

## Técnicas de extracción de entidades con nombre

M. Alicia Pérez Abelleira, Alejandra Carolina Cardoso<sup>1</sup>

<sup>1</sup> Facultad de Ingeniería e Informática e IESIING. Universidad Católica de Salta  
Campo Castañares s/n, A4400 Salta, Argentina  
{aperez, acardoso}@ucasal.net

**Abstract.** La minería de textos tiene un importante potencial, ya que gran parte de la información de las organizaciones está disponible en documentos de texto u otra información no estructurada. Una de las tareas integrales de la minería de textos es la extracción de entidades con nombre (NER). El presente trabajo describe los principales enfoques en uso para esta tarea y los aplica a un problema concreto, la extracción de información de un corpus de 8000 documentos correspondientes a resoluciones rectorales. Los experimentos muestran que los campos aleatorios condicionales (CRFs) son la técnica más adecuada para este problema. El trabajo describe también la arquitectura para la gestión de información no estructurada en la que se enmarca esta tarea y de la que forma parte.

**Keywords:** NER, HMM, CRF, minería de texto, UIMA.

### 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 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 res-puesta 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 en el contenido de los mismos [1].

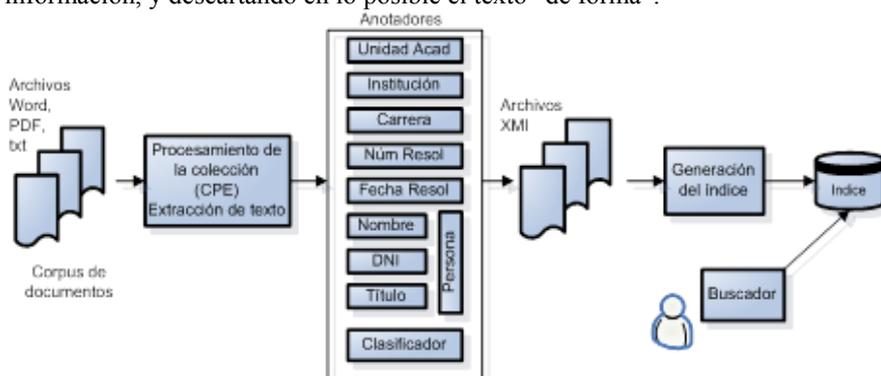
El NER desempeña un papel muy importante en diversos problemas relacionados con la minería de texto, tales como la búsqueda automática de respuestas y la categorización de textos. Nadeau y Sekine [2] y Sarawagi [3] describen aplicaciones de la tarea NER a problemas de atención a clientes, seguimiento 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 [2] 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.

## 2.2 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 [1], 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 eliminando acentos 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. Estos archivos contienen, además de las partes relevantes del texto original, metadatos en forma de anotaciones correspondientes a las entidades existentes 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 búsquedas en el índice. El usuario puede buscar documentos que contengan combinaciones booleanas de entidades, categorías y tokens 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 [4]. 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 utilizando un modelo aprendido automáticamente (ver [1]). Inicialmente los anotadores para NER fueron codificados a mano. El objetivo del presente trabajo ha sido reemplazarlos con modelos aprendidos automáticamente.

### 3 Extracción de Entidades con Nombre

Existen dos enfoques generales para NER: basado en reglas y basado en aprendizaje automático. En nuestro sistema original los anotadores para NER fueron programados utilizando el primer enfoque, con tres técnicas básicas [1]:

- 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, extraídos automáticamente del sistema de gestión de alumnos.
- 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 a su vez entidades detectadas por anotadores.

En cuanto al enfoque basado en aprendizaje automático, podría hablarse de dos grandes familias de técnicas. En primer lugar estarían los algoritmos tradicionales de aprendizaje automático como SVMs, regresión logística, Adaboost [5]. Estos 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 [5] [6].

### 3.1 Modelos Ocultos de Markov (HMMs)

Un HMM [7] modela una secuencia de observaciones  $X = \{x_t\}_{t=1}^T$  mediante la suposición que existe una secuencia subyacente de estados  $Y = \{y_t\}_{t=1}^T$ , 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 (Fig. 2):

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

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

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. La probabilidad conjunta de una secuencia de estados  $y$  y una secuencia de observaciones  $x$  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  $p(y_1)$ .

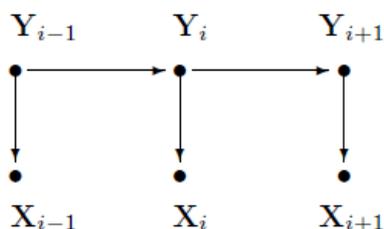


Fig. 2. Modelo gráfico de un HMM

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  $\mathbf{y}$  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  $\mathbf{y}$  tal que maximice  $p(\mathbf{x}, \mathbf{y})$  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  $\lambda_0$
  2. Calcular el nuevo modelo  $\lambda$  usando  $\lambda_0$  y la secuencia de observaciones
  3. Ajustar el modelo  $\lambda_0 \leftarrow \lambda$
- Repetir los pasos 2 y 3 hasta que  $\log p(\mathbf{x}, \mathbf{y}|\lambda) - \log p(\mathbf{x}, \mathbf{y}|\lambda_0) < d$

### 3.2 Implementación de HMMs

Para los experimentos con HMMs hemos utilizado el software libre para NER que forma parte del proyecto LingPipe [8]. 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 subpalabras que son más generales y por tanto más robustas en esta tarea. LingPipe también interpola estimaciones usando el suavizado de Witten-Bell [9]. 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; 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+ [6]. 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 del repositorio *uima.lti.cs.cmu.edu*.

LingPipe proporciona tres variantes para la utilización de HMMs [8], 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 reevalúa (*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 menos preciso.

### 3.3 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  $p(\mathbf{x})$  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. No es necesario representar explícitamente las dependencias entre las variables de entrada y por tanto puede utilizarse 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 [7] [10] modela una distribución conjunta de probabilidades  $p(\mathbf{y}|\mathbf{x})$  sobre las predicciones de etiquetas  $\mathbf{y} = y_1 y_2 \dots y_n$  de los tokens de la frase o texto  $\mathbf{x}$ . Para las tareas de NER una cadena de etiquetas es suficiente para capturar las dependencias entre las mismas; es decir, la etiqueta  $y_i$  del  $i$ -ésimo token 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  $\Lambda = \{\lambda_k\} \in \mathfrak{R}^K$  un vector de parámetros y  $\{f_k(y, y', \mathbf{x}_t)\}_{k=1}^K$  un conjunto de funciones que representan  $K$  características de las observaciones. Entonces podemos definir un CRF como una distribución  $p(\mathbf{y}|\mathbf{x})$  de la forma

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{\sum_{k=1}^K \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)} \quad (2)$$

donde  $Z(\mathbf{x})$  es una función de normalización específica de la instancia  $\mathbf{x}$

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} e^{\sum_{k=1}^K \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)} \quad (3)$$

que utiliza la suma del vector de características sobre todas las posiciones (tokens) de la secuencia.  $Z(\mathbf{x})$  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  $f_k$  correspondiente a la observación es un vector  $\mathbf{x}_t$ , entendiéndose que contiene todos los componentes de la secuencia global  $\mathbf{x}$  necesarios para calcular las características en el instante  $t$ . Por ejemplo, si el CRF usa la siguiente palabra  $x_{t+1}$  como característica, entonces el vector  $\mathbf{x}_t$  incluye la identidad de la palabra  $x_{t+1}$ . Los algoritmos de entrenamiento e inferencia utilizados están descritos en [7].

La implementación que hemos utilizado para NER con CRFs es la propuesta por Finkel et al [11]. Se han realizado adaptaciones similares a las descritas 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](http://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 1 (columna 2), que en general se corresponden con los valores por defecto de parámetros de la herramienta (columna 1).

**Tabla 1.** Características para los modelos CRF

Parám y valor	Característica(s) añadidas
useWord = true	Palabra actual
usePrev true	Palabra anterior
useNext = true	Palabra siguiente
useWordPairs = true	Pares (palabra actual, palabra anterior) y (palabra actual, palabra siguiente)
wordShape=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.)
useNGrams = true maxNGramLeng=6 noMidNGrams=true	n-gramas: prefijos, sufijos de long máx 6 de la palabra
useDisjunctive disjunctionWidth	Presencia de la palabra en la ventana hacia la izquierda de tamaño <i>disjunctionWidth</i> . Id hacia la derecha
useClassFeature= true	En cada caso incluye la clase

## 4 Resultados Experimentales

Del corpus disponible 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, reemplazando los caracteres acentuados por sus equivalentes en ASCII y eliminando los caracteres no ASCII; posteriormente fueron 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.

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 es una tarea larga y tediosa y 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 por descuido ignora entidades.

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 2 muestra los resultados de la evaluación sobre las resoluciones del 2008 de los modelos aprendidos con los datos de 2007 (ambos etiquetados manualmente). Comparando 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 Secc 4.1. A modo de comentario, los anotadores programados tienen resultados bastante buenos en este problema (excepto en el caso de las entidades de tipo ORG, ver Tabla 4). 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.

**Tabla 2.** Resultados (F1, *precision* y *recall*) de los modelos entrenados con los datos de 2007 evaluados con los datos de 2008

Modelo	Todas			PERS			UA			CARR			ORG		
	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 TO- KENSHPAE	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

La Tabla 3 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.

**Tabla 3.** Tiempo necesario para construir los modelos de la Tabla 2 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

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 3.3), 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 [6] [11]. 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).

#### 4.1 Modelos Específicos para las Entidades ORG

De lo anterior queda claro que las técnicas utilizadas no tuvieron resultados adecuados para la extracción de entidades de tipo ORG. Éstas corresponden a instituciones que aparecen en las resoluciones, son externas a la universidad y no suelen repetirse a lo largo del tiempo. Se realizaron algunos experimentos con el objeto de mejorar la extracción de dichas entidades. 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 4 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 4.** Resultados (F1) comparativos de los modelos entrenados para todas las entidades y los entrenados solo para entidades ORG

	Modelo creado con:	
	Todas las entidades	Solo ORG
Lingpipe HMM	0,43	0,44
Lingpipe RESCORING	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 *precision* 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.

#### 4.2 Experimentos con los Parámetros de los Modelos CRF

Dado que los CRFs parecen la técnica más adecuada a esta tarea, se experimentó con 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 1). La Tabla 5 muestra los resultados de experimentar con distintos valores de  $n$  (valor del parámetro *disjunctionWidth*, cuando *useDisjunctive=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 5.** Comparación de modelos CRF para entidades ORG variando el tamaño de la característica *disjunctionWidth*

Valor de <i>disjunctionWidth</i>	F1	<i>precision</i>	<i>recall</i>	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 *disjunctionWidth* 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 *disjunctionWidth* es menor: “*firmado entre la UNIVERSIDAD CATÓLICA DE SALTA, y la EMPRESA <ORG>O.S.P.R.E.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*”

**Tabla 6.** Comparación de modelos CRF (valor F1) variando el tamaño (*disjunctionWidth*) del conjunto de palabras que rodean a la actual

Valor de <i>disjunctionWidth</i>	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

La Tabla 6 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 *disjunctionWidth* en otras entidades no es tan relevante en este caso como en el caso de modelos solo para entidades de tipo ORG.

## 5 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 [1]. 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.

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 de 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. Esta es otra línea de trabajo que estamos explorando.

## 6 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.

## Referencias

1. Perez, A., Cardoso, A. C.: Categorización automática de documentos. In: Simposio Argentino de Inteligencia Artificial, 40 JAIIO, Córdoba (2011)
2. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. *Journal of Linguisticae Investigationes* 30(1) (2007)
3. Sarawagi, S.: Information Extraction. *Foundations and Trends in Databases* 1(3), 261–377 (2007)
4. Ferrucci, D., Lally, A.: Building an example application with the Unstructured Information Management Architecture. *IBM Systems Journal* 45(3) (2004)
5. Tjong Kim Sang, E., De Meulder, F.: Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In: *Proceedings of CoNLL-2003* (2003)
6. Carpenter, B.: Coding Chunkers as Taggers: IO, BIO, BMEWO, and BMEWO+. In: *LingPipe Blog*. Available at: [lingpipe-blog.com/2009/10/14](http://lingpipe-blog.com/2009/10/14)
7. Sutton, R., McCallum, A.: An Introduction to Conditional Random Fields for Relational Learning. In Getoor, L., Taskar, B., eds.: *Introduction to Statistical Relational Learning*. MIT Press, Cambridge (2006) 93-127
8. Alias-i: LingPipe NER tutorial. (Accessed 2009) Available at: <http://alias-i.com/lingpipe/demos/tutorial/ne/read-me.html>
9. Manning, C. D., Schütze, H.: *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA (1999)
10. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *Proceedings ICML-2001* (2001)
11. Finkel, J., Dingare, S., Nguyen, H., Nissim, M., Manning, C., Sinclair, G.: Exploiting Context for Biomedical Entity Recognition: From Syntax to the Web. In: *Joint Workshop on Natural Language Processing in Biomedicine and its Applications, COLING 2004* (2004)

**Agradecimientos.** Este trabajo ha sido financiado en parte por el Consejo de Investigaciones de la Universidad Católica de Salta (Resol Rect 333/11).