasificador	Etiqueta correcta
O	O
B_EVENTO	O
I_EVENTO	B_EVENTO
B_EVENTO	B_EVENTO
I_EVENTO	I_EVENTO

Cuadro 3.5: Primer ejemplo de decisiones tomadas por un clasificador. Se contabilizan $tp = 1$, $fp = 1$, $fn = 1$.

- Un evento está bien reconocido si su comienzo está bien reconocido, pero no necesariamente su extensión.

- Un evento está bien reconocido si su comienzo está bien reconocido y el largo de su extensión no difiere en más de una palabra del correcto.

Está claro que se pueden inventar infinitos criterios según se quiera ajustar la severidad con la que se penalizan los errores. En conferencias como CoNLL-2000 [TKSB00] o CoNLL-2002 [TKS02], que trabajaron con tareas de clasificación secuencial, el criterio que se utilizó es el primero de los que se listaron, denominado *perfect match*. Si bien puede dar medidas bastante inferiores a las que se obtendrían con algo de flexibilidad, goza de mucha popularidad pues al no permitir que el clasificador cometa errores es el menos discutible de los criterios.

En el cuadro 3.5 se puede ver la salida de un clasificador hipotético contrastado con los valores originales de las etiquetas. El clasificador logró identificar bien uno de los eventos pero no el otro. Las medidas serían entonces

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{1}{1+1} = 0.5 \quad , \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{1}{1+1} = 0.5$$

lo que da una medida F de 0.5. En el cuadro 3.6 se presenta un ejemplo un poco más complejo. El clasificador encontró 3 eventos, solo uno de los cuales es correcto. De los 2 eventos correctos, solo logró detectar uno. Se obtienen las medidas

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{1}{1+2} = 0.\overline{3} \quad , \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{1}{1+1} = 0.5$$

Etiqueta clasificador	**Etiqueta correcta**
B_EVENTO	B_EVENTO
I_EVENTO	I_EVENTO
B_EVENTO	I_EVENTO
I_EVENTO	I_EVENTO
O	O
B_EVENTO	B_EVENTO

Cuadro 3.6: Segundo ejemplo de decisiones tomadas por un clasificador. Se contabilizan $tp = 1$, $fp = 2$, $fn = 1$.

Categoría	# Eventos
VERBO	812
NOMBRE	271
ADJETIVO	63
PRONOMINAL	8
OTRO	2

Cuadro 3.7: Cantidad de eventos anotados por cada categoría, para el corpus de entrenamiento.

3.5.3. Separación de eventos según categoría

En el cuadro 3.7 se muestra la cantidad de eventos etiquetados en el corpus de entrenamiento, clasificados según el valor del atributo `CATEGORIA`. La intuición dice que en los verbos no habrá demasiada discusión sobre si constituyen o no un evento, pero en el caso de las demás clases de palabras esta discusión sí se puede dar. Por esto se considera pertinente reportar tanto resultados de desempeño globales como separados para eventos según su categoría. Por la cantidad de eventos de cada tipo con la que se cuenta en el corpus de entrenamiento, no se van a reportar resultados de eventos con categorías distintas a verbos, nombres y adjetivos ya que hay tan pocos datos que los números carecerían de sentido.

A los efectos de evaluar los resultados resta hacer una aclaración respecto a cómo se consideran los tipos de eventos. Para el sistema de evaluación, un evento será verbal, nominal o adjetival si **todas** sus palabras tienen la categoría verbo, nombre o adjetivo (respectivamente), según determinado por un etiquetador gramatical. Esto quiere decir que la cantidad de eventos en cada caso no tiene por qué corresponder exactamente con la cantidad que se contaría al considerar los tipos según el valor del atributo `CATEGORIA`.

3.6. La perspectiva: líneas base y tope

Hasta ahora se ha visto cómo medir de forma objetiva el desempeño de un sistema de reconocimiento de eventos. Un valor numérico como *precision*, *recall* o la medida F puede decir qué tan bien se comporta un sistema frente a un *gold standard*, es decir, un conjunto etiquetado que se considera "ideal". Sin embargo, es pertinente poner en perspectiva estos números pues cada tarea de clasificación secuencial refiere a un problema con ciertas peculiaridades, que pueden hacerlo más fácil o difícil que otros.

Por ejemplo, a priori se podría pensar que si se logró construir un sistema de etiquetado gramatical que asigna la etiqueta correcta el 90 % de las veces, este sistema se comporta razonablemente bien. Lo que no se ha tenido en cuenta es que un sistema sin ninguna inteligencia, que solo asigna la etiqueta gramatical más común a todas las palabras es capaz de lograr estos mismos resultados que, evidentemente, serían bastante más bajos si solamente se consideraran las palabras con ambigüedad [Cha97]. A veces un algoritmo simple puede lograr resultados superficialmente buenos que cuando se someten a un mayor análisis, no lo son.

Se llama **línea base** (del inglés *baseline*) a una cota inferior del resultado que se espera obtener con algún sistema de clasificación para considerar que realiza algún trabajo útil. En general las líneas base se implementan con algún algoritmo bastante simple y sin prestar excesiva atención a los detalles. Por ejemplo, para el caso del etiquetado gramatical una línea base podría ser el ejemplo presentado arriba, un clasificador que asigna siempre la etiqueta gramatical más común. Claro está que de construir un clasificador, por ejemplo basado en aprendizaje automático, se espera que se comporte mejor que la línea base (y si no lo hace, es indicio de que hay algo que reconsiderar).

Habiéndose ocupado de la línea base, resta plantear la pregunta de hasta qué nivel se va a poder seguir mejorando el desempeño del clasificador. Se llama **línea tope** (del inglés *ceiling*) a esta cota superior. La forma más común de establecer una línea tope es determinando qué tan bien la tarea en cuestión es desarrollada por humanos. Incluso luego de invertir mucho tiempo y esfuerzo en el proceso de corrección, muchos errores e inconsistencias existen en los corpus anotados. El lenguaje natural presenta ambigüedad inherente lo que hace que en ciertos casos incluso humanos no puedan ponerse de acuerdo en tareas de anotación. Por ejemplo, sabiendo que para la tarea de etiquetado gramatical anotadores humanos están de acuerdo el 97 % de las veces, un corpus anotado tendrá al menos un 3 % de margen de error, por lo que obtener un 100 % de medida F no tiene sentido pues se estaría modelando ruido.

En lo que sigue se verán formas de calcular las líneas base y tope para la tarea de reconocimiento de eventos.

3.6.1. Línea base

El algoritmo para la línea base se aprovecha en parte de la proporción de eventos que hay anotados con cada categoría, información presentada en el cuadro 3.7. Se ve ahí que la mayoría de los eventos que existen en el corpus son verbos, seguidos de nombres y adjetivos. Lo que se hace es intentar clasificar como eventos palabras con estas tres categorías, ignorando completamente las otras.

En el caso de los verbos, la heurística es etiquetar toda secuencia contigua (posiblemente compuesta por un solo verbo) como un mismo evento. De forma que los verbos siempre serán marcados como eventos. Además, si la palabra “se” se encuentra antes del primer verbo, ésta ingresa al evento.

- *"Los antropólogos forenses* **<evento>delimitaron</evento>** *el predio."*
- *"El criminal* **<evento>se fugó</evento>** *del establecimiento penitenciario."*
- *"El resultado del examen* **<evento>se había hecho desear</evento>**."

En el caso de los nombres y adjetivos la cuestión es un poco más complicada, pues no hay una forma obvia de determinar si es probable que constituyan o no un evento. Dado lo disperso de los datos, quizá no resultaría adecuado un modelo en el que se juzgara si las palabras constituyen o no eventos en base a su frecuencia de aparición en el corpus de entrenamiento. Lo que se decidió hacer es utilizar las terminaciones de las palabras, esperando que algunas terminaciones estén más correlacionadas con la existencia de un evento. Se eligió 4 letras como el largo de las terminaciones, y se ordenaron según una métrica de correlación [CN02]

$$C = \frac{(N_{r+}N_{n-} - N_{r-}N_{n+})\sqrt{N}}{\sqrt{(N_{r+} + N_{r-})(N_{n+} + N_{n-})(N_{r+} + N_{n+})(N_{r-} + N_{n-})}}$$

donde

- N_{r+} (N_{r-}) es la cantidad de veces que la terminación aparece (no aparece) dentro de un evento.
- N_{n+} (N_{n-}) es la cantidad de veces que la terminación aparece (no aparece) fuera de eventos.
- N es la suma de la cantidad de eventos y la cantidad de tokens que están fuera de eventos.

Se realiza este análisis una vez con las terminaciones de los nombres y otra con las de los adjetivos. Según el valor de la correlación se toma cierta cantidad de terminaciones como las más correlacionadas con la existencia de eventos. Para los nombres fueron

```
ción sión anza bios esta onda zcla igro ñeco
queo icie orro sumo alma daño gada taja itro
ueda ioro somo ebra nita gaño lero urla aque
íces erés imio lico cena arma pado icto isis
itio naza enso azón sito reno aude erie eria
abla rres lito gato lido dalo beda ulto bado
```

	Precision (%)	Recall (%)	Medida F (%)
Global	71,67	66,67	69,08
Eventos verbales	73,37	87,32	79,74
Eventos nominales	52,17	37,50	43,64
Eventos adjetivales	75,00	20,00	31,58

Cuadro 3.8: Resultados de la línea base para el corpus de **desarrollo**.

	Precision (%)	Recall (%)	Medida F (%)
Global	67,14	57,32	61,84
Eventos verbales	65,24	79,26	71,57
Eventos nominales	63,33	27,94	38,78
Eventos adjetivales	0,00	0,00	0,00

Cuadro 3.9: Resultados de la línea base para el corpus de **testeo**.

mientras que para los adjetivos fueron

```
osos ulto gnos ácil rasa uito
lava voso fuso egra viva ueno
rnas ciso ible bles olas able
```

Finalmente el algoritmo etiquetará como evento aquellos nombres o adjetivos que tengan alguna de las terminaciones de las listas, según la categoría gramatical de la palabra. En 3.8 y 3.9 se ven los resultados de aplicar el clasificador línea base en los corpus de desarrollo y testeo respectivamente. Se observa una cierta variabilidad en los números, lo que puede deberse al tamaño reducido del corpus con el que se cuenta. En el caso de los adjetivos los resultados son significativamente más pobres, tanto que en el caso del corpus de testeo no se identifica ningún verdadero positivo y por tanto los resultados de todas las medidas dan 0. Un efecto como este podía esperarse puesto que la cantidad de eventos que son adjetivos es significativamente menor que las demás clases.

3.6.2. Línea tope

Para calcular la línea tope se utiliza que la mayoría de los archivos del corpus fueron anotados independientemente por ambas anotadoras, lo que da la posibilidad de realizar comparaciones entre las anotaciones de cada archivo.

	Precision (%)	Recall (%)	Medida F (%)
Global	91, 59	93, 00	92, 29
Eventos verbales	94, 22	97, 11	95, 64
Eventos nominales	85, 82	88, 72	87, 25
Eventos adjetivales	100, 00	74, 19	85, 19

Cuadro 3.10: Medidas de concordancia entre ambas anotadoras (línea tope), tomando los archivos etiquetadas por una de ellas como *gold standard* y las anotaciones de la otra como las de un sistema automático.

Se desea saber qué tan de acuerdo estuvieron ambas anotadoras a la hora de **delimitar** eventos, sin prestar atención a cómo marcaron sus atributos. En general para tareas de este tipo se utiliza una medida de concordancia entre anotadores (en inglés *inter-annotator agreement*) cuyo ejemplo más claro es el **kappa de Cohen**. [Car96] En [WC05] se presenta una tarea de anotación que refiere a casos en los que hay que delimitar texto y se discute cómo medir el acuerdo entre anotadores. Esta tarea es diferente a la de anotar siempre los mismos atributos en ciertas palabras que están determinadas, por lo que se usa una medida de concordancia distinta a el kappa de Cohen, la que se denomina *agr*.

Sean A y B las porciones de texto marcadas como eventos por dos anotadores a y b respectivamente. La medida *agr* dice qué proporción de A también fue marcada por b. Concretamente se computa la concordancia de b a a como

$$agr(a||b) = \frac{|A \text{ concordando con } B|}{|A|}$$

La medida $agr(a||b)$ corresponde al *recall* si a se toma como *gold standard* y b como el sistema de etiquetado, y a la precisión si b es el *gold standard* y a el sistema. Consecuentemente a lo que se vio, se restringe el corpus a todos los archivos para los que se cuenta con anotaciones independientes de cada una de las anotadoras. Luego se toman las anotaciones de una de ellas como referencia[1] y las anotaciones de la otra como el sistema de etiquetado, para obtener las medidas que se muestran en el cuadro 3.10. Para obtener los datos de concordancia entre anotadores del corpus TimeBank de TimeML se utilizó este mismo proceso[2], dando como resultado una concordancia del 78 %

[1] La elección fue arbitraria y de haberse usado la otra anotadora como referencia se obtendrían los valores de *precision* y *recall* intercambiados.

[2] Ver http://timeml.org/site/timebank/documentation-1.2.html.

para eventos. Como se ve en el caso de este trabajo, los valores son significativamente menores para nombres y adjetivos que para verbos, lo que refuerza la intuición de que éstos son más difíciles de clasificar.

Capítulo 4

Implementación

En esta sección se mostrará cómo se implementó la solución descrita en el capítulo 3. Se comenzará viendo la arquitectura del sistema en alto nivel, con una descripción de las tres partes principales que lo componen. Luego se ahondará más en cómo se realizan las primeras dos: el preprocesamiento del corpus y la implementación de los clasificadores, mostrando qué herramientas se utilizaron. A continuación se verá cómo se desarrollaron las partes y se describirán las características de una biblioteca con funciones útiles, que puede ser reutilizable a futuro.

4.1. Arquitectura

El sistema completo consta de tres partes. La primera está enfocada en la generación de los corpus de entrenamiento, desarrollo y testeo a partir de archivos *.txt* y *.xml* que conforman el corpus tal cual fue exportado de Protégé por las anotadoras. Cada archivo *.txt* contiene el texto y su correspondiente *.xml* con las anotaciones con las marcas de los eventos. Estos archivos se leen de manera de combinar el texto con las anotaciones en un formato único, que cuenta con todos los atributos que utilizarán los clasificadores. Se expandirá más sobre este punto en la sección 4.2.2. La segunda parte está integrada por los clasificadores en sí. Éstos se ocupan de leer archivos de texto generados en la parte anterior para realizar el proceso de aprendizaje y poder etiquetar futuras instancias. Naturalmente, son alimentados con el corpus de entrenamiento para realizar el aprendizaje, y luego con los corpus de desarrollo y testeo para generar archivos de texto que contienen dichos corpus etiquetados por el clasificador. La tercera parte

consiste en la evaluación de resultados, es decir, la obtención de métricas a partir de los archivos etiquetados por los clasificadores.

4.2. Preprocesamiento del corpus

Antes de alimentar a un clasificador con un corpus anotado, éste pasa por una etapa llamada preprocesamiento, la que a su vez se puede dividir en dos fases. La primera consiste en segmentar el corpus en oraciones (que finalmente serán las secuencias sobre las que se entrenará el clasificador) y agregar atributos básicos derivados de un analizador morfosintáctico. La segunda fase consiste en la eliminación (o modificación) de información para generar el menor "ruido" posible en la entrada al clasificador, así como el agregado de los restantes atributos. En esta sección, se verá en qué consiste la primera etapa del preprocesamiento, facilitada principalmente por una herramienta llamada Freeling.

4.2.1. Freeling

FreeLing[1] es un programa informático gratuito de análisis lingüístico. Está desarrollado por el *Centre de Tecnologies i Aplicacions del Llenguatge i la Parla* (TALP) de la *Universitat Politècnica de Catalunya* (UPC). Permite múltiples funciones, como división en oraciones, lematización, etc., para el español, catalán, gallego, italiano e inglés, a través de diccionarios y gramáticas específicas para cada lengua.

Las funciones que más se utilizarán en este trabajo serán las de etiquetado gramatical (*POS tagging*), división en *tokens* y división en oraciones. Para la división en *tokens*, se activarán una serie de opciones detalladas en 4.2.2.

4.2.2. Vinculación del texto anotado con la salida de Freeling

Las *suites* que implementan algoritmos de clasificación secuencial por lo general requieren que los archivos que se usan para entrenamiento y testeo estén en cierto formato. Es habitual que sea alguna forma donde el texto esté organizado en columnas separadas por algún carácter de espaciado, donde todas las primeras columnas representan un vector de atributos y la última la etiqueta que se desea aprender. Teniendo esto en cuenta, se aplica el etiquetador gramatical de Freeling para combinar su salida con

[1] http://nlp.lsi.upc.edu/freeling/

los eventos anotados para cada archivo de texto. Se utilizan las tres etiquetas `B_EVENTO`, `I_EVENTO` y `O`. Por ejemplo, un archivo de texto con la oración *"Este año el interior se sublevó contra la autoridad española residente en Montevideo."* se transforma en un archivo de extensión *.oraciones* con el contenido que se muestra en la figura 4.1.

```
token token_ini token_fin lema_freeling tag_freeling tag_evento

Este 69 73 este DD0MS0 O
año 74 77 año NCMS000 O
el 78 80 el DA0MS0 O
interior 81 89 interior NCMS000 O
se 90 92 se P0000000 B_EVENTO
sublevó 93 100 sublevar VMIS3S0 I_EVENTO
contra 101 107 contra SPS00 O
la 108 110 el DA0FS0 O
autoridad 111 120 autoridad NCFS000 O
española 121 129 español AQ0FS0 O
residente 130 139 residente NCCS000 O
en 140 142 en SPS00 O
Montevideo 143 153 montevideo NP00000 O
. 153 154 . Fp O
```

Figura 4.1: Formato por columnas para una oración ejemplo.

Cada oración en el texto original (según detectada por Freeling) aparecerá en el formato que se muestra, separada de las demás por una línea en blanco. Las columnas *token_ini* y *token_fin* representan la posición de inicio y fin respectivamente de cada *token* (según reconocido por Freeling) en el texto original. La columna *lema_freeling* representa el lema de la palabra o token, es decir, su forma canónica. El *tag_freeling* representa la etiqueta gramatical que se le asigna a dicho *token*, según el sistema de etiquetas EAGLES[1]. Posteriormente este formato de archivo *.oraciones* pasará por una segunda etapa de preprocesamiento donde sufrirá más transformaciones, agregando más columnas para representar diversos atributos de la entrada. Además, columnas como

[1] http://nlp.lsi.upc.edu/freeling/doc/userman/parole-es.html

las que marcan el inicio y fin de los *tokens* serán eliminadas ya que no aportan ninguna información a los efectos de reconocer los eventos.

El archivo *.oraciones* puede además tener otras columnas cuyo nombre comience por el carácter ∗. Estos atributos representarán información recuperada a partir de las anotaciones manuales de los eventos, como por ejemplo la categoría asignada (puede servir para ver qué tanto se equivoca el etiquetador gramatical de Freeling) o la clase del evento o índice. Las columnas que comiencen con ∗ serán eliminadas antes de usar el archivo como entrada para alguno de los clasificadores, y solo se usarán para análisis posteriores.

La herramienta Freeling es utilizada a través de su programa invocador *analyzer*, y las siguientes opciones están **habilitadas**

- **Reconocimiento de entidades con nombre**. Ej. "Universidad de la República" será reconocido por Freeling como un solo token *Universidad_de_la_República* y etiquetado apropiadamente.
- **Análisis de afijos**. Ej. "contárselo" será separada en tres tokens *contar*, *se* y *lo*, etiquetados distinto, de manera de obtener más información de la naturaleza de la palabra.
- **Detección de locuciones**. Ej. "a base de" será reconocido como un único token *a_base_de*.
- **Detección de números y cifras**. Ej. "diez mil" será reconocido como un único token *diez_mil*.
- **Detección de cantidades**. Ej. "diez mil metros" será reconocido como un único token *diez_mil_metros* (siempre que se use con la detección de números activada).
- **Detección de fechas**. Ej. "primero de setiembre" será reconocido como un único token *primero_de_setiembre*.

Al construir el archivo *.oraciones* solo se consideran aquellas oraciones que terminan en punto final (un carácter cuya etiqueta gramatical es *Fp*). Esto es para evitar agregar títulos que por el uso de mayúsculas pueden confundir al reconocedor de entidades con nombre de Freeling.

4.3. Los clasificadores

Se eligió desarrollar un clasificador basado en *Conditional Random Fields* (CRF) y contrastarlo con otra técnica diferente como lo son los *Support Vector Machines* (SVM). Como se dijo en la sección 3.2.2, ambas técnicas han dado resultados sumamente prometedores en una variedad de tareas.

4.3.1. Elección de *toolkits*

La implementación manual de algoritmos de clasificación como los CRF o SVM es una tarea altamente compleja. Afortunadamente, existen numerosas *toolkits* que ya proveen su implementación en una forma prácticamente lista para usarse, desde interfaces para ser invocadas con algún lenguaje de programación hasta aplicaciones compiladas que realizan las tareas de clasificación a partir de una entrada en algún formato (usualmente, de texto).

Los algoritmos de aprendizaje utilizan atributos de la entrada en una forma que no es nada práctica; por ejemplo en el caso de los CRF los atributos toman la forma de funciones binarias. Por la cantidad de información que se maneja en la tarea de reconocimiento de eventos, bien se puede llegar a tener cientos de miles de estas *feature functions*, por lo que construirlas a mano es una tarea impensable. Por suerte las *toolkits* que implementan este tipo de algoritmos ya traen una componente para calcular atributos binarios a partir de un formato de archivo por columnas del estilo que se vio en la sección 4.2.2.

Las *toolkits* por lo general construyen un **modelo** que consiste en los numerosos parámetros que se aprenden a partir de los datos de entrada. Este modelo se guarda en un archivo que puede ser binario o de texto, para ser usado más adelante al etiquetar nuevas instancias. Naturalmente, el paso costoso es la generación del modelo, siendo el etiquetado a partir de la información contenida en éste sumamente rápido.

4.3.1.1. CRFSuite

CRFSuite [Oka07] es una implementación de CRF lineales (de primer orden) para clasificación secuencial programada en *C*, que pone foco en la velocidad. La idea de este software es poder correr el algoritmo de aprendizaje de un CRF lo más rápido posible, incluso a expensas de la cantidad de memoria que se utiliza. Esto hace que CRFSuite

sea entre 2.2 y 56.4 veces más rápido que otras implementaciones similares de CRF[1] para la tarea de clasificación secuencial planteada en CoNLL-2000 [TKSB00].

4.3.1.2. Yamcha

Yamcha[2] [KM01] es un software orientado a tareas genéricas de clasificación secuencial. Utiliza un paquete externo que implementa el algoritmo de SVM, siendo los posibles candidatos TinySVM[3] o SVMLight[4]. Un trabajo que utilizó Yamcha resultó el sistema ganador de la tarea de CoNLL-2000 [TKSB00], que consistía en parsing superficial. Los CRF fueron propuestos en 2001 [LMP01] por lo que aún no existían cuando se llevó a cabo CoNLL-2000.

Yamcha por defecto utiliza una ventana de atributos de dos elementos hacia atrás y adelante del elemento actual de la secuencia, así como atributos dinámicos (que se deciden durante el proceso de etiquetado). Como CRFSuite no soporta ninguna de estas características de por sí, fueron deshabilitadas en Yamcha (de manera que solo mire cada *fila* de atributos). Igualmente sí se trabajará con ventanas de atributos estáticos, pero agregando éstos como columnas al archivo. De esta forma las técnicas de CRF y SVM pueden competir en igualdad de condiciones.

4.4. Las partes del sistema y su interacción

Para desarrollar el sistema se eligió utilizar el lenguaje Python[5], más que nada por la facilidad con la que se pueden construir sistemas de procesamiento de texto. Las partes se implementan independientemente como varios programas pequeños en dicho lenguaje, que realizan tareas concretas y como salida generan archivos de texto. Todo funciona en un ambiente GNU/Linux, siendo desarrollado sobre el sistema *Ubuntu 10.10*.

Los programas Python deben seguir cierto orden de ejecución y para ayudar a la tarea se programaron scripts de consola (*bash*) que contienen comandos de invocación.

[1] Según datos del *benchmark* disponible en http://www.chokkan.org/software/crfsuite/benchmark.html
[2] http://chasen.org/~taku/software/yamcha/
[3] http://chasen.org/~taku/software/TinySVM/
[4] http://www.cs.cornell.edu/people/tj/svm_light/
[5] http://www.python.org/

Estos scripts también deben ser ejecutados en cierto orden, y para ayudarse se los ha nombrado de forma numerada.

Los scripts de la primera parte, preprocesamiento del corpus, son

1. `Corpus_1_VinculaFreeling.sh`
   Vincula el corpus "crudo", tal cual fue exportado de Protégé, con información obtenida por la herramienta Freeling. Es el proceso que se vio en la sección 4.2.2.

2. `Corpus_2_Preprocesa.sh`
   Filtro para quedarse solo con aquellas oraciones que terminan en punto final.

3. `Corpus_3_BuscaDesfasajesOraciones.sh`
   Es un script opcional. Sirve para cuando se quieren procesar los corpus de ambas anotadoras, de forma de compararlos. Este script chequea que las oraciones de ambos corpus sean las mismas (ante la eventualidad que los archivos de texto puedan ser levemente diferentes). Si no son, se deben realizar correcciones manualmente.

4. `Corpus_4_GeneraTrainTestDev.sh`
   Genera los corpus de entrenamiento, desarrollo y testeo, con todos los atributos que se usarán.

Una vez que se han ejecutado estos scripts, pueden ejecutarse los clasificadores. Para la segunda y tercera parte se utilizan

1. `Eventos_1_ClasificarYEvaluar.sh`
   Realiza el entrenamiento de cada uno de los clasificadores, el etiquetado posterior del corpus con cada uno de ellos, y la evaluación de resultados.

2. `Eventos_2_GeneraCurvaAprendizaje.sh`
   Idem que el anterior, salvo que está orientado a obtener una medida especial denominada curva de aprendizaje. Se verá más sobre esto en la sección 5.2.

Existe otra parte, que no tiene que ver tanto con realizar la clasificación, sino con obtener medidas útiles a partir del corpus. Para esta parte, intervienen los scripts

1. `Analisis_1_CruzarAnotaciones.sh`
   Etiqueta el corpus tomando las anotaciones de una de las anotadoras como *gold standard*, y las de la otra como la salida de un clasificador automático. Es la base para obtener la línea tope, como se vio en la sección 3.6.2.

2. `Analisis_2_AnalizaCorpusTrain.sh`
   Analiza el corpus de entrenamiento para obtener medidas (como cantidad de tokens, eventos según su categoría, terminaciones más correlacionadas con eventos, etc.).

Los scripts anteriores se apoyan en algunos auxiliares que se listan a continuación

```
Aux_BorraResultadosRenombrados.sh
Aux_ClasificarYEvaluar.sh
Aux_FiltrarEventosYEvaluar.sh
Aux_RenombraResultados.sh
```

Para que todo funcione correctamente, se debe mantener la estructura de directorios a partir de la carpeta donde se encuentran los scripts.

4.5. La biblioteca *pln_inco*

Una de las salidas de la tarea de implementación es una pequeña biblioteca en Python, que es utilizada por la mayoría de los programas que realizan las tareas descritas en la sección anterior. Esta biblioteca contiene funciones útiles para tareas de PLN que necesitan hacer uso de información de eventos (esquema SIBILA) o la herramienta Freeling.

Las principales funciones son

- Invocación al *analyzer* de Freeling (por consola) y lectura de la salida.
- Facilita información de dónde comienzan y terminan los *tokens* según Freeling en el texto original (el programa *analyzer* no lo hace).
- Lectura de los archivos XML exportados de Protégé, con las anotaciones de eventos e índices.
- Vínculo entre anotaciones de eventos con información de Freeling.

- Funciones para manejar secuencias (denominadas oraciones) con atributos, leerlas y guardarlas en archivos de texto en un formato de columnas como el que se vio en la sección 4.2.2.

Todas las funciones se encuentran debidamente comentadas, de manera que se pudo generar documentación utilizando la herramienta *epydoc*[1].

[1] http://epydoc.sourceforge.net/

Capítulo 5

Resultados

Se presentan ahora los resultados de la evaluación de los clasificadores, uno basado en CRFSuite y otro en Yamcha. En primera instancia se muestran los resultados parciales con atributos mientras se experimentaba en el corpus de desarrollo, de forma de encontrar el conjunto de atributos que brinde los mejores resultados posibles. Luego se evalúa sobre el corpus de testeo y se comparan los resultados con los del estado del arte para el idioma inglés.

5.1. Experimentos sobre el corpus de desarrollo

Se parte de un conjunto simple de tres atributos como son el *token*, la etiqueta gramatical completa (formato de etiquetas EAGLES) y la primera letra de dicha etiquetas, que representa la categoría gramatical. En esta primera prueba, no se utiliza ninguna información del contexto. Los resultados que se obtuvieron fueron

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	**79,37** / 71,75	**68,60** / 62,02	**73,60** / 66,53
Eventos verbales	**84,97** / 70,90	91,55 / **94,37**	**88,14** / 80,97
Eventos nominales	71,05 / **75,00**	**42,19** / 37,50	**52,94** / 50,00
Eventos adjetivales	**100,00** / **100,00**	6,67 / **13,33**	12,50 / **23,53**

La línea base para el corpus de desarrollo, del 69,08 % medida F, es superado en el caso de los CRF pero no en los SVM. De todos modos la diferencia es poca. Se comienza a probar atributos de contexto, en el caso de la etiqueta gramatical únicamente. Usando una ventana $[-1, 1]$ para la etiqueta gramatical se logra aumentar las medidas. Se

realizaron pruebas con distintos tamaños de ventana, y se llegó a la conclusión que tamaños de ventana grandes disminuyen la performance. Los mejores resultados se obtuvieron con una ventana $[-2, 2]$

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	$79,06$ / $\mathbf{82,55}$	$71,71$ / $\mathbf{75,19}$	$75,20$ / $\mathbf{78,70}$
Eventos verbales	$85,43$ / $\mathbf{87,26}$	$\mathbf{90,85}$ / $\mathbf{96,48}$	$88,05$ / $\mathbf{91,64}$
Eventos nominales	$60,47$ / $\mathbf{61,90}$	$\mathbf{40,62}$ / $\mathbf{40,62}$	$48,60$ / $\mathbf{49,06}$
Eventos adjetivales	$\mathbf{100,00}$ / $\mathbf{100,00}$	$\mathbf{33,33}$ / $\mathbf{33,33}$	$\mathbf{50,00}$ / $\mathbf{50,00}$

Se ve que utilizando esta simple información del contexto, la línea base es ahora superada ampliamente en ambos casos. Para el atributo *token* se encontró que los mejores resultados se obtienen con ventana $[-2, 1]$. Los resultados se resumen en la siguiente tabla

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	$82,83$ / $\mathbf{83,19}$	$\mathbf{74,81}$ / $72,87$	$\mathbf{78,62}$ / $77,69$
Eventos verbales	$\mathbf{87,10}$ / $84,15$	$95,07$ / $\mathbf{97,18}$	$\mathbf{90,91}$ / $90,20$
Eventos nominales	$\mathbf{64,29}$ / $62,50$	$\mathbf{42,19}$ / $31,25$	$\mathbf{50,94}$ / $41,67$
Eventos adjetivales	$\mathbf{100,00}$ / $\mathbf{100,00}$	$\mathbf{40,00}$ / $33,33$	$\mathbf{57,14}$ / $50,00$

Se agrega ahora como atributo el lema de cada una de las palabras, con una ventana $[-2, 1]$ (al igual que los *tokens*). También se usa una ventana $[-2, 2]$ para la categoría gramatical, para obtener resultados levemente superiores a los anteriores

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	$82,70$ / $\mathbf{83,33}$	$\mathbf{75,97}$ / $73,64$	$\mathbf{79,19}$ / $78,19$
Eventos verbales	$\mathbf{87,26}$ / $84,24$	$96,48$ / $\mathbf{97,89}$	$\mathbf{91,64}$ / $90,55$
Eventos nominales	$59,09$ / $\mathbf{64,52}$	$\mathbf{40,62}$ / $31,25$	$\mathbf{48,15}$ / $42,11$
Eventos adjetivales	$\mathbf{100,00}$ / $\mathbf{100,00}$	$\mathbf{40,00}$ / $\mathbf{40,00}$	$\mathbf{57,14}$ / $\mathbf{57,14}$

Se agregan como atributos el tiempo y modo para los verbos (si las palabras no son verbos éstos adquieren un valor `<NO_VERBO>`), y las primeras dos letras de la etiqueta gramatical, todos con una ventana $[-2, 2]$

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	$\mathbf{84,32}$ / $83,05$	$\mathbf{77,13}$ / $75,97$	$\mathbf{80,57}$ / $79,35$
Eventos verbales	$\mathbf{90,07}$ / $85,80$	$95,77$ / $\mathbf{97,89}$	$\mathbf{92,83}$ / $91,45$
Eventos nominales	$63,04$ / $\mathbf{63,16}$	$\mathbf{45,31}$ / $37,50$	$\mathbf{52,73}$ / $47,06$
Eventos adjetivales	$\mathbf{100,00}$ / $\mathbf{100,00}$	$40,00$ / $\mathbf{53,33}$	$57,14$ / $\mathbf{69,57}$

Los resultados son más de 1 % superiores al caso anterior. Se probó agregar como atributo un indicador que mostraba si la palabra terminaba en una de las terminaciones de 4 letras más correlacionadas según se vio en 3.6.1. Sin embargo se obtuvieron mejores resultados teniendo la terminación de 4 letras como un atributo en sí y dejando que el clasificador infiera cuándo éstas están correlacionadas con la existencia de eventos. Agregando la terminación como atributo en una ventana $[-2, 1]$, la que dio mejores resultados, se obtuvo

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	**84,77** / 82,50	**79,84** / 76,74	**82,24** / 79,52
Eventos verbales	**89,47** / 85,09	95,77 / **96,48**	**92,52** / 90,43
Eventos nominales	**66,67** / 64,29	**53,12** / 42,19	**59,13** / 50,94
Eventos adjetivales	**100,00 / 100,00**	46,67 / **60,00**	63,64 / **75,00**

Intentar hacer lo mismo con el inicio (4 letras) de las palabras parece no tener demasiado beneficio. Para el clasificador CRF el resultado empeora algo más de 1 %, mientras que el SVM mejora levemente. Se probó también con inicios de 3 letras para obtener las mismas conclusiones. Por eso, se decidió que el inicio de las palabras no formaría parte del conjunto final de atributos. Los resultados utilizándolo fueron

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	**84,17** / 83,61	**78,29** / 77,13	**81,12** / 80,24
Eventos verbales	**88,00** / 85,80	92,96 / **97,89**	90,41 / **91,45**
Eventos nominales	**68,63** / 67,50	**54,69** / 42,19	**60,87** / 51,92
Eventos adjetivales	**100,00 / 100,00**	**53,33 / 53,33**	**69,57 / 69,57**

Se vio en la sección 3.4 que en la literatura se suele utilizar atributos bigrama. Se decidió probar lo propio con un bigrama que indique las etiquetas gramaticales, de la forma t_{-1}, t_0 y t_0, t_1. Probablemente sea porque el corpus es pequeño, pero utilizar este atributo hace que los resultados empeoren como se puede ver en la siguiente tabla

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	81,38 / **83,19**	**77,91** / 76,74	79,60 / **79,84**
Eventos verbales	**88,24** / 85,80	95,07 / **97,89**	**91,53** / 91,45
Eventos nominales	58,18 / **65,85**	**50,00** / 42,19	**53,78** / 51,43
Eventos adjetivales	**100,00 / 100,00**	**46,67 / 46,67**	**63,64 / 63,64**

Utilizando otros atributos de bigrama (como w_{-1}, w_0 y w_0, w_1, o t_{-1}, w_0 y t_0, w_1, y alguna otra combinación) se obtuvieron las mismas conclusiones, incluso probando bigramas únicamente con la categoría gramatical. Agregando un atributo que indica el tipo para los nombres (propio, común, otro) con ventana $[-2, 2]$ mejora levemente el desempeño para SVM pero empeora para CRF (respecto a los mejores resultados obtenidos hasta el momento)

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	**84,23** / 82,38	**78,68** / 77,91	**81,36** / 80,08
Eventos verbales	**89,33** / 85,09	94,37 / **96,48**	**91,78** / 90,43
Eventos nominales	**66,00** / 65,22	**51,56** / 46,88	**57,89** / 54,55
Eventos adjetivales	**100,00 / 100,00**	46,67 / **60,00**	63,64 / **75,00**

Agregando además de lo último un atributo que indica la mayúscula inicial para una ventana $[-2, 1]$ se obtiene

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	**84,30** / 83,40	**79,07** / 77,91	**81,60** / 80,56
Eventos verbales	**88,74** / 85,62	94,37 / **96,48**	**91,47** / 90,73
Eventos nominales	66,67 / **68,18**	**53,12** / 46,88	**59,13** / 55,56
Eventos adjetivales	**100,00 / 100,00**	46,67 / **60,00**	63,64 / **75,00**

Con el atributo de la mayúscula inicial, pero quitando el tipo de nombres, se obtienen los mejores resultados hasta el momento para ambos clasificadores

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	**84,77** / 83,33	**79,84** / 77,52	**82,24** / 80,32
Eventos verbales	**89,47** / 85,62	95,77 / **96,48**	**92,52** / 90,73
Eventos nominales	66,67 / **67,44**	**53,12** / 45,31	**59,13** / 54,21
Eventos adjetivales	**100,00 / 100,00**	46,67 / **60,00**	63,64 / **75,00**

Se presentan los valores gráficamente en la figura 5.1. Se ve que el CRFSuite se comporta casi 2 % mejor que Yamcha en la medida F. De estos resultados se concluye que los atributos a utilizar para evaluar sobre el corpus de testeo son

- *token* y lema, ventana $[-2, 1]$.
- la etiqueta gramatical asignada por Freeling, ventana $[-2, 2]$.

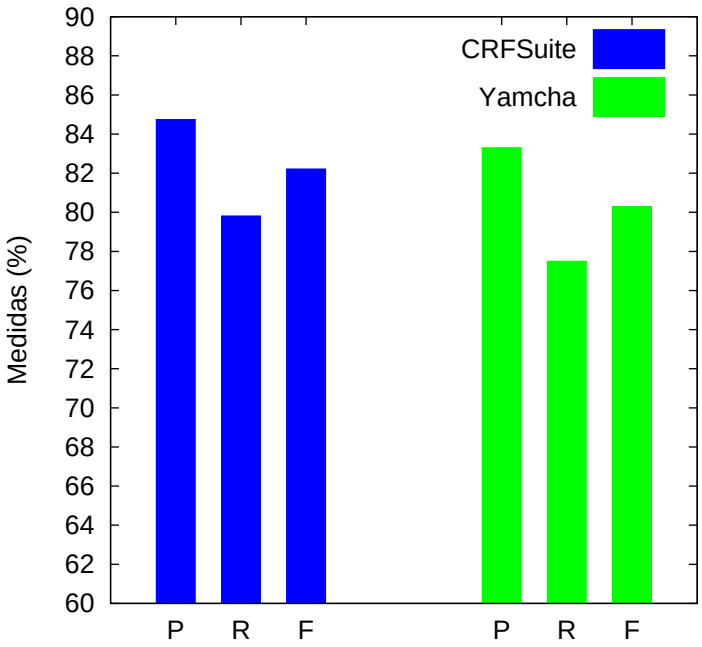

Figura 5.1: Resultados para la mejor combinación de atributos hallada en el **corpus de desarrollo**. A la izquierda las medidas de precisión, *recall* y F para CRFSuite, y a la derecha para Yamcha.

- categoría gramatical (primera letra de la etiqueta gramatical, para todas las categorías), ventana $[-2, 2]$.
- primeras dos letras de la etiqueta gramatical, ventana $[-2, 2]$.
- tiempo y modo (para los verbos), ventana $[-2, 2]$.
- uso de mayúsculas iniciales, ventana $[-2, 1]$.
- terminación de las palabras (4 letras), ventana $[-2, 1]$.

5.2. Evaluación sobre el corpus de testeo

Con el conjunto de atributos finales derivados de los experimentos en la sección anterior, se ejecuta el clasificador sobre el corpus de testeo. Los resultados pueden observarse en el cuadro 5.1 y gráficamente en la figura 5.2. La línea base de $61,84\,\%$ en medida F es superado ampliamente tanto en el caso de CRF como SVM, por $14,88\,\%$

y 18,5 % respectivamente. Al contrario de lo que se observaba en los resultados sobre el corpus de desarrollo, el SVM da medidas más altas en todos los casos excepto los eventos adjetivales. Ya se había hablado en la sección 2.3.3 de que se contaba con relativamente pocos datos, por lo que no es posible aquí llegar a conclusiones certeras de qué técnica se comporta mejor. A ambos clasificadores les falta para alcanzar la línea tope de una medida F de aproximadamente 92,3 %.

	CRFSuite / Yamcha		
	Precision (%)	**Recall (%)**	**Medida F (%)**
Global	81,65 / **84,68**	72,36 / **76,42**	76,72 / **80,34**
Eventos verbales	83,22 / **84,18**	91,85 / **98,52**	87,32 / **90,78**
Eventos nominales	71,79 / **78,95**	41,18 / **44,12**	52,34 / **56,60**
Eventos adjetivales	**100,00 / 100,00**	**50,00 / 50,00**	**66,67 / 66,67**

Cuadro 5.1: Resultados de la evaluación de los clasificadores sobre el **corpus de testeo**.

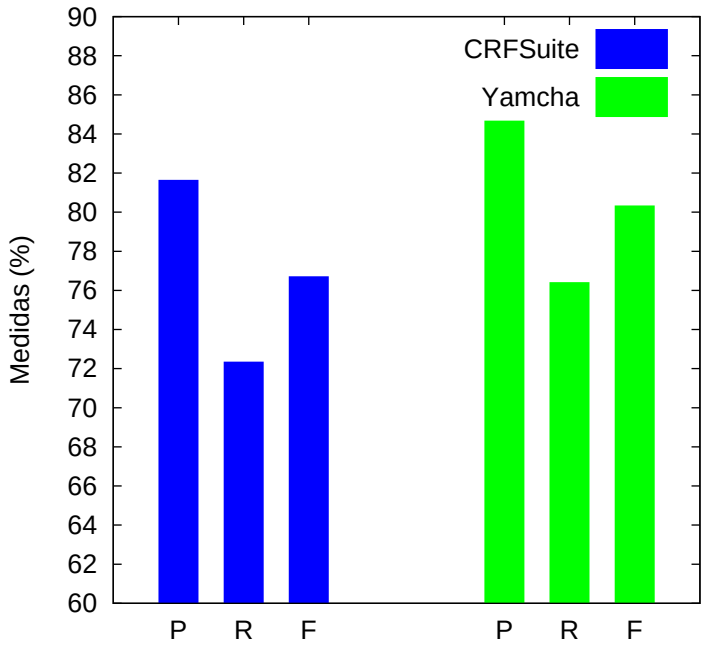

Figura 5.2: Resultados de la evaluación sobre el **corpus de testeo**. A la izquierda las medidas de precisión, *recall* y *F* para CRFSuite, y a la derecha para Yamcha.

En la sección 2.4.2 se presentaron resultados para el reconocimiento de eventos

obtenidos por varios trabajos, para el idioma inglés. EVITA [SKVP05] logra 74,03 % de precisión, 87,31 % *recall* y medida F de 80,12 %. Sim-Evita, implementado por los autores de [BM06], logra una medida F de 73 %. En [BA07] se reporta una medida F de 78,6 % y 80,3 % utilizando la técnica de *word profiling*. El sistema STEP [BM06] logró una medida F de 75,9 %. En [MB08] se llega a una medida F de 76,4 %. Un sistema basado en CRF logra una medida F de 81,4 % en [LSNC10]. La misma información se presenta en el cuadro 5.2.

Sistema	**Medida F (%)**
Este trabajo	76,72 (CRFSuite) / 80,34 (Yamcha)
EVITA [SKVP05]	80,12
Sim-Evita [BM06]	73
Boguraev, Ando [BA07]	78,6 / 80,3 % (con *word profiling*)
STEP [BM06]	75,9
March, Baldwin [MB08]	76,4
Llorens et al. [LSNC10]	81,4

Cuadro 5.2: Comparación de la medida F para la tarea de reconocimiento de eventos, entre distintos sistemas. Notar que se trata solo de un cuadro ilustrativo, pues las medidas no son directamente comparables.

Puede verse que los resultados obtenidos en este trabajo para el español son bastante similares que los que se han logrado para el inglés. Si bien difieren tanto el esquema de anotación (TimeML vs SIBILA), los métodos de evaluación (algunos autores utilizaron *k-fold cross validation* en vez de separar en corpus de desarrollo y testeo) como los atributos y métodos utilizados por cada trabajo, los resultados al fin son muy similares con valores de medida F que se acerca (o supera levemente) al 80 %.

En este trabajo el sistema basado en Yamcha termina obteniendo algo más de 3,6 % de ventaja sobre CRFSuite en la medida F. En [LSNC10] se llega a un resultado opuesto, concluyendo que los CRF tienen mejor poder de generalización que los SVM, sobre todo para eventos nominales. Se presentan los resultados discriminados por la categoría del evento, tal cual se ha hecho aquí. Logran valores de medida F de 91,33 %, 58,42 % y 48,35 % para verbos, nombres y adjetivos respectivamente. Todas las cifras son muy cercanas a las obtenidas por el clasificador Yamcha, excepto los adjetivos donde se obtuvo un resultado superior, pero dada la escasez de datos no se pueden

extraer conclusiones. En este trabajo se supera la línea base por un 19 %, casi el mismo resultado obtenido por Yamcha. Las medidas que se obtuvieron aquí con CRFSuite también son del mismo orden que las presentadas en [LSNC10].

Resulta interesante plantearse cómo se comporta el sistema según la cantidad de datos de entrenamiento. Para eso se construye la denominada **curva de aprendizaje**. La idea es graficar el resultado del sistema (medida F) en función del porcentaje del corpus de entrenamiento completo que se utiliza para entrenar. Se toman los valores usando un 10 %, 20 %, ... y así hasta el corpus completo.

Las figuras 5.3 y 5.4 muestran las curvas de aprendizaje para los corpus de desarrollo y testeo respectivamente. Es clara en la forma de la curva (sobre todo la del corpus de testeo) como al agregar los primeros datos se crece de forma empinada pero luego el crecimiento se desacelera, hasta el punto que puede parecer que agregar más datos no hará demasiada diferencia en los resultados. No obstante el sistema no parece haber llegado al punto en el que la performance no mejora al agregar más datos, dando indicios de que un corpus de mayor tamaño es necesario.

5.2.1. Algunos aciertos y errores cometidos por los clasificadores

A continuación se presentan algunos ejemplos concretos de decisiones tomadas por los clasificadores en secuencias extraídas del corpus de testeo. Se muestran en un formato por columnas los *tokens* y las etiquetas correctas, así como las asignadas por CRFSuite y Yamcha. Además, se incluye la información de los atributos `CATEGORIA` y `CLASE` para los eventos, según las anotaciones manuales.

En el cuadro 5.3 se muestran ejemplos de los errores más comunes, como lo son omitir eventos y marcar eventos inexistentes, en particular los de tipo nominal. La secuencia **(2)** muestra un caso en el que los clasificadores se equivocan para eventos de categoría nombre y adjetivo.

Examinando la salida de los clasificadores en el corpus de desarrollo o testeo, puede verse que la gran mayoría de las veces que ocurren errores éstos son cometidos tanto por CRFSuite como por Yamcha. Esto es indicio de la dificultad inherente al problema de reconocimiento de eventos, ya que dos técnicas que usan paradigmas distintos suelen cometer los mismos errores. En el caso de las secuencias del cuadro 5.3, ambos clasificadores dan la misma salida. En el cuadro 5.4 se pueden ver ejemplos de cuando solamente es uno de los clasificadores el que se equivoca al reconocer eventos.

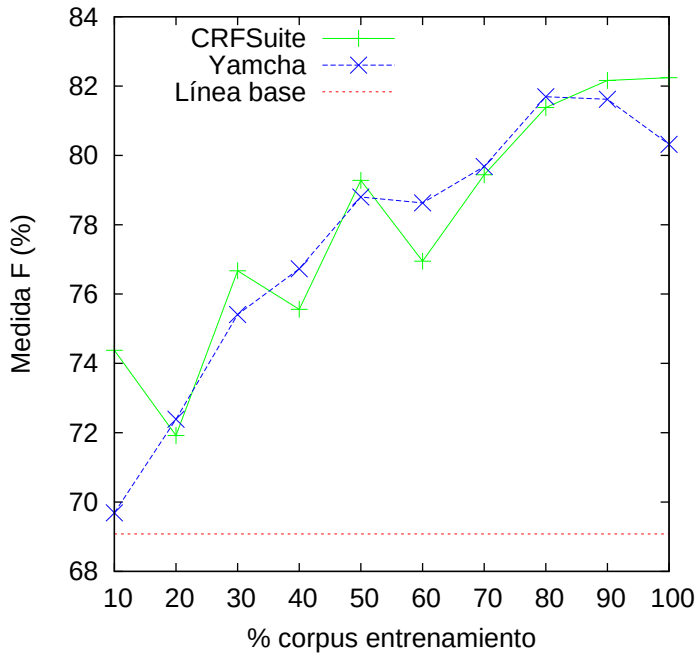

Figura 5.3: Curva de aprendizaje para el **corpus de desarrollo**.

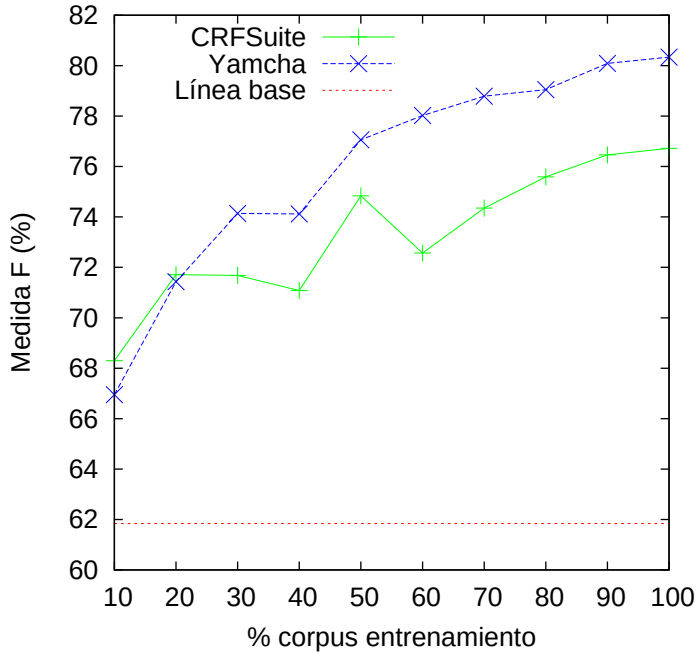

Figura 5.4: Curva de aprendizaje para el **corpus de testeo**.

El cuadro 5.5 muestra errores algo más complejos, en los que interviene la extensión del evento. En la secuencia **(6)** CRFSuite falla al identificar la palabra *conocido* como parte del evento, etiquetándola como un evento nuevo, mientras que Yamcha no comete el error. La secuencia **(7)** muestra un caso complementario, en el que Yamcha no es capaz de identificar que *las* pertenece al evento. Esta división resulta de la separación en *tokens* que realiza Freeling, donde *valorarlas* se separa en dos *tokens*, *valorar* y *las*.

La secuencia **(8)** muestra un caso de una expresión o frase que a su vez constituye un evento, *tiró la toalla*. Esta frase no aparece ninguna vez en el corpus de entrenamiento, por lo que es entendible que ambos clasificadores marquen como evento el verbo pero no el resto de la expresión, al nunca haber presenciado un ejemplo que les dé el indicio necesario.

Token	Etiqueta	CRFSuite	Yamcha	Categoría	Clase
(1) No marcan evento cuando hay, caso nombre					
...					
ellos	O	O	O	-	-
no	O	O	O	-	-
exigirán	B_EVENTO	B_EVENTO	B_EVENTO	VERBO	ACCION_INT_CAUS
su	O	O	O	-	-
marcha	B_EVENTO	O	O	NOMBRE	OCURRENCIA
.	O	O	O	-	-
(2) No marcan evento cuando hay, caso nombre y adjetivo					
...					
en	O	O	O	-	-
ellos	O	O	O	-	-
una	O	O	O	-	-
situación	B_EVENTO	O	O	NOMBRE	OCURRENCIA
que	O	O	O	-	-
creyeron	B_EVENTO	B_EVENTO	B_EVENTO	VERBO	ESTADO_INT
afortunada	B_EVENTO	O	O	ADJETIVO	OCURRENCIA
”	O	O	O	-	-
.	O	O	O	-	-
(3) Marcan evento cuando no hay					
...					
que	O	O	O	-	-
no	O	O	O	-	-
tenían	B_EVENTO	B_EVENTO	B_EVENTO	VERBO	OCURRENCIA
glóbulos	O	B_EVENTO	B_EVENTO	-	-
blancos	O	O	O	-	-
propios	O	O	O	-	-
.	O	O	O	-	-

Cuadro 5.3: Errores de omisión de eventos y de marcación de eventos inexistentes.

Token	Etiqueta	CRFSuite	Yamcha	Categoría	Clase
(4) Se equivoca CRFSuite pero no Yamcha					
...					
derecho	O	O	O	-	-
para	O	O	O	-	-
que	O	O	O	-	-
llevaran_a_cabo	B_EVENTO	B_EVENTO	B_EVENTO	VERBO	ACCION_INT_CAUS
un	O	O	O	-	-
estudio	B_EVENTO	O	B_EVENTO	NOMBRE	OCURRENCIA
.	O	O	O	-	-
(5) Se equivoca Yamcha pero no CRFSuite					
...					
el	O	O	O	-	-
komercni_banka	O	O	O	-	-
tenía	B_EVENTO	B_EVENTO	B_EVENTO	VERBO	EXISTENCIA
una	O	O	O	-	-
plantilla	B_EVENTO	B_EVENTO	O	NOMBRE	ESTADO
de	O	O	O	-	-
13.487	O	O	O	-	-
empleados	O	O	O	-	-
...					

Cuadro 5.4: Ejemplo de errores cometidos por un clasificador pero no por el otro.

Token	Etiqueta	CRFSuite	Yamcha	Categoría	Clase
(6) CRFSuite marca incorrectamente la extensión del evento, Yamcha no					
...					
el	O	O	O	-	-
lugar	O	O	O	-	-
era	B_EVENTO	B_EVENTO	B_EVENTO	VERBO	PERCEPCION
conocido	I_EVENTO	B_EVENTO	I_EVENTO	VERBO	PERCEPCION
de	O	O	O	-	-
antiguo	O	O	O	-	-
...					
(7) Yamcha marca incorrectamente en la extensión del evento, CRFSuite no					
...					
el	O	O	O	-	-
único	O	O	O	-	-
baremo	B_EVENTO	O	O	NOMBRE	OCURRENCIA
para	O	O	O	-	-
valorar	B_EVENTO	B_EVENTO	B_EVENTO	VERBO	OCURRENCIA
las	I_EVENTO	I_EVENTO	O	VERBO	OCURRENCIA
.	O	O	O	-	-
(8) Una expresión que es evento, *"tiró la toalla"*					
...					
tras	O	O	O	-	-
un	O	O	O	-	-
accidente	B_EVENTO	O	O	NOMBRE	OCURRENCIA
tiró	B_EVENTO	B_EVENTO	B_EVENTO	VERBO	OCURRENCIA
la	I_EVENTO	O	O	VERBO	OCURRENCIA
toalla	I_EVENTO	O	O	VERBO	OCURRENCIA
...					

Cuadro 5.5: Errores en la extensión del evento.

Capítulo 6

Conclusiones y trabajo futuro

A lo largo de este proyecto se construyó un sistema de reconocimiento de eventos para textos en idioma español, basado en técnicas de aprendizaje automático. Se utilizó como base un esquema de anotación denominado SIBILA, y un corpus de texto etiquetado para la tarea por dos estudiantes avanzados de lingüística.

Se estudió el problema de reconocimiento de eventos en un marco más general, como instancia particular de una clase de problemas llamados de clasificación secuencial, de manera que éste se reduce a la clasificación de tokens con ciertas etiquetas predefinidas. Se realizó un análisis de varias técnicas que se utilizan para resolver problemas de este tipo, no solo desde un punto de vista pragmático sino intentando comprender su funcionamiento con un nivel razonable de detalle. La salida de este análisis fue un documento de métodos de clasificación secuencial donde se introducen las técnicas estudiadas. Para la tarea se eligió utilizar dos paradigmas de aprendizaje radicalmente diferentes, que han dado buenos resultados en numerosos trabajos del área: los *Conditional Random Fields* (CRF) y los *Support Vector Machines* (SVM).

El corpus etiquetado se dividió en entrenamiento, desarrollo y testeo de manera de utilizarlo tanto para entrenar como para evaluar el desempeño de los clasificadores. También se obtuvieron medidas de las líneas base y tope para el reconocimiento de eventos, para lograr tener cierta perspectiva de los resultados obtenidos por el sistema. Se vio que la definición de atributos relevantes para la tarea era uno de los principales problemas a enfrentar. Utilizando el corpus de desarrollo se hicieron diversas pruebas hasta llegar a una combinación de atributos que diera buenos resultados, y posteriormente se evaluó contra el corpus de testeo. Los resultados fueron muy similares a aquellos obteni-

dos por otros trabajos para el idioma inglés [SKVP05, BM06, BA07, MB08, LSNC10], logrando una medida F de 76,72 % para CRF y 80,34 % para SVM.

Como trabajo para realizar en el futuro puede plantearse un mayor estudio de nuevos atributos que puedan aportar información útil a la tarea. Una limitación de trabajar con el idioma español es que la cantidad de recursos disponibles es significativamente menor a la del inglés. Por ejemplo en [LSNC10] se cuenta con un etiquetador de roles semánticos para usarlos como atributo, pero no se ha encontrado ningún etiquetador de roles para el español. Con el tiempo es posible que aparezcan más recursos para el español, abriendo una nueva gama de posibles atributos. Algo más viable a corto plazo es la realización de un estudio para ver si es factible utilizar información de parsing o *chunking*, por ejemplo la brindada por Freeling. También puede plantearse aprovechar los índices. El problema que se tuvo era que habían muy pocos índices de eventividad en el corpus, pero puede pensarse en construir una gran lista con palabras que pueden ser índices de eventividad para usarse de atributo. El trabajo debe ser extendido para que además de reconocer eventos los clasifique, es decir, aprenda el valor del atributo `CLASE`. La mayoría de los trabajos del área no solo se ocupan del reconocimiento de eventos sino de su clasificación. Es pertinente también lograr determinar otros atributos de los eventos, como factividad, modalidad, tiempo (que no siempre coincide con la conjugación verbal), etc.

Quizá el principal problema enfrentado, como se mencionó en numerosas ocasiones, es el tamaño reducido del corpus con el que se cuenta. Sería prudente realizar un esfuerzo para construir un corpus anotado de mayor tamaño de manera que los resultados obtenidos sean más significativos. Si esto no es posible, existen otras técnicas que pueden aplicarse. Algunas se basan en la clasificación de datos sin etiquetar y funcionan forma iterativa. Los más conocidos son los algoritmos de bootstrapping [KMM08] y co-training [BM98]. Otras técnicas, llamadas *word profiling*, explotan datos sin anotar de forma de mejorar la representación de los atributos [Sch92]. Trabajos relativamente recientes [And04, BA05b] reportan mejoras significativas de performance sobre todo cuando el corpus de entrenamiento es pequeño o diferente del corpus de evaluación. Resultaría de interés el estudio de aplicabilidad de estas técnicas al problema de reconocimiento de eventos.

Es claro que a diferencia de otros problemas como el etiquetado gramatical o reconocimiento de entidades con nombre, la identificación de eventos dista de ser un problema

resuelto. Basta analizar los trabajos que se han realizado para el inglés para ver que, sin importar la técnica que se utilice, se está lejos de alcanzar los niveles de performance humanos.

Referencias

[And04] Rie Kubota Ando. Exploiting unannotated corpora for tagging and chunking. En *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, ACLdemo '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. 68

[BA05a] Branimir Boguraev y Rie Kubota Ando. TimeBank-Driven TimeML Analysis. En Graham Katz, James Pustejovsky, y Frank Schilder, editores, *Annotating, Extracting and Reasoning about Time and Events*, number 05151 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. 15, 17

[BA05b] Branimir Boguraev y Rie Kubota Ando. TimeML-compliant text analysis for temporal reasoning. En *Proceedings of the 19th international joint conference on Artificial intelligence*, páginas 997–1003, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc. 15, 28, 68

[BA07] Branimir Boguraev y Rie Kubota Ando. Effective use of TimeBank for TimeML analysis. En *Proceedings of the 2005 international conference on Annotating, extracting and reasoning about time and events*, páginas 41–58, Berlin, Heidelberg, 2007. Springer-Verlag. 17, 59, 68

[BKL09] Steven Bird, Ewan Klein, y Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing, 2009. 22, 26

[BM97] Eric Brill y Raymond J. Mooney. An overview of empirical natural language processing. *AI Magazine*, 18(4):13–24, 1997. 2

REFERENCIAS

[BM98] Avrim Blum y Tom Mitchell. Combining labeled and unlabeled data with co-training. En *Proceedings of the eleventh annual conference on Computational learning theory*, COLT' 98, páginas 92–100, New York, NY, USA, 1998. ACM. 68

[BM06] Steven Bethard y James H. Martin. Identification of event mentions and their semantic class. En *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, páginas 146–154, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. 17, 29, 59, 68

[Car96] Jean Carletta. Assessing agreement on classification tasks: The Kappa statistic. *Computational Linguistics*, 22:249–254, 1996. 40

[Cha97] Eugene Charniak. Statistical techniques for natural language parsing. *AI Magazine*, 18:33–44, 1997. 36

[CN02] Hai Leong Chieu y Hwee Tou Ng. A maximum entropy approach to information extraction from semi-structured and free text. En *Eighteenth national conference on Artificial intelligence*, páginas 786–791, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. 38

[Des11] Alan Descoins. Métodos de Clasificación Secuencial. 2011. 5, 23

[DRA03] Naomi Daniel, Dragomir Radev, y Timothy Allison. Sub-event based multi-document summarization. En *Proceedings of the HLT-NAACL 03 on Text summarization workshop - Volume 5*, HLT-NAACL-DUC '03, páginas 9–16, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. 4

[Dzi09] D. M. Dziuda. *Data Mining for Genomics and Proteomics: Analyis of Gene and Protein Expression Data*. John Wiley & Sons, Inc., New York, NY, USA, 2009. 25

[Fer01] Lisa Ferro. TIDES: Instruction manual for the annotation of temporal expressions. *The MITRE Corporation*, 2001. 15

[Fis09] Wolfgang Fischl. Part of speech tagging - a solved problem? Seminar in Artificial Intelligence, 2009. 21

[Guy03] Isabelle Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003. 22

[HAJ90] Xuedong Huang, Yasuo Ariki, y Mervyn Jack. *Hidden Markov Models for Speech Recognition*. Columbia University Press, New York, NY, USA, 1990. 21

[HZCPn04] Xuming He, Richard S. Zemel, y Miguel Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. En *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*, CVPR'04, páginas 695–703, Washington, DC, USA, 2004. IEEE Computer Society. 24

[JM08] Daniel Jurafsky y James H. Martin. *Speech and Language Processing*. Pearson Prentice Hall, second edition, 2008. 21

[JT03] Dominic R. Jones y Cynthia A. Thompson. Identifying events using similarity and context. En *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, páginas 135–141, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. 17

[Kee09] S. Sathiya Keerthi. CRF versus SVM-Struct for Sequence Labeling. 2009. 24

[KH03] Sanjiv Kumar y Martial Hebert. Discriminative fields for modeling spatial dependencies in natural images. En *In NIPS*. MIT Press, 2003. 24

[KM01] Taku Kudo y Yuji Matsumoto. Chunking with support vector machines. En *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, páginas 1–8, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. 48

[KMM08] Z Kalal, J Matas, y K Mikolajczyk. Weighted sampling for large scale boosting. *BMVC*, 2008. 68

[Lan94] Pat Langley. Selection of relevant features in Machine Learning. En *In Proceedings of the AAAI Fall symposium on relevance*, páginas 140–144. AAAI Press, 1994. 22

[LCWG05] Yan Liu, Jaime Carbonell, Peter Weigele, y Vanathi Gopalakrishnan. Segmentation conditional random fields (scrfs): A new approach for protein fold recognition. En *Proc. of the 9th Ann. Intl. Conf. on Comput. Biol. (RECOMB*, páginas 14–18. ACM Press, 2005. 24

[LMP01] John D. Lafferty, Andrew McCallum, y Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. En *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, páginas 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. 24, 48

[LSNC10] Hector Llorens, Estela Saquete, y Borja Navarro-Colorado. TimeML events recognition and classification: learning CRF models with semantic roles. En *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, páginas 725–733, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. 18, 20, 29, 59, 60, 68

[MB08] Olivia March y Timothy Baldwin. Automatic event reference identification. En *Proceedings of the Australasian Language Technology Association Workshop 2008*, páginas 79–87, Hobart, Australia, December 2008. 18, 30, 59, 68

[NFM00] Natalya Fridman Noy, Ray W. Fergerson, y Mark A. Musen. The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. En *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, EKAW '00, páginas 17–32, London, UK, 2000. Springer-Verlag. 12

[Ogr06] Philip V. Ogren. Knowtator: A Protégé plug-in for annotated corpus construction. En Robert C. Moore, Jeff A. Bilmes, Jennifer Chu-Carroll,

y Mark Sanderson, editores, *HLT-NAACL*. The Association for Computational Linguistics, 2006. 12

[Oka07] Naoaki Okazaki. CRFsuite: a fast implementation of Conditional Random Fields (CRFs). http://www.chokkan.org/software/crfsuite/, 2007. 47

[PCI+03] James Pustejovsky, José Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, y Graham Katz. TimeML: Robust specification of event and temporal expressions in text. En *in Fifth International Workshop on Computational Semantics (IWCS-5*, 2003. 2, 8, 16

[PFM04] Fuchun Peng, Fangfang Feng, y Andrew McCallum. Chinese segmentation and new word detection using conditional random fields. En *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Morristown, NJ, USA, 2004. Association for Computational Linguistics. 24

[PHS+03] James Pustejovsky, Patrik Hanks, Roser Saurí, A. See, Robert Gaizauskas, Andrea Setzer, Dragomir R. Radev, Beth Sundheim, David Day, Lisa Ferro, y M. Lazo. The TIMEBANK Corpus. En *Corpus Linguistics*, páginas 647–656, 2003. 16

[Pus02] James Pustejovsky. TERQAS: Time and Event Recognition for Question Answering Systems. En *ARDA Workshop*, 2002. 4

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. En *Proceedings of the IEEE*, páginas 257–286, 1989. 21

[Sch92] H. Schütze. Dimensions of meaning. En *Proceedings of the 1992 ACM/IEEE conference on Supercomputing*, Supercomputing '92, páginas 787–796. IEEE Computer Society Press, 1992. 68

[Set05] Burr Settles. Abner: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21:3191–3192, July 2005. 24

REFERENCIAS

[SG00a] A. Setzer y R. Gaizauskas. Annotating events and temporal information in newswire texts. En *Proceedings of the Second International Conference on Language Resources and Evaluation*, páginas 1287–1294, Athens, 2000. 16

[SG00b] A. Setzer y R. Gaizauskas. Building a temporally annotated corpus for information extraction. En *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC2000) Workshop on Information Extraction Meets Corpus Linguistics*, páginas 9–14, Athens, 2000. 16

[SKVP05] Roser Saurí, Robert Knippen, Marc Verhagen, y James Pustejovsky. Evita: a robust event recognizer for QA systems. En *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, páginas 700–707, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. 4, 5, 17, 18, 59, 68

[SP03] Fei Sha y Fernando Pereira. Shallow parsing with conditional random fields. En *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, páginas 134–141, Morristown, NJ, USA, 2003. Association for Computational Linguistics. 24

[SS05] Kengo Sato y Yasubumi Sakakibara. Rna secondary structural alignment with conditional random fields. *Bioinformatics*, 21:237–242, January 2005. 24

[TAK02] B. Taskar, P. Abbeel, y D. Koller. Discriminative probabilistic models for relational data. En *Proc. Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, Edmonton, Canada, 2002. 24

[TKS02] Erik F. Tjong Kim Sang. Introduction to the conll-2002 shared task: language-independent named entity recognition. En *Proceedings of the 6th conference on Natural language learning - Volume 20*, COLING-02, páginas 1–4, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. 34

[TKSB00] Erik F. Tjong Kim Sang y Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. En Claire Cardie, Walter Daelemans, Claire Nedellec, y Erik Tjong Kim Sang, editores, *Proceedings of CoNLL-2000 and LLL-2000*, páginas 127–132. Lisbon, Portugal, 2000. 34, 48

[TKSDM03] Erik F. Tjong Kim Sang y Fien De Meulder. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. En *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, páginas 142–147, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. 14

[Vap95] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995. 24

[Ven67] Zeno Vendler. *Linguistics and philosophy*, capítulo Verbs and times, páginas 97–121. Cornell University Press, Ithaca, NY., 1967. 15

[VNT96] Stephan Vogel, Hermann Ney, y Christoph Tillmann. HMM-based word alignment in statistical translation. En *In COLING '96: The 16th Int. Conf. on Computational Linguistics*, páginas 836–841, 1996. 21

[WC05] Janyce Wiebe y Claire Cardie. Annotating expressions of opinions and emotions in language. En *Language Resources and Evaluation (formerly Computers and the Humanities*, 2005. 40

[WMR08] Dina Wonsever, Marisa Malcuori, y Aiala Rosá. SIBILA: Esquema de anotación de eventos. Reporte Técnico RT 08-11, PEDECIBA, 2008. 8, 16

[WMR09] Dina Wonsever, Marisa Malcuori, y Aiala Rosá. Factividad de los eventos referidos en textos. Reporte Técnico RT 09-12, PEDECIBA, 2009. 10

[ZS02] GuoDong Zhou y Jian Su. Named entity recognition using an HMM-based chunk tagger. En *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, páginas 473–480, Morristown, NJ, USA, 2002. Association for Computational Linguistics. 21