Categorización Automática de Documentos

M. Alicia Pérez Abelleira, Alejandra Carolina Cardoso[1]

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

Abstract. La clasificación de documentos de texto es una aplicación de la minería de textos que pretende extraer información de texto no estructurado. Su interés se justifica porque se estima que entre el 80% y el 90% de los datos de las organizaciones son no estructurados. Por otro lado, la búsqueda semántica permite al usuario especificar en una consulta no solamente términos que deben aparecer en el documento, sino conceptos y relaciones, que pueden detectarse mediante el análisis de texto. El objetivo de este trabajo es implementar un buscador semántico que aproveche el resultado de algoritmos de aprendizaje automático supervisado y semi-supervisado para la categorización o clasificación de documentos. El dominio de aplicación es un corpus de más de 8000 documentos que contienen nueve años de resoluciones rectorales de la Universidad Católica de Salta en distintos formatos.

Keywords: categorización de documentos, buscador semántico, aprendizaje semisupervisado, minería de texto, UIMA.

1 Introducción

El conocimiento es cada vez más un recurso de importancia estratégica para las organizaciones y su generación, codificación, gestión, divulgación aportan al proceso de innovación. Todos estos aspectos se incluyen en la llamada *gestión del conocimiento*. La cantidad de documentos de diversos tipos disponibles en una organización es enorme y continúa creciendo cada día. Estos documentos, más que las bases de datos, son a menudo un repositorio fundamental del conocimiento de la organización, pero a diferencia de éstas la información no está estructurada. La minería de textos tiene como objetivo extraer información de texto no estructurado, tal como entidades (personas, organizaciones, fechas, cantidades) y las relaciones entre ellas. Por otro lado, la búsqueda semántica permite al usuario especificar en una consulta no solamente términos que deben aparecer en el documento, sino esas entidades y relaciones extraídas mediante el análisis de texto.

La categorización de documentos de texto es una aplicación de la minería de texto que asigna a los documentos una o más categorías, etiquetas o clases, basadas en el contenido. Es un componente importante de muchas tareas de organización y gestión de la información. El enfoque tradicional para la categorización de textos en que los expertos en el dominio de los textos definían manualmente las reglas de clasificación ha sido reemplazado por otro basado en técnicas de aprendizaje automático, o en combinaciones de éste con otras técnicas.

Nuestro trabajo se centra en desarrollar técnicas para la categorización automática de documentos según su contenido que avancen el estado del arte en nuestro medio, aplicando el aprendizaje automático a la minería de texto. El objetivo final es implementar un buscador semántico que aproveche el resultado de algoritmos de aprendizaje para la clasificación de documentos. El dominio de aplicación es un corpus de más de 8000 documentos que contienen 9 años de resoluciones rectorales de la Universidad Católica de Salta en distintos formatos (Word, texto plano, PDF).

Este artículo comienza describiendo la información no estructurada, repositorio fundamental del conocimiento de una organización, y las arquitecturas para la gestión de información no estructurada. La Sección 3 muestra nuestra instanciación del modelo general para el problema de la clasificación y búsqueda de resoluciones rectorales. En la Sección 4 se exploran diferentes algoritmos para la categorización de documentos y se describen los experimentos realizados para determinar su adecuación a nuestro dominio. Concluye el trabajo con el funcionamiento del motor de búsqueda semántica (Sección 5) y algunas conclusiones.

2 Información Estructurada y No Estructurada

La información estructurada se caracteriza por tener un significado que pretende no ser ambiguo y que está representado explícitamente en la estructura o formato de los datos. El ejemplo típico es una base de datos relacional. De la información no estructurada podría decirse que su significado no está implicado en su forma y por tanto precisa interpretación para aproximar o extraer el mismo. Los documentos en lenguaje natural o hasta de voz, audio, imágenes, etc. entran en esta categoría. El interés por extraer significado de la información no estructurada se debe a que se estima que entre el 80% y el 90% de los datos de las organizaciones son no estructurados [1]. Aunque muchas organizaciones han invertido en tecnologías para minería de datos estructurados procedentes de sus bases de datos y sistemas transaccionales, en general no han intentado capitalizar sus datos no estructurados o semi-estructurados.

Una aplicación de gestión de la información no estructurada (UIM por sus siglas en inglés) típicamente es un sistema que analiza grandes volúmenes de información no estructurada con el fin de descubrir, organizar y entregar conocimiento relevante al usuario final. La información no estructurada puede ser mensajes de correo electrónico, páginas web o documentos generados con una variedad de procesadores de texto, como en el caso de las resoluciones rectorales de nuestra universidad. Estas aplicaciones utilizan para el análisis una variedad de tecnologías en las áreas del procesamiento del lenguaje natural, recuperación de la información, aprendizaje automático, ontologías y hasta razonamiento automático.

El resultado del análisis generalmente es información estructurada que se hace accesible al usuario mediante aplicaciones adecuadas. Un ejemplo puede ser la generación de un índice de búsqueda y la utilización de un buscador que facilita el acceso a documentos de texto por tema, ordenados según su relevancia a los términos o conceptos de la consulta del usuario.

Existen diversas arquitecturas para el desarrollo de aplicaciones UIM. Para nuestro trabajo hemos utilizado UIMA [2], una arquitectura software basada en componentes que surgió como proyecto de investigación de IBM y fue puesta a disposición de la comunidad como software libre.

3 Arquitectura del Sistema

Conceptualmente suele verse a las aplicaciones de UIM con dos fases: una de análisis y otra de entrega de la información al usuario. En la fase de análisis se recogen y analizan colecciones de documentos. Los resultados del análisis 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, 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 las herramientas de software libre POI (poi.apache.org) y *tm-extractors* (www.textmining.org). 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".

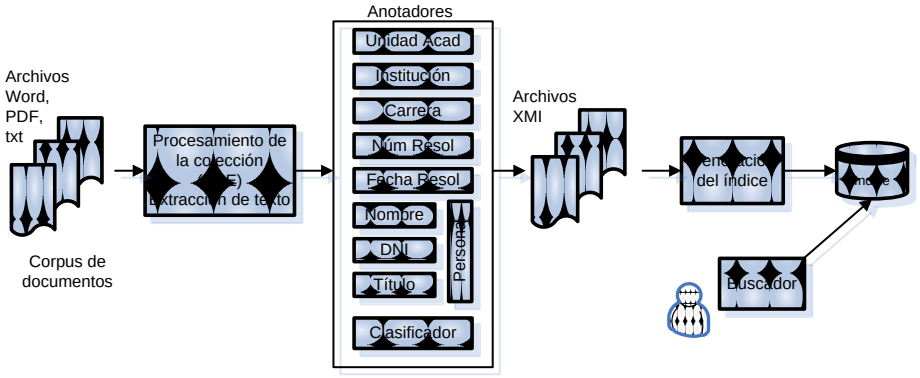

Fig. 1. Arquitectura del sistema.

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). Además con la ayuda de un clasificador aprendido automáticamente del corpus de resoluciones, como se explica en la Sección 4, 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, llamado a concurso docente, o designación de tribunal de concurso.

El resultado de la fase de análisis es un conjunto de archivos en formato XMI (Sección 3.3). 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 de forma que el usuario pueda buscar documentos que contengan combinaciones booleanas de tokens, entidades y categorías mediante un motor de búsqueda semántica.

3.1 Análisis a Nivel de Documento

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 los anotadores de nuestro sistema realizan reconocimiento de entidades con nombre (NER), a saber: personas, unidades académicas, carreras, instituciones, fechas, número y año de las resoluciones. Además, para detectar entidades correspondientes a personas se agregan otras (nombres propios, DNIs y títulos) obtenidas por los anotadores correspondientes.

Las técnicas utilizadas para el reconocimiento de entidades son [3,4]:

- 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 y *gazetteers* (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 [5] basado en la implementación que hace *Lingpipe* del algoritmo Aho-Corasick [3].
- 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. Todos los atributos son a su vez entidades detectadas por anotadores.

Además de los anotadores mencionados, se utiliza el anotador de UIMA que detecta tokens usando una sencilla segmentación basada en los espacios en blanco, y crea anotaciones con tokens y sentencias.

Aparte de todos estos anotadores, existe otro que asigna la categoría de documento en base al modelo aprendido automáticamente, como se describe en la Sección 4

3.2 Análisis a Nivel de Colección

Además se realiza análisis al nivel de la colección de documentos. Un caso particular es un bucle de realimentación, que produce recursos estructurados a partir del análisis de una colección de documentos y luego usa esos recursos para permitir el

subsiguiente análisis de documentos [2]. Tal estructura de realimentación se ha utilizado para implementar el aprendizaje automático de categorías (Fig. 2).

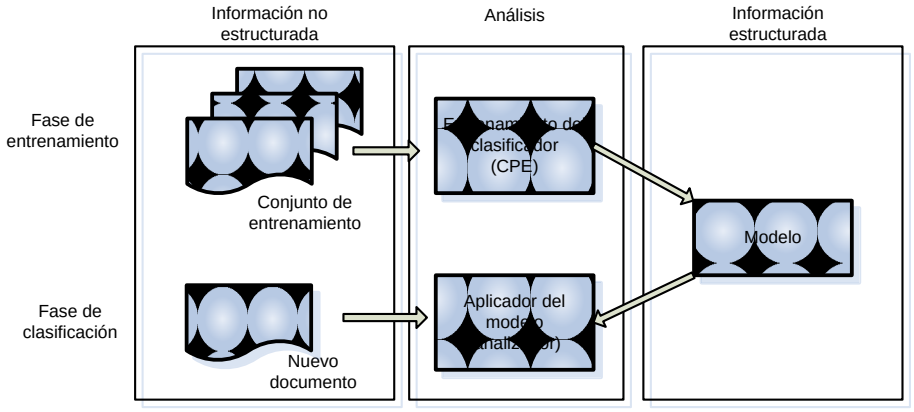

Fig. 2. Implementación del aprendizaje automático en la arquitectura de UIMA.

En la fase de entrenamiento, un motor de procesamiento de colecciones (CPE) analiza una colección de documentos, denominada conjunto de entrenamiento. Este CPE invoca a un analizador que utilizando algoritmos de aprendizaje automático produce un modelo que, basado en los atributos del documento, es capaz de asignarle una categoría. El modelo forma parte de un nuevo anotador, el clasificador de la Fig. 1, que se encarga de aplicar el modelo. En la fase de clasificación, el modelo es consultado por el clasificador al analizar un nuevo documento y asignarle una categoría, que se convierte en una anotación más sobre el documento. La Sección 4 describe los algoritmos de aprendizaje utilizados para generar el modelo.

3.3 Formato de la Información Estructurada

Un importante principio de arquitectura en UIMA es que todos los analizadores operan sobre una estructura de datos estándar, el CAS, que incluye el texto y las anotaciones. El CAS es el principal repositorio de información estructurada y utiliza como lenguaje el estándar XMI (*XML Metadata Interchange*) [6]. XMI proporciona un formato en XML para el intercambio de modelos UML, lo que además hace posible la interoperabilidad con otras aplicaciones. En XMI el texto se incluye en el SofA (*Subject of Analysis*) del CAS como atributo *sofaString* y las anotaciones en el CAS tienen prefijos particulares en el espacio de nombres de XMI, como por ejemplo *<mio:Carrera>*. El espacio de nombres queda definido al declarar un sistema de tipos como parte del dominio en UIMA. La Fig. 3 muestra un ejemplo del resultado del proceso de análisis. Dos *SofAs* recogen el encabezado, del que sólo se extrae la fecha y el número de resolución, y el cuerpo de la misma. La primera anotación del tipo *SourceDocument Information* guarda información sobre el archivo, para poder recuperarlo posteriormente, por ejemplo, para mostrarlo al usuario como resultado de una búsqueda. A continuación aparecen varias anotaciones, algunas con varios

campos. La anotación *Clase* contiene la categoría *DesigPlanta* asignada a esta resolución por el modelo aprendido.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xmi:XMI … xmi:version="2.0">
…
 <cas:Sofa xmi: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:Sofa xmi:id="13" sofaNum="1" sofaID="_InitialView" mimeType="text" sofaString="la
presentacion efectuada por las autoridades de la Escuela Universitaria de Educacion Fisica,
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 Educacion Fisica…" />
…
 <examples:SourceDocumentInformation xmi:id="25" sofa="13" begin="0" end="0"
uri="file:/D:/UIMA/docs/RES.%20%20Nº0470-08.txt" offsetInSource="0" documentSize="2280" />
 <mio:UA xmi:id="48" sofa="13" begin="177" end="208" confidence="30.0"
componentId="de.julielab.jules.lingpipegazetteer.GazetteerAnnotator"
specificType="UnidadAcademica" />
 <mio:Carrera xmi:id="72" sofa="13" begin="398" end="430" confidence="0.0"
componentId="de.julielab.jules.lingpipegazetteer.GazetteerAnnotator" specificType="Carrera" />
 <mio:NumeroResol xmi:id="33" sofa="1" begin="0" end="19" nroResol="RESOLUCION N
470/08" numero="470" anio="2008" />
 <mio:FechaResol xmi:id="40" sofa="1" begin="196" end="248" anio="2008" mes="MAYO"
dia="27" fechaResolCompleta="VEINTISIETE DE MAYO DE DOS MIL OCHO" />
 <mio:Clase xmi:id="28" sofa="1" begin="0" end="0" valor=" DesigDocPlanta" />
…
</xmi:XMI>
```

Fig. 3. Ejemplo de texto anotado

4 Aprendizaje Automático para la Categorización de Documentos

El enfoque dominante actualmente para el problema de categorización de textos se basa en técnicas de aprendizaje automático supervisado: se construye automáticamente un clasificador mediante aprendizaje inductivo de las características o atributos de las categorías a partir de un conjunto de documentos previamente clasificados que sirven como conjunto de entrenamiento. Una importante limitación del aprendizaje supervisado es que se precisa gran cantidad de ejemplos (documentos) etiquetados para poder alcanzar una precisión apropiada. El etiquetado de hasta miles de documentos ha de ser realizado por una persona, especialista en el área de interés de los documentos, y por tanto es un proceso muy costoso.

En cambio los algoritmos de aprendizaje semi-supervisado pueden aprender de un número limitado de ejemplos de entrenamiento etiquetados utilizando adicionalmente documentos no etiquetados, generalmente fácilmente disponibles. En un trabajo anterior [7] hemos experimentado en una variedad de dominios con tres enfoques representativos de este tipo de algoritmos: la aplicación de *expectation maximization* (EM) al uso de datos etiquetados y no etiquetados; los algoritmos de *co-training*; y el algoritmo *co-EM* que combina características de los dos anteriores. En nuestros experimentos la implementación de *co-training* ha dado los mejores resultados para la categorización de textos, por lo que enfocamos en ella nuestra descripción.

La categorización de texto suele desarrollarse en tres etapas: pre-procesamiento de los datos, construcción del clasificador y categorización de los nuevos documentos, A continuación se describen los dos primeros pasos. El tercero es realizado mediante un anotador (ver Sección 3).

4.1 Pre-Procesamiento de los Datos

En esta fase se transfiere el texto original, resultante de la tokenización, a un formato compacto que será utilizado en las fases siguientes. Incluye:
- Lematización (*stemming*), que reduce una palabra a su raíz. Se ha utilizado el algoritmo propuesto en Snowball (snowball.tartarus.org).
- Eliminación de palabras de función (artículos, preposiciones, conjunciones, etc.) y otras dependientes del dominio. En nuestro caso pueden ser 'Universidad Católica de Salta', 'rector', 'resuelve', etc. Esta eliminación está basada en un diccionario y ocurre después de la lematización.
- Selección de atributos, para reducir la dimensionalidad del espacio de datos eliminando los que parezcan irrelevantes, como por ejemplo palabras que aparezcan en todos los documentos.
- Asignar distintos pesos a los atributos (o el mismo peso). Es común utilizar tf–idf para evaluar la importancia de una palabra en el corpus de documentos. Estamos realizando experimentos para elegir la opción más adecuada en nuestro corpus.

Con los pasos anteriores cada documento de la colección se convierte a una representación compacta adecuada para los algoritmos de aprendizaje, tal como el formato *arff*. El filtro *StringToWordVector* de Weka (www.cs.waikato.ac.nz/ml/weka/) puede ser configurado para aplicar varias de las transformaciones anteriores.

4.2 Construcción del Clasificador

Los documentos, transformados en conjuntos de atributos y valores por la etapa anterior, se utilizan para construir un clasificador que asignará categorías a nuevos documentos. Hemos evaluado una variedad de algoritmos, incluyendo algunos semi-supervisados. A continuación se describen brevemente algunos.

a. SMO: El aprendizaje de máquinas de vectores soporte es un método supervisado que ha demostrado buenas propiedades para la categorización de documentos. Hemos utilizado la implementación en Weka del algoritmo de optimización minimal secuencial para entrenar máquinas de vectores soporte usando un kernel polinomial [8].

b. *Co-training*: Blum y Mitchell [9] introdujeron la idea de *co-training*, al reconocer que los atributos de algunos conjuntos de datos pueden descomponerse naturalmente en dos subconjuntos, y cada uno de ellos sirve efectivamente para clasificar cada documento, es decir, son redundantemente predictivos. Cada uno de esos subconjuntos actúa como una perspectiva diferente de la tarea de clasificación. Nuestra implementación genera los modelos para los dos conjuntos de datos en cada iteración con el algoritmo SMO. *Co-training* depende de que las dos perspectivas (conjuntos de atributos A y B) sean redundantes e independientes - para cada

instancia, los atributos de A son condicionalmente independientes de los de B dada la clase de la instancia. Aunque parezca una suposición poco realista, hay resultados que muestran que funciona bien en ciertos conjuntos de datos que no satisfacen completamente estos requisitos, y esta consideración sigue siendo una pregunta abierta [9, 10]. Nuestros experimentos han estado dirigidos a entender el comportamiento en la categorización de textos en nuestro problema particular.

c. *Expectation maximization* (EM): Está basado en una técnica sencilla pero efectiva para clasificar textos, Naive Bayes, que sólo aprende de datos etiquetados. La idea es utilizar Naive Bayes para aprender clases para un pequeño conjunto etiquetado y después ampliarlo a un conjunto grande de datos no etiquetados utilizando el algoritmo EM de *clustering* iterativo [11, 8]. El procedimiento tiene dos pasos: primero entrenar un clasificador Naive Bayes usando los datos etiquetados. Segundo, aplicarlo a los datos no etiquetados para etiquetarlos con las probabilidades de clase (el paso E o *expectation*). Tercero, entrenar un nuevo clasificador usando ahora las etiquetas de todos los datos (el paso M o maximización). Cuarto, repetir estos pasos hasta llegar a la convergencia (cuando el cambio en las probabilidades asignadas es menor que un cierto umbral). Nuestra implementación tomó como punto de partida la de Mooney (www.cs.utexas.edu/users/ml/risc).

4.3 Configuración de los Experimentos

Para evaluar nuestro enfoque hemos utilizado un corpus de 1000 resoluciones rectorales pertenecientes al año 2007 a las que se han asignado una de 21 categorías manualmente. De este corpus hemos extraído conjuntos de entrenamiento de diversos tamaños para experimentar comparando el comportamiento de los algoritmos supervisados (SMO) y no supervisados (*co-training*, EM) en función del número de ejemplos disponibles. Estos algoritmos fueron seleccionados en base a resultados de experimentos anteriores [7].

El corpus ha sido preprocesado utilizando las utilidades de filtrado de Weka con los pasos descritos en la sección 4.1. En este preprocesamiento también hemos variado los valores de diversos parámetros, especialmente las maneras de seleccionar atributos y de asignar pesos a los atributos.

- Para reducir la dimensionalidad se ha realizado la selección de los 100 atributos más relevantes. Se ha experimentado con dos maneras de seleccionar estos atributos: los 100 más relevantes en la colección completa para todas las clases, y los 100 más relevantes en la colección completa para cada una de las clases. Esta segunda opción selecciona en la práctica entre 700 y 900 atributos.
- Los dos mecanismos más populares para asignar pesos a los atributos son binario y tfidf. En el primer caso, los pesos son 0 o 1 indicando ausencia o presencia de una palabra (el atributo) en el documento. Tfidf asigna mayor peso a los atributos que aparecen más frecuentemente, es decir, los considera representativos del contenido del documento, pero además equilibra esto reduciendo el peso de un atributo si aparece en muchos documentos, es decir, es menos discriminante.

Para evaluar la calidad de la clasificación de cada modelo en cada variación del corpus, algoritmo, y parámetros de preprocesamiento hemos utilizado la medida F1.

4.4 Resultados y Discusión

La Fig. 4 muestra los valores de la medida F1 en función del número de ejemplos de entrenamiento disponibles. SMO precisa ejemplos etiquetados; *co-training* y EM, al ser semi-supervisados, pueden aprovechar ejemplos no etiquetados también, por lo que en el experimento la mitad del conjunto de entrenamiento estaba etiquetada y la otra no. Para cada algoritmo modificamos los mecanismos de reducción de la dimensionalidad y de asignación de pesos a los atributos. En los casos (3) y (4) se seleccionaron los 100 atributos más relevantes para cada una de las 21 clases, resultando en un total de entre 614 y 933 atributos (dependiendo del conjunto de entrenamiento); en (1) y (2) los 100 atributos más relevantes para todas las clases. Solamente en los casos (2) y (4) se aplica la transformación tfidf.

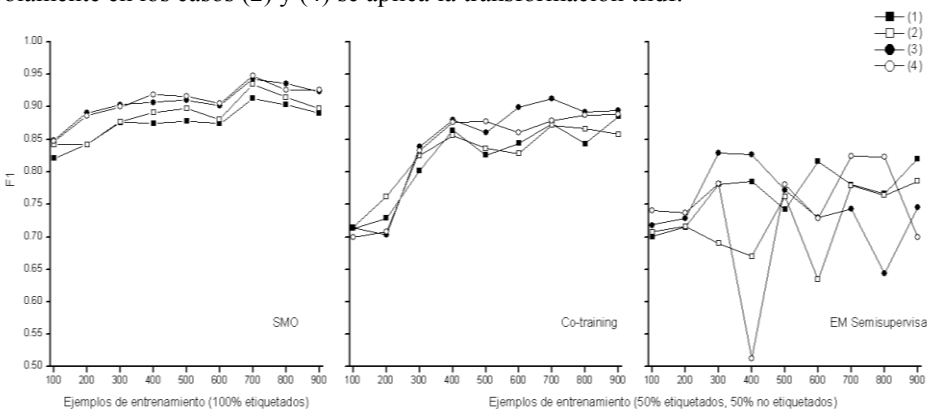

Fig. 4. Variación de F1 con el número de ejemplos de entrenamiento según las formas de reducir de la dimensionalidad y de asignar pesos a los atributos

En general los resultados son mejores (de 1% a 6% mejores en el caso de SMO y *co-training*) cuando se eligen los atributos más relevantes para cada una de las 21 categorías (líneas 3 y 4). Esta selección lleva a un conjunto de atributos mucho mayor, lo cual aumenta el tiempo de aprendizaje del modelo considerablemente en el caso de *co-training*, aunque no así en el caso de SMO. No obstante, dado que el modelo sólo se construye una vez, éste no es un factor tan importante a la hora de elegir el algoritmo. Por otro lado, la utilización de tfidf (gráficos 2 y 4) supone algo de mejora solamente cuando se utiliza SMO.

La Fig. 5(a) compara los valores de F1 para los tres algoritmos en el caso (3), es decir, seleccionando los 100 atributos para cada una de las 21 clases y sin usar tfidf. Puede verse cómo SMO produce mejores resultados, especialmente con menos ejemplos de entrenamiento disponibles. Como se indicó, el conjunto de entrenamiento de los algoritmos semi-supervisados estaba sólo parcialmente etiquetado. Por lo tanto, en cierto modo SMO tenía a su disposición más información (más ejemplos etiquetados). Por ello además comparamos en la Fig. 5(b) los tres algoritmos solamente frente al número de instancias de entrenamiento etiquetadas. (Los algoritmos semi-supervisados reciben además un número igual de instancias no etiquetadas). En este caso no hay tanta diferencia en el rendimiento de SMO y *co-*

training, pero vemos que el uso de los ejemplos no etiquetados no supone una ventaja en este dominio (a diferencia de otros dominios como los explorados en [7]).

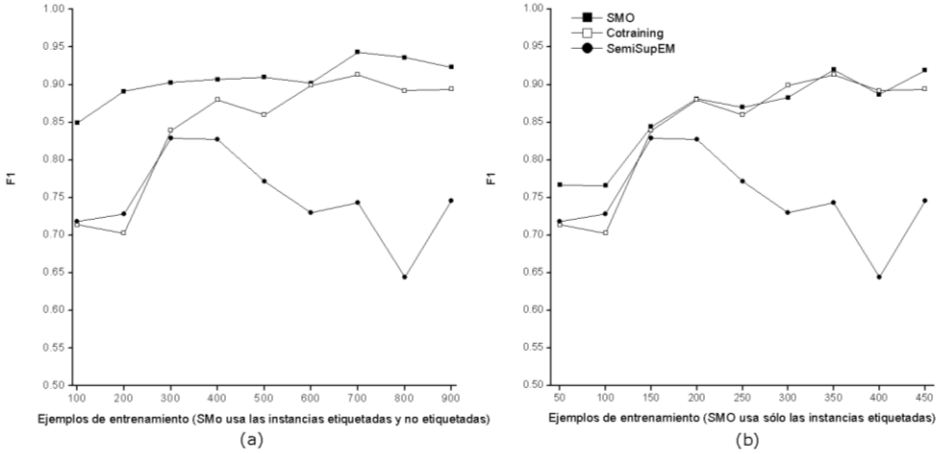

Fig. 5. Comparación de los tres algoritmos

5 Búsqueda Semántica

Nuestro objetivo final es construir un buscador semántico que utilice los meta-datos obtenidos automáticamente por los anotadores y las categorías asignadas por los modelos aprendidos. Para ello hemos experimentado adaptando dos APIs de software libre diferentes para implementar motores de búsqueda: *SemanticSearch* y *Lucene*. Mientras que la segunda es más ampliamente utilizada para construir motores de búsqueda en general, la primera ha sido especialmente construida para su integración con UIMA. Por esto nuestro prototipo inicial está construido sobre *SemanticSearch* y ofrece una interfaz básica para el buscador semántico. Una segunda implementación con Lucene es la base del trabajo de un alumno de grado que está construyendo una interfaz amigable y más completa basada en web para el buscador.

Para construir las consultas hemos adoptado el lenguaje de consultas *XML Fragments* [12] debido a su expresividad y a la disponibilidad de un motor de búsquedas como parte del componente *SemanticSearch* [13], cuya interfaz de búsqueda permite los operadores estándar, tales como búsqueda de texto libre y los operadores AND, OR, NOT y comodines para combinar las consultas.

Una consulta en el lenguaje *XML Fragments* consta de una estructura XML sub-especificada que combina consultas de palabras con consultas de información anotada. Esto permite buscar conceptos más específicos, e incluso relaciones entre objetos (por ejemplo, “la persona y la unidad académica deben aparecer en la misma frase” o “una persona que pertenece a cierta unidad académica” o “una unidad académica vinculada a una institución” porque aparecen cerca en el documento).

Las operaciones que se pueden realizar con *XML Fragments* incluyen [12]:

- conceptualización, que generaliza un literal (string) a un concepto apropiado del sistema de tipos. Por ejemplo, la consulta “institución” devuelve documentos en que

aparezca esa palabra, mientras que <*Institución/*> busca ocurrencias de la anotación institución, aunque la palabra "institución" misma no aparezca en el documento.
- restricción: restringe las ocurrencias de etiquetas XML indicando qué palabras deben aparecer. Por ejemplo <*Institución*> *ingenier* <*/Institución*> devuelve ocurrencias del Consejo Profesional de Ingenieros mientras que <*UnidadAcadémica*> *ingenier* <*/UnidadAcadémica*> devuelve documentos sobre la Facultad de Ingeniería.
- relación: una anotación representa la relación entre los términos de la consulta. Ej. <*FechaResol*>*2007 Septiembre* <*FechaResol*> encuentra las resoluciones de sept. del 2007 (y no simplemente resoluciones de 2007 en que aparezca "septiembre").

Obviamente el usuario final debe poder formular la parte estructurada de las consultas mediante una interfaz apropiada.

6 Conclusiones

La minería de textos, y la categorización de documentos en particular, son un campo de investigación y aplicación prometedor dado que más y más las organizaciones están interesadas en aprovechar el gran cuerpo de conocimiento no estructurado de que disponen. Las técnicas de recuperación de la información y de aprendizaje automático, combinadas con la potencia de la arquitectura UIMA y su reuso de componentes, facilitan la tarea de extracción de conocimiento. En particular, hemos investigado e implementado anotadores que extraen del texto entidades, relaciones entre ellas, y la información necesaria para asignar automáticamente categorías a documentos de texto en base a un corpus relativamente pequeño de ejemplos.

El enfoque utilizado para la extracción de entidades, en particular instituciones y carreras, funciona relativamente bien, pero su calidad depende de la calidad del diccionario y de la precisión en la equiparación de las ocurrencias del texto con las del diccionario. A menudo una misma entidad, por ejemplo el nombre de una carrera, aparece de muy diversas formas en el texto. El diccionario debe incluirlas todas o variaciones suficientemente cercanas. Una clara posibilidad de trabajo futuro es flexibilizar este enfoque, posiblemente aplicando algoritmos de aprendizaje automático para detectar estas entidades.

La clasificación automática de documentos utilizando el modelo aprendido es de bastante calidad comparada con las etiquetas asignadas manualmente. En base a los experimentos realizados, el algoritmo semi-supervisado *cotraining* tiene un rendimiento bastante bueno en cuanto a los resultados de la clasificación y requiere menos ejemplos etiquetados para aprender al poder aprovechar los ejemplos no etiquetados. No obstante los modelos aprendidos por el algoritmo SMO (máquinas de vectores soporte) son muy buenos para este problema sin necesidad de tener muchos ejemplos etiquetados de entrenamiento. Por tanto lo hemos elegido para la implementación en este problema.

Hemos desarrollando un prototipo que integra todas las anotaciones poniéndolas a disposición de un buscador semántico y así en última instancia de un usuario que pueda efectuar consultas y visualizar los documentos resultantes de las mismas mediante una interfaz apropiada. El prototipo utiliza la API *SemanticSearch* y

presenta una interfaz básica suficiente como prueba de concepto. Se está construyendo un sistema más robusto y una interfaz más amigable utilizando la API *Lucene* en un proyecto de grado asociado a este trabajo de investigación.

El conocimiento y la experiencia obtenida de este proyecto son base para futuras implementaciones que pueden hacer extensivas estas técnicas a otros problemas de clasificación e interpretación de textos, y en una perspectiva más amplia a otros problemas de gestión del conocimiento.

Referencias

1. Moore, C.: Diving into Data, Infoworld, http://www.infoworld.com/article/02/10/25/021028feundata_1.html, 25 de octubre (2002)
2. Ferrucci, D., Lally, A.: Building an Example Application with the Unstructured Information Management Architecture. IBM Systems Journal 43, 3, 455--475 (2004)
3. Alias-I: LingPipe 3.8.2. http://alias-i.com/lingpipe (2008)
4. Feldman, R., Sanger, J.: The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data, Cambridge University Press, New York (2007).
5. Tomanek, K., Wermter, J.: JULIE Lab Lingpipe Gazetteer Annotator Version 2.1. Jena University Language & Information Engineering (JULIE) Lab. www.julieblab.de (2008)
6. OMG: XML Metadata Interchange (XMI), v 2.1.1 (2007)
7. Pérez, M. A., Cardoso, A. C.: Comparación de Algoritmos de Aprendizaje Semi-Supervisado. En V Jornadas de Ciencia y Tecnología de Facultades de Ingeniería del NOA, Salta (2009)
8. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed., Morgan Kaufmann (2005)
9. Blum, A., Mitchell, T.: Combining Labeled and Unlabeled Data with Co-Training. En Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT' 98. ACM, New York, NY, 92--100 (1998)
10. Nigam, K., Ghani, R.: Analyzing the Effectiveness and Applicability of Co-Training. En Proceedings of the Ninth international Conference on Information and Knowledge Management. CIKM '00. ACM, New York, 86--93 (2000)
11. Nigam, K., McCallum, A. K., Thrun, S., Mitchell, T.: Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning 39 (2-3) 103--134 (2000)
12. Chu-Carroll, J., Prager, J., Czuba, K., Ferrucci, D., Duboue, P.: Semantic Search via XML Fragments: a High-Precision Approach to IR. En Proceedings of the 29th Annual international ACM SIGIR Conference on Research and Development in information Retrieval, SIGIR '06. ACM, New York, NY, 445--452 (2006)
13. Hampp, T., Lang, A.: Semantic Search in WebSphere Information Integrator OmniFind Edition: The Case for Semantic Search. IBM Developer Works (2005)

Agradecimientos. Este trabajo ha sido financiado en parte por el Consejo de Investigaciones de la Universidad Católica de Salta (Resol Rect 723/08). Las autoras agradecen la colaboración de la Prof Constanza Diedrich por su asesoramiento sobre la taxonomía de resoluciones rectorales y de los alumnos David Zamar e Iván Ramos.