QUERANDO!: Un Agente de Filtrado de Documentos Web

Sergio A. Gómez[1] **Laura Lanzarini**[2]

Laboratorio de Investigación y Desarrollo en Informática[3]
Facultad de Informática - Universidad Nacional de La Plata

Resumen

La gran cantidad de información en la WWW requiere de métodos superiores de asistencia al usuario con respecto a la funcionalidad provista por los motores de búsqueda actuales. Enfoques recientes en el área de agentes inteligentes ayudan a los usuarios a solucionar este problema mediante la adquisición de perfiles de filtrado de documentos.
El presente artículo describe un agente de filtrado de páginas web llamado *Querando!* capaz de aprender perfiles representativos de las preferencias del usuario, a través de una red neuronal basada en la Teoría de la Resonancia Adaptativa Difusa.
El modelo utilizado en este agente se caracteriza por permitir el perfeccionamiento del perfil a través del tiempo sin requerir reentrenamiento con documentos previos.
Finalmente, se discute la implementación de *Querando!* y se enumeran los resultados obtenidos que avalan dicho enfoque comparándolo con otras soluciones existentes.

Palabras claves: Agentes inteligentes, algoritmos de clustering, filtrado de documentos, redes neuronales, Teoría de la Resonancia Adaptativa Difusa.

1 Introducción

La cantidad de información publicada en formato electrónico supera la capacidad de los usuarios de encontrarla y procesarla. En ambientes de información como la World Wide Web, la necesidad de ayudar a los usuarios a filtrar documentos es fundamental debido a la vastedad de la misma.

Hasta hace unos años atrás, un usuario prospectivo de información en la WWW tenía dos opciones para hallar documentos HTML relevantes a sus necesidades de información. Una opción era explorar manualmente el digrafo de documentos HTML; esta actividad generalmente se conoce como *surfing*. La otra opción era hacer una consulta a un motor de búsqueda como *Altavista* o *Lycos*. El resultado de esta actividad era una lista de enlaces a sitios relevantes almacenados en la forma de un *bookmark* en un navegador.

Por otro lado, enfoques recientes en el área de agentes de la información plantean la adquisición automática de perfiles de filtrado de usuario mediante técnicas de *machine learning* (veáse sección 5). Dichos algoritmos toman como entrada un conjunto de documentos web (particionados en relevantes e irrelevantes por el usuario) y son capaces de predecir la relevancia o irrelevancia de un documento nunca antes visto. Luego, el perfil de filtrado así obtenido

[1]Lic. en Informática. Jefe de Trabajos Prácticos Dedicación Simple. Fac. de Informática. Universidad Nacional de La Plata. Becario de Entrenamiento Comisión de Investigaciones Científicas de la Provincia de Buenos Aires. E-mail: sergiog@info.unlp.edu.ar.

[2]Lic. en Informática. Profesor Titular Dedicación Exclusiva. Fac. de Informática. Universidad Nacional de La Plata. E-mail: laural@info.unlp.edu.ar.

[3]Calle 50 y 115 1er. Piso, (1900) La Plata, Argentina, Tel./Fax +(54)(221)422-7707. URL: http://lidi.info.unlp.edu.ar

puede usarse como una representación de los gustos del usuario. La actividad de adquisición de perfiles de filtrado es muy difícil por razones explicadas en [Gómez01, Mostafa97].

Las redes neuronales pueden utilizarse para esta tarea debido a sus cualidades naturales para tratar con presencia de ruido e incompletitud en los datos de entrada [Bigus98, Hyötyniemi96, Kohonen00]. Una red neuronal es capaz de particionar los datos de entrada en un conjunto de clusters basado en un criterio de similitud [Freeman93, Rasmussen92, Skapura96, Wasserman89]. De esta manera, cada cluster puede etiquetarse como relevante o irrelevante y así llevar a cabo el filtrado de los documentos.

Sin embargo, las soluciones convencionales adolecen del defecto de requerir reentrenamiento de los algoritmos de clasificación ante la presencia de nuevos documentos. La red neuronal basada en la Teoría de la Resonancia Adaptativa (ART) soluciona este problema al resolver el *dilema de la estabilidad-plasticidad* el cual le permite aprender nuevos patrones a medida que éstos se le presentan sin olvidar los aprendidos previamente [Freeman93].

En este trabajo se utiliza una variante de la red ART basada en la lógica difusa (FART) [Lavoie99], la cual acepta entradas analógicas y desarrolla una clasificación; las entradas familiares activan su categoría mientras que entradas no familiares activan el aprendizaje adaptativo de una categoría existente o la creación de una nueva.

En este artículo, se describe una experiencia en la construcción del agente prototipo *Querando!* para filtrar y buscar páginas en la WWW. En el prototipo construido, el trabajo del agente es aprender la manera en que el usuario elige páginas de una serie de resultados de búsqueda a fin de adquirir un perfil de los intereses de dicho usuario. Este perfil puede usarse para sugerir qué otros enlaces deben explorarse. El aporte de este trabajo consiste en usar la red de la Teoría de la Resonancia Adaptativa Difusa para poder hacer clasificación de documentos; además, se plantea una representación de documentos que no requiere el uso de diccionarios de frecuencias de apariciones de términos.

El resto del artículo se halla estructurado de la siguiente manera. Primero, se describen las mediciones experimentales para probar la idoneidad de las representaciones propuestas para los documentos HTML. Segundo, se discuten los aspectos de implementación del agente *Querando!*. Tercero, se enumeran los trabajos relacionados hallados en la literatura de agentes y se comparan con el realizado aquí. Finalmente, se enumeran las conclusiones y se discuten futuras líneas de investigación.

2 Clustering de Documentos

2.1 Representación de Documentos

Uno de los problemas básicos del filtrado y la recuperación de información es la relación difusa entre el formato de los documentos (texto libre) y su contenido semántico. Por lo tanto, surge la necesidad de especificar para cada documento una representación numérica vectorial que caracterice su contenido [Frakes92a]. De esta forma, quedará establecido el espacio de patrones donde podrán aplicarse diferentes medidas de similitud. En esta sección, se explica cómo se logra la construcción de un vector de características que modele a los documentos HTML que clasificará el agente Querando!.

2.1.1 Preprocesamiento

Previo a la construcción del vector de características correspondiente a un documento dado, es necesario realizar un preprocesamiento del mismo a fin de quitar la información redundante.

De esta forma se eliminan los marcadores HTML (exceptuando las *claves meta*), el código de estilo y scripts [Gulbransen98, Raggett98] y los marcadores correspondientes a sentencias PHP [Bakken99]. Los términos restantes son filtrados utilizando un diccionario negativo (lista de palabras que no contienen información como `the`, `of`, `this`, `that`, ...) [FoxC92]. Finalmente se aplica un algoritmo de stemming basado en la eliminación de sufijos [Frakes92b].

En el presente trabajo, la obtención del vector de características a partir de un documento HTML se realiza a través de un parser desarrollado ad-hoc para este fin[4].

2.1.2 Representación utilizando Contadores de Claves

La representación de documentos utilizando contadores de claves está basada en la teoría de recuperación y filtrado de información donde los documentos, requerimientos y perfiles se representan con un conjunto de términos de indexación pesados con algún criterio [Frakes92a]. De esta manera, el usuario debe especificar un conjunto de claves de filtrado. Es decir que, dado un perfil de filtrado formado por N claves $k_1, \ldots, k_N$, la representación de un documento HTML o de texto D se obtiene de la siguiente manera:

1. Preprocesar el documento D para obtener D_2. Este proceso es el descripto en 2.1.1.

2. Contar cuántas veces aparecen las claves de filtrado $k_1, \ldots, k_N$, en D_2 para obtener un vector de contadores $v = (c_1, \ldots, c_N)$.

3. A las entradas c_i de v $(i = 1, \ldots, N)$ distintas de 0 aplicarles la función $F(x) = (1 + e^{-(x-5)})^{-1}$ (función sigmoide desplazada hacia la derecha del eje cartesiano en 5 unidades) para obtener un vector difuso $vf = (F(c_1), \ldots, F(c_N))$. Esta operación tiene como objetivo escalar las entradas del vector de contadores para adecuarlas al funcionamiento de la FART.

4. Asignar con 1 a las componentes i de vf tal que sus claves de filtrado k_i estuvieran en la lista de claves meta. El significado de esta operación es tomar las claves especificadas por el autor del documento HTML (si las hubiera) como las más importantes de todas[5].

2.2 Algoritmos de Clustering

Querando! es capaz de realizar una partición en clases de un flujo de documentos. Las clases obtenidas son entonces etiquetadas por el usuario como relevantes o irrelevantes. Así, un documento nunca visto antes puede asignarse dinámicamente a una clase existente o producir la creación de una nueva clase de documentos. Los documentos se representan internamente con *un vector de características*; las cuales conforman patrones numéricos que serán la entrada de un algoritmo de clustering [Freeman93, Maravall94, Rasmussen92, Skapura96, Wasserman89]. Los algoritmos de clustering toman un conjunto de patrones representados por vectores numéricos y generan, en forma no supervisada, un agrupamiento de los mismos de acuerdo a un criterio de similitud [Rasmussen92].

El objetivo es aplicar las redes neuronales al filtrado de documentos basado en el clustering de los mismos. Las pruebas preliminares para estudiar la viabilidad de este enfoque se hicieron

[4]Si bien esta labor se podría haber hecho usando alguna herramienta de diseño de compiladores como LEX o YACC [Aho83], las rutinas necesarias se implementaron completamente.

[5]El lenguaje HTML contiene un marcador llamado `META` que sirve para que los programadores HTML agreguen explicitamente claves de indexación para los motores de búsqueda de la web [Raggett98, p. 57]. Por ejemplo: `<META name="keywords" lang="en" content="vacation, Greece, sunshine" >`.

usando el método de las K-medias (KM) [Maravall94], el modelo de redes neuronales de contrapropagación (CPN) [Freeman93, Rao95, Skapura96, Wasserman89] y el modelo finalmente utilizado que está basado en la teoría de la resonancia adaptativa difusa (FART) [Lavoie99].

Aquí, se da una descripción sintética de los algoritmos mencionados.

El KM realiza una partición de los datos de entrada basado en la minimización de la dispersión estadística de las distancias euclídeas interclase de los vectores; requiere, además, la especificación de la cantidad de clases esperadas, la que hace que este algoritmo sea muy sensible a este parámetro. En las mediciones realizadas sólo se usó la capa competitiva de la CPN (dejando de lado la capa de Grossberg) la cual es capaz de aprender una caracterización de los datos de entrada en la forma de los centroides de clases halladas en los mismos. La red FART resuelve el *dilema de la estabilidad-plasticidad* que la hace capaz de aprender nuevos patrones sin olvidar los ya aprendidos; esta red realiza una partición de los datos de entrada en la forma de hiperrectángulos de manera que la presencia de un nuevo patrón de entrada produce la adaptación de una clase existente para aprenderlo o dispara la creación de una nueva clase.

Debido a que tanto KM como CPN requieren varias pasadas por la entrada y re-entrenamiento en el caso de la aparición de nuevos patrones mientras que en la FART no hay diferencia entre las etapas de entrenamiento y producción, la postura de este trabajo es que la FART es la opción adecuada para modelar el algoritmo de aprendizaje de un agente de filtrado. A pesar de su inaptitud para la implementación final del agente, KM y CPN son métodos muy estudiados y, por lo tanto, son capaces de usarse como medidas comparativas del desempeño de la red FART.

2.3 Análisis de Similitud entre Documentos

Se realizaron mediciones experimentales para verificar la representación propuesta en 2.1. En esta sección se describen el conjunto de documentos usados y los resultados obtenidos junto con un análisis de los mismos.

2.3.1 Conjunto de Documentos

Los documentos usados en las mediciones pertenecen a tres clases temáticas y son los siguientes:

- *Relojes Breitling*: Este conjunto de documentos pertenece al sitio de los relojes de los entusiastas de la aviación *Breitling*[6]. Los archivos que componen este conjunto son: *Breitling Produits1.htm*, *Breitling Produits2.htm*, *Breitling Produits3.htm*, *Breitling Produits4.htm*, *Breitling Produits5.htm*, *Breitling Produits6.htm*, *Breitling Produits7.htm* y *Breitling Produits8.htm*.

- *Running Times*: Este conjunto de documentos pertenece a la revista de aficionados al *jogging Running Times*[7]. Los archivos son: *oatips26.htm*, *oamareco.htm*, *hydration_g.htm*, *mar_tips.htm*, *jgmarsimpl.htm*, *marmind.htm*, *halmar.htm*, *marbenji.htm*, *jglongrn.htm*, *halworldbest.htm*, *mara_rw_off_10princ.htm*, *mara_rw_off_adv.htm*, *mara_rw_off_int.htm*, *mara_rw_off_beg.htm* y *mara_rw_official.htm*.

- *Handheld Palm Pilot*: Este conjunto de documentos fue tomado del sitio de *3Com* sobre el asistente personal *Palm Pilot*[8]. Los archivos que componen este conjunto son: *palm1.htm*, *palm2.htm*, *palm3.htm*, *palm4.htm*, *palm5.htm* y *palm6.htm*.

[6]URL: `www.breitling.com`
[7]URL: `www.runningtimes.com`
[8]URL: `www.palm.com`

2.3.2 Matriz de Similitud entre Documentos

Con respecto a la representación de los documentos usados, las claves utilizadas para la obtención de los contadores de claves fueron: *marathon, running, breitling, watches, time, palm, pilot, computer*. Los vectores de características de los mismos se hallan en la tabla 1.

La similitud entre dos documentos da una métrica del "parecido" entre ellos [Rasmussen92]. En la tabla 2 se detalla la matriz de similitud entre los documentos mencionados. En ambos casos la medida de similitud usada fue la distancia euclídea. La matriz que se muestra es incompleta por una cuestión de espacio ya que la matriz completa tiene 29^2 entradas.

Documento	*Título*	*Vector de características*
Breitling Produits1	Breitling Produits: Home Page	(0.000 0.000 1.000 0.000 0.000 0.000 0.047 0.000)
Breitling Produits2	Breitling Produits: Models	(0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000)
Breitling Produits3	Breitling Produits: Catalogue	(0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000)
Breitling Produits4	Breitling Produits: Chronoliner	(0.000 0.000 1.000 0.000 0.000 0.000 0.047 0.000)
Breitling Produits5	Breitling Produits: Chronomat GT Watch	(0.000 0.000 1.000 0.017 0.017 0.000 0.000 0.000)
Breitling Produits6	Breitling Produits: Chronomat Vitesse Watch	(0.000 0.000 1.000 0.017 0.017 0.000 0.000 0.000)
Breitling Produits7	Breitling Produits: Chronomat Blackbird Watch	(0.000 0.000 1.000 0.017 0.017 0.000 0.000 0.000)
Breitling Produits8	Breitling Produits: Chronomat Longitud Watch	(0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000)
oatips26	Tips for 26	(1.000 1.000 0.000 0.000 0.047 0.000 0.000 0.000)
oamareco	The Perfect Marathon Recovery	(1.000 1.000 0.000 0.000 0.119 0.000 0.000 0.000)
hydration_g	Fluid tips for the long run	(1.000 1.000 0.000 0.000 0.017 0.000 0.000 0.000)
mar_tips	40 marathon tips	(1.000 1.000 0.000 0.268 0.999 0.731 0.000 0.000)
jgmarsimpl	Marathoning Simplified	(1.000 1.000 0.000 0.000 0.047 0.000 0.000 0.000)
marmind	Mind over Marathon	(1.000 1.000 0.000 0.000 0.731 0.000 0.000 0.000)
halmar	Hal Higdon's Marathon Training Plan	(1.000 1.000 0.000 0.000 0.017 0.000 0.000 0.000)
marbenji	The Path to Marathon Success	(1.000 1.000 0.000 0.017 0.999 0.000 0.000 0.000)
jglongrn	Long May You Run	(1.000 1.000 0.000 0.000 0.047 0.000 0.000 0.000)
halworldbest	World's Best Marathon Program	(1.000 1.000 0.000 0.000 0.993 0.000 0.000 0.000)
mara_rw_off_10princ	10 Principles of Marathon Training	(1.000 1.000 0.000 0.000 0.119 0.000 0.000 0.000)
mara_rw_off_adv	Marathon Training for Advanced Runners	(1.000 1.000 0.000 0.000 0.017 0.000 0.000 0.000)
mara_rw_off_int	Marathon Training for Intermediate Runners	(1.000 1.000 0.000 0.000 0.017 0.000 0.000 0.000)
mara_rw_off_beg	Marathon Training for Beginners	(1.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000)
mara_rw_official	Official Runner's World Marathon Training Programs	(1.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000)
palm5.htm	Palm OS v3.1.1 Update	(0.000 0.017 0.000 0.000 0.000 1.000 0.017 0.119)
palm2.htm	E-mail Conduit Update	(0.000 0.000 0.000 0.000 0.000 0.880 0.017 0.119)
palm3.htm	Palm III Support	(0.000 0.000 0.000 0.000 0.047 0.993 0.017 0.119)
palm4.htm	Palm IIIe Connected Organizer FAQ	(0.000 0.000 0.000 0.000 0.000 1.000 0.047 0.880)
palm1.htm	Palm Desktop 3.0.1 Software	(0.000 0.017 0.000 0.000 0.000 1.000 0.119 0.952)
palm6.htm	Palm Support	(0.000 0.000 0.000 0.000 0.000 0.999 0.017 0.119)

Tabla 1: Representación de los documentos separados por clase. Las claves utilizadas para la obtención de los contadores de claves fueron: *marathon, running, breitling, watches, time, palm, pilot, computer*

El éxito de los algoritmos de clustering se basa en que los elementos pertenecientes a una misma clase son "parecidos" aplicando un criterio de similitud dado.

En el presente trabajo, los elementos a agrupar están definidos por vectores de características construidos como se indica en 2.1 y el criterio de similitud utilizado es la distancia euclídea entre vectores.

2.3.3 Resultados Obtenidos

Los resultados de la aplicación de los algoritmos de clustering se enumeran en la tabla 3.

Los parámetros usados en los métodos de clustering son:

El KM se ejecutó con un número de clases $K = 3$ y alcanzó la convergencia en 2 iteraciones por los 29 vectores de la población, lo que hace un total de 58 pasos. Los centroides y las dispersiones de las clases obtenidos son:

	0	1	2	3	4	5	6	7	8
0	† 0.000	† 0.000	† 0.047	1.732	1.736	1.732	1.419	1.338	1.415
1	† 0.000	† 0.000	† 0.047	1.732	1.736	1.732	1.419	1.338	1.415
2	† 0.047	† 0.047	† 0.000	1.733	1.736	1.732	1.419	1.338	1.415
3	1.732	1.732	1.733	‡ 0.000	‡ 0.071	‡ 0.029	1.726	1.671	1.732
4	1.736	1.736	1.736	‡ 0.071	‡ 0.000	‡ 0.101	1.730	1.674	1.733
5	1.732	1.732	1.732	‡ 0.029	‡ 0.101	‡ 0.000	1.726	1.670	1.732
6	1.419	1.419	1.419	1.726	1.730	1.726	◇ 0.000	◇ 0.120	◇ 0.051
7	1.338	1.338	1.338	1.671	1.674	1.670	◇ 0.120	◇ 0.000	◇ 0.122
8	1.415	1.415	1.415	1.732	1.733	1.732	◇ 0.051	◇ 0.122	◇ 0.000

Tabla 2: Matriz de similitud (una parte) entre documentos. Hay tres documentos por cada clase, están en forma consecutiva y las clases están distinguidas con †, ‡ y ◇ respectivamente. Los documentos medidos son *Breitling Produits1, Breitling Produits2, Breitling Produits4, oatips26, oamareco, hydration_g, palm5, palm2* y *palm3*. Se puede apreciar que la distancia entre documentos es menor cuando los documentos se hallan en la misma clase. Esta cualidad de la representación es lo que hace posible el funcionamiento de los métodos de clustering.

- Clase 0: (0.000 0.000 1.000 0.004 0.004 0.000 0.014 0.000) y $\sigma = 0.0005$
- Clase 1: (1.000 1.000 0.000 0.019 0.278 0.048 0.000 0.000) y $\sigma = 0.1973$
- Clase 2: (0.000 0.005 0.000 0.000 0.007 0.978 0.039 0.385) y $\sigma = 0.1454$

La CPN se usó con tasa de aprendizaje = 0.01, cantidad de clases esperadas = 10, tiempo de entrenamiento 40 pasadas por cada uno de los 29 patrones lo que da 1160 *epochs*. La inicialización de los centroides se hizo con los primeros 10 vectores.

Los centroides para las clases halladas por la CPN son[9]:

- Clase 1: (0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000)
- Clase 3: (0.000 0.000 0.998 0.000 0.000 0.000 0.047 0.000)
- Clase 5: (0.000 0.000 0.999 0.017 0.017 0.000 0.000 0.000)
- Clase 6: (0.000 0.000 0.999 0.000 0.000 0.000 0.017 0.000)
- Clase 8: (0.673 0.673 0.000 0.009 0.160 0.023 0.000 0.000)
- Clase 9: (0.067 0.071 0.000 0.003 0.036 0.820 0.029 0.275)

La FART fue utilizada con los siguientes parámetros: $\alpha = 0.0$, velocidad de aprendizaje $\beta = 1.0$ (aprendizaje rápido [Lavoie99]) y test de vigilancia $\rho = 0.8$.

Las tablas 4.a y 4.b muestran las distancias interclases para el KM y la CPN. En el caso de la CPN se obvian los valores para las unidades que no codificaron vectores.

2.3.4 Análisis de los Resultados Obtenidos

La tabla 2, la tercera columna de la tabla 3 y las tablas 4.a y 4.b demuestran que la representación seleccionada es adecuada para la aplicación de los diferentes algoritmos de clustering, ya que la distancia entre documentos pertenecientes a una misma clase es mucho menor a la distancia interclases.

La tabla 3 muestra que los tres algoritmos son capaces de particionar correctamente a los documentos en clases disjuntas. Sin embargo, la diferencia entre ellos radica en la manera en que adquieren el perfil del usuario.

Los algoritmos KM y CPN obtienen los representantes de cada clase mediante un proceso iterativo aplicado sobre el conjunto completo de vectores de entrada. Mas allá de que ambos

[9]Se omiten los vectores que no codificaron centroides de los datos.

Documento	Clase KM	Distancia al centroide K-Medias	Clase CPN	Distancia centroide CPN	Clase FART	Test de vigilancia	Historia de resonancias
Breitling Produits1	0	0.033	3	0.000	0	1.000	0
Breitling Produits2	0	0.015	1	0.000	0	0.994	0
Breitling Produits3	0	0.015	1	0.000	0	0.994	0
Breitling Produits4	0	0.033	3	0.000	0	0.994	0
Breitling Produits5	0	0.023	5	0.000	0	0.989	0
Breitling Produits6	0	0.023	5	0.000	0	0.989	0
Breitling Produits7	0	0.007	6	0.000	0	0.989	0
Breitling Produits8	0	0.015	1	0.000	0	0.989	0
oatips26	1	0.236	8	0.137	1	1.000	0:0.610
oamareco	1	0.167	8	0.091	1	0.991	
hydration_g	1	0.265	8	0.157	1	0.987	
mar_tips	1	1.024	8	0.570	2	1.000	1:0.752
jgmarsimpl	1	0.236	8	0.137	1	0.987	
marmind	1	0.455	8	0.306	1	0.910	
halmar	1	0.265	8	0.157	1	0.910	
marbenji	1	0.723	8	0.439	1	0.875	
jglongrn	1	0.236	8	0.137	1	0.875	
halworldbest	1	0.716	8	0.436	1	0.875	
mara_rw_off_10princ	1	0.167	8	0.091	1	0.875	
mara_rw_off_adv	1	0.265	8	0.157	1	0.875	
mara_rw_off_int	1	0.265	8	0.157	1	0.875	
mara_rw_off_beg	1	0.283	8	0.169	1	0.872	
mara_rw_official	1	0.283	8	0.169	1	0.872	
palm5.htm	2	0.267	9	0.251	3	1.000	0:0.722, 1:0.482, 2:0.542
palm2.htm	2	0.284	9	0.245	3	0.982	
palm3.htm	2	0.270	9	0.252	3	0.972	
palm4.htm	2	0.496	9	0.404	3	0.878	
palm1.htm	2	0.573	9	0.437	3	0.860	
palm6.htm	2	0.267	9	0.255	3	0.860	

Tabla 3: Clasificación de los documentos con K-medias, CPN y FART.

	0	1	2
0	0.000	1.754	1.451
1	1.754	0.000	1.754
2	1.451	1.754	0.000

	1	3	5	6	8	9
1	0.000	0.047	0.025	0.017	1.390	1.327
3	0.047	0.000	0.053	0.029	1.390	1.326
5	0.025	0.053	0.000	0.031	1.388	1.326
6	0.017	0.029	0.031	0.000	1.390	1.326
8	1.390	1.390	1.388	1.390	0.000	1.206
9	1.327	1.326	1.326	1.326	1.206	0.000

Tabla 4: Distancias intercentroides para el KM y CPN respectivamente.

métodos requieren alguna información de la cantidad de clases a formar, su mayor problema se basa en la necesidad de recalcular completamente el perfil del usuario para poder incorporar una nueva clase.

Por su lado, la FART tiene la capacidad de adaptarse a los patrones de entrada perfeccionando el conocimiento adquirido a medida que el usuario ingresa nueva información.

3 Agente de Filtrado *Querando!*

Las mediciones experimentales descriptas en la sección anterior fueron realizadas con el objetivo de medir la viabilidad de usar la red de la FART como modelo de un agente de filtrado de documentos HTML. El agente, llamado *Querando!*, aprende las preferencias del usuario con respecto a un conjunto de documentos presentados. Luego, cuando el agente se expone a nuevos documentos será capaz de dar un juicio de relevancia sobre los mismos basado en la clasificación aprendida. En el caso de desconocer la relevancia del documento, el agente preguntará al usuario.

Los documentos a filtrar pueden obtenerse de tres fuentes:

1. *Reservorios de enlaces*: El usuario puede especificar una URL correspondiendo a un reservorio de enlaces (una página con enlaces recolectados por algún usuario todos apuntando a documentos en un tema dado).

2. *Consultas a motores de búsqueda*: El usuario especifica una cadena de búsqueda y el agente tiene codificado conocimiento para interactuar con varios motores de búsqueda de la WWW. En general esta opción fue pensada para la etapa inicial de definición de clusters.

3. *Una URL y una profundidad de búsqueda*: El digrafo formado por documentos y enlaces HTML se recorre en anchura con una profundidad determinada, generando un reservorio de enlaces efímero.

Las páginas de enlaces son aumentadas con controles para que el usuario pueda expresar su juicio de relevancia sobre los documentos referenciados. El agente realizará entonces una clasificación de los documentos; cada una de las clases obtenidas será etiquetada como relevante, irrelevante o no definida. De esta manera, se aprende una caracterización de un conjunto de documentos presentados al usuario que puede usarse para continuar la sesión de filtrado corriente, continuarla en el futuro o conducir una sesión de filtrado futura al almacenar el conocimiento obtenido como un perfil de filtrado de usuario. En la práctica, los perfiles de usuario son el conjunto de pesos de una red junto con sus parámetros respectivos.

El agente fue construido como un conjunto de documentos HTML basados en formularios HTML con validación usando JavaScript [Gulbransen98] y procesamiento en un servidor central vía guiones CGI [Dwight97] programados en C++ [Brokken95, Kernigham85]. De esta manera, la aplicación es accesible desde cualquier punto del globo disponiendo de un browser. Los datos de la aplicación se hallan almacenados en una base de datos basada en ODBC. La interacción con servidores de la WWW se realiza teniendo en cuenta directivas de *nettiquete* [Koster93] respecto del tiempo entre *hits* para prevenir su sobrecarga.

4 Resultados Obtenidos

Se ha comprobado que la representación de los documentos utilizando contadores de claves resulta adecuada para esta aplicación. Obviamente, el éxito de esta representación es dependiente del criterio del usuario para seleccionar las claves respectivas.

En cuanto a la arquitectura seleccionada, los resultados obtenidos con la FART han demostrado ser tan satisfactorios como los de otros métodos de clustering. Sin embargo, a diferencia de ellos, la FART permite a Querando! perfeccionar el perfil del usuario a medida que éste utiliza el agente.

Las pruebas realizadas con 30 usuarios diferentes demuestran que Querando! es capaz de separar en clases un conjunto de documentos con una precisión del superior al 95%. Los parámetros usados en estas pruebas fueron $\alpha = 0$, $\beta = 1$ (aprendizaje rápido) [Lavoie99] y $\rho = 0.87$. La cantidad de documentos necesarias para que la FART aprenda un concepto varía entre 5 y 10 documentos por clase.

5 Comparación con Trabajos Relacionados

5.1 Background

Históricamente, en el feedback de relevancia en recuperación de información se pueden hallar los primeros intentos en mejorar una consulta en base a la respuesta de un usuario [Harman92].

Aplicaciones tales como las arañas son ejemplos conocidos de programas que recorren el grafo de la WWW para construir índices de los documentos hallados [Cheong96, Hardy96].

Por otro lado, existen varias implementaciones de agentes de filtrado de documentos basados en el enfoque de obtener en forma automática un perfil de filtrado que represente un concepto. En [Lashkari97, Maes94] se describe un agente de filtrado de correo electrónico para la aplicación Eudora. En [Bigus98] se describe la implementación de un agente de filtrado de grupos de noticias usando las redes de backpropagation y mapa autoorganizativo de Kohonen [Freeman93, Rao95, Skapura96]. Otros trabajos plantean el clustering de documentos similares usando el mapa autoorganizativo de Kohonen [Hyötyniemi96, Kohonen00].

[Perkowitz96] piensan que la construcción de agentes a medida no escala al ritmo de crecimiento de Internet y proponen al agente *ILA*[10] capaz de aprender a modelar recursos por su cuenta. También en esta línea de trabajo, se halla el *Internet Softbot* [Etzioni97]. En [Kushmerick98], se plantea la traducción automática de recursos de información a KIF [Genesereth92, Genesereth94a, Genesereth94b, Genesereth97].

Otros enfoques proponen la construcción automática de perfiles de filtrado basados en la distribución estadística de los términos en los documentos relevantes y/o irrelevantes [Balabanovic98, Billsus97, Pazzani97a, Pazzani97b].

Trabajos similares son los enfoques para mantener directorios de recursos [Cohen96] y asistencia del usuario en la navegación web [Wang97].

5.2 Discusión

En esta sección se discuten los resultados en este trabajo y se comparan con los mencionados en el punto anterior. Las siguientes consideraciones caben con respecto a los agentes de la literatura que realizan funciones comparables a las del agente Querando!:

- El filtrador de noticias de Bigus [Bigus98], al utilizar los modelos de redes neuronales de propagación hacia atrás [Freeman93, Rao95, Skapura96, Wasserman89] y red de Kohonen [Freeman93, Rao95, Skapura96, Wasserman89], requiere reentrenamiento cuando cambia el escenario de uso. La red FART, al resolver el dilema de estabilidad-plasticidad [Freeman93, Lavoie99] no tiene este inconveniente. El filtro de Bigus está implementado como una aplicación stand-alone; esto tiene dos consecuencias: a) puede incrementar la eficiencia del agente ya que el usuario no tiene que compartir la CPU con otros usuarios; y b) no tiene la flexibilidad de la implementación basada en web en cuanto al acceso desde cualquier lugar del mundo.

- En cuanto a las capacidades de *spider*, Querando! es capaz de recorrer la estructura de grafo de la WWW. Sin embargo, en contraposición a Harvest [Hardy96], éste no fue el objetivo principal del trabajo. Si bien el sistema almacena las páginas visitadas, esto sólo se implementó como una necesidad a la hora de trabajar sin conexión a la red; además, el sistema sólo almacena documentos HTML y de texto, dejando de lado otros formatos de datos como fotos, películas y sonidos.

[10] Internet Learning Agent

- En contraposición a los agentes basados en el clasificador bayesiano [Balabanovic98, Billsus97, Pazzani97a, Pazzani97b], el uso de la red neuronal FART elimina la necesidad de construir diccionarios de frecuencias de apariciones de términos en la WWW.
- La función del agente puede pensarse como un caso particular del agente de mantenimiento de directorios de recursos de Cohen [Cohen96], ya que Querando! es capaz de aprender un concepto a través de la ejemplificación del mismo mediante la presentación de documentos. Sin embargo, por la forma en que está planteada la interfaz, Querando! no maneja hipótesis de mundo cerrado [Lloyd87]; esto quiere decir, que si el agente no sabe clasificar un documento no va a contestar que éste es irrelevante sino que va a preguntar una calificación para el mismo y lo agregará a su base de conocimiento (pesos de la red neuronal).
- La interfaz de Querando! es similar a la de los trabajos de Michael Pazzani [Billsus97, Pazzani97a, Pazzani97b] y de Marko Balabanovic [Balabanovic98].

6 Conclusiones y Líneas de Trabajo Futuras

Se ha presentado un agente de filtrado de páginas web llamado Querando! capaz de aprender perfiles representativos de las preferencias del usuario, basado en una red neuronal FART.

A diferencia de otras soluciones existentes, Querando! aprende permanentemente del usuario, mejorando sus respuestas a medida que se le incorpora información.

Hay una variedad de direcciones en las cuales esta investigación puede orientarse en el futuro. Estas pueden separarse en dos clases:

1. Representación y clasificación de documentos:
   - En este trabajo, se dejaron deliberadamente de lado diversos formatos de documentos que son igualmente interesantes para el filtrado como por ejemplo: formato de texto enriquecido (RTF), Postcript (PS), Abobe Portable Document Format (PDF), DVI, etc. Tampoco se consideraron los formatos comprimidos, por ejemplo: ARJ, ZIP, etc.
   - Tampoco fueron tenidas en cuenta características de los documentos tales como el contenido multimedia, compuesto por fotos, películas y sonidos. Estas características pueden también formar parte del perfil del usuario.
   - Las técnicas de datamining también se pueden aplicar a la obtención de información en grandes bases de documentos [Glymour96, Kohonen00].
2. Implementación de agentes y sistemas multiagentes:
   - Una nueva implementación del agente Querando! basada en un programa standalone que permitiría utilizar el agente sin necesidad de un servidor central.
   - La implementación actual del agente está dirigida a un único usuario. Una forma de extender este trabajo puede orientarse al filtrado cooperativo, haciendo que varios usuarios den forma a un perfil de filtrado.
   - Otra línea de trabajo futura puede hallarse en la implementación de un sistema de filtrado multiagente que, en lugar de utilizar un único programa que realice el filtrado de los documentos, utilice varios logrando reducir el tiempo de procesamiento [Gómez99, Huhns97].

Referencias

[Aho83] Aho, Alfred V.; Sethi, Ravi; Ullman, Jeffrey D. *Compilers. Principles, Design and Tools.* Addison-Wesley Publising Co. Reading, Massachusetts, 1983.

[Bakken99] Bakken, Stig Sæther; Aulbach, Alexander; Schmid, Egon ; Winstead, Jim; Wilson, Lars Torben; Lerdorf, Rasmus; Suraski, Zeev. *PHP3 Manual.* The PHP Documentation Group, 1999.

[Balabanovic98] Balabanovic, M.; Shoham, Y.; Yun, Y. *An Adaptive Agent for Automated Web Browsing.* 1998. `http://elib.stanford.edu/Dienst/UI/2.0/Describe/stanford.cs%2fCS-TN-97-52`

[Bigus98] Bigus, J. P.; Bigus, J. *Constructing Intelligent Agents with Java.* 1998. Wiley Computer Publishing. Ed. John Wiley & Sons. ISBN: 0-471-14135-3.

[Billsus97] Billsus, D.; Pazzani, M. *Learning Probabilistic User Models.* In workshop notes of Machine Learning for User Modeling, Sixth International Conference on User Modeling, Chia Laguna, Sardinia, 2-5 June 1997. `http://www.ics.uci.edu/~pazzani/Publications/ProbUserModels.ps`

[Brokken95] Brokken, F.; Kubat, K. *C++ Annotations.* 1995. ICCE, State university of Gronningen, Netherlands. ISBN 90 367 0470 7.

[Cohen96] Cohen, W.; Singer, N. *Learning to Query the Web.* In The 1996 AAAI Workshop on Internet-Based Information Systems. 1996. `http://www.research.att.com/~wcohen/postscript/aaai-ws-96.ps`

[Cheong96] Cheong, F. C. *Internet Agents: Spiders, Wanderers, Brokers, and Bots.* 1996. Ed. New Riders Publishing.

[Dwight97] Dwight, J; Erwin, M.; Niles, R. *Using CGI. Second Edition.* QUE Corp. 1997.

[Etzioni97] Etzioni, O.; Weld, D. *A Softbot-Based Interface to the Internet.* 1997. Readings in Agents, págs. 77–81. Morgan Kaufmann.

[FoxC92] Fox, C. *Lexical Analysis and Stop Lists.* In Information Retrieval. Págs. 28–43. Frakes, W & Baeza-Yates eds. Prentice-Hall, 1992. Upper Saddle River, New Jersey.

[Frakes92a] Frakes, W.; Baeza-Yates, R. *Information Retrieval. Data Structures & Algorithms.* Ed. Prentice Hall. 1992.

[Frakes92b] Frakes, W. *Stemming Algorithms.* En Information Retrieval. Data Structures & Algorithms. W. Frakes & R. Baeza-Yates editores. Págs. 131–160. Ed. Prentice Hall. 1992.

[Freeman93] Freeman, J; Skapura, D. Redes Neuronales. *Algoritmos, aplicaciones y técnicas de programación.* 1993. Ed. Addison-Wesley/Díaz de Santos.

[Genesereth92] Genesereth, M.; Fikes, R. *Knowledge Interchange Format Version 3.0 Reference Manual.* Technical Report Logic-92-1, Computer Science Department, Stanford University, June 1992. `http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps`

[Genesereth94a] Genesereth, M.; Singh, N. *A Knowledge Sharing Approach to Software Interoperation.* 1994. Computer Science Department, Stanford University.

[Genesereth94b] Genesereth, M.; Singh, N.; Syed, M. *A Distributed and Anonymous Knowledge Sharing Approach to Software Interoperation.* 1994. Computer Science Dept. Stanford University. Stanford.

[Genesereth97] Genesereth, M.; Ketchpel, S. *Software Agents.* Logic Group, Computer Science Departement. Stanford University, 1997.

[Gómez99] Gómez, Sergio A. *Una Taxonomía de Cambios para el Grafo de Espacios de Nombres Contextuales para Sistemas Multi-Agente en Contextos Múltiples.* V Congreso Argentino de Ciencias de la Computación. Tandil, 1999.

[Gómez01] Gómez, Sergio A.; Lanzarini, Laura. *Aplicación de Redes Neuronales al Filtrado de Documentos.* III Workshop de Investigadores en Ciencias de la Computación. WICC 2001, San Luis, 2001. Pp. 131-133.

[Glymour96] Glymour, C.; Madigan, D.; Pregibon, D.; Smyth, P. *Statistical Themes and Lessons for Data Mining.* Journal of Data Mining and Knowledge Discovery, 1, 25-42 (1996). Kluwer Academic Publishers, Boston. `http://bayes.stat.washington.edu/PAPERS/dami.ps`

[Gulbransen98] Gulbransen, D.; Rawlings, K. *HTML Dinámico. Edición Especial.* QUE, Prentice Hall. 1998.

[Hardy96] Darren R. Hardy; Michael F. Schwartz; Duane Wessels. *Harvest User's Manual. Version 1.4 patchlevel 2.* January 31, 1996. Technical Report CU-CS-743-94. Department of Computer Science. University of Colorado. Boulder, Colorado 80309-0430. `http://harvest.cs.colorado.edu/`

[Harman92] Harman, D. *Relevance Feedback and Other Query Modification Techniques.* In Information Retrieval. Págs. 241–262. Frakes, W. & Baeza-Yates, R. eds. Prentice-Hall, 1992. Upper Saddle River, New Jersey.

[Huhns97] Huhns, M.; Singh, M. *Readings in Agents.* 1997. Morgan Kaufmann Publishers, Inc.

[Hyötyniemi96] Hyötyniemi, H. *Text Document Classification with Self-Organizing Maps.* 1996. `http://www.hut.fi/~hhyotyni/HH3/HH3.ps`

[Kernigham85] Kernigham, B.; Ritchie, D. *El lenguaje de programación C.* Ed. Prentice-Hall Hispanoamericana, 1985.

[Kohonen00] Kohonen, T.; Kaski, S.; Lagus, K.; Salojärvi, J.; Honkela, J.; Paatero, Vesa; Saarela, Antti. *Self Organization of a Massive Document Collection.* IEEE Transactions on Neural Networks, Vol. 11, No. 3, May 2000, Pp. 574–585.

[Koster93] Martijn Koster. *Guidelines for Robot Writers.* 1993. `http://info.webcrawler.com/mak/projects/robots/guidelines.html`

[Kushmerick98] Nicholas Kushmerick. *Wrapper induction: Efficiency and expressiveness.* School of Computer Applications, Dublin City University Draft of May 1, 1998. `http://www.compapp.dcu.ie/~nick/research/download/kushmerick-wi-journal.ps`

[Lashkari97] Lashkari, Y.; Metral, M.; Maes, P. *Collaborative Interface Agents.* 1997. Readings in Agents, págs. 111–116. Morgan Kaufmann.

[Lavoie99] Lavoie, P.; Crespo, J.; Savaria, Y. *Generalization, Discrimination, and Multiple Categorization Using Adaptive Resonance Theory.* IEEE Transactions on Neural Networks, Vol. 10, No. 4, July, 1999. Publisher Item Identifier S 1045-9227(99)05271-8.

[Lloyd87] Lloyd, John. *Foundations of Logic Programming.* Springer-Verlag, 1987.

[Maes94] Maes, P. *Agents that Reduce Work and Information Overload.* Communications of the ACM. July, 1994. Vol. 17, No. 7.

[Maravall94] Maravall Gómez Allende, Darío. *Reconocimiento de Formas y Visión Artificial.* 1994. Ed. Addison-Wesley Iberoamericana.

[Mostafa97] Mostafa, J.; Mukhopadhyay, S.; Lam, W.; Palakal, M. *A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation.* ACM Transactions on Information Systems, Vol. 15, No. 4, October 1997, Pages 368–399. ACM 1046-8188/97/1000-0368. `/pubs/articles/journals/tois/1997-15-4/p368-mostafa/p368-mostafa.pdf`

[Pazzani97a] Pazzani, M.; Billsus, D. *Learning and Revising User Profiles: The Identification of Interesting Web Sites.* Machine Learning 27, 313-331. 1997. Ed. Kluwer Academic Publisers. `http://www.ics.uci.edu/~pazzani/Publications/SW-MLJ.pdf`

[Pazzani97b] Pazzani, M; Nguyen, L; Mantik, S. *Learning from hotlists and coldlists: Towards a WWW information filtering and seeking agent.* Department of Information and Computer Science. University of California, Irvine. 1997. `http://www.ics.uci.edu/~pazzani/TAI/ColdListfinal.html`

[Raggett98] Raggett, D.; Le Hors, A.; Jacobs, I. *HTML 4.0 Specification. W3C Recommendation.* 1998. `http://www.w3.org/TR/1998/REC-html40-19980424`

[Perkowitz96] Perkowitz, Mike; Etzioni, Oren. *Category Translation: Learning to understand information on the Internet.* Seattle, 1996. `ftp://ftp.cs.washington.edu/pub/ai/`

[Rao95] Rao, V.; Rao, H. *C++ Neural Networks and Fuzzy Logic, Second Edition.* MIS Press, 1995.

[Rasmussen92] Rasmussen, E. *Clustering Algorithms.* Information Retrieval. Data Structures & Algorithms. Pp. 419–442. 1992. In Frakes, W.; Baeza-Yates, R. (Eds.) Prentice Hall.

[Skapura96] Skapura, David. *Building Neural Networks.* ACM Press, Addison-Wesley. New York, 1996.

[Wang97] Wang Baldonado, Michelle; Winograd, Terry. *SenseMaker: An Information-Exploration Interface Supporting the Contextual Evolution of a User's Interests.* In Proceedings of the Conference on Human Factors in Computing Systems, pp. 11-18. ACM Press, New York, Atlanta, Ga. March, 1997. `http://www-diglib.stanford.edu/cgi-bin/WP/get/SIDL-WP-1996-0048`

[Wasserman89] Wasserman, P. *Neural Computing. Theory and Practice.* 1989. Ed. Anza Research.