Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Clasificación automática de papers de Ciencias de la Computación

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Alejandro Scherz
alejsch@gmail.com
LU: 255/04

Director: José Castaño - jcastano@dc.uba.ar

Buenos Aires, 2018

Clasificación automática de papers de Ciencias de la Computación

El presente trabajo consiste en el desarrollo de un sistema de clasificación automática de papers, que permite asignarles la categoría adecuada de acuerdo al contenido de los mismos mediante la aplicación de técnicas de machine learning. Una herramienta de estas características puede ahorrar a los publicadores de papers el tiempo significativo que implica atravesar un proceso de selección de categorías jerárquicas para clasificarlos correctamente, y de esta manera facilitar su búsqueda una vez que se han publicado en los repositorios bibliográficos online.

Para la implementación de este sistema se obtuvo una colección de artículos de todo tipo del sitio web de la Librería Digital de la ACM (Association for Computing Machinery) [ws@a] a partir de la cual se generaron los datasets de entrenamiento que se utilizarán junto con una herramienta llamada MALLET [ws@c] para determinar la categoría de papers que aún no han sido clasificados, estudiando el caso particular de las últimas taxonomías que presenta dicho sitio (versiones de 1998 y 2012).

El sistema, que se encuentra implementado en Python 2.7 [ws@d] e interactúa con la API de Java [ws@b] de la herramienta MALLET, provee una interfaz interactiva (tanto gráfica como en consola), en la cual se puede seleccionar diversas opciones, entre ellas la etapa de clasificación a ejecutar, los campos del paper tenidos en cuenta para el entrenamiento, los algoritmos de clasificación a utilizar, y la taxonomía a considerar para la predicción de las categorías.

Palabras claves: Clasificación jerárquica de textos, machine learning, data mining, ACM.

Automatic classification of papers of Computer Science

The present work consists in the development of a system of automatic classification of papers, which allows them to be assigned the appropriate category according to their content through the application of machine learning techniques. Such a tool can save paper publishers the significant time it takes to go through a process of selecting hierarchical categories to sort them correctly, and thus facilitate their search once they have been published in online bibliographic repositories.

For the implementation of this system, a collection of articles of all kind was obtained from the website of the Digital Library of the ACM (Association for Computing Machinery) [ws@a] , from which the training datasets were generated that will be used together with a tool called MALLET [ws@c] to determine the category of papers that have not yet been classified, studying the particular case of the latest taxonomies presented by the site (1998 and 2012 versions).

The system, which is implemented in Python 2.7 [ws@d] and interacts with the Java [ws@b] API of the MALLET tool, provides an interactive interface (both graphical and console mode), in which various options can be selected, including the classification stage to execute, the paper fields taken into account for the training, the classification algorithms to use, and the taxonomy to consider for the prediction of the categories.

Keywords: Hierarchical text classification, machine learning, data mining, ACM.

Agradecimientos

En primer lugar quiero agradecer a mi director José Castaño, por su importante guía a lo largo de todo el proceso, y sobre todo por tenerme la mayor de las paciencias hasta la conclusión final de la tesis, pese a todos los diferentes inconvenientes externos que tuve que pasar durante el transcurso de la misma, que desafortunadamente hicieron que la última etapa se demore un tiempo mucho mayor del que estaba planeado en un principio.

Por otro lado me gustaría mencionar a todos los docentes que me tocaron a lo largo de mi carrera, ya que su simpatía, amabilidad, predisposición y dominio de sus asignaturas hicieron no sólo que mi recorrido por las diferentes cursadas fuera más ameno, sino también que yo haya logrado aprender muchísimas cosas que al día de hoy llevo a la práctica en el ámbito profesional.

Tambien dedico unas palabras a mis compañeros de cursada, con quienes en muchos casos hemos entablado una relación de amistad, y hemos compartido varios fines de semana hasta última hora, ya sea para terminar un trabajo práctico, o estudiar juntos para algún examen parcial o final.

Párrafo aparte para mis viejos, quienes me dieron la fuerza y energía necesarias para poder llevar adelante esta etapa final y no dejar que me rinda en ningún momento, y también por haberme brindado su ayuda constante.

Y finalmente, al resto de mi familia, mis amigos, y mis compañeros de trabajo, por todos los buenos momentos compartidos, que hicieron este proceso más llevadero.

A la memoria de mis abuelos Raquel y Jacobo.

Índice general

Índice de figuras

1

Introducción

1.1. Motivación

Los artículos científicos (también llamados coloquialmente *papers*) son trabajos de investigación destinados a la publicación en revistas especializadas. Tienen como objetivo difundir, de manera clara y precisa, los resultados de una investigación realizada sobre un área determinada del conocimiento. También buscan fomentar el desarrollo de métodos experimentales innovadores.

Los papers están cuidadosamente redactados para expresar de un modo claro y sintético lo que se pretende comunicar, e incluir las citas y referencias, indispensables para contextualizar, verificar y poder reproducir los resultados originales que se dan a conocer en el mismo.

La estructura típica de todo paper suele componerse de las siguientes secciones:

- Título: debe especificarse con exactitud según lo referente al texto.
- Autores: deben escribirse después del título de forma secuencial y separados por comas.
- Resumen (o abstract): presentan el contexto y propósito del estudio, los procedimientos básicos (ej: selección de sujetos del estudio o animales de laboratorio, métodos observacionales o analíticos, etc), los descubrimientos y conclusiones principales.
- Palabras clave (o keywords): destacan los principales tópicos a los que hará referencia el artículo.
- Publicación (o journal): revista científica en la que se presenta el artículo.
- Contenido: se presenta en un esquema organizado, generalmente compuesto por: introducción, materiales y métodos, resultados, discusión y bibliografía.

En la figura 1.1 se muestra un ejemplo de estas secciones.

Un aspecto importante para publicar papers es el de proveer información e indexación adecuadas para los mismos, de acuerdo al sistema de clasificación que se utilice en la fuente académica en que se presenten. Esto es beneficioso para los autores de dichos papers, debido a que una categorización apropiada provee al lector una referencia rápida del contenido, facilitando la búsqueda tanto de literatura relacionada como de citas a sus trabajos en toda clase de recursos en línea.

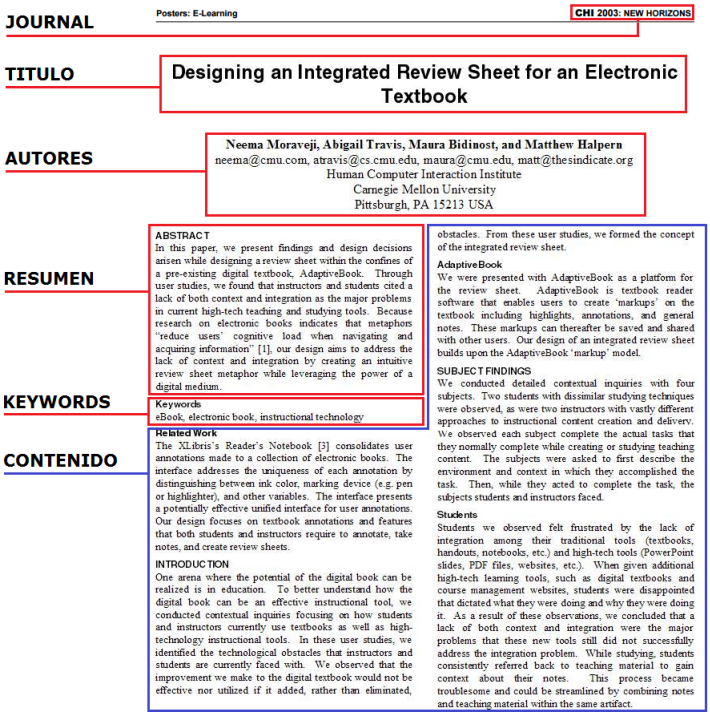

Fig. 1.1: Ejemplo de la primera hoja de un paper, con sus diferentes secciones.

No obstante, estos sistemas de clasificación requieren que sean los mismos autores quienes apliquen las categorías a sus papers en forma manual, ya que son precisamente los expertos en cuanto al tipo de contenido que aspiran a publicar. Debido a esto, se ven obligados a familiarizarse con las taxonomías que ofrecen dichos sistemas, para luego atravesar un proceso de selección que determinará, para cada nivel de jerarquía, cuales son los tópicos que más se ajustan a los papers en cuestión, lo que puede significar una pérdida de tiempo considerable.

Dada esta situación, se vuelve de suma importancia la posibilidad de realizar este proceso de clasificación de forma automática. Con este fin, en el presente trabajo se desarrolla un clasificador basado en técnicas de aprendizaje automático o machine learning, que por medio del propio contenido de una colección de papers de entrenamiento, puede predecir con un cierto nivel de performance la categoría correcta para los papers que aún no se encuentran clasificados.

Como existe una amplia variedad de áreas científicas, para el caso del presente trabajo nos concentraremos en el caso particular de los papers relacionados al campo de las Ciencias de la Computación, para el cual ya existen diversos repositorios bibliográficos como ACM DL, dblp, EBSCO, SCOPUS, y arXiv, entre otros. Este trabajo se centra específicamente en el sistema de clasificación aportado por la ACM en sus distintas taxonomías.

1.2. Trabajos previos

La clasificación automática de textos siempre ha sido un tema de investigación importante desde la existencia de documentos digitales, ya que ésta permite manejar la enorme cantidad de información disponible en la web [Iko05] . Está basada principalmente en técnicas de *machine learning*, las cuales construyen automáticamente un clasificador que aprende las características o *features* de las categorías a partir de un conjunto preclasificado de documentos [Seb02] . Estas técnicas juegan un rol muy importante en la extracción y resumen de información, recuperación de texto, entre otros.

Se ha estudiado una cantidad considerable de enfoques para ayudar al proceso de clasificación de textos, el cual se puede representar a grandes rasgos en 5 etapas principales, las cuales incluyen: representación y preprocesado de los datos [Kim06] [Por80] , selección y transformación de características, construcción de modelos de espacio vectorial, aplicación de algoritmos de clasificación (siendo los mas populares Naive Bayes, SVM, Redes Neuronales y Maxima Entropia [Tin11] [Dum00] [Rui02] [Nig99] [Ng01]), y evaluación de la performance del clasificador generado [Yan99] .

Por otra parte, si bien la mayoría de trabajos existentes se han basado en datasets de dominio público como los que posee la UCI (University of California Irvine) para medir el rendimiento algorítmico - como es el caso de Reuters-21578, MEDLINE y DigiTrad, entre otros -, también han habido otros trabajos que utilizaron una colección de documentos de la ACM [Zha04] [San09] , aplicando además un enfoque de clasificación jerárquica local, en contraposición a otros artículos que se basaban en la jerarquía plana convencional, sin tener en cuenta una taxonomía que permita agrupar las unidades de conocimiento de una forma estructural.

En el caso del presente trabajo, realizaremos un estudio similar que permita comparar estos dos enfoques jerárquicos, haciendo uso de una herramienta llamada MALLET [ws@c] , la cual provee una interfaz para clasificar textos en forma eficiente.

1.3. Estructura del informe

Primero se introducen en el capítulo 2 los conceptos generales de los sistemas de clasificación que se verán a lo largo de esta tesis.

Luego en el capítulo 3 se presentan en detalle los métodos de clasificación de textos con aprendizaje automático o machine learning, analizando dos algoritmos en particular: Naive Bayes y Máxima Entropía, además de la ya mencionada herramienta MALLET, que

usará el clasificador para ejecutar dichos algoritmos.

En el capítulo 4, por otra parte, se describe el clasificador automático propiamente dicho, especificando los diferentes métodos posibles (jerarquía global y local), y también analizando las etapas en las que se divide el procesamiento del mismo; por otro lado, también se expone la interfaz de la aplicación implementada para el presente trabajo.

Los resultados obtenidos se exhiben en el capítulo 5, junto con un análisis de los mismos para los algoritmos que se evalúan en esta tesis.

Finalmente, el capítulo 6 muestra las conclusiones obtenidas a partir de este trabajo y se discuten algunas alternativas de mejoramiento para trabajos a futuro, y en la sección final se exhiben las referencias bibliográficas a todos los artículos en los que se ha basado el presente trabajo.

2

Conceptos Generales

2.1. ACM

2.1.1. Association for Computing Machinery (ACM)

La Asociación de los Sistemas Informáticos (Association for Computing Machinery, en sus siglas ACM) fue fundada en 1947 como la primera sociedad científica y educativa acerca de la Computación. Su objetivo es el de publicar diversos papers, revistas y periódicos científicos relacionados con la Computación; patrocina conferencias en varias áreas del campo (principalmente los grupos de investigación dedicados a la computación gráfica -SIGGRAPH- y a la comunicación de datos -SIGCOMM-) y otros eventos relacionados con las Ciencias de la Computación, como por ejemplo la competencia internacional de programación ACM-ICPC (International Collegiate Programming Contest); publica una extensiva biblioteca digital y una referencia de la literatura de la Computación. La ACM también ofrece seguros y otros servicios a sus miembros de los Estados Unidos.

La Librería Digital de ACM (ACM Digital Library, en sus siglas ACM DL), por su parte, es una colección de textos completos de todas las publicaciones en la ACM, que comprenden sus artículos, revistas y procedimientos de conferencias. Es un repositorio bibliográfico inmenso que posee aproximadamente más de un millón de entradas, y cuya base de datos contiene trabajos clave que abarcan todo tipo de géneros y categorías, escritos por las editoriales más importantes que conciernen al área de la Computación. La versión web se puede acceder con el link que está indicado en la bibliografía [ws@a] .

2.1.2. ACM Computing Classification System (CCS)

El Sistema de Clasificación de Computación de la ACM (ACM Computing Classification System, en sus siglas CCS) en su versión 2012 se ha implementado como una ontología poli-jerárquica que puede ser utilizada en aplicaciones web semánticas. Esta

versión reemplaza a la tradicional de 1998, la cual ha servido como el estándar de facto para el campo de la Computación. Actualmente está en proceso de integración con las capacidades de búsqueda y las interfaces de tópicos visuales de la ACM. Depende de un vocabulario semántico como la fuente de categorías y conceptos que refleja el estado del arte de la disciplina de la Computación, y es receptivo a cambios estructurales a medida que va evolucionando en el tiempo. La ACM provee una herramienta dentro del formato de interfaz visual para facilitar la aplicación de las categorías del CCS 2012 a los papers venideros, y un proceso para asegurar que el CCS continúe actualizado y relevante. Este nuevo sistema de clasificación jugará un rol clave en el desarrollo de una interfaz de búsqueda de personas en la ACM DL para complementar su búsqueda bibliográfica tradicional.

El árbol completo de clasificación del CCS está disponible en forma gratuita para propósitos educativos y de investigación en los formatos XML, Microsoft Word y HTML. Mientras que en la ACM DL, el sistema es presentado tanto en un formato de interfaz visual que facilita la navegación y el feedback, como en un archivo de texto plano con toda la información necesaria.

2.2. Taxonomias

2.2.1. Taxonomia de CCS

Para poder asignar una categorización a los papers que aspiran a ser publicados, el CCS ofrece una taxonomía conceptual que fue variando a lo largo de los años, de la que en esta tesis se estudia en particular las dos últimas versiones, 1998 y 2012. Estas se encuentran conformadas por un arbol de categorias, de 4 y 6 niveles de jerarquía respectivamente, en los que cada una de las áreas del conocimiento en computación ha sido identificada. Cada una de éstas áreas, a su vez, está subdividida en varias unidades de conocimiento que contienen los temas relevantes al área. La filosofía que se sigue para este método de organización es la siguiente:

- El “corazón” del CCS es un árbol, la forma más sencilla en la cual se representa una estructura jerárquica en un formato de publicación lineal.
- El árbol de clasificación está restringido a una cantidad fija de niveles codificados con letras y números, con el fin de que sea capaz de reflejar la estructura esencial de una disciplina en un periodo extendido.
- Los subject descriptors (el último nivel no codificado del árbol) proveen suficiente detalle para hacer frente a nuevos desarrollos en un campo determinado. Se han convertido en parte permanente del árbol, dado que no es fácil eliminarlos sin alterar las referencias a los trabajos que clasificaron previamente.
- La cantidad de uso de los términos del CCS permite determinar cuáles serán tanto los términos a eliminar, como las secciones a expandir. Por ejemplo, para la versión de 1998, se modificó la estructura general, pero sólo en sus niveles más bajos para ser compatible con sus versiones anteriores.

El árbol de la versión de 1998, por ejemplo, consiste en 11 nodos de primer nivel, y 1 o 2 niveles bajo cada uno de ellos (ver figura 2.1). A su vez, el conjunto de hijos de todos los nodos comienza con un nodo “General” y termina con un nodo “Misceláneos” (ver figura 2.3). Los nodos de primer nivel están designados por letras de A a K, mientras que el segundo y tercer nivel tienen una designación que combina letras y números.

Las 11 áreas definidas en la versión de 1998 fueron transformadas en 14 áreas para la nueva versión 2012 de la taxonomía. Aquí se muestran las diferencias entre la versión final de ambas ontologías, en su primer nivel:

CCS 1998	
A	General Literature
B	Hardware
C	Computer Systems Organization
D	Software
E	Data
F	Theory of Computation
G	Mathematics of Computing
H	Information Systems
I	Computing Methodologies
J	Computer Applications
K	Computing Milieux

Fig. 2.1: Tabla de contenidos del CCS de 1998.

CCS 2012	
A	General and Reference
B	Hardware
C	Computer Systems Organization
D	Networks
E	Software and its Engineering
F	Theory of Computation
G	Mathematics of Computing
H	Information Systems
I	Security and Privacy
J	Human-centered Computing
K	Computing Methodologies
L	Applied Computing
M	Social and Professional Topics
N	Proper nouns: People, technologies and companies

Fig. 2.2: Tabla de contenidos del CCS de 2012.

Y este es un ejemplo de la localización de categorías dentro del primer árbol:

- A General Literature
  - A.0 GENERAL
  - A.1 INTRODUCTORY AND SURVEY
  - A.2 REFERENCE
  - A.m MISCELLANEOUS
- B Hardware
  - B.0 GENERAL
  - B.1 CONTROL STRUCTURES AND MICROPROGRAMMING
    - B.1.0 General
    - B.1.1 Control Design Styles
    - B.1.2 Control Structure Performance Analysis and Design Aids
    - B.1.3 Control Structure Reliability, Testing, and Fault-Tolerance
    - B.1.4 Microprogram Design Aids

    ...

Fig. 2.3: Fragmento expandido del CCS de 1998.

Para el uso actual de la clasificación, los nodos de primer nivel nunca se utilizan para clasificar material; si un paper trata un tópico de un nivel general, se tomará dicho nodo (ej: para *B: Hardware* se toma B.0). Sin embargo, si cubre diversos tópicos de sus nodos hermanos, pero no necesariamente todos, se toma el nodo general para los niveles inferiores (ej: para *K.7: The Computing Profession*, *K.7.0: General* puede servir para un artículo general del mismo, pero también puede tratar específicamente alguno de estos: *K.7.1: Occupations*, *K.7.2: Organizations*, o *K.7.3: Testing, Certification, and Licensing*).

En cuanto a los descriptores, estos se encuentran asociados con la mayoría de hojas del árbol (aunque difícilmente con las de General o Misceláneos). Además de los que forman parte del CCS, también se pueden incluir descriptores implícitos a un nodo apropiado (ejemplo: 'C++' para *D.3.2: Language Classifications*, 'OS/2' para *D.4.0: Operating Systems, General*, y 'Bill Gates' y 'Grace Murray Hopper' para *K.2: History of Computing*).

3

Machine Learning

3.1. Introducción

El Aprendizaje Automático o Machine Learning es la rama de la Inteligencia Artificial que se dedica al estudio de los agentes/programas que aprenden o evolucionan basados en su experiencia, para realizar una tarea determinada cada vez mejor. El objetivo principal de todo proceso de aprendizaje es utilizar la evidencia conocida para poder crear una hipótesis y poder dar una respuesta a nuevas situaciones no conocidas.

Hoy en día existe una amplia variedad de aplicaciones que utilizan agentes basados en aprendizaje en numerosas ramas de la industria y de la ciencia. A continuación se pueden enumerar una serie de usos prácticos que están vigentes en la actualidad:

- En el Procesamiento del Lenguaje Natural se utiliza para el análisis sintáctico y morfológico de los textos, extracción de Información, clasificación automática de documentos y agrupamiento semántico, entre otras tareas (en esta tesis nos concentraremos en este último punto);
- En los Sistemas de Recuperación de Información (como buscadores de Internet) se utilizan técnicas de ML para mejorar la performance de sus búsquedas y confeccionar rankings personalizados según la experiencia de los usuarios;
- En el diagnóstico médico, para asistir a profesionales según la historia clínica y los síntomas que presenta un paciente, y determinar el riesgo de vida de un accidentado y decidir entre el envío de una ambulancia, un particular o asistencia telefónica;
- En Biología, para la clasificación de especies, reconocimiento de tumores o arritmias o patrones en cadenas de ADN;
- En Finanzas e Industria bancaria, para definir modelos de riesgo y fraude crediticio que califican a los solicitantes según sus antecedentes bancarios, y de predicción de comportamiento en el mercado de valores;

- En el análisis de imágenes, para reconocer escritura manuscrita, identificar direcciones y remitentes de documentos legales, detectar objetos dentro de una imagen (como personas, rostros, accidentes geográficos);
- En juegos de estrategia (Backgammon y Damas), para software que aprende a realizar movimientos válidos y contribuir a mejorar su propia performance;
- Y finalmente en Robótica, para regular el desplazamiento de robots y planificación de tareas.

3.2. Clasificación de Textos

3.2.1. Clasificadores lineales

En el área del aprendizaje automático y la estadística, la clasificación es el problema de identificar la pertenencia de un conjunto de objetos a una clase o grupo determinados mediante la utilización de las características o features de éstos. Un ejemplo podría ser el de asignar un diagnóstico a un paciente de acuerdo a los valores que presenten su análisis clínicos (género, presión sanguínea, presencia de síntomas, etc).

Para el caso concreto del aprendizaje automático, un problema de clasificación se puede considerar como una instancia de aprendizaje supervisado, en la cual se encuentra disponible un subconjunto de objetos que ya se encuentra identificado, a diferencia de los métodos no supervisados como el de *clustering*, el cual agrupa la información en categorías en base a alguna medida inherente de similaridad o distancia.

Un ejemplo de clasificador estadístico supervisado es el llamado *clasificador lineal*, el cual implementa un método de decisión basado en una combinación lineal de las características de los objetos observados, que se representa con un vector numérico (feature vector).

Formalmente, sea x un feature vector de entrada de un clasificador, entonces la salida del mismo se define como:

$$y = f(w \cdot x) = f(\sum_j w_j \cdot x_j) \tag{3.1}$$

donde w es un vector con pesos, y f es una función que convierte el producto vectorial de ambos vectores en la salida deseada (en otras palabras, w es una transformación lineal que mapea x en $\mathbb{R}$). Dicho vector w es "aprendido" a partir de muestras de entrenamiento, mientras que f mapea los valores a una clase u otra dependiendo de un cierto umbral, mediante una probabilidad de pertenencia a ésta.

Para el caso de una clasificación binaria, la operación de un clasificador lineal equivale a dividir un espacio de entrada multidimensional en un hiperplano, donde los puntos de un lado de éste pertenecen a una clase y los restantes a la otra.

Estos tipos de clasificadores muestran buen rendimiento para problemas prácticos como el de clasificar documentos de texto, que involucran una gran cantidad de variables (donde x suele depender del número de ocurrencias de cada palabra en un documento) alcanzando niveles de precisión comparables a clasificadores no-lineales, y con una eficiencia mayor en cuanto a tiempos de ejecución.

Observar que la ecuación 3.1 se corresponde con la definición de un perceptrón simple en redes neuronales si la funcion f es una función sigmoide.

3.2.2. Modelos generativos y discriminativos

Estos clasificadores estadísticos se basan en el Teorema de Bayes, que relaciona las probabilidades de una hipótesis h antes y después de considerar una evidencia e. Formalmente, esta probabilidad condicional se expresa de la siguiente forma:

$$P(h \mid e) = \frac{P(h) \cdot P(e \mid h)}{P(e)} \tag{3.2}$$

donde:

- $P(h|e)$ es la probabilidad de la hipótesis dada la evidencia (a posteriori);
- $P(e)$ es la probabilidad de la evidencia bajo toda hipótesis (a priori);
- $P(e|h)$ es la probabilidad de la evidencia dada la hipótesis (verosimilitud);
- $P(h)$ es la probabilidad de la hipótesis bajo toda evidencia (marginal).

Aplicando esta expresión a nuestro caso, un ejemplo de hipótesis y evidencia podría ser el siguiente:

- h = “El documento pertenece a una clase dada C”
- e = “El documento contiene las características D = $\{D_1, ..D_n\}$”

Para determinar los parámetros de un clasificador lineal existen dos clases de métodos: estos son los modelos generativos y discriminativos.

Los modelos generativos, por un lado, modelan la forma en la cual los objetos fueron generados para poder categorizar las observaciones, mediante funciones de probabilidad condicional, particularmente $P(e|h)$ y $P(h)$. Algunos ejemplos de algoritmos que los emplean son:

- Latent Dirichlet Allocation (LDA), el cual asume un modelo de categorías con distribución de Dirichlet.

- Naïve Bayes, el cual utiliza modelos multinomiales o de Bernoulli multivariables..

Por otro lado, los modelos discriminativos buscan categorizar las observaciones mediante la creación de fronteras de decisión que intentan discriminar dichas observaciones, calculando $P(h|e)$ directamente. A continuación se exhiben algunos casos:

- Regresión logística, que genera un estimador de w de máxima verosimilitud, asumiendo que los objetos fueron generados por un modelo que depende de la salida.
- Perceptrón, que se basa en redes neuronales e intenta corregir errores en el entrenamiento.
- Support vector machine (SVM), que maximiza el margen entre el hiperplano de decisión y los objetos en el entrenamiento.

Debido a que estos métodos son netamente estadísticos, suponemos que dependen fuertemente del contexto, y esto significará que si las características más comunes de cada clase varían mucho entre dominios, la clasificación de documentos de un dominio no entrenado tendrá resultados pobres. En el capítulo del clasificador automático se realizará una etapa de preprocesamiento, donde se obtienen las características más relevantes a fin de atenuar este tipo de dificultades.

3.3. Algoritmos de Clasificación

3.3.1. Naïve Bayes

En términos simples, un clasificador de Bayes ingenuo o Naïve Bayes asume que la existencia (o no) de una característica particular no está relacionada con la existencia (o no) de cualquier otra característica, dada la clase variable. Por ejemplo, si una fruta puede ser considerada como una manzana si es roja, redonda y de alrededor de 7 cm de diámetro, entonces dicho clasificador considera que cada una de estas características es independiente de la probabilidad de que esta fruta sea una manzana, esté o no presente otro tipo de características.

Una ventaja del clasificador de Bayes ingenuo es que solo se requiere una pequeña cantidad de datos de entrenamiento para estimar los parámetros (las medias y las varianzas de las variables) necesarias para la clasificación. Como las variables independientes se asumen, solo es necesario determinar las varianzas de las variables de cada clase y no toda la matriz de covarianza.

Sea C la variable discreta que define las categorías representando a las clasificaciones de las taxonomías, y D = $\{D_1, ..., D_n\}$ el conjunto de características del documento a clasificar, entonces el modelo probabilístico para el clasificador se definirá de la siguiente manera:

$$P_{NB}(C \mid D) = P(C \mid D_1, ..., D_n) \tag{3.3}$$

Es decir, la probabilidad de que dado un documento D con características D_i pertenezca a la clase C = c, está condicionada naturalmente por la probabilidad de ocurrencia de cada D_i. Utilizando el Teorema de Bayes, se puede reescribir:

$$P(C \mid D_1, ..., D_n) = \frac{P(C) \cdot P(D_1, ..., D_n \mid C)}{P(D_1, ..., D_n)} \tag{3.4}$$

Si observamos el numerador, aplicando primero la definición de probabilidad compuesta (pc), y luego la regla de la cadena (rc), obtenemos el siguiente resultado:

$$\begin{aligned} P(C) \cdot P(D_1, ..., D_n \mid C) &\overset{pc}{=} P(C, D_1, ..., D_n) \\ &\overset{rc}{=} P(C) \cdot P(D_1 \mid C) \cdot ... \cdot P(D_n \mid C, D_1, .., D_{n-1}) \end{aligned} \tag{3.5}$$

En este punto es donde el método obtiene el adjetivo de "ingenuo": asume que las características son probabilísticamente independientes, utilizando la definición de independencia condicional (ic):

$$\forall i \neq j, P(D_i \mid C, D_j) = P(D_i \mid C) \tag{3.6}$$

y bajo esta asunción, finalmente queda:

$$\begin{aligned} P(C) \cdot P(D_1 \mid C) \cdot ... \cdot P(D_n \mid C, D_1, .., D_{n-1}) &\overset{ic}{=} P(C) \cdot P(D_1 \mid C) \cdot ... \cdot P(D_n \mid C) \\ &= P(C) \cdot \prod_i P(D_i \mid C) \end{aligned} \tag{3.7}$$

Cabe aclarar que dicho supuesto no parece intuitivo, ya que en cualquier texto sus términos son generalmente dependientes. A pesar de esta fuerte decisión, el método ha sido extensamente utilizado en clasificación de textos con resultados satisfactorios. Si se reemplaza en 3.3 lo obtenido en 3.7 nos queda que:

$$P_{NB}(C \mid D) = \frac{P(C) \cdot \prod_i P(D_i \mid C)}{P(D_1, ..., D_n)} \tag{3.8}$$

De esta manera, partiendo del cálculo de una probabilidad "a posteriori" para determinar el grado de pertenencia a una clase, obtuvimos una expresión equivalente que se basa exclusivamente en la evidencia y los datos obtenidos en etapa de entrenamiento, y es con esta expresión que Naïve Bayes realiza la clasificación de acuerdo a la clase para la

que obtenga mayor probabilidad. Formalmente, siendo $\{D_1, ..., D_n\}$ las características del documento D, la clasificación es realizada de la siguiente manera:

$$y = \underset{c \in C}{argmax} \frac{P(C = c) \cdot \prod_i P(D_i \mid C = c)}{P(D_1, ..., D_n)} \tag{3.9}$$

El valor de $P(C)$ puede ignorarse en el caso de que se trabaje con un corpus balanceado (cantidad similar de papers en promedio para cada clase), ya que esto evita polarizaciones hacia una categoría u otra dadas simplemente por una mayor cantidad de papers de esa categoría (en el caso más extremo, por ejemplo si se tienen 1000 papers en B.1 y solo 1 en B.2, muchos de estos se orientarán hacia la primer clase). Por otro lado, $P(D_1, ..., D_n)$ es un dato conocido - la frecuencia de las características en el documento a clasificar - y que por ser conocido y constante para todas las clases, se suele ignorar; y más relevantemente, que la productoria es un dato fácilmente obtenible luego de haber corrido al clasificador con un conjunto de entrenamiento.

3.3.2. Ejemplo

Supongamos que se tiene la siguiente tabla donde se decide si se puede jugar al tenis o no bajo ciertas circunstancias climáticas, como por ejemplo el pronostico, el tiempo, la humedad y el viento:

Día	P (Pronóstico)	T (Tiempo)	H (Humedad)	V (Viento)	J (Jugar tenis)
1	Sol	Calor	Alta	Debil	No
2	Sol	Calor	Alta	Fuerte	No
3	Nubes	Calor	Alta	Debil	Si
4	Lluvia	Templado	Alta	Debil	Si
5	Lluvia	Frío	Normal	Debil	Si
6	Lluvia	Frío	Normal	Fuerte	No
7	Nubes	Frío	Normal	Fuerte	Si
8	Sol	Templado	Alta	Debil	No
9	Sol	Frío	Normal	Debil	Si
10	Lluvia	Templado	Normal	Debil	Si
11	Sol	Templado	Normal	Fuerte	Si
12	Nubes	Templado	Alta	Fuerte	Si
13	Nubes	Calor	Normal	Debil	Si
14	Lluvia	Templado	Alta	Fuerte	No
15	Sol	Frío	Alta	Fuerte	??

Supongamos que se desea saber si es posible jugar dadas las condiciones climáticas del último día, para determinarlo creamos la instancia correspondiente:

x=(P=Sol, T=Frio, H=Alta, V=Fuerte)

Usando la definicion en 3.9 es necesario realizar los siguientes calculos:

- $P(J = Si) = 9/14$

- $P(J = No) = 5/14$
- $P(P = Sol|J = Si) \cdot P(T = Frio|J = Si) \cdot P(H = Alta|J = Si) \cdot P(V = Fuerte|J = Si) = (2/9)(3/9)(3/9)(3/9)$
- $P(P = Sol|J = No) \cdot P(T = Frio|J = No) \cdot P(H = Alta|J = No) \cdot P(V = Fuerte|J = No) = (3/5)(1/5)(4/5)(3/5)$
- $P(P = Sol, T = Frio, H = Alta, V = Fuerte) = (5/14)(4/14)(7/14)(6/14)$

Finalmente, las probabilidades buscadas se calculan de esta manera:

- $P_{NB}(J = Si|x) = \frac{(9/14)\cdot(2/9)(3/9)(3/9)(3/9)}{(5/14)(4/14)(7/14)(6/14)} \approx 0{,}242$
- $P_{NB}(J = No|x) = \frac{(5/14)\cdot(3/5)(1/5)(4/5)(3/5)}{(5/14)(4/14)(7/14)(6/14)} \approx 0{,}942$

3.3.3. Máxima Entropía

El clasificador de máxima entropía o MaxEnt, al igual que el de Naive Bayes, también se basa en la definición de un modelo probabilístico para determinar el grado de pertenencia de una entrada a una clase determinada. En este caso, MaxEnt pertenece a la familia de los clasificadores log-lineales, los cuales extraen un conjunto de características de la entrada, para luego combinar el logaritmo de sus probabilidades de modo lineal y utilizar el resultado como un exponente.

Se basa en la regresión logística multinomial, la cual generaliza el método de regresión logística para problemas multiclase, es decir, con más de dos posibles resultados discretos. Dicho de otra forma, se trata de un modelo que sirve para predecir las probabilidades de los diferentes resultados posibles de una distribución categórica como variable dependiente, dado un conjunto de variables independientes (que pueden ser de valor real, binario, categórico-valorado, etc.)

Sea un documento D con características $\{D_1, ..., D_n\}$ y una categoría C. Dicho modelo es el siguiente:

$$P_{ME}(C \mid D) = \frac{1}{Z(D)} \cdot exp(\sum_i \beta_{C,i} \cdot f_i(C, D)) \tag{3.10}$$

donde Z(D) es un factor de normalización para que 3.10 sea efectivamente una distribución de probabilidad (es decir, que la suma de todas las probabilidades sea igual a 1), y está definida de la siguiente manera:

$$Z(D) = \sum_{c \in C} exp(\sum_i \beta_{c,i} \cdot f_i(c, D)) \tag{3.11}$$

y $f_i(c, D)$ es una función indicador que indica si en D se cumple una cierta propiedad ϕ_i para la clase c; se utiliza para restringir al modelo probabilístico tal que lo observado en el entrenamiento se refleje en el modelo utilizado para clasificar nuevos documentos. En general, se define de esta manera:

$$f_i(c, D) = \alpha(\phi_i(D) \wedge c = C_j) \tag{3.12}$$

donde la función binaria α es la siguiente:

$$\alpha(x) = \begin{cases} 1 & \text{si } x = True \\ 0 & c.c. \end{cases} \tag{3.13}$$

Los $\beta_{c,i}$ son los parámetros que indican el peso de cada D_i, donde valores elevados indican una mayor probabilidad de pertenencia a C. La obtención de los valores se realiza resolviendo un problema de optimización a través del algoritmo iterativo BFGS (Broyden-Fletcher-Goldfarb-Shanno).

El modelo definido de esta manera elige valores para $\beta_{c,i}$ de manera tal que se maximiza la entropía de la distribución obtenida, si se tiene en cuenta además la restricción de que los valores esperados de D_i con respecto al modelo son iguales a los valores esperados con respecto a los datos de entrenamiento. La idea detrás de este planteo es simple: maximizar la cantidad de información extraída en la etapa de entrenamiento, sin hacer ningún tipo de asunción respecto de los datos y manteniendo consistencia con respecto a los mismos. Esta es una diferencia clave con respecto a Naive Bayes, que asume independencia de características, y por eso responde mejor en ciertos contextos donde hay dependencia fuerte.

3.3.4. Ejemplo

A continuacion se muestra un problema particular de tagging de secuencias, que es el de identificar la categoría gramatical de cada una de las partes de una oración:

Secretariat	is	expected	to	race	tomorrow
SusSing	VrbPre3	VrbPart	A	??	Adv

Además se han definido las siguientes funciones indicadoras para evaluar las características de la entrada:

$$f_1(c, x) = \alpha(x_i = race \wedge c = SusSing) \qquad f_2(c, x) = \alpha(x_{i-1} = A \wedge c = VrbBase)$$
$$f_3(c, x) = \alpha(suf(x_i) = ing \wedge c = VrbGeru) \qquad f_4(c, x) = \alpha(islower(x_i) \wedge c = VrbBase)$$
$$f_5(c, x) = \alpha(x_i = race \wedge c = VrbBase) \qquad f_6(c, x) = \alpha(x_{i-1} = A \wedge c = SusSing)$$

La instancia que se desea determinar en este caso es la siguiente:

$x = (\text{"}race\text{"})$

De esta manera se computan las variables con el algoritmo de máxima entropía para el cálculo de probabilidades de que dicha categoría sea $VrbBase$ (Verbo Base) o $SusSing$ (Sustantivo Singular):

Variable / i	1	2	3	4	5	6
$\beta_{SusSing,i}$	0.8					-1.3
$f_i(SusSing, x)$	1	0	0	0	0	1
$\beta_{VrbBase,i}$		0.8		0.01	1	
$f_i(VrbBase, x)$	0	1	0	1	1	0

Similarmente al algoritmo anterior, se realizan los cálculos para los términos de 3.10:

- $exp(\sum_i \beta_{SusSing,i} \cdot f_i(SusSing, x)) = exp(\sum_{i=1,6} \beta_{SusSing,i}) = e^{0,8+(-1,3)}$
- $exp(\sum_i \beta_{VrbBase,i} \cdot f_i(VrbBase, x)) = exp(\sum_{i=2,4,5} \beta_{VrbBase,i}) = e^{0,8+0,01+0,1}$
- $Z(x) = e^{0,8+(-1,3)} + e^{0,8+0,01+0,1}$

Finalmente, las probabilidades buscadas se calculan de esta manera:

- $P_{ME}(C = SusSing|x) = \frac{e^{0,8+(-1,3)}}{e^{0,8+(-1,3)}+e^{0,8+0,01+0,1}} \approx 0{,}197$
- $P_{ME}(C = VrbBase|x) = \frac{e^{0,8+0,01+0,1}}{e^{0,8+(-1,3)}+e^{0,8+0,01+0,1}} \approx 0{,}803$

3.4. Herramienta MALLET

MALLET, contracción de "MAchine Learning for LanguagE Toolkit", es un paquete de software de código abierto implementado en el lenguaje de programación Java, que incluye librerías para el procesamiento estadístico de lenguajes naturales, clasificación de documentos, clusterización, modelado de tópicos, extracción de información, y otras técnicas de machine learning aplicadas a texto.

Esta herramienta fue desarrollada principalmente por Andrew McCallum, investigador de la Universidad de Massachusetts (UMass), con la asistencia de estudiantes de posgrado y profesores, tanto de esta Universidad como de la de Pennsylvania (UPenn). Contiene herramientas sofisticadas para la clasificación de documentos, como por ejemplo rutinas eficientes para convertir texto a características (features), una amplia variedad de algoritmos (incluyendo Naïve Bayes, Máxima Entropía y Árboles de Decisión, entre otros), y código para evaluar el rendimiento de un clasificador mediante la utilización de diversas métricas de uso común.

Además de la clasificación, MALLET posee herramientas para *tagging de secuencias*, y para aplicaciones como *named entity recognition* (NER) en un texto, utilizando algoritmos como Modelos de Markov Ocultos, Modelos de Máxima Entropía, y los Campos Aleatorios Condicionales (CRF); estos métodos se implementan en un sistema extensible para autómatas de estado finito. Por otro lado, para analizar grandes colecciones de texto

sin etiquetar existen los llamados *modelos tópicos*, los cuales "descubren"las categorias que ocurren en un conjunto de documentos. Para esto las herramientas de MALLET contienen implementaciones eficientes y basadas en muestreo de Latent Dirichlet Allocation (LDA), Pachinko Allocation, y LDA Jerárquico.

Muchos de los algoritmos en MALLET dependen de la optimización numérica, por lo que se incluye, entre otros métodos, una eficiente implementación de BFGS (Broyden–Fletcher–Goldfarb–Shanno) de Memoria Limitada, el cual es un método iterativo para resolver problemas de optimización no lineal sin restricciones.

En adición a las sofisticadas aplicaciones de Machine Learning, MALLET incluye rutinas para transformar documentos de texto en representaciones numéricas que se pueden procesar eficientemente. Este proceso se implementa a través de un sistema flexible de tubos o *pipes*, el cual se encarga de diversas tareas como el *tokenizing* de cadenas (división de las mismas en elementos o tokens), la eliminación de *stopwords* (palabras superfluas), y finalmente la conversión de secuencias en vectores de conteo.

Si bien MALLET provee herramientas de importación de datos por línea de comandos, al requerir en nuestro caso mayor flexibilidad dado que el sistema desarrollado debe integrarse a las mismas, se hace conveniente hacer uso de su API de importación. Esta API se basa en dos tipos de clases: datos, que estan en forma de listas de instancias, y clases que operan con datos, iteradores y tubos o *pipes*.

Una instancia de MALLET consiste de 4 campos, que pueden ser de cualquier tipo de datos:

- ID: Contiene un valor que identifica la instancia, usualmente con un texto.
- Etiqueta: Se usa principalmente en aplicaciones de clasificación. Suele ser un valor dentro de un conjunto finito de etiquetas, llamado *LabelAlphabet*.
- Datos: Generalmente un *FeatureVector* (pares de valores de features desordenados) o un *FeatureSequence* (lista de features ordenados, como una secuencia de palabras).
- Fuente: Representa el estado original de la instancia, suele ser un archivo o texto que contiene el texto sin preprocesar; aunque frecuentemente esta vacío.

Las instancias se suelen guardar como listas de instancias (*InstanceList*), el cual es el formato estándar de MALLET. Al momento de importar datos, lo usual es pasar las instancias a través de una serie de pasos de procesamiento, que se representan con la interfaz de pipes. Asi, un *pipeline* típico de importación comienza con un iterador, que varía entre *FileIterator* (para datos de un archivo por instancia) y *CsvIterator* (para datos de una línea por archivo), el cual se pasa por un objeto *SerialPipes* que realiza una secuencia de pipes.

4

Clasificador automático

Con el objetivo de clasificar un conjunto de papers en sus categorías correspondientes de manera automática, se ha desarrollado una aplicación en el lenguaje de programación Python, en la cual se hace uso de la herramienta MALLET para que, dado un conjunto de papers y una configuración de parámetros específica, se intente predecir la categoría a la que dichos papers pertenecen.

4.1. Métodos de clasificación jerárquica

Para encarar el problema de clasificar documentos en categorías que respetan una organización jerárquica, existen dos posibles enfoques: global y local. A continuación se explicará con un mayor detalle en qué consiste cada uno de estos métodos, tomando como referencia el sistema de categorías de la ACM.

4.1.1. Método global

Este enfoque consiste en transformar dicha estructura jerárquica en una plana, de manera que todas las categorías pertenezcan a un único nivel. De esta forma, las subcategorías 0,1,2 pertenecientes a las categorías generales A,B,..,K aparecen como categorías independientes A.0,A.1,A.2, ..., B.0,B.1,B.2, ..., K.0,K.1,K.2 (ver figura 4.1), y podemos utilizar un único clasificador para entrenarse en estas categorías. La principal ventaja de este enfoque es que podemos utilizar cualquier clasificador que acepte la tarea de aprender diferentes categorías.

Sin embargo, al no hacer uso de la jerarquía esto resulta en una pérdida de información respecto a las relaciones de pertenencia que esta implica, y por otro lado se obtienen buenos resultados mayormente cuando la cantidad de categorías es relativamente baja.

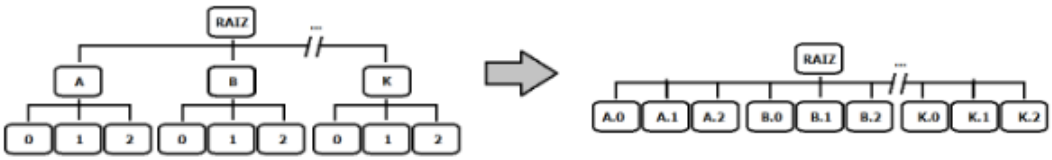

Fig. 4.1: Transformación de una estructura jerárquica a una estructura plana.

Dado que utilizaremos los algoritmos de Naive Bayes y Máxima Entropía, que trabajan con estructuras planas y al aplicarlos en forma directa no se producen resultados de gran precisión (ver sección de resultados), es necesario emplear otro enfoque para no perder la información de la jerarquía, el cual veremos a continuación en detalle.

4.1.2. Método local

En este método, en cambio, se realiza la descomposición del problema de clasificación en subproblemas más pequeños, cuyas soluciones luego se combinan para resolver el problema principal. Para entenderlo mejor veremos un ejemplo concreto para la clasificación de papers de categorías A y B:

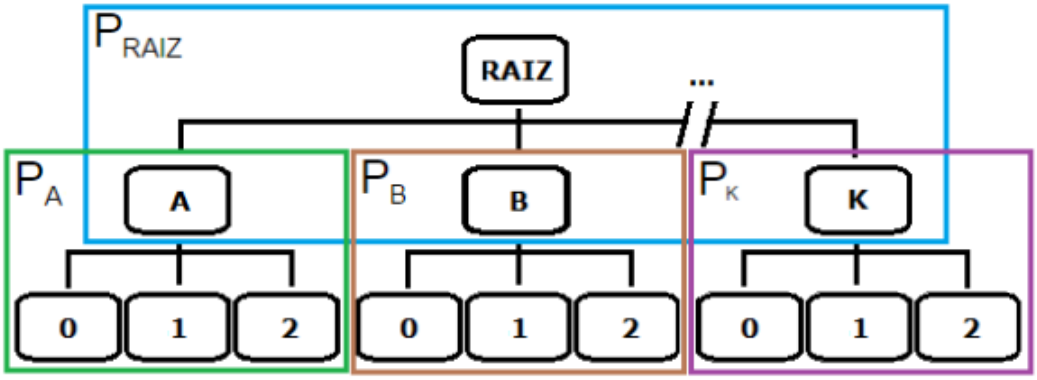

Fig. 4.2: Descomposicion del problema en los subproblemas $P_{RAIZ}, P_A, ..., P_K$.

Como se ve en la figura 4.2, primero es necesario generar un subclasificador para el nodo raíz (resuelve el problema P_{RAIZ}), el cual determina cual de las categorías del primer nivel $\{A, B, .., K\}$ es la más relevante para el documento.

Asumiendo que la categoría resultante fue A, se repite el procedimiento anterior, pero ahora para determinar cuál de las categorías hijas del segundo nivel es la correspondiente (resuelve el problema P_A), mientras que para el caso que la resultante haya sido B, se resuelve de manera similar el problema P_B. Este mecanismo se repite de idéntica forma

hasta llegar a alguna de las "hojas", en cuyo caso al reconstruir el camino realizado se puede determinar la categoría a la que pertenece un paper dado. Ejemplo: Si $P_{RAIZ} \rightarrow A$ y $P_A \rightarrow 1$, la categoría final es A.1.

Este método presenta ventajas respecto al anterior, no solo por tener en cuenta la estructura jerárquica, sino porque además es menos costoso computacionalmente construir nodos subclasificadores en lugar de sólo uno que concentre todo el conocimiento; en particular, si ingresa un nuevo paper al conjunto de entrenamiento, basta con recalcular sólo los nodos de la rama a la que pertenece la categoría asociada, sin la necesidad de tener que actualizar los nodos del resto del árbol.

4.1.3. Clasificadores en cascada

Para realizar una comparación de la performance en los métodos previamente mencionados, en el sistema implementado se utiliza un mecanismo de clasificación en *cascada*: es decir, se construye un árbol de nodos subclasificadores donde cada uno resuelve el problema de determinar un nivel particular, para luego enviar la respuesta al nivel inmediato siguiente. Esto a su vez se evalúa partiendo desde cada uno de los niveles de la jerarquía hasta el último posible. Para definirlo formalmente, dado un nivel inicial **i** y uno final **f**, el clasificador en cascada asociado se notará como $C_{i \rightarrow f}$.

Si trasladamos esta definición al ejemplo anterior exhibido en el método local, puede interpretarse como un caso particular de un clasificador $C_{1 \rightarrow 2}$, donde la primera etapa de la cascada determina que el primer nivel es A, y la segunda determina que el siguiente es *1*. De manera que lógicamente la categoría a la que evalua este método será *A.1*.

Cada uno de estos clasificadores, por su parte, no sólo debe evaluar los nodos del árbol que determinan cada nivel parcial de la jerarquía, sino también principalmente el correspondiente al punto de entrada del mismo árbol: este es el de la categoría ficticia *RAIZ*. Dichos nodos son los que requieren mayor complejidad para ser construidos, ya que deben recibir todos los papers conocidos en su etapa de entrenamiento, de manera de poder determinar el nivel inicial de un paper desconocido que se desee clasificar. Por otra parte, el resto de los nodos subclasificadores sólo requieren ser entrenados con el subconjunto de papers conocidos que concierne a dicha categoría, ya que su dominio se encuentra restringido únicamente a su rama de jerarquía.

Dado que las taxonomías presentan una cantidad considerable de niveles de clasificación, para el caso del presente trabajo se ha decidido limitar el estudio a los primeros 3 niveles de dichas taxonomías, por lo tanto los clasificadores en cascada que fueron evaluados corresponden a $C_{1 \rightarrow 3}$, $C_{2 \rightarrow 3}$ y $C_{3 \rightarrow 3}$, donde el último caso, al concentrar toda la evaluación en un único nodo de nivel 3, corresponde precisamente al método global.

4.2. Obtención de los documentos

A fin de utilizar las taxonomías de las últimas versiones de la ACM, es decir tanto la de 1998 como la de 2012, se realizó un proceso de *web crawling* del sitio de la ACM DL, de manera similar al realizado en el trabajo de Santos [San09] . Este proceso consiste en ejecutar un script que inspecciona y recolecta información de dicho sitio para obtener una colección de tamaño considerable de artículos, la cual permitirá armar los datasets de entrenamiento y testing para el clasificador. Estos se encuentran almacenados en formato *shelve*, que es estructuralmente similar a un diccionario.y se utiliza en Python para serializar objetos.

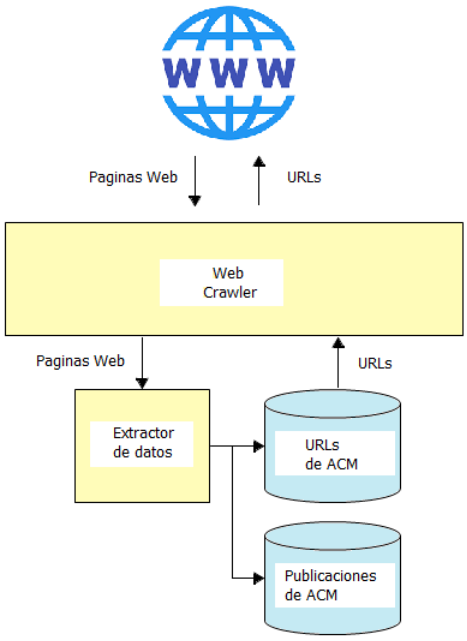

Fig. 4.3: Esquema de etapas del proceso del web crawling

4.3. Etapas del clasificador

El funcionamiento del clasificador implementado se puede dividir en 5 etapas fundamentales, para las cuales se ha generado una estructura de carpetas que contendrá la información a la que hará uso cada una de las mismas. Se pueden distinguir dos grandes fases, que son la de entrenamiento y la de testeo, donde en la primera se genera un dataset a partir de la información obtenida de los papers conocidos, que se usa para entrenar los clasificadores; por otro lado, en la segunda se realiza la clasificación propiamente dicha para un conjunto de papers desconocidos.

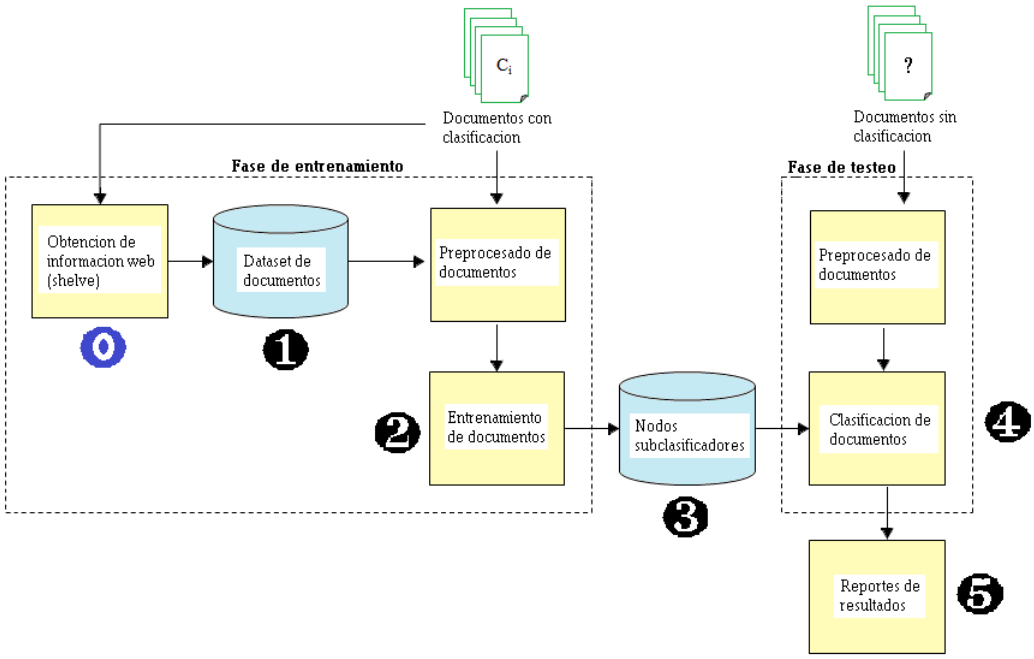

Fig. 4.4: Esquema de etapas del clasificador automático

En cuanto a la estructura de carpetas, esta se compone de la siguiente manera:

- `00_input` : contiene los papers y los árboles de categorías que fueron extraídos de la ACM.
- `01_dataset` : contiene los datasets generados a partir de los papers (ver sección 4.3.1).
- `02_training` : contiene la estructura de entrenamiento para MALLET (ver sección 4.3.2).
- `03_clasif` : contiene los clasificadores para cada categoría y nivel específicos (ver sección 4.3.3).
- `04_testing` : contiene la estructura de testeo para MALLET (ver sección 4.3.1).
- `05_result` : contiene los resultados de clasificación para cada algoritmo (ver sección 4.3.5.

4.3.1. Armado del Dataset inicial

En primer lugar se procesan todos los archivos shelve en la carpeta `00_input` , los cuales fueron obtenidos mediante el método de crawling mencionado anteriormente para

cada categoría en particular, para reunirlos íntegramente en un único dataset en formato CSV, lo que permitirá facilitar los procesos de lectura posteriores. El dataset se guarda en `01_dataset/dataset.csv` .

Por otro lado, se han generado archivos que contienen los árboles completos de categorías en `00_input/ccs_1998.csv` y `00_input/ccs_2012.csv` , los cuales se utilizarán más adelante para optimizar el tiempo de búsqueda de las mismas.

Una vez generado el dataset, el mismo pasa por una etapa de preprocesamiento, en la cual se normalizan los datos y se eliminan las características irrelevantes de los mismos, a fin de conservar únicamente los términos que serán la entrada de los métodos de clasificación en las siguientes secciones. Dicho preprocesamiento emplea las siguientes técnicas:

- Normalizado de términos: consiste en el pasaje de todo el texto a minúsculas, y luego la conversión de los caracteres del mismo a la codificación *ASCII* estándar.
- Segmentación (o *tokenización*): se divide el texto en unidades simples (llamadas *tokens*), donde todos los caracteres no alfabéticos se toman como separadores.
- Eliminado de *stopwords*: se quitan del texto las palabras superfluas (como artículos y preposiciones), tomando como base una lista de palabras comunes del idioma inglés.
- *Stemming*: se reducen las palabras a su raíz quitándoles todos los posibles sufijos, mediante el algoritmo de Porter-Stemmer.

Por último, dicho dataset se guarda en `01_dataset/dataset_pre_<campos>.csv` , donde *campos* corresponde a la combinacion de atributos de los papers que considera la configuracion elegida. Los campos disponibles son: título, abstract, keywords y journal.

4.3.2. Generación de Estructura para Entrenamiento

Aquí es donde se produce la información de entrada que recibirán las librerías de la herramienta MALLET. Como la misma representa los documentos en listas de instancias (ver ejemplo en figura 4.5), se puede optar por dos métodos distintos: uno considera una instancia por cada archivo en una carpeta dada, y el otro, una por cada línea de un archivo dado. Dado que en esta implementación se usará la información de cada archivo para diferentes configuraciones, se empleará el último método mencionado, donde cada línea de un archivo de entrada debe respetar el formato `[id] [categoria] [datos]` . A continuacion veremos un ejemplo:

Listing 4.1: training_1998_ta_1_RAIZ.itr

```
1189647.1189651 K global network organ futur inform system (...)
642748.642765 D declar specif model design manag softwar (...)
1158825.1159064 B robust on-chip commun high-perform comput (...)
```

Fig. 4.5: Lista de instancias

De esta manera, para cada nivel de jerarquía se genera una estructura de archivos tanto para entrenamiento como para testeo, donde a cada paper del dataset se le asignará a una ubicación específica de acuerdo a la categoría que el mismo posee. En este caso, la estructura se guarda en la carpeta `02_training` .

Si se toma por ejemplo el paper "The global network organization of the future", el cual según la ACM tiene id 1189647.1189651 y categoría K.6.1 ("Project and People Management"), y una configuración que usa la taxonomía 1998, el trainer Naive Bayes, y el título y abstract como las características a considerar, los archivos donde se almacenará este paper en la estructura de entrenamiento serán:

Nivel	Nodo(s)	Destino
1	RAIZ	`training_1998_ta_1_RAIZ.itr` (id=1189647.1189651, cat=K)
2	K	`training_1998_ta_2_K.itr` (id=1189647.1189651, cat=K.6)
3	K.6	`training_1998_ta_3_K.6.itr` (id=1189647.1189651, cat=K.6.1)

Fig. 4.6: Tabla de archivos de training por nivel

4.3.3. Generación de Subclasificadores

En esta etapa se utiliza la herramienta MALLET, primero para importar los datos de los papers a un formato binario que ésta entiende, y luego para generar los subclasificadores que permitirán predecir cada nivel de la categoría en forma individual. Además para cada nivel se generará un subclasificador extra con la categoría ficticia RAÍZ, que serán los puntos de entrada para comenzar a clasificar a partir de dichos niveles.

Continuando con la configuración mencionada en el paso anterior, se generarán los siguientes nodos subclasificadores para las diferentes alternativas de testeo en la carpeta `03_clasif` :

Nivel	Nodo(s)			Subclasificador
1	RAIZ			clasif_1998_ta_1_RAIZ_NaiveBayes.cla
2	A	...	K	clasif_1998_ta_2_A_NaiveBayes.cla ... clasif_1998_ta_2_K_NaiveBayes.cla
3	A.0	...	K.8	clasif_1998_ta_3_A.0_NaiveBayes.cla ... clasif_1998_ta_3_K.8_NaiveBayes.cla

Fig. 4.7: Tabla de nodos subclasificadores por nivel

4.3.4. Generación de Estructura para Testeo

Finalmente se llega a la etapa donde se realiza el proceso de testing, en el cual se clasifica un conjunto dado de papers mediante la generación de archivos en forma secuencial, donde tal como se mencionó en la sección de clasificación jerárquica, en cada iteración se obtienen las categorías parciales partiendo del primer nivel, para luego procesar los resultados sucesivamente hasta llegar al último posible. En este caso, la estructura se guarda en la carpeta `04_testing` .

A continuación veremos con mayor detalle el proceso descrito en el párrafo anterior, enfocándonos en particular en el mismo paper (1189647.1189651), para el caso jerárquico con nivel RAIZ 1. Supongamos que se parte de un archivo de entrada que contiene 3 papers, incluido el del ejemplo. Dado que es el punto inicial, debemos utilizar el clasificador de la categoría simbólica RAÍZ para el primer nivel, el cual se encuentra en el archivo `clasif_1998_ta_1_RAIZ_NaiveBayes` . Se muestran entonces los archivos de entrada y salida respectivamente:

Listing 4.2: testing_1998_ta_0_RAIZ_NaiveBayes_IN.its

```
1189647.1189651 global network organ futur inform system (...)
642748.642765 declar specif model design manag softwar (...)
1158825.1159064 robust on-chip commun high-perform comput (...)
```

Listing 4.3: testing_1998_ta_1_RAIZ_NaiveBayes_OUT.its

```
1189647.1189651  B 7.28E-70  C 3.45E-39  K 0.99      J 2.86E-42  (...)
642748.642765    B 9.30E-13  C 2.67E-9   K 1.41E-12  J 5.40E-13  (...)
1158825.1159064  B 0.84      C 0.15      K 2.69E-11  J 1.27E-11  (...)
```

En el archivo anterior se puede ver como MALLET devuelve para cada paper, todas las categorías posibles junto con su probabilidad asociada, de manera que para determinar el primer nivel se elige la que tiene el valor más alto, el cual para el caso del paper de ejemplo será K.

El proceso es similar para las siguientes categorías, eligiendo en cada caso el subclasificador adecuado para cada iteración. Continuando con este ejemplo en particular,

se generará un archivo de entrada que contendrá tanto a este paper como a todos los que hayan caído en esa categoría. Luego, repitiendo el procedimiento anterior, se elige el clasificador `clasif_1998_ta_2_K_NaiveBayes` para obtener la salida para el segundo nivel:

Listing 4.4: testing_1998_ta_2_K_NaiveBayes_IN.its

```
1189647.1189651 global network organ futur inform system (...)
1599503.1599554 design distribut scaffold model complex (...)
```

Listing 4.5: testing_1998_ta_2_K_NaiveBayes_OUT.its

```
1189647.1189651  K.6 0.99      K.3 5.02E-31  K.4 2.88E-7  (...)
1599503.1599554  K.6 2.22E-21  K.3 0.96      K.4 1.86E-13 (...)
```

Como se puede ver la categoría más probable es K.6, por lo cual pasando a la última iteración usaremos el clasificador `clasif_1998_ta_3_K.6_NaiveBayes` , para finalmente obtener los archivos correspondientes de entrada y salida:

Listing 4.6: testing_1998_ta_3_K.6_NaiveBayes_IN.its

```
1189647.1189651 global network organ futur inform system (...)
```

Listing 4.7: testing_1998_ta_3_K.6_NaiveBayes_OUT.its

```
1189647.1189651  K.6.5 2.90E-17  K.6.1 1.0  K.6.3 9.0E-39  (...)
```

Y así se llega a la categoría final para este paper, la cual efectivamente es K.6.1, que coincide con la que tenía originalmente.

El procedimiento descrito arriba se repite de manera similar para todas las categorías, hasta llegar a recorrer todos los nodos del árbol. Luego de reconstruir las categorías en que ha sido clasificado cada paper, éstas se guardan en un archivo, que contiene para cada uno de ellos la categoría original, seguidas de los resultados con las predicciones para cada nivel RAIZ, del primero al tercero.

Listing 4.8: testing_1998_ta_0_RAIZ_NaiveBayes_OUT.its

```
1189647.1189651 K.6.1 ,K.6.1 ,K.1.1
1599503.1599554 K.3.1 ,K.3.1 ,K.4.2
(...)
```

4.3.5. Generación de Resultados

En la etapa final del clasificador se genera un reporte que contiene toda la información y resultados obtenidos en las etapas anteriores, los cuales permiten evaluar la performance conseguida por el procedimiento que realiza la aplicación con una configuración de parámetros dada.

Para dicho reporte, además, se calcula la matriz de confusión para las predicciones realizadas, junto con diferentes métricas de precisión y exhaustividad, que son empleadas a menudo en todo tipo de sistemas de clasificación n-aria. La estructura de un reporte se divide en 3 secciones: CLASIF, DATOS y STATS. A continuación se verá un ejemplo de salida de este reporte, para cada una de estas secciones.

La primera sección (CLASIF) muestra la parametrización de cada uno de los subclasificadores generados por MALLET, con la cantidad de papers de entrenamiento y validación (trap y valp), junto con la medida de accuracy propia de estos (tram) para cada categoría en particular, tomando en cuenta únicamente sus propios datos.

Listing 4.9: result_1998_tkj_MaxEnt.out (fragmento CLASIF)

```
     |err  cant  trap  utrp  valp  tstp  tram  tstm
-----+-------------------------------------------------
RAIZ |.    411   0.9   .     0.1   .     1.0   .
E.0  |.    0     0.9   .     0.1   .     .     .
E.1  |.    24    0.9   .     0.1   .     1.0   .
E.2  |.    2     0.9   .     0.1   .     1.0   .
E.3  |.    129   0.9   .     0.1   .     1.0   .
E.4  |.    248   0.9   .     0.1   .     1.0   .
E.5  |.    4     0.9   .     0.1   .     1.0   .
E.m  |.    4     0.9   .     0.1   .     1.0   .
```

La segunda sección (DATOS) muestra las tablas de predicciones y métricas de clasificación para cada nivel, que permiten evaluar la performance de los algoritmos para cada categoría en particular.

La primera es una matriz de confusión, la cual es una tabla cuyas filas son las categorías originales, y cuyas columnas son las categorías que predice el clasificador (por ejemplo, se observa que para E.1 se clasificaron 6 papers correctamente, mientras que 1 se clasificó como E.3 y 5 como E.4).

Listing 4.10: result_1998_tkj_MaxEnt.out (fragmento DATOS - PREDICCIONES)

```
     |E.0  E.1  E.2  E.3  E.4  E.5  E.m
-----+-------------------------------------
E.0  |.    .    .    .    .    .    .
E.1  |.    6    .    1    5    .    .
E.2  |.    .    .    .    1    .    .
E.3  |.    .    .    57   5    .    .
E.4  |.    .    .    20   99   .    .
E.5  |.    1    .    .    .    1    .
E.m  |.    .    .    .    2    .    .
```

La segunda es una tabla con varias métricas calculadas a partir de la primera, de las que se destacan PPV (positive predictive value o *precision*) y TPR (true positive rate o *recall*), que miden precisión y exhaustividad respectivamente. Estas se calculan a partir de las 4 primeras, que representan todas las posibles combinaciones de casos: los positivos evaluados correctamente (TP) o incorrectamente (FP), y los negativos evaluados correctamente (TN) o incorrectamente (FN).

Dada una categoría, se pueden distinguir los documentos de un dataset clasificado en dos conjuntos: los *relevantes*, que son los que efectivamente pertenecen a ésta, y los *seleccionados*, que son los que fueron clasificados en ésta. De manera que, en función de éstos, la precisión determina la probabilidad de que un documento sea relevante a la categoría en que fue clasificado (en otras palabras, cuántos documentos seleccionados son relevantes), y el recall indica la probabililidad de que un documento sea clasificado en la categoría relevante para el mismo (es decir, cuántos documentos relevantes fueron seleccionados).

$$PPV = \frac{|\{Seleccionados\} \cap \{Relevantes\}|}{|\{Seleccionados\}|} = \frac{TP}{TP + FP} \tag{4.1}$$

$$TPR = \frac{|\{Seleccionados\} \cap \{Relevantes\}|}{|\{Relevantes\}|} = \frac{TP}{TP + FN} \tag{4.2}$$

Listing 4.11: result_1998_tkj_MaxEnt.out (fragmento DATOS - METRICAS)

	TP	FN	FP	TN	PPV	NPV	TPR	TNR	ACC	F1
E.0	.	.	.	198	.	1.0	.	1.0	1.0	.
E.1	6	6	1	185	0.857	0.969	0.5	0.995	0.965	0.632
E.2	.	1	.	197	.	0.995	.	1.0	0.995	.
E.3	57	5	21	115	0.731	0.958	0.919	0.846	0.869	0.814
E.4	99	20	13	66	0.884	0.767	0.832	0.835	0.833	0.857
E.5	1	1	.	196	1.0	0.995	0.5	1.0	0.995	0.667
E.m	.	2	.	196	.	0.99	.	1.0	0.99	.

Ambos valores se utilizan para calcular la métrica *F1-score*, la cual pondera estos dos valores para cuantificar una media de performance para cada categoría en forma individual. Es un caso particular de la formula genérica de *F-measure* para $\beta = 1$, la cual se computa de la siguiente forma:

$$F_\beta = (1 + \beta^2) \cdot \frac{PPV \cdot TPR}{\beta^2 \cdot PPV + TPR} \tag{4.3}$$

De aquí se desprende que el F1-score es aproximadamente el promedio de ambos valores cuando son cercanos (en otras palabras la media armónica), que para el caso de dos números coincide con la media geométrica al cuadrado dividida por la media aritmética; de esta forma los valores precision y recall tienen peso equivalente.

Finalmente, la tercera seccion (STATS) es la que evalua la efectividad de los clasificadores para cada método de evaluación jerárquica: $C_{1\to3}$, $C_{2\to3}$ y $C_{3\to3}$, descomponiendo las categorías predichas de cada resultado en cada nivel de jerarquía. Para entenderlo con un ejemplo, si para el caso cuyo nivel RAIZ es 1 ($C_{1\to3}$), la categoría original es A.1.2 y la predicha fue A.1.3, entonces se contará un acierto para las dos primeras columnas (Niv 1 y Niv 2), pero no así en la tercera columna (Niv 3).

Listing 4.12: result_1998_tkj_MaxEnt.out (fragmento STATS)

```
                |Niv 1                Niv 2                Niv 3
----------------+-------------------------------------------------------------
C(1->3)         |198/198 (100.0%)  163/198 (82.32%)  163/198 (82.32%)
C(2->3)         |198/198 (100.0%)  162/198 (81.82%)  162/198 (81.82%)
C(3->3)         |198/198 (100.0%)  161/198 (81.31%)  161/198 (81.31%)
```

En este caso puede verse que en el caso donde la raíz inicia en el primer nivel (jerarquía local) tiene una performance levemente superior al caso que inicia en el ultimo nivel (jerarquía global).

Por último, esta operación también genera un reporte visual, en el cual se muestran gráficos de performance para cada nivel, tomando como ejes las categorías parciales y la métrica de F1-score para cada una de las mismas. Esta herramienta fue usada para generar los gráficos que se exponen más adelante en la sección de resultados.

4.4. Interfaz del sistema

El sistema de clasificación desarrollado provee dos tipos de interfaces interactivas: una gráfica y una textual, que pueden ser invocadas desde la línea de comandos mediante el intérprete de Python. La primera, al ser una interfaz visual, otorga al usuario una forma de manipular la configuración de manera sencilla, mientras que la segunda permite ingresar los parámetros de manera directa mediante el ingreso de un archivo de configuración como dato de entrada, lo que facilita las pruebas por lotes. Estos son ejemplos de las diferentes alternativas de ejecución:

- Lanzar modo gráfico:

```
$ python Tesis.py gui
```

- Lanzar modo textual:

```
$ python Tesis.py cmd ./tesis/_cfg_/test.cfg
```

Es importante aclarar que para el correcto funcionamiento de esta aplicación es necesario tener instalado el lenguaje Python (en particular la versión 2.7), así como los paquetes y librerías de los que ésta depende, según el sistema operativo donde se ejecute. Para facilitar esta última tarea, Python provee una herramienta llamada *pip* que se encarga de instalar automáticamente un paquete, dado el nombre que identifica al mismo. Este es un ejemplo de instalación para el paquete *sklearn-pandas*, el cual se utiliza para la manipulación de datasets:

```
$ pip install -U sklearn-pandas
```

4.4.1. Interfaz gráfica

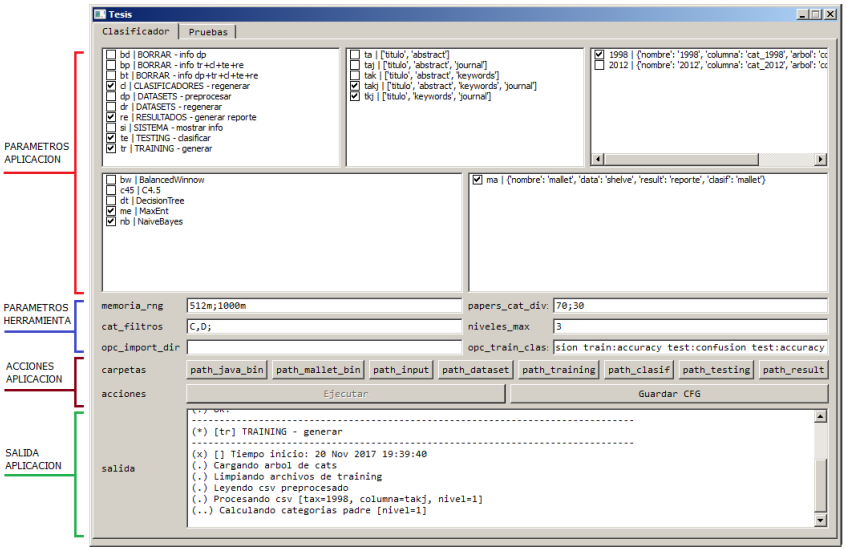

Fig. 4.8: Pantalla principal de la interfaz grafica del clasificador.

A continuación se explicarán en detalle las distintas secciones en que se compone la pantalla principal de la interfaz gráfica, las cuales fueron indicadas en la figura anterior:

- Parametros de la aplicación: Son los que determinan las operaciones y la configuración de los procesos que ejecutará la aplicación. Estos son los **comandos**, que indican cuáles etapas del clasificador se ejecutan; las **columnas**, que indican las combinaciones de campos relevantes a tener en cuenta; las **taxonomías** de la ACM que se usarán para las fases de training y testing; los **trainers** o algoritmos de clasificación; y finalmente las **herramientas** externas que provean métodos para clasificar (esto último fue diseñado para ser extendido).

- Parametros de la herramienta: Son los que utiliza la herramienta de clasificación para su configuración interna. Entre estos se incluyen: el intervalo de **memoria** a reservar para el espacio de *heap* que utiliza la máquina virtual de Java; los **porcentajes de división** de datos en training y testing; las categorías a **filtrar** de forma incluyente o excluyente; la cantidad máxima de **niveles** de profundidad a procesar; asi como también parametros internos que dependen de la herramienta.

- Acciones de la aplicación: Es un conjunto de botones que permiten realizar todas las acciones necesarias del clasificador. Estas incluyen: en la barra superior, accesos rápidos a las carpetas de archivos de cada etapa; y en la barra inferior, los botones de ejecución y de guardado de la configuración actual (para esto último se utiliza el archivo de configuración default, ubicado en `./tesis/_cfg_/global.cfg`).

- Salida de la aplicación: Dado que esta aplicación interactúa con una herramienta externa, para obtener la salida de la misma fue necesario redirigir los canales que

utiliza el sistema por defecto (*STDOUT* y *STDERR*) hacia la propia interfaz del sistema. De esta forma, en esta sección se captura dicha salida para poder visualizarla de forma directa.

4.4.2. Interfaz textual

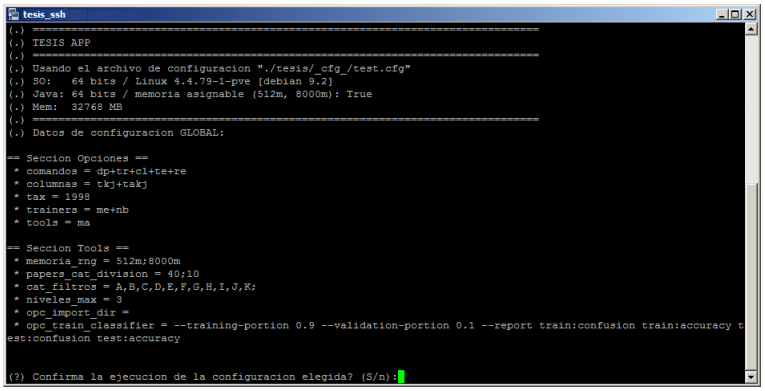

Fig. 4.9: Pantalla principal de la interfaz textual del clasificador.

Por otro lado, y como se mencionó en la sección anterior, el sistema además provee una interfaz en modo texto para cuando sea necesaria su ejecución desde una terminal de consola, como por ejemplo en sistemas donde no se tenga acceso a un entorno gráfico. Esta opción, además, es de gran utilidad para realizar una serie de pruebas en secuencia, facilitando todos los parámetros necesarios directamente por la línea de comandos.

Las opciones que ofrece esta interfaz son idénticas a la anterior, con la diferencia de que en este caso deben ingresarse en un archivo de configuración, el cual contiene toda la información necesaria que utilizará el sistema para una ejecución determinada. A continuación veremos un ejemplo de la estructura que debe tener este tipo de archivos:

Listing 4.13: test.cfg

```
[PathsWindows]
path_java_bin = /ProgramData/Oracle/Java/javapath
path_tool_bin = /PROGRAMS/Proyectos/PYTHON/Eclipse/tesispapers/src/
    tesis/_res_/mallet-2.0.7/bin
path_input = /PROGRAMS/Proyectos/PYTHON/Eclipse/tesispapers_res/00
    _input
path_dataset = /PROGRAMS/Proyectos/PYTHON/Eclipse/tesispapers_res/01
    _dataset
path_training = /PROGRAMS/Proyectos/PYTHON/Eclipse/tesispapers_res/02
    _training
path_clasif = /PROGRAMS/Proyectos/PYTHON/Eclipse/tesispapers_res/03
    _clasif
path_testing = /PROGRAMS/Proyectos/PYTHON/Eclipse/tesispapers_res/04
    _testing
path_result = /PROGRAMS/Proyectos/PYTHON/Eclipse/tesispapers_res/05
    _result
```

```
[PathsLinux]
path_java_bin = /usr/bin
path_tool_bin = /home/ale/Proyectos/PYTHON/Eclipse/tesispapers/src/
    tesis/_res_/mallet-2.0.7/bin
path_input = /home/ale/Proyectos/PYTHON/Eclipse/tesispapers_res/00
    _input
path_dataset = /home/ale/Proyectos/PYTHON/Eclipse/tesispapers_res/01
    _dataset
path_training = /home/ale/Proyectos/PYTHON/Eclipse/tesispapers_res/02
    _training
path_clasif = /home/ale/Proyectos/PYTHON/Eclipse/tesispapers_res/03
    _clasif
path_testing = /home/ale/Proyectos/PYTHON/Eclipse/tesispapers_res/04
    _testing
path_result = /home/ale/Proyectos/PYTHON/Eclipse/tesispapers_res/05
    _result

[Tools]
memoria_rng = 512m;4000m
papers_cat_division = 70;30
cat_filtros = C,D;
niveles_max = 3
opc_import_dir =
opc_train_classifier = --training-portion 0.9 --validation-portion 0.1
    --report train:confusion train:accuracy test:confusion test:
    accuracy

[Opciones]
comandos = te+re
columnas = tkj+takj
tax = 1998
trainers = me+nb
tools = ma
```

Como se ve en el ejemplo de arriba, un archivo de configuración que recibe el clasificador se compone de 4 secciones principales:

- `[PathsWindows]` : aquí se definen todas las rutas relevantes para el caso en que se ejecute en el sistema operativo Windows, tanto a cada etapa del ciasificador como a las herramientas con las que este interactúa, más precisamente Java y MALLET.
- `[PathsLinux]` : similar al anterior, pero en particular para las rutas relevantes en caso de utilizar el sistema operativo Linux.
- `[Tools]` : aquí se configuran todos los parámetros relevantes a la herramienta de clasificación, como el rango de memoria asignada, división de papers de entrenamiento y testing, etc.
- `[Opciones]` : por último se definen las acciones a ejecutar por el clasificador. Corresponde a la seccion de Parámetros de Aplicación de la interfaz gráfica mencionada en el punto anterior.

Para el caso de múltiples parámetros de una misma opción, es posible combinarlos en forma secuencial mediante el uso del operador de suma `+` . Por otro lado, para elegir todas las combinaciones posibles para una opción determinada se puede usar `!` .

Si se ingresan las opciones de ejecución que muestra el archivo de ejemplo, las acciones a realizar serán en cada caso serían las siguientes:

- `comandos = te+re` : Ejecuta los comandos de testing y reporte de resultados
- `columnas = tkj+takj` : Columnas relevantes (t=titulo, a=abstract, k=keywords, j=journal)
- `tax = 1998` : Taxonomías de la ACM a utilizar (en este caso 1998)
- `trainers = me+nb` : Algoritmos de clasificación (donde nb=NaiveBayes y me=MaxEnt)
- `tools = ma` : Herramienta externa (donde ma=MALLET)

5

Resultados

Los resultados que se muestran en esta sección fueron generados mediante el sistema anteriormente mencionado en una computadora que cuenta con las siguientes características de hardware y software:

- Microprocesador Intel® Core™ I5.
- Memoria RAM de 32 GB.
- Sistema operativo Debian Linux 9.2, de 64 bits.

Para generarlos, se configuraron los parámetros del sistema de forma que cada nodo clasificador utilice el 70 % de los papers de su categoría específica para training y el 30 % restante para testing, y al momento de entrenar se reserva el 10 % para validación.

Dado que la cantidad de papers obtenidos del sitio web de la ACM-DL es considerablemente alta (aproximadamente 100000), la herramienta MALLET debe realizar un uso extensivo de la memoria para generar los clasificadores más complejos, en particular los que emplean el método de jerarquía global. Por lo tanto, fue necesario configurar la máquina virtual de Java (JVM) asignándole un máximo de 8 GB para el *heap size*, de manera que el sistema pueda disponer de la cantidad de memoria necesaria para llevar a cabo correctamente esta tarea.

A continuación se presentan los resultados de los experimentos en las figuras 5.1 hasta 5.8, donde se probaron diferentes combinaciones de los parámetros disponibles para comparar y evaluar nuestra implementación.

En todas las figuras, las líneas de colores representan la performance de los clasificadores evaluados, los cuales son $C_{1\to3}$, $C_{2\to3}$ y $C_{3\to3}$, identificados con el verde, azul y rojo respectivamente. En el eje horizontal se encuentran las categorías, y en el eje vertical se encuentran los valores del F1-score calculado para cada una de ellas (ver fórmula en la sección 4.3).

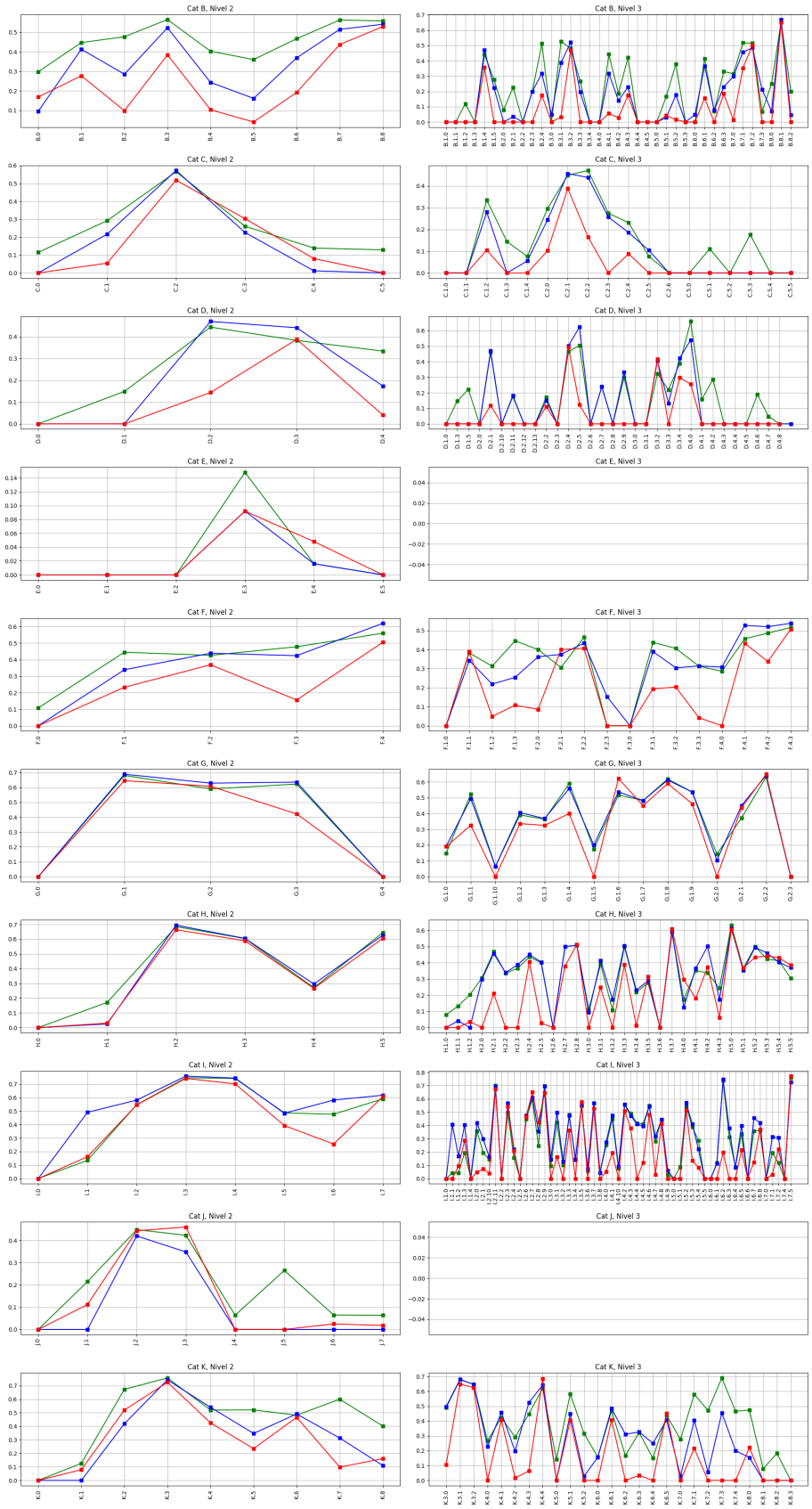

Fig. 5.1: Resultados para Naive Bayes (Tax 1998, Titulo+Keywords+Journal)

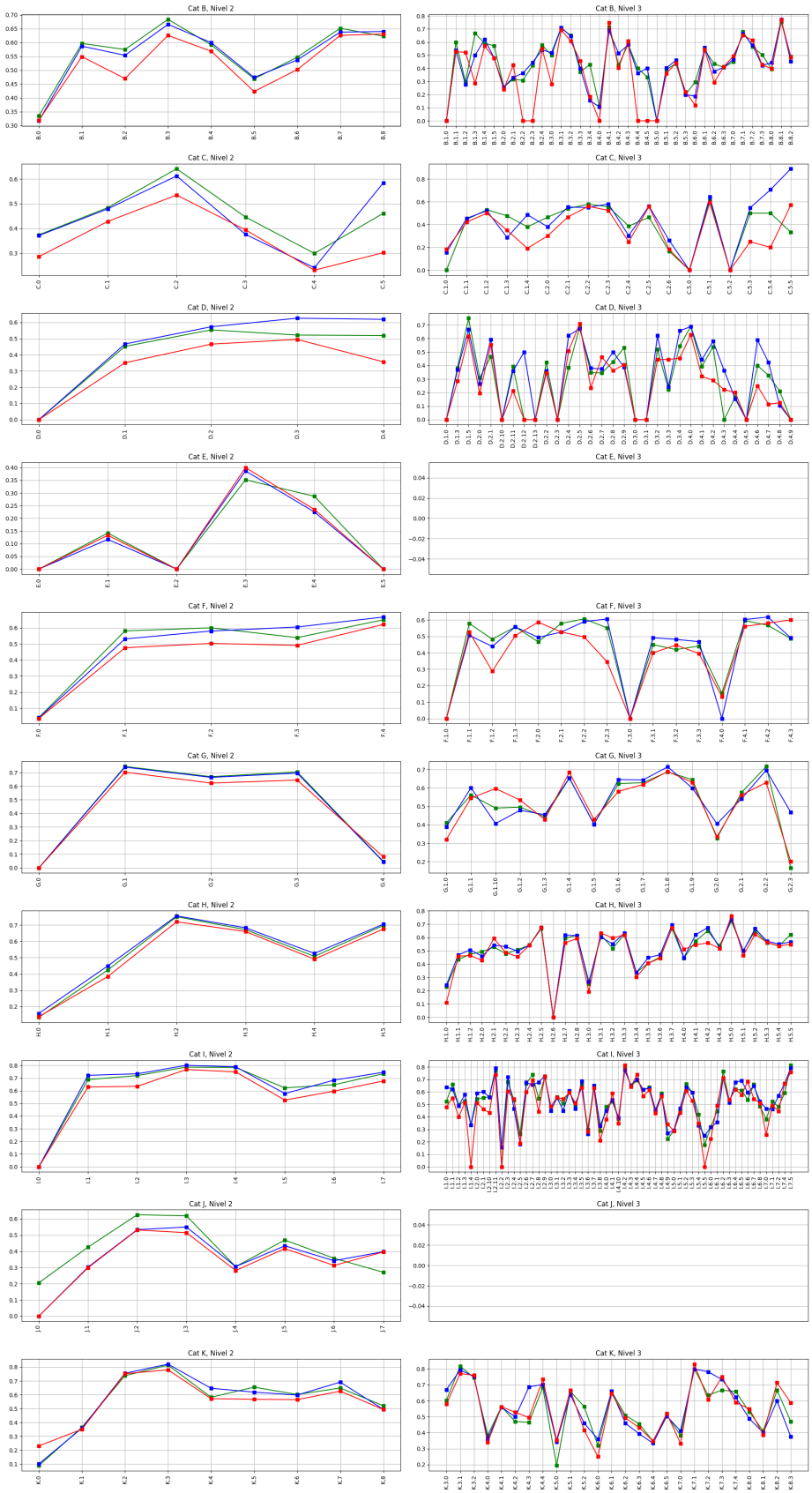

Fig. 5.2: Resultados para Maxima Entropia (Tax 1998, Titulo+Keywords+Journal)

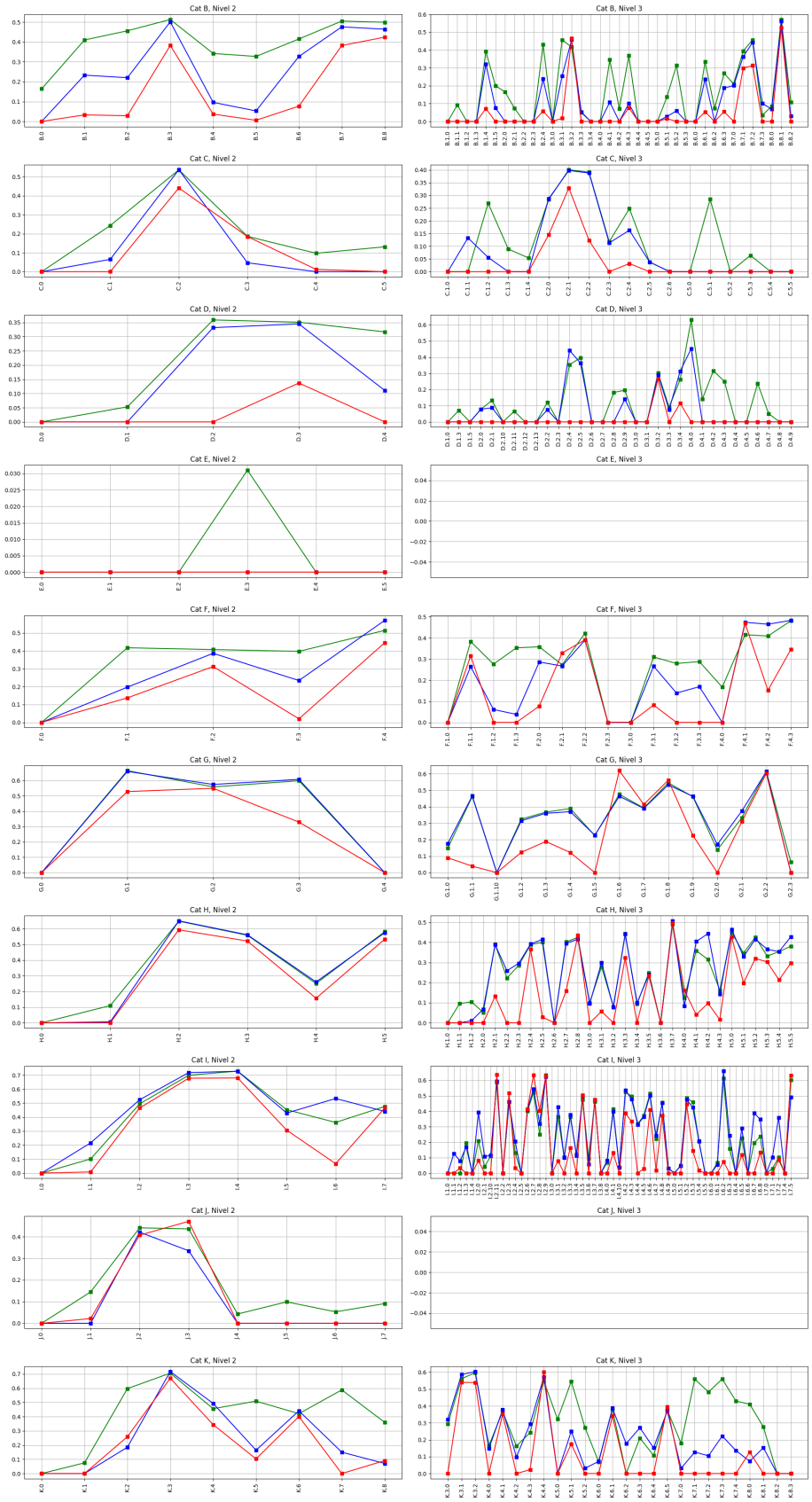

Fig. 5.3: Resultados para Naive Bayes (Tax 1998, Titulo+Keywords+Abstract+Journal)

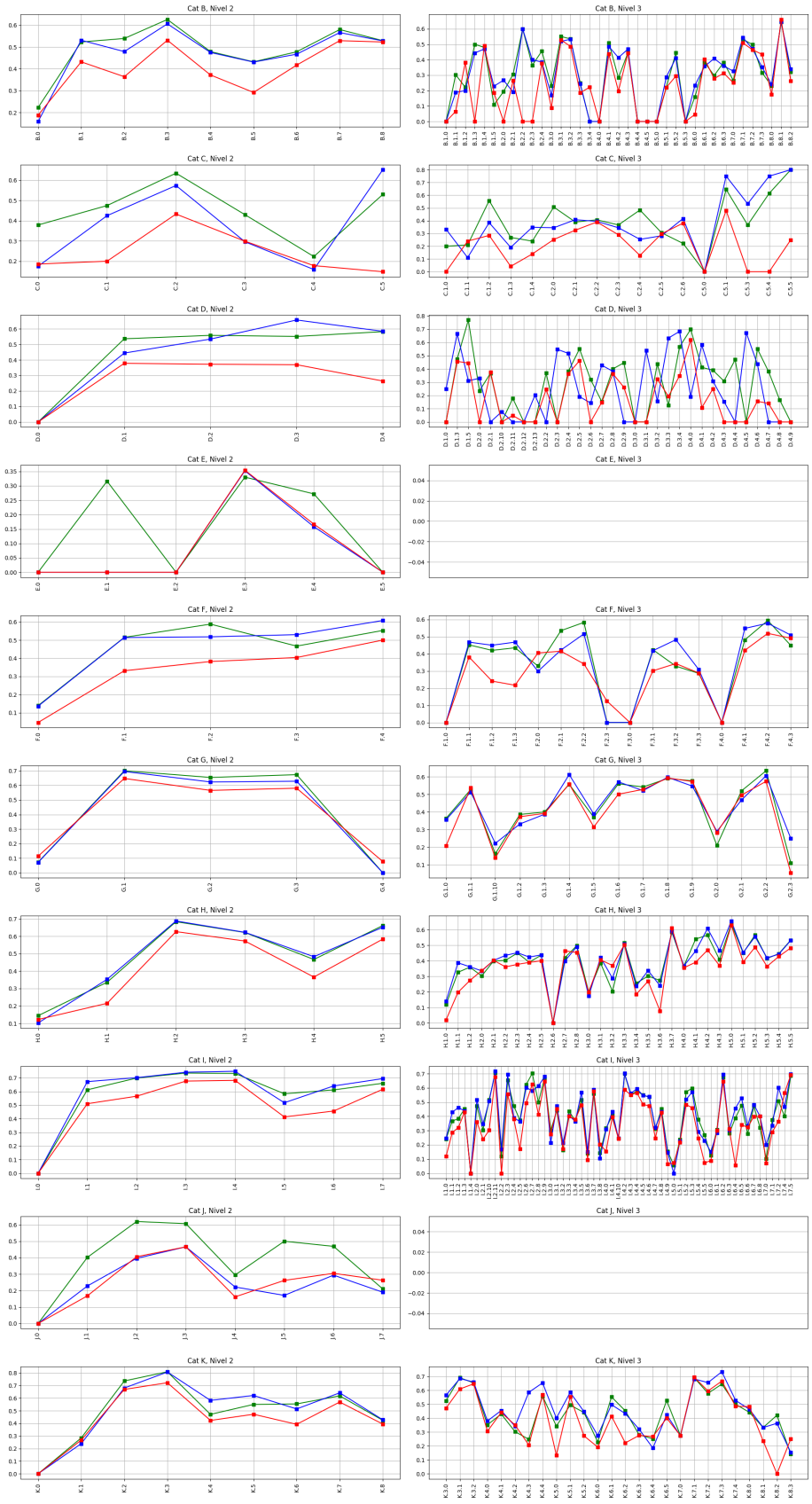

Fig. 5.4: Resultados para Maxima Entropia (Tax 1998, Titulo+Keywords+Abstract+Journal)

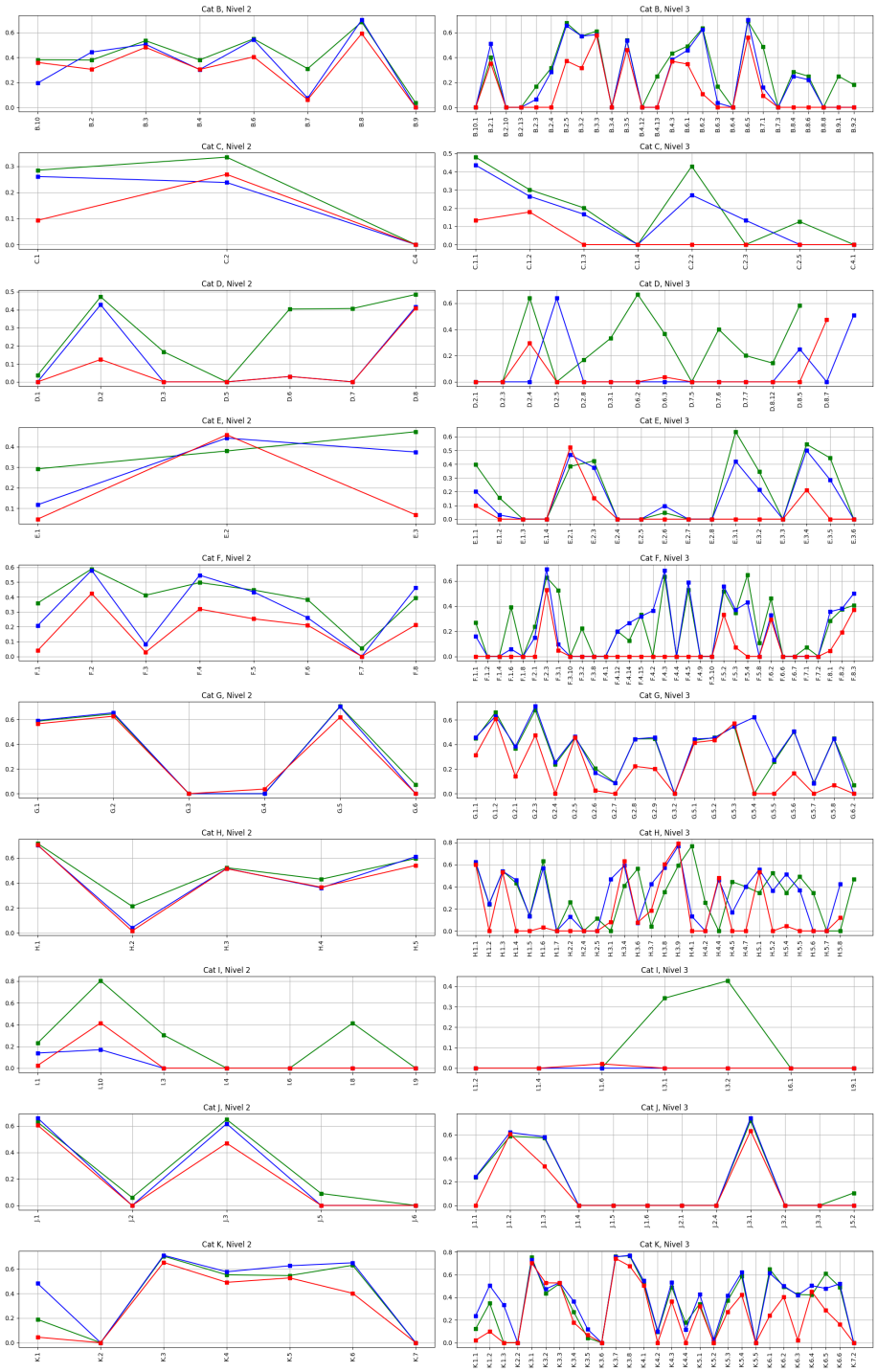

Fig. 5.5: Resultados para Naive Bayes (Tax 2012, Titulo+Keywords+Journal)

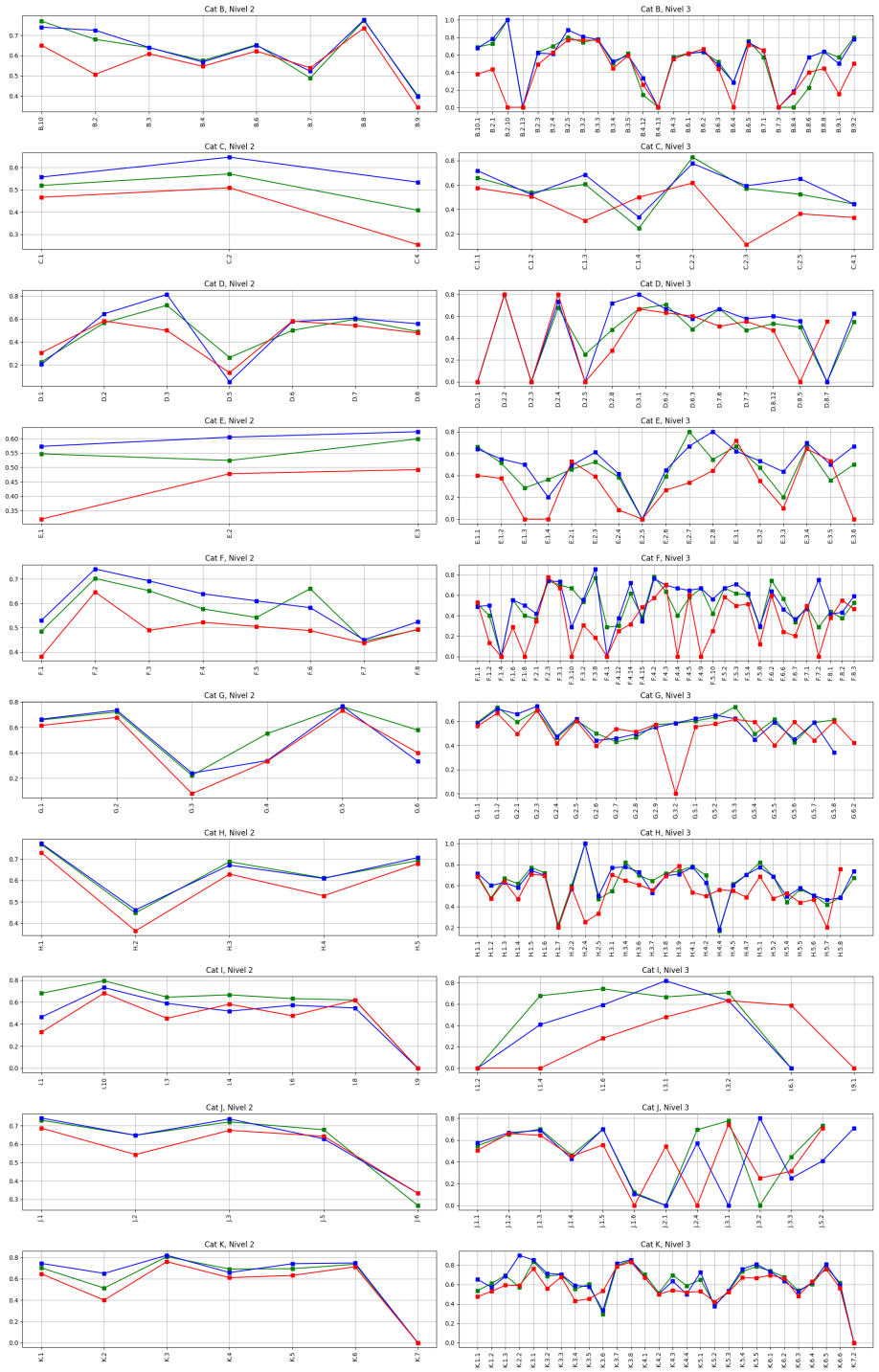

Fig. 5.6: Resultados para Maxima Entropia (Tax 2012, Titulo+Keywords+Journal)

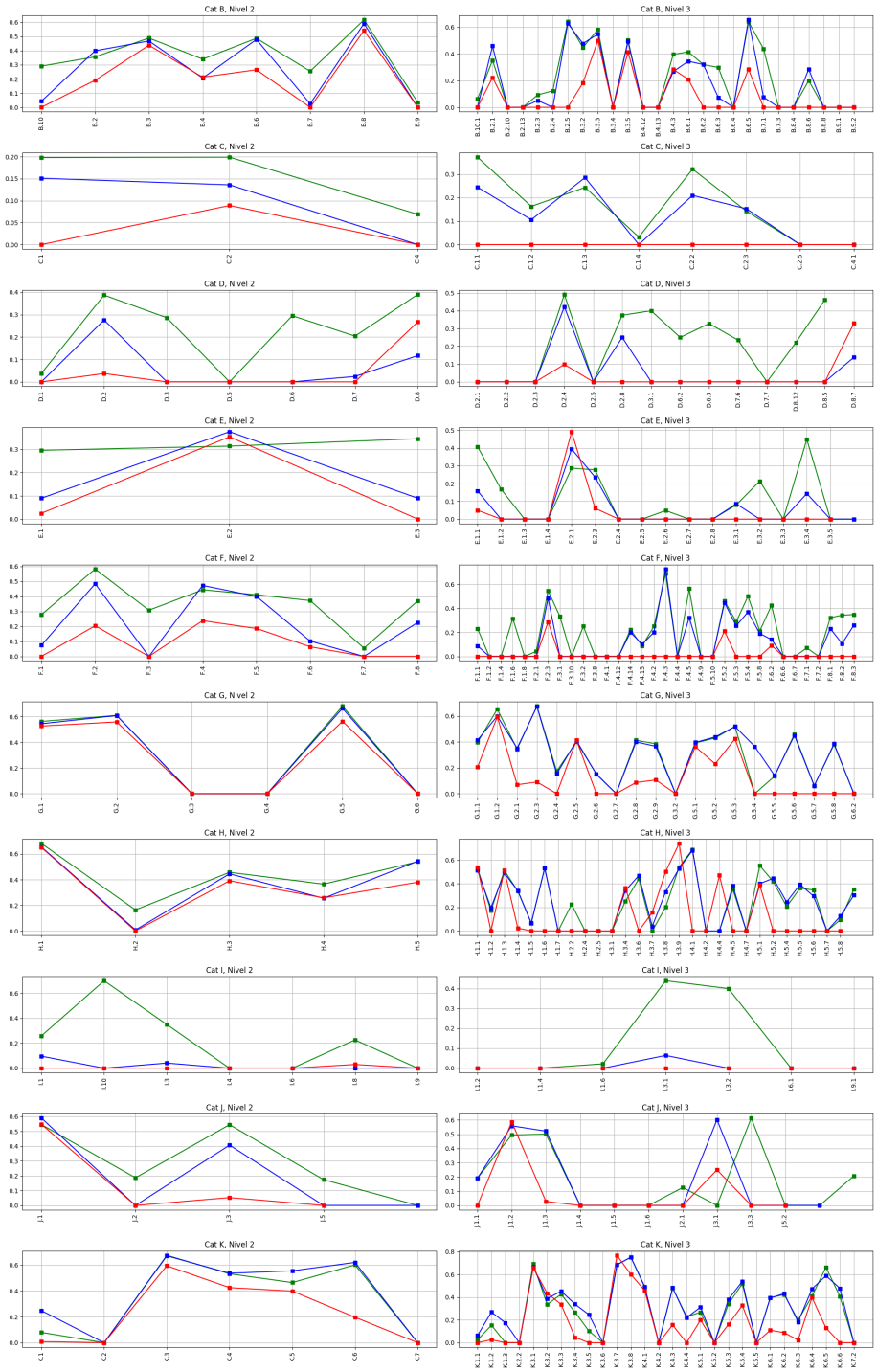

Fig. 5.7: Resultados para Naive Bayes (Tax 2012, Titulo+Keywords+Abstract+Journal)

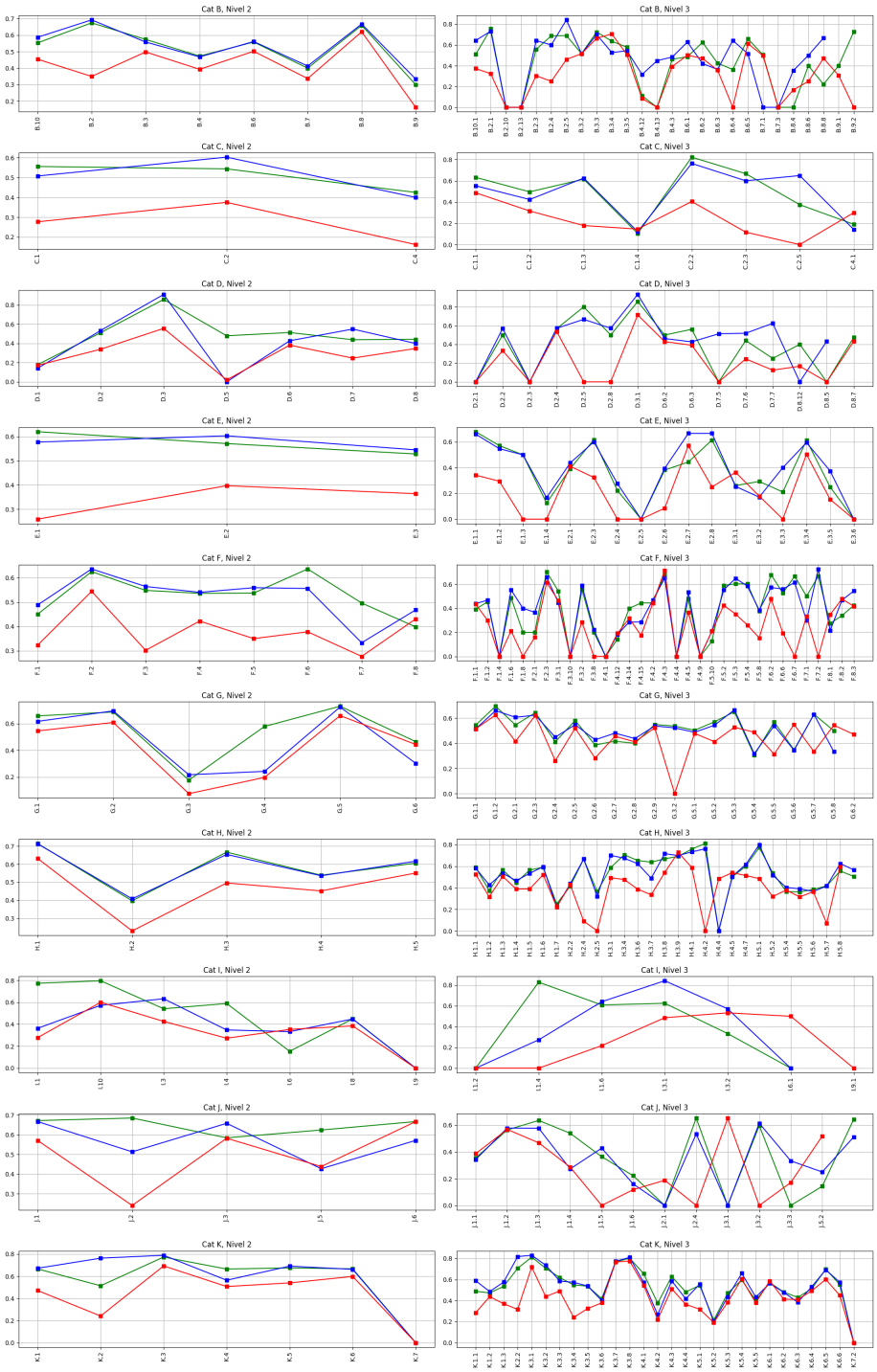

Fig. 5.8: Resultados para Maxima Entropia (Tax 2012, Titulo+Keywords+Abstract+Journal)

Taxonomía	Campos	Algoritmo	Clasificador	Nivel 1	Nivel 2	Nivel 3
1998	takj	MaxEnt	$C_{1\to3}$	77.06 %	60.87 %	45.4 %
1998	takj	MaxEnt	$C_{2\to3}$	73.79 %	59.72 %	43.92 %
1998	takj	MaxEnt	$C_{3\to3}$	67.09 %	50.88 %	38.87 %
1998	takj	NaiveBayes	$C_{1\to3}$	68.7 %	51.28 %	35.02 %
1998	takj	NaiveBayes	$C_{2\to3}$	69.06 %	51.15 %	34.24 %
1998	takj	NaiveBayes	$C_{3\to3}$	62.78 %	41.91 %	27.64 %
1998	tkj	MaxEnt	$C_{1\to3}$	78.63 %	65.28 %	54.07 %
1998	tkj	MaxEnt	$C_{2\to3}$	77.74 %	65.46 %	53.93 %
1998	tkj	MaxEnt	$C_{3\to3}$	73.76 %	60.4 %	50.84 %
1998	tkj	NaiveBayes	$C_{1\to3}$	72.1 %	55.6 %	40.81 %
1998	tkj	NaiveBayes	$C_{2\to3}$	72.6 %	55.75 %	40.49 %
1998	tkj	NaiveBayes	$C_{3\to3}$	67.97 %	49.16 %	34.76 %
2012	takj	MaxEnt	$C_{1\to3}$	78.77 %	61.89 %	49.27 %
2012	takj	MaxEnt	$C_{2\to3}$	75.98 %	61.15 %	48.41 %
2012	takj	MaxEnt	$C_{3\to3}$	68.68 %	50.82 %	41.42 %
2012	takj	NaiveBayes	$C_{1\to3}$	70.56 %	49.98 %	37.0 %
2012	takj	NaiveBayes	$C_{2\to3}$	68.54 %	49.3 %	36.33 %
2012	takj	NaiveBayes	$C_{3\to3}$	63.72 %	40.88 %	30.66 %
2012	tkj	MaxEnt	$C_{1\to3}$	79.81 %	66.56 %	57.53 %
2012	tkj	MaxEnt	$C_{2\to3}$	80.02 %	67.44 %	58.1 %
2012	tkj	MaxEnt	$C_{3\to3}$	75.15 %	61.38 %	53.88 %
2012	tkj	NaiveBayes	$C_{1\to3}$	73.83 %	54.74 %	42.86 %
2012	tkj	NaiveBayes	$C_{2\to3}$	73.05 %	54.95 %	42.86 %
2012	tkj	NaiveBayes	$C_{3\to3}$	69.51 %	48.56 %	38.22 %

Fig. 5.9: Tabla de performance de clasificadores por nivel

Observando las performances en la tabla de arriba, las cuales fueron obtenidas en las ejecuciones con diferentes parametrizaciones (5.9), puede notarse que los clasificadores en cascada $C_{1\to3}$ y $C_{2\to3}$ superan en todos los niveles - ya sea por pequeño o amplio margen - al clasificador de jerarquía global $C_{3\to3}$, siempre utilizando para cada caso el mismo conjunto de entrenamiento y testeo.

Podemos decir que por lo general es esperable que a mayor nivel de detalle tengamos menor porcentaje de aciertos, ya que las características que definen a los niveles más bajos de la jerarquía tienden a poseer un indice alto de ambiguedad y a compartir una gran cantidad de conceptos clave. Sin embargo consideramos que la precisión es más importante en los primeros niveles, ya que es prioritario coincidir con la unidad de conocimiento principal a la que pertenece cualquier documento. Por este motivo concentramos nuestra atención en los niveles 1 y 2.

En cuanto al método algorítmico, también se desprende de la tabla que el algoritmo de Máxima Entropía tiene mayor efectividad que el de Naive Bayes en todas las pruebas, permitiendo así concluir que, si bien ambos métodos son similares en cierta medida, el primero tiene mejores resultados al considerar justamente las dependencias entre las ca-

racterísticas de los documentos, sin hacer ningún tipo de asunción sobre éstas, mediante el maximizado de la informacion extraída en el entrenamiento.

Por otra parte, otro detalle que se puede apreciar es que entre las combinaciones de campos elegidas (titulo, abstract, keywords y journal), las que consiguieron porcentajes más altos fueron las que no tomaron en cuenta el campo de abstract. Con lo cual, lejos de aportar mayor información al algoritmo de clasificación, incluir dicho campo entre las características resulta en un empeoramiento de la performance general, ya que genera una distorsión en el análisis de términos relevantes de los documentos.

Pasando a las taxonomías analizadas (1998 y 2012), se observa que en la más reciente de ambas los resultados son levemente superiores. Esto puede deberse a que la taxonomía de 2012 ha mejorado su organización respecto de la anterior, ya que ha sufrido importantes cambios estructurales que permiten tener una mejor comprensión de las categorías que comprenden las unidades de conocimiento de la Computación, y de esta manera los algoritmos pueden identificar de una manera más precisa la pertenencia de un documento a su categoría original.

Finalmente, algo que pudimos observar es que hay categorías de nivel 2 que tenían una dificultad intrínseca ya que siempre dan malos resultados sin importar que técnica usemos. Estas categorías son las de General y Misceláneos. Este problema se solucionó al fusionar estas 2 subcategorías en cada nivel (es decir, las que finalizan tanto en *.0* como en *.m*) ya que son categorías muy similares, o bien no está claramente definido qué es lo que las caracteriza.

6

Conclusiones y trabajos futuros

La clasificación de textos ha recibido la atención de los investigadores desde mediados de los años 90. Sin embargo, la clasificación jerárquica aún ofrece desafíos interesantes, los cuales dan lugar tanto para nuevas investigaciones como para la optimización de contribuciones que ya se encuentran disponibles.

En este trabajo se ha hecho una revisión general de los problemas de clasificación jerárquica, y un subconjunto de sus posibles soluciones. Se ha diseñado para tal fin una aplicación que permita automatizar el proceso de clasificación de artículos científicos y facilitar la configuración de los parámetros de la misma para poder comunicarse de manera sencilla con herramientas de clasificación externa, utilizando para este caso particular la implementacion en Java de MALLET, la cual posee eficientes metodologías para realizar este tipo de tareas.

Se ha recolectado para esto una colección de papers clasificada de acuerdo al esquema del sitio de la ACM, la cual fue obtenida mediante un proceso de *web crawling*, y posteriormente preprocesada usando un conjunto de técnicas variadas de aplicación general, como es el caso de normalizado, tokenización, stemming y remoción de stopwords, entre otras. Se han evaluado dos métodos de clasificación lineal, Naive Bayes y Máxima Entropía, los cuales fueron parametrizados con diferentes configuraciones (campos relevantes de los papers, porcentajes de training/testing, taxonomías, etc).

Como se puede ver en los resultados, la mayor efectividad de aciertos se obtiene combinando la taxonomía más reciente de las estudiadas (2012), el conjunto de campos de título, keywords y journal (sin tener en cuenta el abstract), y el algoritmo de Máxima Entropía. También se puede ver que a rasgos generales, tanto en el algoritmo de Naive Bayes como el de Máxima Entropía, la efectividad es mayor con el clasificador en cascada de jerarquía local por sobre el de jerarquía global, lo que resalta la importancia de preservar la información de las relaciones jerárquicas entre las categorías dentro de una taxonomía en particular.

También es importante remarcar que el método presentado no sólo tiene un mayor

rendimiento que el estándar, sino que es más eficiente en cuanto a tiempos de ejecución, ya que la construcción de los nodos subclasificadores en todos los casos demanda un orden de complejidad considerablemente menor que concentrar toda la información en un único nodo raíz.

Por otra parte, gracias a la cantidad de experimentos realizados, también se puede concluir que esta combinación no depende de la cantidad de papers ni de la cantidad de términos utilizados; sin embargo se debe destacar que estas conclusiones en principio solo pueden considerarse válidas para la colección de papers extraídos de la librería de la ACM.

En adición al trabajo realizado en esta tesis, se presentan algunas propuestas para trabajos futuros:

- Investigar métodos para recuperarse de errores en un nivel dado, es decir, prevenir la propagación de errores a niveles inferiores de la jerarquía, la cual es la desventaja principal del enfoque jerárquico local.
- Investigar medidas para evaluar la clasificación jerárquica, en cuanto a la distancia en el árbol entre la categoría predicha y la correcta.
- Realizar experimentos usando un enfoque jerárquico que combine los subclasificadores en diferentes niveles en un único modelo predictivo mediante la utilización de técnicas "meta-algorítmicas", como por ejemplo disminuir la varianza (*boosting*), o el sesgo (*bagging*) u optimizar la fuerza predictiva (*stacking* o *ensemble*).
- Utilizar algoritmos de clasificacion de etiquetado múltiple (ejemplos: *Binary Relevance*, *Label Powerset*) para poder incorporar las clasificaciones secundarias que poseen los papers.
- Tener en cuenta grupos de colaboradores que participaron en la redacción de los papers, es decir, incorporar la información de los temas de investigación a los que pertenecen estos colaboradores.

Bibliografía

[Dum00] Susan Dumais. Hierarchical Classification of Web Content. *SIGIR '00 Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000.

[Iko05] E. K. Ikonomakis. Text Classification Using Machine Learning Techniques. *WSEAS Transactions on Computers*, 2005.

[Kim06] S. Kim. Some effective techniques for naïve bayes text classification. *IEEE Transactions on Knowledge and Data Engineering*, 2006.

[Ng01] Andrew Ng. On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in Neural Information Processing Systems 14*, 2001.

[Nig99] Kamal Nigam. Using Maximum Entropy for Text Classification. 1999.

[Por80] M.F. Porter. An algorithm for suffix stripping. *Program 14*, 1980.

[Rui02] Miguel Ruiz. Hierarchical Text Categorization Using Neural Networks. *Information Retrieval*, 2002.

[San09] António Paulo Santos. Multi-label Hierarchical Text Classification using the ACM Taxonomy. 2009.

[Seb02] Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 2002.

[Tin11] S.L. Ting. Is Naive Bayes a Good Classifier for Document Classification. *International Journal of Software Engineering and Its Applications*, 2011.

[ws@a] ACM Digital Library. `http://dl.acm.org/dl.cfm`.

[ws@b] Lenguaje de programacion Java. `https://www.java.com/es`.

[ws@c] MALLET, MAchine Learning for LanguagE Toolkit. `http://mallet.cs.umass.edu`.

[ws@d] Lenguaje de programacion Python. `https://www.python.org`.

[Yan99] Yiming Yang. An Evolution of statistical Approaches to Text Categorization. *Information Retrieval 1*, 1999.

[Zha04] Baoping Zhang. Combining Structural and Citation-Based Evidence for Text Classification. *CIKM '04 Proceedings of the thirteenth ACM international conference on Information and knowledge management*, 2004.