Cuadernos de la Facultad de Ingeniería e Informática
Universidad Católica de Salta, vol. 7, 2012: 51-71

Extracción de entidades con nombre

M. Alicia Pérez[1], A. Carolina Cardoso[1]

Resumen

El uso de la minería de textos está aumentando en la actualidad ya que las organizaciones quieren aprovechar el potencial de la gran cantidad de información de que disponen en forma de documentos de texto u otra información no estructurada; este tipo de datos supone un porcentaje considerable de los datos con que cuentan las organizaciones. Una de las tareas integrales para la minería de textos es la extracción de entidades con nombre (NER). El presente trabajo describe los principales enfoques en uso para esta tarea, centrándose especialmente en los específicos para el aprendizaje de secuencias. Estas técnicas se aplican a un problema concreto, la extracción de información de un corpus de 8000 documentos correspondientes a resoluciones rectorales, de los que se extraen nombres de personas, diversos departamentos académicos y otras organizaciones vinculadas a la universidad. El trabajo describe la arquitectura para la gestión de información no estructurada en la que se enmarca esta tarea y de la que forma parte, en la que las entidades extraídas permiten la búsqueda semántica de información. Los experimentos muestran que los campos aleatorios condicionales (CRFs) son la técnica más adecuada para el problema de extracción de entidades con nombre.

1. Introducción

El interés en la *minería de textos* ha crecido enormemente en los últimos años, debido a la creciente cantidad de documentos disponibles en forma digital y la también creciente necesidad de organizarlos y aprovechar el conocimiento contenido en ellos. La minería de textos es el proceso de extraer información y conocimiento interesante y no trivial de texto no estructurado. Es un campo relativamente reciente con un gran valor comercial que se nutre de las áreas de recuperación de la información (IR), minería de datos, aprendizaje automático, estadística y procesamiento del lenguaje natural. La minería de textos incluye una serie de tecnologías, entre otras: extracción de la información, seguimiento de temas (*topic tracking*), generación automática de resúmenes de textos, categorización, agrupamiento, vinculación entre conceptos, visualización de la información, y respuesta automática de preguntas. El presente trabajo se centra en el reconocimiento de entidades con nombre (NER), uno de los problemas básicos de la minería de textos y complementa un trabajo anterior en la categorización de documentos y en la búsqueda semántica del contenido

[1] Facultad de Ingeniería e Informática e IESIING, Universidad Católica de Salta.
aperez@ucasal.net, acardoso@ucasal.net

de los mismos (Pérez & Cardoso, 2011).

El NER es tarea base de todo sistema de IR y desempeña un papel muy importante en otros problemas relacionados como la búsqueda automática de respuestas y la categorización de textos. Nadeau y Sekine (2007) y Sarawagi (2008) describen aplicaciones de la tarea NER a problemas de atención a clientes, seguimiento de noticias, limpieza y preparación de datos para un almacén de datos, búsqueda en bases de datos documentales, especialmente de biología, etc. Los tipos de entidades más estudiados son los nombres propios: de personas, de lugares y de organizaciones. Mientras que los primeros sistemas de NER se basaban en reglas cuidadosamente desarrolladas, los más recientes utilizan aprendizaje automático supervisado para generar reglas que etiqueten automáticamente las entidades. En general la literatura indica (Nadeau&Sekine, 2007) que la elección adecuada de las características del texto relevantes en un problema dado puede tener mayor importancia incluso que la del algoritmo de aprendizaje, que el problema de NER depende fuertemente del dominio de aplicación y que técnicas con buenos resultados en un dominio o incluso idioma no necesariamente se trasladan bien a otros.

El presente artículo presenta la arquitectura del sistema y las técnicas utilizadas para la tarea NER, para describir los resultados experimentales de evaluación de dichas técnica, y terminar con líneas de trabajo futuras y conclusiones.

2. La arquitectura

Conceptualmente las aplicaciones de gestión de información no estructurada suelen organizarse en dos fases. En la fase de análisis se recogen y analizan colecciones de documentos y los resultados se almacenan en algún lenguaje o depósito intermedio. La fase de entrega hace accesible al usuario el resultado del análisis, y posiblemente el documento original completo mediante una interfaz apropiada. La Fig. 1 muestra la aplicación de este esquema a nuestro dominio (Pérez & Cardoso, 2011), en el que partimos de más de 8000 resoluciones rectorales en archivos de texto de distinto tipo: Word, PDF, texto plano. Previo al análisis, se procede a la extracción del texto de cada archivo utilizando herramientas de software libre (*poi.apache.org* y *tm-extractors*). El texto se normaliza remplazando los acentos por sus equivalentes en ASCII para facilitar los procesos de búsqueda y equiparación de cadenas. También se divide en partes la resolución extrayendo el encabezado (texto que contiene el número y la fecha de la resolución) y el cuerpo con la mayor parte de la información, y descartando en lo posible el texto «de forma».

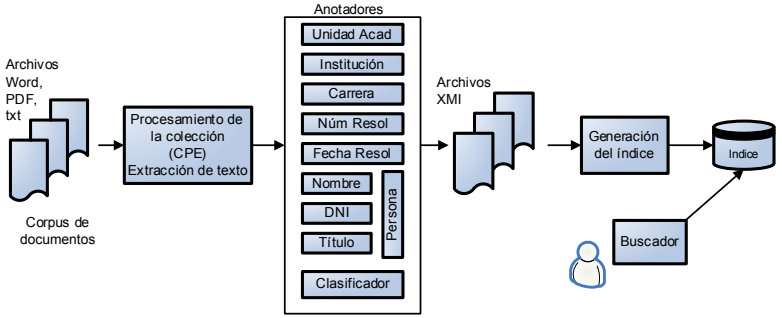


Fig.1. Arquitectura del sistema

La fase de análisis incluye tokenización y detección de entidades en documentos individuales tales como personas, fechas, organizaciones, unidades académicas y datos sobre la resolución (fecha y número). Es aquí donde se ubica la tarea NER objeto de este trabajo. Además con la ayuda de un clasificador aprendido automáticamente del corpus de resoluciones se anota cada documento con una categoría. Existen 21 categorías que fueron obtenidas del personal especializado en la elaboración de resoluciones. Algunos ejemplos son: designación de planta docente, convenio de pasantías, convenio de colaboración, o llamado a concurso docente.

El resultado de la fase de análisis es un conjunto de archivos en formato XMI (XML MetadataInterchange) (OMG, 2007). Estos archivos contienen, además de las partes relevantes del texto original, metadatos en forma de anotaciones correspondientes a las entidades y a la categoría de documentos. Estos archivos serán procesados para construir el índice de un motor de búsqueda que contiene los tokens (en nuestro caso, las palabras que aparecen en el texto) y las entidades y categorías extraídas automáticamente.

En la fase de entrega existe una interfaz para hacer consultas de búsqueda en el índice. El usuario puede buscar documentos que contengan combinaciones booleanas de tokens, entidades y categorías mediante un motor de búsqueda semántica.

Las dos fases están desarrolladas sobre UIMA(Unstructured Information Management Architecture), una arquitectura basada en componentes para construir sistemas de procesamiento de información no estructurada (Ferrucci&Rally, 2004). En UIMA, el componente que contiene la lógica del análisis se llama anotador. Cada anotador realiza una tarea específica de extracción de información de un documento y genera como resultado anotaciones, que son añadidas a una estructura de datos denominada CAS (*common analysis structure*). A su vez, esas anotaciones pueden ser utilizadas por otros anotadores. Los anotadores pueden ser agrupados en anotadores agregados. La mayoría de nuestros anotadores realizan reconocimiento de entidades con nombre (NER), a saber: personas, unidades académicas, carreras, instituciones; además hay otros que extraen fechas, número y año de las resoluciones. Para detectar entidades correspondientes a personas se agregan otras (nombres propios, DNIs y títulos) obtenidas por los anotadores correspondientes. Un último anotador asigna la categoría de documento en base al modelo aprendido automáticamente (Pérez & Cardoso, 2011). Inicialmente los anotadores para NER fueron codificados a mano. El objetivo del presente trabajo ha sido remplazarlos con modelos aprendidos automáticamente.

3. Extracción de entidades con nombre

En esta sección se describe la tarea NER: basados en reglas y con las grandes familias de técnicas utilizadas para resolverla.

3.1. Introducción

Para entender un texto y extraer información del mismo, las personas, lugares y cosas que aparecen tienen un papel muy importante. Por ello tratar de identificar las partes de habla correspondientes, es decir, los nombres, y hacerlos disponibles para otras aplicaciones es una tarea útil, y es la que elegimos para el presente trabajo. Esta tarea puede ser realizada por un buen analizador sintáctico (*parser*) pero a menudo es suficiente enfocarse en un subconjunto de los nombres, es decir, aquellos que se refieren a entidades específicas:

los nombres propios, que también se conocen como entidades con nombre. Este es el término que utilizaremos de aquí en adelante. Por otro lado el análisis sintáctico completo de una frase es un proceso computacionalmente costoso, mientras que hay formas más eficientes de extraer simplemente las entidades con nombre. Las mismas técnicas pueden también utilizarse para extraer otras entidades, como fechas y cantidades (20 de Septiembre; 34 pesos) pero no son el objeto de este trabajo.

Desde el punto de vista de las técnicas disponibles es interesante mencionar que aunque en un texto o corpus hay muchas entidades con nombre, en general las instancias específicas de las mismas son raras. Por ejemplo, en nuestro corpus de resoluciones rectorales casi todas las resoluciones contienen varios nombres (incluyendo apellido) de personas, pero la probabilidad de que una misma persona aparezca en varias resoluciones no es grande, excepto en el caso del nombre de las autoridades firmantes de la resolución, que suelen ser las mismas y por lo tanto su consideración no es útil. Por tanto las técnicas apropiadas deben ser capaces de extraer, en este caso, nombres de persona en general, y no de aprender a detectar, o memorizar, nombres específicos.

El contexto del texto puede ayudar a detectar estas entidades con nombre, así como otras pistas como el uso de mayúsculas o minúsculas, los títulos (ej Dr. o Lic.), o las entidades cercanas (por ejemplo, en un convenio suelen aparecer dos entidades cercanas en el texto correspondientes a las organizaciones que lo subscriben).

La Figura 2 presenta un ejemplo del resultado de la extracción de entidades con nombre. La Figura 3 muestra el mismo texto en el formato XMI (ver Sección 2) como resultado del proceso de análisis del documento al que el texto pertenece por nuestro prototipo en UIMA (Pérez & Cardoso, 2011).

La fuente de datos no estructurados:

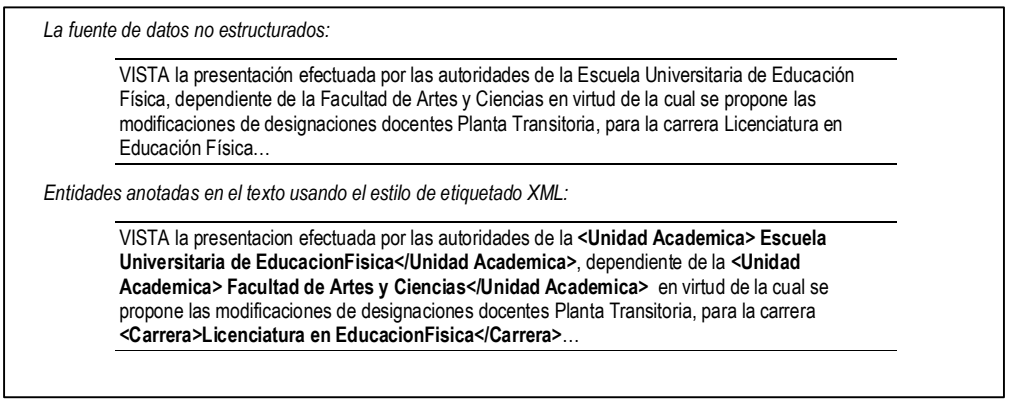


Figura 2. Ejemplo de texto con las entidades con nombre reconocidas y anotadas. Las entidades aparecen en negrita encerradas en etiquetas que indican el tipo de entidad asignado.

```
<?xmlversion="1.0" encoding="UTF-8" ?>
<xmi:XMI …xmi:version="2.0">
…
 <cas:Sofaxmi:id="1"sofaNum="2"sofaID="encabezado"mimeType="text"
       sofaString="ResoluciOn N 470/08 En el Campo Castanares, sito en la Ciudad de
Salta, Capital de la Provincia del mismo nombre, Republica Argentina, sede de la
Universidad Catolica de Salta, a los veintisiete dias el mes de mayo del ano dos mil
ocho:" />
 <cas:Sofaxmi:id="13"sofaNum="1"sofaID="_InitialView"mimeType="text"
       sofaString="la presentacion efectuada por las autoridades de la Escuela
Universitaria de EducacionFisica, dependiente de la Facultad de Artes y Ciencias en
virtud de la cual se propone las modificaciones de designaciones docentes Planta
Transitoria, para la carrera Licenciatura en EducacionFisica…" />
…
 <examples:SourceDocumentInformationxmi:id="25" sofa="13" begin="0" end="0"
    uri="file:/D:/UIMA/docs/RES.%20%20Nº0470-08.txt"offsetInSource="0"
       documentSize="2280"lastSegment="false" />
 <mio:UAxmi:id="48" sofa="13" begin="177" end="208" confidence="30.0"
    componentId="de.julielab.jules.lingpipegazetteer.GazetteerAnnotator"
    specificType="UnidadAcademica" />
 <mio:Carreraxmi:id="72" sofa="13" begin="398" end="430" confidence="0.0"
    componentId="de.julielab.jules.lingpipegazetteer.GazetteerAnnotator"
    specificType="Carrera" />
 <mio:NumeroResolxmi:id="33"sofa="1"begin="0"end="19"
       nroResol="RESOLUCION N 470/08" numero="470"anio="2008" />
 <mio:FechaResolxmi:id="40"sofa="1"begin="196"end="248"anio="2008"
mes="MAYO"
       dia="27"fechaResolCompleta="VEINTISIETE DE MAYO DE DOS MIL OCHO" />
<mio:Clasexmi:id="28"sofa="1"begin="0"end="0" valor="DesigDocPlanta" />
…
```

Figura 3. Ejemplo de texto anotado (Pérez & Cardoso, 2011).

3.2 Métodos de extracción de entidades con nombre

Los métodos para extraer entidades con nombre se suelen ubicar en un espacio a lo largo de dos dimensiones (Sarawagi, 2007), que se describen en las dos subsecciones siguientes.

3.2.1 Dimensión 1: codificados a mano o basados en aprendizaje automático

Un sistema codificado a mano precisa de expertos humanos que definan pautas para la extracción automática de las entidades, y que suelen estar expresadas en forma de reglas, de expresiones regulares, o hasta de fragmentos de código que realizan la extracción. En general para ello el experto debe serlo tanto en el dominio del corpus como en la codificación o programación en la plataforma o lenguaje de implementación.

En el otro extremo están los sistemas basados en aprendizaje automático que requieren de ejemplos de entidades, etiquetadas

como tal en el texto, a partir de los cuales las técnicas de aprendizaje automático construyen modelos que realizan la extracción. El etiquetado de entidades es generalmente realizado a mano por un experto en el corpus, que a menudo debe tomar decisiones adecuadas al objetivo del sistema. Por ejemplo, en el texto «Fundación Roberto Romero», se debe decidir si etiquetar o no «Roberto Romero» como persona (además del etiquetar el texto completo como organización), lo cual depende en general de la tarea a la que vaya destinado el sistema. El experto frecuentemente comete errores. Estas etiquetas podrían también ser generadas automáticamente por un sistema como el definido en el párrafo anterior.

3.2.2 Dimensión 2: Basados en reglas o estadísticos.

Los métodos basados en reglas realizan la extracción usando condiciones o predicados que deben equipararse precisamente al texto. Los métodos estadísticos toman decisiones basadas en un conjunto de parámetros (condiciones) ponderados que forman una función o modelo. Las reglas son más fáciles de entender y construir por una persona, pero los modelos estadísticos son más robustos cuando hay ruido y más adecuados para aprender automáticamente.

3.2.3 Evolución de las técnicas de extracción de entidades con nombre

Los primeros sistemas eran basados en reglas codificadas a mano. Este proceso era tedioso y a la vez complejo, y normalmente no fácil de extender cuando el dominio de aplicación cambiaba. Por ello se comenzaron a aplicar las técnicas de aprendizaje automático de reglas a partir de ejemplos. Al ir creciendo el contexto de las aplicaciones, para incluir textos con ruido las reglas fueron perdiendo su poder y comenzaron otras técnicas para construir representaciones más complejas. Entre ellas están los modelos generativos, basados en modelos ocultos de Markov (HMMs), y los modelos condicionales como los campos aleatorios condicionales (conditional random fields o CRFs). En la actualidad se utiliza una diversidad de técnicas, seleccionando las más apropiadas en cada aplicación. Sarawagi (2007) y Nadeau y Sekine (2007) presentan sendos panoramas de estas técnicas.

3.2.4 Desafíos

Las aplicaciones reales de extracción de entidades con nombre enfrentan una serie de desafíos (Sarawagi, 2007). El principal es la precisión (*accuracy*) en la extracción, debida principalmente a:

- Diversidad de características del texto que se deben combinar para aumentar la precisión en la tarea de reconocimiento, que es inherentemente compleja. Cada una de esas características (tales como uso de mayúsculas, parte de la oración en que se encuentra una palabra, palabras que la rodean, similitud con palabras de un diccionario,…) por sí misma tiene muy poco poder de cara al reconocimiento. De ahí la importancia de su combinación. La manera óptima de combinarlas depende del dominio, del tipo de entidad a reconocer, y es un problema de investigación abierto.

- Dificultad para descubrir qué entidades el modelo o técnica no ha sido capaz de detectar. La precisión de detección de entidades se calcula en base a dos componentes, normalmente expresados por sus nombres en inglés:

· Precisión: mide el porcentaje de instancias recuperadas que son correctas

· Recall: porcentaje de instancias correctas en el texto que son efectivamente detectadas

Al construir y evaluar un sistema es fácil detectar manualmente los casos en que el sistema marcó incorrectamente en el texto algo

que en realidad no es una entidad. Así el modelo puede ajustarse hasta hacer prácticamente desaparecer esos errores, aumentando su precisión. Sin embargo es más difícil detectar qué entidades el sistema no ha sido capaz de encontrar en una gran masa de texto no estructurado que no haya sido previamente etiquetado a mano. Por ello conseguir que recall sea alta es más problemático.

- Complejidad de las estructuras que se han de extraer: a medida que van surgiendo tareas de aplicación para la extracción de entidades con nombre, va apareciendo la necesidad de tipos de entidades más complejos, o compuestos, o que no aparecen consecutivos en el texto sino dispersos a lo largo del mismo, con referencias simplificadas o acortadas a la entidad. El problema es detectar o hasta definir cuáles son los límites (comienzo o fin) de las entidades en el texto.

Un problema relacionado al de la precisión es el manejo de errores. Dado que la precisión en general no es perfecta, los errores son inevitables. Para ello en algunos sistemas se asocia con cada entidad extraída un grado de confianza que se relaciona con la probabilidad de que la entidad sea correcta.

Otros desafíos son el tiempo de ejecución de la aplicación; y la variación a lo largo del tiempo del contenido y el estilo de los textos que se han de procesar, que puede causar que los modelos aprendidos en el pasado dejen de ser los más adecuados para el texto actual y deban ser actualizados durante el ciclo de vida de la aplicación.

4. Técnicas basadas en reglas

Las reglas pueden referirse a una serie de características disponibles para cada palabra o token, típicamente (Sarawagi, 2007):

- la cadena de caracteres que representa el token, es decir, la palabra misma
- características relacionadas con la ortografía del token: mayúscula inicial o en toda la palabra, puntuación, uso de espaciado, etc.
- la parte de habla que es el token
- diccionarios en que el token aparece
- anotaciones asignadas al token por fases anteriores del proceso

En nuestro sistema original los anotadores para NER fueron programados utilizando tres técnicas básicas (Pérez & Cardoso, 2011):

- Equiparación con expresiones regulares que capturan el patrón que siguen las entidades (ejemplos son la detección de DNIs, fecha y número de las resoluciones).

- Equiparación con diccionarios (ejemplos son las carreras, unidades académicas, instituciones, títulos y nombres propios). El diccionario de nombres propios consta de más de 1300 nombres y fue extraído automáticamente del sistema de gestión de alumnos. El enfoque basado en componentes de UIMA nos ha permitido adaptar el Gazetteer Annotator de Julie Lab (Tomanek & Wermter, 2008) basado en la implementación que hace Lingpipe del algoritmo Aho-Corasick (Alias-i, 2009).

- Equiparación con plantillas: para detectar entidades correspondientes a personas se utiliza una plantilla que describe a la persona mediante los siguientes atributos: nombre1, nombre2, apellido(s), DNI, título. Sólo nombre1 y apellido(s) son obligatorios. Estos elementos son entidades detectadas por anotadores, mientras que el reconocimiento del grupo se implementó mediante un autómata finito (en Java), también en el cuerpo de un anotador.

5. Técnicas basadas en aprendizaje automático

Podría hablarse de dos grandes familias de técnicas para aprender automáticamente modelos para este problema. En primer lugar estarían los algoritmos tradicionales de aprendizaje automático como SVMs, regresión logística, Adaboost (Tjong&Meulder, 2003). Es-

tos han sido superados por algoritmos específicos para el aprendizaje de secuencias, y es en estos en que nos hemos concentrado. En general el éxito de los sistemas depende de la elección de características (propiedades de los tokens usadas para construir el modelo) para el problema dado. Los algoritmos del primer grupo requieren una cantidad considerable de características bien elegidas. Los modelos que aprenden de secuencias en general utilizan menos características, por ejemplo la ubicación de los límites de la entidad (principio, fin, token intermedio) para cada tipo de entidad, como en el caso de los modelos ocultos de Markov (HMMs). Los campos aleatorios condicionales (CRFs), también para el aprendizaje de secuencias, surgieron posteriormente y permiten aprovechar un conjunto mucho más rico de características. En general la elección de características tiene gran importancia para el éxito de un sistema, tanta o más que la elección de técnica(Tjong&Meulder, 2003; Carpenter, 2006).

5.1. Introducción

Los métodos estadísticos convierten la tarea de extracción en un problema con dos partes (Sarawagi, 2007):

- Descomponer adecuadamente el texto no estructurado.
- Etiquetar cada pedazo o secuencia de pedazos como una entidad con nombre de un tipo determinado.

La forma más frecuente de descomponer el texto es en forma de tokens. Existen diversas técnicas de tokenización, pero la mayoría utilizan delimitadores (espacios, comas, puntos) para dividir el texto. El modelo asigna a cada token una etiqueta de entidad (o no), y la entidad se conforma como una secuencia de tokens con la misma etiqueta. Otra forma de descomposición es en grupos de palabras, llamados segmentos chunks. En la fase de etiquetado se asigna la etiqueta al chunk, que comprende toda la entidad con nombre.

Si se utiliza la descomposición en tokens, el texto no estructurado se ve como una secuencia de tokens y el problema es asignar una etiqueta a cada token. Sea la secuencia de token s $x=x_1, x_2,...x_n$. Cada x_i ha de ser clasificado con una etiqueta de un conjunto Y, produciendo la secuencia de etiquetas $y=y_1, y_2,...y_n$. El conjunto está formado por los tipos de entidades y una etiqueta especial para «otro» indicando que el token no pertenece a una entidad. La Figura 4 muestra un ejemplo, en que a los tokens de una frase se les han asignado etiquetas indicando que corresponden a una unidad académica, o a una persona. La etiqueta O indica que el token no pertenece a una entidad.

Nótese que en algunas etiquetas pueden ser compuestas. Si en el ejemplo de la Figura 4 mostramos las etiquetas generadas al nivel más bajo, el resultado sería el mostrado en la Figura 5 donde T indica un título, N un nombre de persona y A un apellido. La etiqueta persona está compuesta por cero o más títulos, uno o más nombres, un apellido y un DNI opcional.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x	DESIGNAR	al	DR.	Juan	García	como	integrante	del	IESIING	en	el	ambito	de	la	Facultad	de	Ingeniería
y	*O*	*O*	*P*	*P*	*P*	*O*	*O*	*O*	*UA*	*O*	*O*	*O*	*O*	*O*	*UA*	*UA*	*UA*

Figura 4. Ejemplo de asignación de etiquetas a una secuencia de tokens.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x	DESIGNAR	al	DR.	Juan	García	como	integrante	del	IESIING	en	el	ambito	de	la	Facultad	de	Ingeniería
y	*O*	*O*	*T*	*N*	*A*	*O*	*O*	*O*	*UA*	*O*	*O*	*O*	*O*	*O*	*UA*	*UA*	*UA*

Figura 5. Ejemplo de etiquetas componentes de las etiquetas de la Figura 4

5.1.1 Codificación

Existen dos estilos de representación de las entidades con nombre, es decir, de elección de las etiquetas del conjunto Y:

- Dentro/fuera: la etiqueta señala si el token pertenece a una entidad (está dentro de la entidad, I, donde I representa el tipo de entidad) o no (está fuera, O). En general se denomina codificación IO. La Figura 6 muestra el ejemplo anterior usando este sistema de codificación.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x	DESIGNAR	al	DR.	Juan	García	como	integrante	del	IESIING	en	el	ambito	de	la	Facultad	de	Ingeniería
y	*O*	*O*	*P*	*P*	*P*	*O*	*O*	*O*	*UA*	*O*	*O*	*O*	*O*	*O*	*UA*	*UA*	*UA*

Figura 6. Ejemplo de etiquetado con estilo dentro/fuera

- Principio/fin: la etiqueta señala si el token es el primero de una entidad (B), o está en el resto de la entidad (I), o no pertenece a una entidad (O). Esta codificación suele llamarse BIO y la Figura 7 muestra el ejemplo anterior con este estilo.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x	DESIGNAR	al	DR.	Juan	García	como	integrante	del	IESIING	en	el	ambito	de	la	Facultad	de	Ingeniería
y	*O*	*O*	*B-P*	*I-P*	*I-P*	*O*	*O*	*O*	*B-UA*	*O*	*O*	*O*	*O*	*O*	*B-UA*	*I-UA*	*I-UA*

Figura 7. Ejemplo de etiquetado con estilo principio/fin.

Ambos estilos son usados en la literatura, con variaciones de los mismos, y junto a otras representaciones (Carpenter, 2009). El estilo BIO es preferido en muchos sistemas y corpora porque permite distinguir entre dos entidades del mismo tipo seguidas inmediatamente. En nuestro sistema hemos experimentado con ambos tipos (Sección 6).

5.1.2. Espacio de características

En general, la asignación de etiqueta a un token a la hora de detectar si pertenece o no a una entidad se realiza en función de una serie de características (features) del mismo que capturan diversas propiedades del token y de su contexto (los tokens que lo rodean). La Tabla 1 muestra las características más utilizadas en este tipo de problemas (Nadeau & Sekine, 2007; Sarawagi, 2007). La Sección 5.4 mostrará las seleccionadas en este proyecto.

Característica	Ejemplos
La misma palabra minúsculas	La palabra como aparece (Facultad es parte de una UA)La palabra toda en
Mayúsculas/minúsculas	Comienza con mayúscula (Juan Garcia)
	Toda en mayúscula (IESIING)
	Contiene letras en mayúscula (Dr.)
Puntuación	Termina con un punto (Dr.)
	Contiene un punto (I.B.M.)
	Contiene un apóstrofe, guión, etc (O'Farrell)
Dígitos	Patrón de dígitos (DNI es 99.999.999)
	Números romanos (Juan Pablo II)
Patrones	Expresar el token según un patrón, capturando la «forma» o «silueta» de la palabra (Garcia -> Aa; I.B.M.->A-A-A-) (Minkov et al, 2005)
Partes de la palabra	Subcadenas, sufijos, prefijos de la palabra de hasta una longitud dada
Posición	Posición de la palabra en la frase
Categoría gramatical	Parte del habla (POS) de la palabra: verbo, adjetivo, etc
En diccionario	Diccionario para cada entidad (ej diccionario de nombres comunes, lista de unidades académicas de la universidad)Diccionario de palabras que suelen formar parte de una entidad (títulos de personas, palabras que suelen aparecer en el nombre de una organización, como Universidad, Fundación...)
Palabras que rodean a la palabra	La palabra siguiente, la palabra anteriorEl conjunto de n palabras a la izquierda y a la derecha de la actual La secuencia de n palabras que la rodean (secuencia implica ordenamiento)Usar esas palabras solo si tienen una longitud mínima.La etiqueta POS de la palabra anterior o siguiente
Pares de palabras	Característica formada por una palabra y la anterior o la siguiente

5.2 Modelos Ocultos de Markov (HMMs)

Un HMM (Sutton &Mc. Callum, 2006) modela una secuencia de observaciones $X=\{x_t\}_{t=1}$, mediante la suposición que existe una secuencia subyacente de estados $Y=\{Y_t\}_{t=1}$, pertenecientes a un conjunto finito de estados S. En el problema de NER cada observación x_t es la identidad de la palabra en la posición t y cada estado y_t es la etiqueta de tipo de entidad (persona, organización,... u otro). Para que el modelado de la distribución conjunta p(y, x) sea tratable un HMM supone dos cosas (Figura 8):

(a) cada estado depende solamente en su predecesor inmediato, es decir cada estado y_t es independiente de sus antepasados $y_1, y_2, ... y_{t-2}$ dado su estado previo y_{t-1}.

(b) cada variable observada x_t depende solo del estado actual y_t.

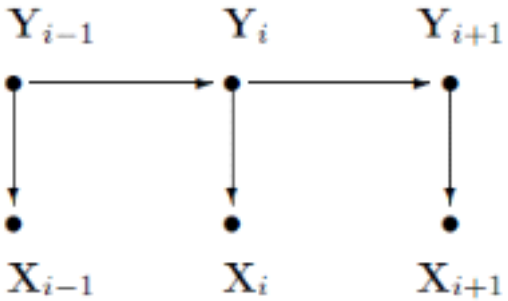


Figura 8. Modelo gráfico de un HMM.

Con estas suposiciones un HMM puede especificarse usando tres distribuciones de probabilidad: (a) la distribución $p(y_1)$ sobre estados iniciales; (b) la probabilidad de transición $p(y_t|y_{t-1})$; y (c) la distribución de las observaciones $p(x_t|y_t)$ o emisiones. Es decir, la probabilidad conjunta de una secuencia de estados $\mathbf{y}$ y una secuencia de observaciones $\mathbf{x}$ puede expresarse como el producto:

$$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^{T} p(y_t|y_{t-1}) p(x_t|y_t) \quad (1)$$

donde $p(y_1|y_0)$ es la distribución .

Aplicar un HMM a un problema, como NER, supone dos tareas: entrenamiento o aprendizaje, y reconocimiento o decodificación (inferencia):

- Decodificación: dada una secuencia de palabras y un HMM entrenado en un corpus, encontrar la secuencia de etiquetas de mayor probabilidad (encontrar que maximice la expresión anterior). El método tradicional es el algoritmo de Viterbi de programación dinámica.

-Entrenamiento: construir el modelo más probable a partir de una secuencia de tokens, es decir entrenarlo tal que se maximice la probabilidad de las observaciones del conjunto de entrenamiento. Para entrenar el modelo hay que determinar las distribuciones (b) y (c). No existe un método analítico para elegir tal que maximice p (**x,y**)pero se puede maximizar localmente mediante un algoritmo iterativo de escalada, como forward-backward o Baum-Welch, un caso especial de EM (Expectation Maximization). El entrenamiento basado en EM tiene los siguientes pasos generales:

1. Inicializar el modelo λ
2. Calcular el nuevo modelo l usando l_0 y la secuencia de observaciones
3. Ajustar el modelo $\lambda_0 \leftarrow \lambda$

Repetir los pasos 2 y 3 hasta que

$$\log p(\mathbf{x}, \mathbf{y}|\lambda) - \log p(\mathbf{x}, \mathbf{y}|\lambda_0) < d$$

5.2.1. Implementación de HMMs usando LingPipe

Para los experimentos con HMMs hemos utilizado el software libre para NER que forma parte del proyecto LingPipe (Alias-i, 2009). Normalmente en un HMM las emisiones (etiquetas) se estiman como distribuciones multinomiales y se utiliza una técnica de suavizado para el caso de tokens que no aparecieron anteriormente (por ejemplo, que no aparecieron en el conjunto de entrenamiento) a los que se asignaría probabilidad 0. Sin embargo, LingPipe estima las probabilidades de emisión usando modelos de lenguaje basados

en n-gramas de caracteres, uno para cada etiqueta. Tradicionalmente, cuando aparece una palabra no vista antes, un modelo HMM produce un valor de probabilidad por defecto, con lo que aumenta el número de errores de asignación. Al usar n-gramas a nivel de caracteres, los modelos pueden usar sub-palabras que son más generales y por tanto más robustas en esta tarea. LingPipe también interpola estimaciones usando el suavizado de Witten-Bell (Manning&Schutze, 1999) En este trabajo usamos los valores por defecto para n (máximo orden de los n-gramas) y para el parámetro de interpolación en los modelos de lenguaje: el valor es 8.0 en ambos casos. En la fase de decodificación LingPipe utiliza por defecto el algoritmo de Viterbi, y además dispone de otros decodificadores más sofisticados, que no hemos utilizado.

Para utilizar LingPipe con UIMA hemos convertido las anotaciones del formato usado por UIMA al formato estándar IO (cada línea es un token con su etiqueta), aunque el HMM subyacente utiliza el esquema más detallado BMEWO+ (Carpenter, 2009). En la conversión se ha obtenido un solo archivo para una colección de archivos XMI y en el caso de anotaciones anidadas se conserva solo la más externa, adecuado ya que las anotaciones que nos interesan PERS, ORG, CARR y UA son precisamente las que contendrían otras (como Nombre, Apellido, DNI, etc.). No se permiten anotaciones que se superpongan; esto originaría que un token pertenezca a dos anotaciones distintas, lo cual no puede representarse en el esquema IO. Solo se considera el cuerpo de la resolución, y no el encabezado, ya que en éste no aparecen las entidades que nos interesan.

Para la inferencia se ha creado un anotador que utiliza el modelo HMM generado por LingPipe para producir las anotaciones en el formato de UIMA, es decir, el paso recíproco al anterior, adaptando un anotador existente en el repositorio *uima.lti.cs.cmu.edu*.

LingPipe proporciona tres variantes para la utilización de HMMs (Alias-i, 2009), que hemos utilizado en los experimentos:

- *CharLmHmmChunker*, que funciona según lo descrito en esta sección.

- *CharLmRescoringChunker*: más preciso, pero también más lento. Utiliza el anterior para generar hipótesis que después revalúa (*rescoring*) usando modelos de lenguaje con caracteres a mayores distancias.

- *TokenShapeChunker*: con un modelo generativo que predice conjuntamente el próximo token y la etiqueta basándose en los dos tokens anteriores y la etiqueta anterior. Las palabras desconocidas se remplazan con características relacionadas con la forma de la palabra. Es muy rápido, pero en general no tan preciso como los anteriores.

5.3. Modelos generativos y modelos discriminantes

En este momento es interesante entender la distinción entre modelos generativos y modelos discriminantes. Los primeros están basados en modelar la distribución conjunta , mientras que los segundos se basan en modelar la distribución condicional . Los modelos ocultos de Markov son generativos y los CRFs son discriminantes. A continuación se analizarán brevemente las diferencias entre ambos y las ventajas de los segundos para el problema de NER (Sutton & McCallum, 2006). Estas ventajas son las que dieron pie a los modelos descritos en las siguientes secciones, en particular los CRFs.

Como hemos dicho, los modelos generativos asignan una probabilidad conjunta a las secuencias emparejadas de observaciones **x** y etiquetas **y**. Los parámetros normalmente se obtienen por entrenamiento tratando de maximizar la probabilidad conjunta de los

ejemplos de entrenamiento. Para definir una probabilidad conjunta sobre secuencias de observaciones y etiquetas, un modelo generativo se encuentra con la dificultad de modelar $p(\mathbf{x})$ cuando el problema contiene un rico conjunto de características independientes entre sí: el modelo debe enumerar todas las secuencias posibles de observaciones, para lo cual suele precisar una representación en que las observaciones son objetos atómicos (palabras en el caso de la tarea de NER). En particular, no es práctico representar características que interactúen o dependencias entre observaciones distantes entre sí en la secuencia, ya que el problema de inferencia en esos modelos se convierte en intratable (Lafferty et al, 2001). Por ejemplo, en el problema de NER, un HMM, que es generativo, utiliza una sola característica de la entrada, la palabra misma. Pero muchas palabras, por ejemplo los nombres propios, no aparecen en el conjunto de entrenamiento, por lo que la decisión basada en la palabra no sirve mucho, y hay que utilizar otras características como el uso de mayúsculas, las palabras cercanas, pertenencia en diccionarios, etc., en fin, las características mencionadas en la Sección 5.1.2. Para poder incluir en un modelo generativo estas características tenemos dos opciones: hacer el modelo complejo para incluir las dependencias entre las entradas, o suponer que las entradas son independientes.

Los modelos discriminantes, o condicionales, se adaptan mejor a la representación de conjuntos ricos y no necesariamente independientes de características de la entrada. Modelan directamente la distribución condicional, es decir, las probabilidades de las secuencias de etiquetas posibles dada una secuencia de observaciones. Por tanto no necesitan esforzarse en conocer la forma de $p(\mathbf{x})$, lo que por otra parte no es necesario para la tarea de clasificación, ya que la secuencia en ese momento es una dada. Además esta probabilidad condicional de la secuencia de etiquetas puede depender en características arbitrarias, no independientes entre sí, de la secuencia de observaciones, sin obligar al modelo a describir la distribución de esas características y sus dependencias. Las características en el caso de NER, como hemos visto, pueden ser las observaciones mismas (palabras) o propiedades de ellas. La probabilidad de transición entre dos etiquetas puede depender no solo en la observación sino también en observaciones pasadas y futuras, si están disponibles. Por el contrario los modelos generativos deben suponer independencia estricta entre las observaciones y sus características, para ser tratables.

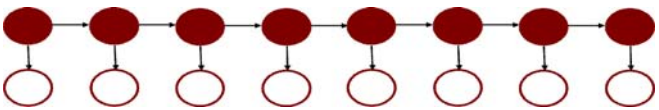
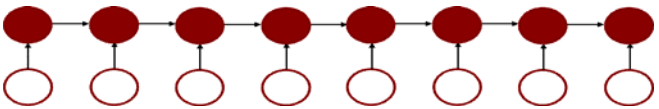
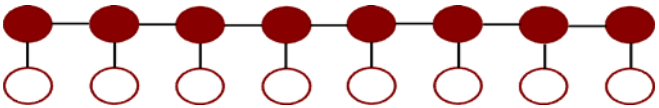
Esto puede explicar por qué se ha observado que los CRFs tienden a ser más robustos que los modelos generativos cuando se violan las suposiciones sobre independencia: los CRFs hacen suposiciones de independencia entre las $\mathbf{y}$ pero no entre las $\mathbf{x}$.

Los primeros modelos discriminantes para modelar secuencias, y en particular para nuestro problema, propuestos en la literatura fueron los modelos de Markov de máxima entropía (MEMMs) (Ratnaparkhi, 1999). En un MEMM cada estado fuente tiene un modelo exponencial que toma como entrada las observaciones y sus características y produce una distribución sobre los estados siguientes posibles. (Es el modelo de la probabilidad condicional del estado siguiente dado el estado actual). Estos modelos exponenciales se entrenan con un método iterativo usando técnicas de entropía máxima. Los MEMMS y otros modelos no generativos basados en clasificadores del próximo estado, tiene una debilidad, el problema del bias de las etiquetas (Lafferty et al, 2001). Esta debilidad sugirió la idea de los CRFs. La diferencia principal entre ambos es que un MEMM usa un modelo exponencial para cada estado de la probalilidad condicional del próximo estado dado el estado actual, mien-

tras que un CRF tiene un único modelo exponen-cial de la probabilidad conjunta de la secuencia completa de etiquetas dada la secuencia de observaciones. Esto permite que los pesos de las distintas características en distintos estados puedan ser balanceados entre sí.

La tabla siguiente muestra las estructuras gráficas de los HMMs, MEMMs y CRFs y resume sus características distintivas.

Tabla 2. Estructuras gráficas y características comparadas de los HMMs, MEMMs y CRFs (Finkel, 2007; Lafferty et al, 2001).

Modelos ocultos de Markov (HMMs) 	Generativos: encontrar los parámetros que maximicen $p(\mathbf{y},\mathbf{x})$ Asume que las características son independientes Al etiquetar x_i se tienen en cuenta observaciones futuras
Modelos de Markov de máxima entropía (MEMMs) 	Discriminantes: encontrar los parámetros que maximicen $p(\mathbf{y},\mathbf{x})$ No asumen que las características son independientes No se tienen en cuenta observaciones futuras.
Campos aleatorios condicionales (CRFs) 	Discriminantes: encontrar los parámetros que maximicen $p(\mathbf{y},\mathbf{x})$ No asumen que las características son independientes Al etiquetar y_i se tienen en cuenta observaciones futuras *Reúne las mejores características de los dos modelos anteriores*

5.4 Campos Aleatorios Condicionales

Modelar la distribución conjunta $p(\mathbf{y}|\mathbf{x})$ se complica cuando se usa un conjunto rico de características de los datos porque requiere modelar la distribución que puede incluir dependencias complejas y así llevar a modelos intratables; por otro lado, ignorarlas puede degradar la capacidad de predicción de los modelos. Los campos aleatorios condicionales (CRFs) surgieron como una solución a este problema, modelando directamente la distribución condicional $p(\mathbf{y}|\mathbf{x})$ que es suficiente para la tarea de clasificación. Las dependencias entre las variables de entrada no necesitan ser representadas explícitamente y así se puede utilizar un conjunto amplio de características de la entrada. A continuación describimos los CRFs y las técnicas utilizadas para construir modelos en este formalismo.

Un CRF (Sutton &McCallum, 2006; Lafferty et al, 2001) modela una única distribución conjunta de probabilidades $p(\mathbf{y}|\mathbf{x})$ sobre las predicciones de etiquetas $\mathbf{y}=y_1y_2...y_n$ de los tokens de la frase o texto $\mathbf{x}$. Para las tareas de NER una cadena es suficiente para capturar las dependencias entre las etiquetas. Esto quiere decir que la etiqueta y_i del i-

ésimotoken solo está influenciada por las etiquetas de los tokens adyacentes. O sea, una vez que se fija la etiqueta y_{i-1} la etiqueta y_{i-2} no influye sobre la etiqueta y_i.

Sea $\Lambda=\{\lambda_k\}\varepsilon\Re^k$ un vector de pará-metros y un $\{f_k(y,y',\mathbf{x}_t)\}_k^k$ un conjunto de funciones qe representan las características de las observaciones. Entonces podemos definir un CRF como una distribución de la forma $p(\mathbf{y}|\mathbf{x})$

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(x)} e^{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)}$$

donde Z es una función de normalización específica de la instancia

$$Z(\mathbf{x}) = \sum_{y} e^{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)}$$

que utiliza la suma del vector de características sobre todas las posiciones (tokens) de la secuencia. es una suma sobre todas las secuencias de estados posibles, que en general es exponencialmente grande, pero puede calcularse eficientemente usando el algoritmo forward-backward.

Para representar que cada característica puede depender de observaciones (tokens) anteriores o posteriores el argumento de correspondiente a la observación es un vector , entendiéndose que contiene todos los componentes de la secuencia global necesarios para calcular las características en el instante . Por ejemplo, si el CRF usa la siguiente palabra como característica, entonces elvector incluye la identidad de la palabra . Los algoritmos de entrenamiento e inferencia utilizados están descritos en (Sutton &McCallum, 2006).

La implementación que hemos utilizado para NER con CRFs es la propuesta por Finkel*et al* (2005). Se han realizado adaptaciones similares a las descriptas en el caso de LingPipe para utilizar esta herramienta con UIMA: el anotador para generar anotaciones UIMA a partir del CRF se ha adaptado el disponible en *www.florianlaws.de* modificándolo para la versión actual de UIMA y del software CRF. Mientras no se indique lo contrario, se han aprendido modelos CRFs usando las características descritas en la Tabla 3, que en general son los valores por defecto proporcionados.

Tabla 3. Características para los modelos CRF

	Característica(s) añadidas
Use Word = true	Palabra actual
Use Prev true	Palabra anterior
Use Next = true	Palabra siguiente
Use Word Pairs = true	Pares (palabra actual, palabra anterior) y (palabra actual, palabra siguiente)
Word Shape=chris2useLC	Forma de la palabra actual y de las palabras que la rodean (codifica atributos longitud, uso de mayúsculas al principio o en toda la palabra, etc)
Use N Grams = true Max N Gram Leng = 6 No Mid N Grams=true	n-gramas: prefijos, sufijos de longmáx 6 de la palabra
Use Disjunctive Disjunction Width	Presencia de la palabra en la ventana hacia la izquierda de tamaño *disjunctionWidth*. Id hacia la derecha
Use Class Feature= true	En cada caso incluye la clase

6. **Resultados experimentales**

Del corpus de más de 8000 resoluciones, para los experimentos descriptos aquí hemos utilizado como conjunto de entrenamiento las correspondientes al año 2007 (1626 documentos, con un total de 356.718 tokens) y como conjunto de prueba las del año 2008 (764 documentos con 165.128 tokens). Los documentos fueron preprocesados usando un tokenizador estándar,que separa los tokens usando los espacios en blanco, remplazando los caracteres acentuados por sus equivalentes en ASCII, y eliminando los caracteres no ASCII, y posteriormente convertidos del formato XMI (anotaciones de UIMA) al formato IO precisado por los algoritmos utilizados. Hicimos también experimentos con el formato BIO con resultados similares. No se utilizó lematización, ya que en nuestros experimentos iniciales con NER basado en reglas, ésta empeoraba la tarea de reconocimiento.

Tabla 4.Resultados (F1, *precision* y *recall*) de los modelos entrenados con los datos de 2007 evaluados con los datos de 2008

	Todas			PERS			UA			CARR			ORG		
Modelo	F1	prec	rec	F1	prec	rec	F1	prec	rec	F1	prec	rec	F1	prec	rec
Lingpipe HMM	0,83	0,83	0,84	0,80	0,77	0,83	0,91	0,92	0,90	0,88	0,86	0,89	0,43	0,43	0,44
Lingpipe RESCORING	0,82	0,82	0,82	0,79	0,76	0,82	0,90	0,92	0,88	0,87	0,85	0,89	0,40	0,39	0,40
Lingpipe TOKENSHAPE	0,85	0,84	0,86	0,87	0,83	0,91	0,91	0,92	0,89	0,87	0,85	0,89	0,49	0,48	0,51
Stanford-NER CRF	0,86	0,89	0,84	0,83	0,84	0,81	0,91	0,94	0,89	0,89	0,90	0,88	0,62	0,72	0,55

Los dos conjuntos fueron etiquetados a mano para poder evaluar las técnicas descriptas. A modo orientativo, el conjunto 2007, presenta 1875 entidades de tipo PERS (Personas), 4375 de tipo UA (Unidad Académica), 2912 CARR (Carrera) y 627 ORG (Institución externa a la universidad).Hay que tener en cuenta que el etiquetado manual está sujeto a errores. Algunos errores que hemos observado repetidas veces son:

- inconsistencia en los criterios: por ejemplo, el etiquetador puede decidir en unos casos que el título (Dr., Lic.) y/o el DNI de una persona son parte de la entidad persona y en otros casos que no lo son.

- el etiquetador al usar la herramienta de marcado no delimita correctamente la entidad (por ejemplo, omite incluir el primer o el último carácter).

- el etiquetador omite por descuido etiquetar entidades, dado que es una tarea larga y tediosa.

Dado que la comparación del etiquetado de un modelo con el manual se hace automáticamente, puede ser que el modelo marque una entidad correctamente y el humano no, lo cual penaliza los resultados del modelo.

La Tabla 4 muestra los resultados de la evaluación de los modelos aprendidos con los datos de 2007 sobre las resoluciones del 2008 (ambos etiquetados manualmente). Con carácter de comparación hemos incluido en la primera fila los resultados con los anotadores programados descriptos en la Sección 4. Comparan-

do los resultados se observa que los modelos CRF son los mejores para esta tarea, seguidos de la técnica basada en Token Shape de LingPipe, que es la más «económica» en tiempo y memoria, tanto en las fases de entrenamiento como de inferencia. La ventaja de CRF es especialmente importante en el caso de las entidades de tipo ORG, lo que se analizará en la Sección6.2. Observamos también que los anotadores programados tienen resultados bastante buenos en este problema (excepto en el caso de las entidades de tipo ORG).No obstante el esfuerzo de escribir, probar y depurar el código de estos anotadores fue considerable, en comparación con el de aprender automáticamente a partir de los datos, aunque en este segundo caso precisamos de la existencia de datos ya anotados para el entrenamiento.

La Tabla 5 muestra el tiempo de entrenamiento necesario de cada uno de los modelos anteriores con la clase *Calendar* de Java) y el tamaño de los mismos a fines comparativos.

Nótese que los modelos CRF, los más efectivos en general, precisan mucho más tiempo de entrenamiento. Aunque parte de la diferencia podría deberse a la implementación, la causa principal es el uso de un conjunto más rico de características, que hace que los algoritmos de entrenamiento de los modelos CRF sean computacionalmente intensivos, especialmente en tiempo, comparados con los basados en HMMs. Esto ocurre también en el caso de la inferencia (etiquetado). Aunque hemos utilizado como base las características por defecto para construir CRFs (Sección 5.4), una limitación de los modelos de CRF es el esfuerzo necesario para ajustar el conjunto de características a cada problema para obtener el mejor resultado posible (Carpenter, 2006; Finkel et al, 2004). Por el contrario los modelos HMM solo precisan dos elementos para predecir la etiqueta de la palabra actual: la palabra en sí y la etiqueta de la palabra anterior (o un conjunto de caracteres).

Tabla 5. Tiempo necesario para construir los modelos de la Tabla 4 y tamaño de los mismos

Modelos	Tiempo	Tamaño del modelo
Lingpipe HMM	7,0 s	2,355 MB
LingpipeRESCORING	27,3 s	19,882 MB
Lingpipe TOKENSHAPE	6,7 s	2,130 MB
Stanford-NER CRF	1659,8 s	2,341 MB

6.1. Influencia del formato de los datos

En algunos lugares de la literatura se habla de las ventajas de uno u otro esquema de codificación de los datos (Sección 5.1.1) (Carpenter, 2009). En nuestro caso hemos experimentado solo con los formatos IO y BIO. Nótese que los modelos HMM de LingPipe requieren (internamente) el formato BMEWO+ aunque externamente acepte tanto BIO como IO, y los traduzca a BMEWO+. El software para entrenar CRFs que hemos usado solo acepta datos en formato IO. De hecho uno de sus autores

indica (citado en Carpenter, 2009) indican que este formato parece ser suficiente. El único caso en que hemos encontrado una diferencia significativa ha sido en el de la técnica basada en TokenShape de LingPipe para la detección de entidades de tipo ORG. Usando el formato IO el valor de F1 fue 0,49 (Tabla 3) mientras que usando BIO este valor pasó a ser 0,56.

6.2 Modelos específicos para las entidades ORG

De los resultados anteriores queda claro que las técnicas utilizadas no tuvieron resultados adecuados para la extracción de entidades de tipo ORG, correspondiente a instituciones que aparecen en las resoluciones, son externas a la universidad y no suelen repetirse a lo largo del tiempo. Por tanto no se cuenta con un diccionario de las mismas. Se realizaron algunos experimentos con el objeto de mejorar la extracción de entidades ORG. En primer lugar se evaluó si la construcción de un modelo específico para este tipo de entidad podría mejorar el rendimiento, ya que los modelos anteriores fueron construidos para todas las entidades (un solo modelo para todas). La Tabla 6 muestra estos resultados, evaluando el rendimiento de los modelos solamente en la detección de entidades ORG. En general el modelo abocado solo a la detección de entidades ORG tiene un rendimiento algo mejor que el modelo general.

Tabla 6. Resultados (F1) comparativos de los modelos entrenados para todas las entidades y los entrenados solo para entidades ORG

	Modelocreado con:	
	Todaslasentidades	Solo ORG
Anotadoresprogramados	0,16	0,16
Lingpipe HMM	0,43	0,44
LingpipeRESCORING	0,40	0,42
Lingpipe TOKENSHAPE	0,49	0,56
Stanford-NER CRF	0,62	0,64

Puede destacarse también que para los modelos construidos para entidades ORG con técnicas de HMM *precisión* y *recall* tienen valores similares, mientras que para el mejor modelo (CRF) *precision* es de 0,78 y *recall* es 0,55. Concluimos que para nuestro problema es conveniente entrenar un solo modelo CRF para las entidades PERS, UA y CARR y un modelo separado para las entidades ORG. Dado que los modelos CRF tienen una gran ventaja para este problema específico, se continuó experimentando con sus parámetros.

6.3 Experimentos con los parámetros de los modelos CRF

Dado que los CRFs parecen la técnica más adecuada a esta tarea, se experimentócon algunos parámetros, llegando a la conclusión que el único parámetro que afecta los resultados de

manera significativa es el que regula el uso, como una característica, del conjunto (disyunción) de las n palabras a la izquierda y las n palabras a la derecha de la actual (ver Tabla 2). La Tabla 7 muestra los resultados de experimentar con distintos valores de n (*disjunction Width*, cuando *use Disjunctive =true*) para las entidades de tipo ORG. Se han añadido el tiempo empleado en construirlo (tiempo de entrenamiento) y la velocidad con que el modelo construido se aplica al etiquetado de los datos de prueba. Puede observarse que los modelos con valores mayores de n precisan más tiempo de entrenamiento y son más lentos para etiquetar nuevas entidades, aunque la diferencia no es considerable.

Tabla 7. Comparación de modelos CRF para entidades ORG variando el tamaño de la característica *disjunction Width*

Valor de *disjunction Width*	F1	*precision*	*recall*	Tiempo de entrenam (segs)	Veloc de etiquetado (palab/seg)
4 (default)	0,64	0,78	0,55	339	6222
8	0,67	0,80	0,57	324	6049
12	0,70	0,80	0,62	336	5982
16	0,67	0,79	0,58	361	5456

Inspeccionando los resultados de cada modelo, para analizar por qué aumenta su *recall*, encontramos que hay ciertas entidades de tipo ORG que son detectadas solo cuando *disjunction Width* es mayor que 8. Los siguientes son algunos fragmentos de ejemplos de esta situación en que está anotada la entidad ORG detectada correctamente, pero no lo es si el valor de *disjunction Width* es menor: «*firmado entre la UNIVERSIDAD CATÓLICA DE SALTA, y la EMPRESA <ORG>O.S.P.RE.R.A. </ORG>, de fecha*» (la entidad O.S.P.R.E.R.A. está formada por catorce tokens); «*entre la UNIVERSIDAD CATÓLICA DE SALTA y la <ORG>FUNDACIÓN DOCTOR ESTEBAN LAUREANO MARADONA</ORG>, que se incorporan*»; «*suscripto entre la UNIVERSIDAD CATÓLICA DE SALTA y la <ORG> UNIVERSIDAD PABLO DE OLAVIDE </ ORG>, DE SEVILLA, de fecha*»

La Tabla 8 amplía este análisis al conjunto completo de entidades, es decir, construyendo un solo modelo para todos los tipos de entidades, a diferencia del caso anterior. Vemos que el efecto de la variación de *disjunctionWidth* las entidades no es tan relevante en este caso como en el caso de modelos solo para entidades de tipo ORG.

Tabla 8. Comparación de modelos CRF variando el tamaño (*disjunction Width*) del conjunto de palabras que rodean a la actual

	Todas las entidades	PERS	UA	CARR	ORG
4	0,86	0,83	0,91	0,89	0,62
8	0,88	0,87	0,91	0,90	0,64
12	0,88	0,89	0,92	0,91	0,64

7. Trabajo futuro

Las técnicas para NER que hemos utilizado son de aprendizaje supervisado, y requieren una colección grande de ejemplos de entrenamiento etiquetados manualmente. Es por ello que más recientemente se ha prestado atención a los algoritmos semi-supervisados o no supervisados que requieren respectivamente menos ejemplos o ningún ejemplo (Pérez & Cardoso, 2011). Pretendemos aplicar al problema de NER la experiencia anterior con este tipo de algoritmos, prometedores y foco de investigación actual debido a la gran disponibilidad de ejemplos no etiquetados en la web.

Una de las posibilidades para mejorar los resultados de reconocimiento de NER es la combinación de varios clasificadores o modelos. Para ello puede usarse un esquema de votación. Por ejemplo, Kozareva et al (2007) combinan el resultado de modelos HMM, MEMM y basado en memoria mediante una estrategia de votación para el reconocimiento de entidades en castellano, mejorando el rendimiento de los clasificadores individuales. Sin embargo no comparan los resultados con los de un modelo CRF.Florian et al (2003) combinan modelos HMM, MEMM, un clasificador por maximización de riesgo robusto y otro basado en aprendizaje por transformación, evaluando distintas formas de combinarlos en diferentes condiciones. Gaillard et al (2009) describen cómo realizar la integración de varios modelos en la plataforma UIMA. En este caso último caso los modelos aprendidos fueron generados por sistemas comerciales.

El trabajo que hemos desarrollado hasta ahora puede integrarse para tareas más complejas, como la búsqueda de respuestas a preguntas propuestas por un usuario (sistemas QA). En esta tarea deben comprenderse preguntas sencillas en lenguaje natural para detectar qué entidad o concepto se busca, traducir la pregunta a una consulta del buscador semántico, que devuelve un conjunto documentos, y transformar esa respuesta en algo que se pueda presentar al usuario: por ejemplo, colocando las respuestas más relevantes primero, marcando la frase o párrafo del documento donde está ubicada la respuesta, o hasta extrayendo la misma.

8. Conclusiones

Este trabajo ha explorado una variedad de técnicas para la extracción de entidades con nombre. De los experimentos se ha observado que los modelos CRF son los más adecuados para esta tarea aplicada a un corpus de 8000 documentos que contienen resoluciones rectorales. El trabajo forma parte de una línea de investigación sobre la minería de textos, de importante potencial en la actualidad, ya que una gran parte de la información que manejan las organizaciones está disponible en documentos de texto u otra información no estructurada. Aunque la curva de aprendizaje para estas técnicas y herramientas es pronunciada, la experiencia adquirida es una buena base para aplicaciones más sofisticadas e integradas de las técnicas de aprendizaje automático a la minería de textos.

9. Referencias bibliográficas

Alias-i, LingPipe NER tutorial, http://alias-i.com/lingpipe/demos/tutorial/ne/readme.html (acceso Mayo 2012), 2009

Carpenter, B., Why do you Hate CRFs? LingPipe Blog. Disponible en http://lingpipe-blog.com /2006 /11/22

Carpenter, B., Coding Chunkers as Taggers: IO, BIO, BMEWO, and BMEWO+. LingPipe Blog. Disponible en lingpipe-blog.com/2009/10/14

Ferrucci, D., A. Rally, Building an example application with the Unstructured

Information Management Architecture, *IBM Systems Journal* 45:3, 2004

Finkel, J., S. Dingare, H. Nguyen, M. Nissim, C. Manning, G. Sinclair, Exploiting Context for Biomedical Entity Recognition: From Syntax to the Web. *Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, COLING 2004

Finkel, J., T. Grenager, C. Manning, Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling.*Proc of the 43nd Annual Meeting of the Association for Computational Linguistics*, 363-370, 2005

Finkel, J., Named Entity Recognition and the Stanford NER software, 2007. Disponible en nlp.stanford.edu/software/jenny-ner-2007.ppt

Florian, R., A. Ittycheriah, J. Hongyan, T. Zhang, Named Entity Recognition through Classifier Combination, En *Proceedings of CoNLL-2003*, pp 168-171, 2003

Gaillard, B., S. Guillemin-Lanne, G. Jacquet, C. Martineau, A. Migeotte, Combining NER Systems via a UIMA-based platform, en *1st French-speaking meeting around the framework Apache UIMA*, Nantes, Julio 2009

Kozareva, Z., O. Ferrández, A. Montoyo, R. Muñoz, A. Suárez, J. Gómez, Combining data-driven systems for improving Named Entity Recognition, *Data & Knowledge Engineering* 61, pp 449–466, 2007

Lafferty, J., A. McCallum, F. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data. En *Proceedings ICML-2001*

Minkov, E., R. C. Wang, W. Cohen, Extracting Personal Names from Email: Applying Named Entity Recognition to Informal Text, Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Vancouver, Canada, Association for Computational Linguistics, 443-450, 2005

Nadeau, D., S. Sekine, A survey of named entity recognition and classification, en *Journal of Linguistica e Investigationes,* 30:1, 2007

OMG, XML Metadata Interchange (XMI), v 2.1.1, 2007

Pérez, A.,A. C. Cardoso, Categorización automática de documentos. *Simposio Argentino de Inteligencia Artificial, 40 JAIIO,* 2011

Ratnaparkhi, A., Learning to parse natural language with maximum entropy models, Machine Learning, 34, 1999

Sarawagi, S., Information Extraction, en*Foundations and Trends in Databases*, 1:3, 261–377, 2007

Sutton, C., A. McCallum, An Introduction to Conditional Random Fields for Relational Learning. En *Introduction to Statistical Relational Learning.*Eds L. Getoor& B. Taskar. MIT Press, 93-127, 2006

Tjong Kim Sang, E., F. De Meulder, Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition, *Proceedings of CoNLL-2003*

Tomanek, K., J. Wermter, JULIE Lab Lingpipe Gazetteer Annotator v. 2.1. Jena Univ Language & Information Engineering (JULIE) Lab. disponible en www.julielab.de, 2008