Universidad de las Ciencias Informáticas

Facultad 10

Título: Propuesta del Módulo Decisor del Motor de Clasificación Inteligente por Contenidos (MOCIC).

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Autor: Indira Tamarit Muñoz
Ana Miranda Bermudez

Tutor: Ing. Kiuver Kaddiel Ibañez Castro

Ciudad de la Habana, 11 de junio de 2009

“Saber no es suficiente; tenemos que aplicarlo. Tener voluntad no es suficiente: tenemos que implementarla”

Johann Wolfgang von Goethe.

Declaración de autoria

Declaramos que somos los únicos autores del trabajo titulado:

_____

_____

y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Indira Tamarit Muñoz	Ana Miranda Bermudez
Autor	Autor

Ing. Kiuver Kaddiel Ibañez Castro

Tutor

Resumen

En la Universidad de las Ciencias Informáticas se está desarrollando, desde el año 2005, a petición de la Oficina de Seguridad para las Redes Informáticas, un sistema de Filtrado de Paquetes por Contenido que pretende regular (aceptar o denegar) el acceso de usuarios a determinados contenidos de Internet y brindarles una navegación segura, adaptándose a las normas y políticas de las instituciones en que pueda instalarse. Para el proceso de filtrado utiliza, principalmente, una Base de Datos de URLs Categorizadas, mediante la cual se toma la decisión de autorizar o denegar el acceso al los recursos solicitados y la Política de Uso Aceptable de Internet definida por la organización. Dado el gran volumen que posee Internet actualizar manualmente esta base de datos resulta imposible; para automatizar esta tarea surge el Motor de Categorización Inteligente por Contenidos, que es el encargado de automatizar el proceso de clasificación de documentos HTML provenientes de Internet. Para categorizar un documento HTML en su totalidad es necesario hacerlo primero de manera independiente, tomando el texto y las imágenes por separado y clasificándolos mediante los módulos especializados en realizar esta tarea, pero ese criterio no es suficiente, se necesita de un módulo que sea capaz de unificar todos esos resultados y determinar de manera ”inteligente”la categoría más probable a la que pertenece el documento HTML. En este documento se propone un módulo para clasificar de manera automática e “inteligente” los documentos HTML con vistas a su integración en el Motor de Categorización Inteligente por Contenidos. Su aplicación permitirá crear listas de URLs categorizadas, permitiendo mantener actualizada la base de datos de FILPACON y obtener/descartar (en línea) listas de URLs categorizadas, para clientes adscritos al servicio.

Palabras claves: Aprendizaje Automático, Redes Neuronales Artificiales, mecanismo de decisión.

Índice general

Índice de figuras

Índice de cuadros

Introducción

Internet se ha convertido en poco tiempo en la herramienta tecnológica más revolucionaria y poderosa de todas, influyendo prácticamente en todos los niveles de la actividad humana. El Internet o la denominada “Autopista de la información” es un sistema mundial de redes de computadoras, un conjunto integrado por las diferentes redes de cada país del mundo, que le permite a un usuario en cualquier computadora, en caso de contar con los permisos apropiados, acceder a los disímiles servicios que ofrece la “red de redes”, los cuales, por ser tan populares, cómodos e incluso gratis, propician un aumento desenfrenado de la cantidad de usuarios que ven en ellos la necesidad de utilizarlos para obtener gran cantidad de información. Se calcula que existen en el mundo más de 6000 millones de personas que acceden a Internet, consultando y aportando en esta gigantesca “biblioteca” mundial, que es considerada además como una fuente inagotable del conocimiento humano.

La World Wide Web (WWW) es tal vez el punto más visible de Internet y hoy en día el más usado. La WWW puede definirse básicamente como tres cosas: hipertexto, que es un sistema de enlaces que permite saltar de unos lugares a otros; multimedia, que hace referencia al tipo de contenidos que puede manejar (textos, gráficos, vídeos, sonidos y otros) e Internet, la base sobre la que se transmite la información [2].

En la Web se encuentra la mayor fuente de información que haya existido jamás. Sus contenidos son inagotables y prácticamente no hay ningún tema sobre el que no pueda encontrarse información exhaustiva. Millones de páginas están permanentemente a disposición de las personas que quieran consultarlas, lo que convierte a la Web en una herramienta de valor incalculable.

La Red de Redes y específicamente la Web tiene otra gran ventaja, como es la libertad de expresión. Gracias a ello, la información que puede encontrarse tiene un carácter libre y plural, lo cual -teóricamente-

supone un valor añadido. Sin embargo, como en cualquier otro medio, hay personas que hacen mal uso de tal libertad, lo que puede resultar extremadamente peligroso. Contenidos sobre terrorismo, pedofilia, inducción al suicidio, violencia descontrolada, aberraciones sexuales, pornografía, etc., también forman parte de Internet. La cuestión es: ¿resulta conveniente que todo el mundo pueda acceder a todo tipo de contenidos?

Pese a que algunos países han tratado de regular los contenidos a los que los usuarios pueden tener o no acceso, lo cierto es que, dado el carácter global de Internet, la tarea resulta difícil; esto implica la necesidad de utilizar herramientas específicas y locales. Los sistemas de filtrado de contenido son una solución técnica utilizada para regular el acceso a materiales inadecuados en la red, entre los más reconocidos a escala mundial se encuentran OPTENET, IBM Proventia Web Filter, Netsweeper y POESIA, siendo este último el único libre.

Actualmente en nuestro país existen más de 1 400 000 usuarios de redes informáticas, incluyendo en esta estadística a aquellas personas que hacen uso del correo electrónico, ya sea de alcance nacional o internacional, los que navegan por la Red Cuba o quienes acceden plenamente a Internet [1], pero toda nueva tecnología trae consigo riesgos y por tanto es necesario mantener un estricto control para evitar el intercambio y diseminación de información nociva; es por ello que a petición de la Oficina de Seguridad para las Redes Informáticas (OSRI), en la Universidad de las Ciencias Informáticas (UCI) se está desarrollando un proyecto llamado Filtrado de Paquetes por Contenido (FILPACON).

FILPACON es un sistema de filtrado Web que permite regular (aceptar o denegar) el acceso de usuarios a determinados contenidos de Internet y brindarles una navegación segura, adaptándose a las normas y políticas de las instituciones en que pueda instalarse. Para el proceso de filtrado utiliza, principalmente, una Base de Datos de URLs Categorizadas (BDUC) y la Política de Uso Aceptable de Internet definida por la organización.

Dado el gran volumen que posee Internet actualizar manualmente la BDUC es una tarea prácticamente inviable. Para automatizar este proceso se utilizará el Motor de Categorización Inteligente por Contenidos (MOCIC), que se encuentra en desarrollo.

MOCIC está compuesto por un conjunto de módulos que mayoritariamente estarán dotados de Inteligencia Artificial (IA), que de forma unida, permitirán automatizar el proceso de clasificación de contenidos y será utilizado para categorizar documentos HTML provenientes de Internet.

Para categorizar un documento HTML en su totalidad es necesario hacerlo primero de manera independiente, tomando el texto y las imágenes por separado y clasificándolos mediante los módulos especializados en realizar esta tarea. Estos devuelven una respuesta al Controlador con la categoría del contenido, pero esta información no es suficiente, para categorizar el documento HTML se necesita de un módulo que sea capaz de unificar todos esos resultados y determinar de manera ”inteligente”la categoría más probable a la que pertenece el documento HTML y el encargado de realizar ese proceso será el Decisor.

La situación problémica está dada por:

- Se necesita de una herramienta informática que de manera automática e “inteligente”, sea capaz de clasificar los documentos HTML, para fundamentalmente crear listas de URLs categorizadas, permitiendo mantener actualizada la base de datos de FILPACON y obtener/descartar (en línea) listas de URLs categorizadas, para clientes adscritos al servicio.
- Se requiere del Módulo Decisor para el funcionamiento de MOCIC.

Surgiendo como problema científico: ¿Qué mecanismo de decisión utilizar para que MOCIC pueda categorizar los documentos HTML de manera automática e “inteligente”?

Para darle solución a este problema se plantea como idea a defender que empleando los mecanismos de decisión para el Módulo Decisor de MOCIC, este clasificará de manera automática e “inteligente” los documentos HTML.

Se enmarcó el objeto de estudio en el mecanismo de decisión del sistema de filtrado POESIA y como campo de acción los Algoritmos de Aprendizaje Automático para la clasificación de documentos HTML.

En el presente trabajo se plantea como objetivo general proponer el diseño del Módulo Decisor de MOCIC, para categorizar los documentos HTML automática e “inteligentemente”, desglosándose en los siguientes objetivos específicos:

- Estudiar el estado del arte del mecanismo decisor del sistema de filtrado POESIA.
- Definir el funcionamiento del Módulo Decisor.
- Evaluar la efectividad de los algoritmos seleccionados.

Para darle cumplimiento a los objetivos planteados se han definido las siguientes tareas investigativas:

- Estudio del mecanismo de decisión de POESIA.
- Estudio de Algoritmos de Aprendizaje Automático para la solución de problemas de categorización.
- Estudio de la integración del Módulo Decisor con MOCIC.
- Implementación de un prototipo funcional.
- Estudio de las funciones para la evaluación de la clasificación de los algoritmos seleccionados.

Diseño metodológico

Para el desarrollo de la presente investigación se emplearon los siguientes métodos:

Métodos teóricos

Analítico-Sintético: Ayudó a procesar el marco referencial de la tesis a partir de la sistematización del conocimiento científico relacionado con el objeto de estudio, permitió reconocer las múltiples relaciones y componentes del problema abordado por separado, para luego integrarlas en un todo como se presenta en la realidad y fue la vía a través de la cual se realizó la interpretación de la información recogida a través de la aplicación de los instrumentos que se seleccionaron a fin de poderse llegar a las conclusiones correspondientes.

Inductivo-Deductivo: Aportó la determinación del problema y la diferenciación de las tareas desarrolladas en el proceso investigativo permitiendo que a partir del estudio de la bibliografía referente a los mecanismos decisores de los sistemas de filtrado autónomos, se llegara a establecer lo común en ellos y a partir de allí se logró proceder a la implementación del prototipo funcional que permitió la evaluación de los algoritmos seleccionados. Además proporcionó el establecimiento de las

relaciones entre los hechos analizados y las explicaciones y conclusiones a las que se arribó en la presente investigación.

Histórico-Lógico: Permitió que se analizara el desarrollo histórico del objeto de estudio y encontrar la lógica interna del desarrollo, así como todas las publicaciones posibles editadas en Cuba y en el extranjero sobe los mecanismos decisores de los sistemas de filtrado autónomos, el empleo de Algoritmos de Clasificación Supervisada en la clasificación de documentos HTML, funciones para la evaluación de la efectividad de los algoritmos de clasificación y las bibliotecas libres existentes para este fin, todo ello contextualizado en el tiempo.

Modelación: Este se utilizó para diseñar un prototipo funcional de forma que reflejara lo mejor posible la realidad que se presenta en el proceso de decisión.

Métodos empíricos

Medición: Mediante este método se realizó la determinación de las diferentes variables que permitieron realizar la evaluación de la efectividad de los algoritmos analizados mediante el prototipo funcional propuesto.

Estructura del contenido

El documento se encuentra organizado en dos capítulos tal como se describe a continuación:

Capítulo 1: *Fundamentación Teórica*. En este capítulo se realiza un estudio del mecanismo de Decisión del sistema de filtrado POESIA. Se abordarán temas referentes al Aprendizaje Automático y dentro de este al Aprendizaje Supervisado y No Supervisado. Se describirá el funcionamiento de los Algoritmos de Aprendizaje Automático más utilizados en la clasificación de documentos HTML. Se analizarán las funciones para la evaluación de la clasificación y se comentará sobre las herramientas utilizadas a lo largo de la investigación.

Capítulo 2: *Diseño del Módulo Decisor*. En este capítulo se realiza una descripción de la arquitectura de MOCIC, con el fin de aclarar la integración del Módulo Decisor en el mismo. Se conforma el vector de características mediante la respuesta de los módulos de clasificación y se describe la forma en que serán normalizados los datos. Se proponen los algoritmos de clasificación que serán utilizados y se

selecciona la colección de entrenamiento para entrenar el sistema. A partir de la implementación del prototipo funcional se evalúan los algoritmos propuestos y se prueba la integración del Módulo Decisor con el sistema.

Capítulo 1

Fundamentación teórica

1.1. Introducción

En este capítulo se presenta una breve revisión del mecanismo de decisión del sistema de filtrado POESIA. Se fija el marco teórico para poder establecer definiciones formales y presentar aspectos relacionados con el aprendizaje automático. Se analizan en detalle los algoritmos más empleados en la solución de problemas de clasificación. Se dan a conocer los parámetros que se tendrán en cuenta para la evaluación de la clasificación y las herramientas que servirán de apoyo en la implementación del prototipo funcional, evaluación de la efectividad y la confección de este documento.

1.2. Estado del arte

Un filtro de contenido es uno o más elementos de software que operan juntos para regular (permitir o denegar) el acceso de los usuarios a determinados materiales que se encuentran en Internet. FILPACON constituye una solución de este tipo, uno de sus componentes fundamentales es MOCIC, que realiza la función de recuperación y clasificación de la información, que actualmente se encuentra en desarrollo. Ante la necesidad de un Módulo Decisor en MOCIC, que será el encargado de entregar un resultado de categorización combinando varios factores, entre ellos el resultado de algunos módulos que componen el motor (Clasificador de Enlaces, Clasificador de Textos, Clasificador de Desnudez, Clasificador de Rostros, Clasificador de Objetos, Reconocimiento Óptico de Caracteres), se precisa del estudio de soluciones similares que puedan existir para dicho módulo.

1.2.1. Ámbito internacional

En el marco particular de los sistemas de filtrado autónomos, la existencia de un motor de recuperación y clasificación de la información y por tanto de un módulo decisor como parte fundamental del mismo, se hace necesario para llenar sus base de datos.

Estos sistemas de filtrado utilizan técnicas de IA lo que reduce la dependencia de las listas de URLs, logrando que el filtro en cuestión se adapte a la naturaleza dinámica de Internet. A continuación se muestra una pequeña lista de algunos sistemas de filtrado autónomos:

- POESIA [1]
- OPTENET [2]
- IBM Proventia Web Filter [3]
- Netsweeper [4]
- CyberPatrol [5]

Con la excepción de POESIA los filtros mencionados son software privativos, presentando las siguientes desventajas:

- El motor de recuperación y clasificación de la información no es un producto que está separado del filtro.
- Son diseñados como una solución a la medida del propio filtro, imposibilitando el uso de estos en otro entorno.
- Al no tener acceso al código fuente es imposible su mantenimiento.
- Alta dependencia de los propietarios.

[1] http://www.poesia-filter.org
[2] http://www.optenet.com/es/webfilter.asp?c=1
[3] http://www.virtual.com/pages/partners_iss.aspx
[4] http://www.netsweeper.com/Developers/Categorization
[5] http://www.cyberpatrol.com

POESIA es un proyecto diseñado para ser aplicado en el sistema operativo GNU/Linux bajo la licencia libre GPL. Dicho proyecto ha supuesto la creación de un filtro de contenidos pornográficos, violentos y xenófobos basado en el filtrado tanto de contenidos textuales como de las imágenes.

El mecanismo de decisión es el último eslabón de la cadena del proceso de filtrado, este agrupa la información proveniente de los diferentes componentes especializados y debe dar la última palabra sobre la aprobación de la página solicitada.

Las entradas al mecanismo de decisión las constituyen las salidas de los demás componentes, las cuales provienen de:

- Componente de Filtrado de lenguaje.
- Componente de Filtrado de imágenes.
- Componente de Filtrado de PICS.
- Componente de Filtrado de JavaScript.
- Componente de Filtrado de URL.

No es tarea de los componentes de filtrado enviar mensajes al mecanismo de decisión, realmente, es una tarea del componente orientador enviar al mecanismo de decisión las salidas recibidas de los otros componentes.

El objetivo del sistema de filtrado está basado en las siguientes características:

1. fiabilidad
2. rapidez
3. configurable

Dado que los dos primeros puntos son obviamente claros, el último punto será explicado: diferentes usuarios pueden tener diferentes necesidades en relación a qué debe ser filtrado, por eso al mecanismo de filtrado no se le pueden aplicar reglas de filtrado generales, sino que deben ser flexibles y adaptables. Más que nada el mecanismo de filtrado debe desarrollarse de acuerdo al uso y las necesidades del

usuario final, por lo tanto debemos elegir implementar el mecanismo de decisión mediante plugin, lo que significa que puede ser posible y simple hacerlo, adicionando en un nuevo estado un nuevo mecanismo de decisión, implementado con un estilo de filtrado diferente.

El punto más importante del mecanismo de decisión es el manejo de la falta de sincronización de la respuesta proveniente de los componentes. A primera vista, esto puede amenazar la posibilidad de tener una red neuronal para implementar la lógica del mecanismo de decisión, ya que esperar por todos los filtrados para completar su evaluación puede significar una demora considerable, es por esto que el equipo de desarrollo de POESIA reconoce la necesidad de que el mecanismo decisor de una respuesta aun cuando no todos los componentes de filtrado hayan concluido el proceso.

Para respetar la fiabilidad requerida al menos uno de los estilos de filtrado será implementado usando uno de los modelos estándar de aprendizaje automático.

Los mecanismos que se deben elegir para el diseño e implementación, son los plugin siguientes.

- Mecanismo de decisión basado en la teoría Bayesiana.
- Mecanismo de decisión basado en redes neuronales.
- Mecanismo de decisión basado en reglas.

A pesar de todas las ventajas que nos brinda POESIA este tiene la desventaja que su última versión disponible data del año 2006, lo que supone un bajo mantenimiento del software y su mecanismo de decisión no se ajusta a la arquitectura de MOCIC, por lo que no puede ser reutilizable [2].

1.2.2. Ámbito nacional

En nuestro país, FILPACON constituye la única solución de su tipo, implicando novedad en cada uno de sus componentes. No se tienen referencias de un motor de clasificación inteligente de contenidos basado en técnicas de IA, desarrollado por ninguna institución del país. Por lo antes expresado se llega a la conclusión de que a escala nacional no existe una solución a nuestro problema.

1.3. Aprendizaje automático

Hasta finales de los años 80 del siglo pasado, el enfoque más popular dentro de la clasificación automática era el de la “Ingeniería del Conocimiento”, consistente en la definición manual de un conjunto de reglas que codificaban el conocimiento experto humano sobre como clasificar un documento dado un conjunto prefijado de clases. En la década siguiente este enfoque fue perdiendo popularidad, especialmente en la comunidad científica, en favor del paradigma del aprendizaje automático [3].

El Aprendizaje Automático es una rama de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento. En muchas ocasiones el campo de actuación del Aprendizaje Automático se solapa con el de la Estadística, ya que las dos disciplinas se basan en el análisis de datos.

El Aprendizaje Automático puede definirse como la disciplina que permite desarrollar sistemas capaces de realizar una tarea de un modo automático y de forma que el desempeño de dicha tarea resulte mejor con experiencia que sin ella. El aprendizaje automático consiste en la programación de sistemas informáticos de forma que se optimice un determinado criterio de rendimiento, empleando datos de entrenamiento o con ayuda de una experiencia pasada. Se define un modelo a partir de un conjunto de parámetros y, mediante un aprendizaje, se ejecuta un programa que optimiza dichos parámetros, usando un conjunto de datos que representan experiencia pasada.

Las ventajas que aporta el aprendizaje automático a la clasificación son unas tasas de precisión comparables a las ofrecidas por un experto humano, así como un ahorro en términos de trabajo potencial humano, ya que no se requiere del conocimiento de un experto en un determinado tema para la construcción de un clasificador, ni en el caso en que se desee cambiar el conjunto de categorías.

En general existen dos enfoques principales dentro del aprendizaje automático: a) aprender para poder generar un nuevo conocimiento o comportamientos para un determinado sistema y, b) aprender para tratar de mejorar el comportamiento de un sistema. En el primer caso se suelen utilizar técnicas basadas

en razonamiento inductivo, mientras que la segunda suele estar relacionada con la utilización de técnicas analíticas. Ambos enfoques pueden emplearse también conjuntamente [3].

El Aprendizaje Automático tiene una amplia gama de aplicaciones, incluyendo motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, juegos y robótica.

Dentro del aprendizaje automático, y dependiendo de si se dispone o no de datos etiquetados, se puede distinguir entre: aprendizaje supervisado y no supervisado.

1.3.1. Aprendizaje supervisado

El aprendizaje supervisado se construye sobre un conocimiento a priori. En la clasificación de documentos HTML se debe disponer de una colección de entrenamiento (conjunto de documentos HTML de ejemplo para cada una de las categorías en las que se quiere clasificar). Después de una etapa de entrenamiento, el sistema queda ajustado de tal modo que ante nuevos ejemplos, el algoritmo es capaz de clasificarlos en alguna de las categorías existentes. Cuanto mayor sea el conjunto de datos etiquetados (colección de entrenamiento) mayor será la información potencial disponible y, previsiblemente, mejor resultará el aprendizaje [3].

La *clasificación* puede ser formalizada como la tarea de aproximar una *función objetivo* desconocida $\Phi : I \times C \rightarrow \{T, F\}$ (que describe cómo instancias del problema deben ser clasificadas de acuerdo a un experto en el dominio) por medio de una función $\Theta : I \times C \rightarrow \{T, F\}$ llamada el *clasificador*, donde $C = \{c_1, \ldots, c_{|c|}\}$ es un conjunto de categorías predefinido, e I es un conjunto de instancias del problema.

Comúnmente cada instancia $i_j \in I$ es representada como una lista $A = \langle a_1, a_2, \ldots, a_{|A|} \rangle$ de valores característicos, conocidos como *atributos*, i.e. $i_j = \langle a_{1j}, a_{2j}, \ldots, a_{|A|j} \rangle$. Si $\Phi : i_j \times c_i \rightarrow T$, entonces i_j es llamado un *ejemplo positivo* de c_i, mientras si $\Phi : i_j \times c_i \rightarrow F$ éste es llamado un ejemplo negativo de c_i.

Para generar automáticamente el clasificador de c_i es necesario un proceso inductivo, llamado el *aprendiz*, el cual por observar los atributos de un conjunto de instancias preclasificadas bajo c_i o $\bar{c}_i$, adquiere los atributos que una instancia no vista debe tener para pertenecer a la categoría. Por tal motivo, en la

construcción del clasificador se requiere la disponibilidad inicial de una colección Ω de ejemplos tales que el valor de $\Phi(i_j, c_i)$ es conocido para cada $\langle i_j, c_i \rangle \in \Phi \times C$. A la colección usualmente se le llama *conjunto de entrenamiento* (Tr). En resumen, al proceso anterior se le identifica como *aprendizaje supervisado* debido a la dependencia de Tr [4].

1.3.2. Aprendizaje no supervisado

La clasificación no supervisada consiste en realizar una división sin contar con un tutor o información de apoyo, es decir, se realiza el proceso automático extrayendo características comunes y/o relaciones existentes. Este tipo de clasificación también se denomina auto-adaptativa ya que se va modificando de forma tal que se adecua al conjunto de datos de entrada del que parte.

Los modos de esta clasificación son búsqueda de:

- **Nivel de semejanza entre conceptos** buscando la cercanía de los mismos. Aplicada a la recuperación de información. Significa buscar comparación entre los conceptos para estructurar la propia información.
- **Extracción de características y sus relaciones**, indagar en las características que existen en los conceptos y sus relaciones. Dichos conceptos parecidos deberían dar los mismos resultados.
- **Analizar y extraer principales características**, escoger las peculiaridades necesarias y fundamentales que nos den la suficiente información; para la recuperación es importante ya que se restringen los datos a tener en cuenta para realizar la recuperación.
- **Modelización**, tratar de agrupar y buscar el prototipo que represente a todo un grupo.

El modelo básico en el que se basa la no supervisión:

- **Regla de Hebb**: la conexión entre dos neuronas conectadas es mayor si ambas están activadas y menor si no lo están.
- **Interacción lateral**: cada neurona tiene relación excitatoria con un vecindario cercano e inhibitoria con uno lejano imitando el cerebro.

El aprendizaje y entrenamiento se basa en los siguientes conceptos:

- **Auto-adaptación**: a través de la aplicación de los algoritmos se van modificando sus parámetros para resolver el problema.
- **Competición**: potenciar a los ganadores en detrimento de los perdedores.

Las ventajas y desventajas de la no supervisión son:

- Encontrar la solución innovadora.
- No encontrar una buena solución.
- Dificultad en la modelización y parametrización [5].

1.4. Algoritmos basados en Redes Neuronales Artificiales

Las redes neuronales consisten normalmente en un número de elementos de procesamiento o neuronas interconectadas. Las conexiones son arreglos entre neuronas y la naturaleza de éstas determina la estructura de la red. Como la fortaleza de las conexiones es ajustada o entrenada para alcanzar un comportamiento deseado, la red es gobernada por sus algoritmos de aprendizaje. Las redes neuronales pueden ser clasificadas de acuerdo a sus estructuras o algoritmos de aprendizaje [6].

Ventajas de las Redes Neuronales

La principal razón del uso de las redes neuronales radica en el gran número de aplicaciones exitosas. El éxito en las aplicaciones se debe principalmente a las ventajas que las redes neuronales tienen sobre otro tipo de modelos computacionales. A continuación se mencionan algunas de estas ventajas:

Aprendizaje Adaptable: La capacidad adaptable es una de las características más atractivas de las redes neuronales. Aprenden a llevar a cabo ciertas tareas mediante un entrenamiento con ejemplos ilustrativos. Como las redes neuronales pueden aprender a diferenciar patrones mediante ejemplos y entrenamiento, no es necesarios que elaboremos modelos a priori ni necesitamos especificar funciones de distribución de probabilidad.

Tolerancia a fallos: Las redes neuronales son los primeros métodos computacionales con la capacidad inherente de tolerancia a fallos. Comparados con los sistemas computacionales tradicionales, los cuales pierden su funcionalidad en cuanto sufren un pequeño error de memoria, en las redes neuronales, si se produce un fallo en un pequeño número de neuronas, aunque el comportamiento del sistema se ve influenciado, no sufre una caída repentina. Hay dos aspectos distintos respecto a la tolerancia a fallos: primero, las redes pueden aprender a reconocer los patrones con ruido, distorsionados o incompletos, esta es una tolerancia a fallos respecto a los datos. Segundo, pueden seguir realizando su función (con cierta degradación) aunque se destruya parte de la red. La razón por la que las redes neuronales son tolerantes a fallos es que tienen su información distribuida en las conexiones entre neuronas, existiendo cierto grado de redundancia en este tipo de almacenamiento. La mayoría de los ordenadores algorítmicos y sistemas de recuperación de datos almacenan cada pieza de información en un espacio único, localizable y direccionable. Las redes neuronales almacenan información no localizada. Por tanto, la mayoría de las interconexiones entre los nodos de la red tendrán unos valores en función de los estímulos recibidos, y se generará un patrón de salida que represente la información almacenada.

Operación en tiempo real: Una de las prioridades de la mayoría de las áreas de aplicación es la necesidad de realizar grandes procesos con datos de forma muy rápida. Las redes neuronales se adaptan bien a esto debido a su implementación paralela. Para que la mayoría de las redes neuronales puedan operar en un entorno de tiempo real, la necesidad de cambio de los pesos de las conexiones o entrenamiento es mínima. Por tanto, las redes neuronales son una excelente alternativa para el reconocimiento y clasificación de patrones en tiempo real.

Fácil inserción dentro de la tecnología existente: Debido a que una red puede ser rápidamente entrenada, comprobada, verificada y trasladada a una implementación hardware de bajo costo, es fácil insertar redes neuronales para aplicaciones específicas dentro de sistemas existentes [7].

Aplicaciones

La aplicación de las redes neuronales artificiales se ha extendido a diversas disciplinas, debido principalmente a sus características de adaptabilidad, confiabilidad y auto-organización. Una ventaja destacable para el procesamiento de datos radica en la posibilidad de su implementación con tecnologías específicas disponibles, que posibilitan su operación en tiempo real.

Las redes neuronales han sido utilizadas en problemas de categorización en numerosas ocasiones. Es posible entrenar una red neuronal para que dada una entrada determinada (un vector de representación) produzca una salida deseada (la categoría a la que corresponde ese documento) [3].

1.4.1. Multilayer Perceptron

El perceptrón multicapa es una red neuronal artificial formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón (también llamado perceptrón simple). El perceptrón multicapa puede ser totalmente o localmente conectado.

En el primer caso cada salida de una neurona de la capa i es entrada de todas las neuronas de la capa $i+1$, mientras que el segundo, cada neurona de la capa i es entrada de una serie de neuronas (región) de la capa $i+1$. Las capas pueden clasificarse en tres tipos:

- Capa de entrada: Constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se produce procesamiento.
- Capas ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y las salidas pasan a neuronas de capas posteriores.
- Capa de salida: Neuronas cuyos valores de salida se corresponden con las salidas de toda la red.

La propagación hacia atrás (también conocido como retropropagación del error o regla delta generalizada), es un algoritmo utilizado en el entrenamiento de estas redes, por ello, el perceptrón multicapa también es conocido como red de retropropagación.

Características

El Multilayer Perceptron se caracteriza por presentar una no-linealidad en la salida, capas de neuronas ocultas y un alto grado de conectividad. Es de entrenamiento supervisado. Utiliza el algoritmo de retropropagación del error, que está basado en la regla de aprendizaje por corrección de error, considerada como una generalización del algoritmo de los cuadrados mínimos (LMS), utilizado en filtrado adaptivo mediante redes lineales simples.

Su operación consta de dos fases, una directa y una inversa o de retroceso. En la fase directa, se ingresa el patrón de actividad en la capa de entrada de la red (vector de entrada), que recorre todas las capas subsiguientes. Se obtiene la respuesta real de la red en la capa de salida. En esta fase, los pesos sinápticos de la red permanecen fijos. En la fase inversa, los pesos sinápticos son ajustados de acuerdo con la regla de corrección del error. Esta regla, conocida como método de Levenberg-Marquardt, minimiza el cuadrado de las diferencias entre la respuesta o salida deseada y la salida real de la red [8].

Limitaciones

El Perceptrón Multicapa no extrapola bien, es decir, si la red se entrena mal o de manera insuficiente, las salidas pueden ser imprecisas.

La existencia de mínimos locales en la función de error dificulta considerablemente el entrenamiento, pues una vez alcanzado un mínimo, el entrenamiento se detiene aunque no se haya alcanzado la tasa de convergencia fijada.

Cuando caemos en un mínimo local sin satisfacer el porcentaje de error permitido se puede considerar: cambiar la topología de la red (número de capas y número de neuronas), comenzar el entrenamiento con unos pesos iniciales diferentes, modificar los parámetros de aprendizaje, modificar el conjunto de entrenamiento o presentar los patrones en otro orden.

Aplicaciones

El perceptrón multicapa se utiliza para resolver problemas de asociación de patrones, segmentación de imágenes, compresión de datos, etc.

1.4.2. Modelo de Kohonen

Existen evidencias que demuestran que en el cerebro hay neuronas que se organizan en muchas zonas, de forma que las informaciones captadas del entorno a través de los órganos sensoriales se representan internamente en forma de mapas bidimensionales.

Aunque en gran medida esta organización neuronal está predeterminada genéticamente, es probable que parte de ella se origine mediante el aprendizaje, ésto sugiere que el cerebro podría poseer la capacidad inherente de formar mapas topológicos de las informaciones recibidas del exterior, de hecho esta teoría podría explicar su poder de operar con elementos semánticos: algunas áreas del cerebro simplemente podrían crear y ordenar neuronas especializadas o grupos con características de alto nivel y sus combinaciones, en definitiva se construirían mapas especiales para atributos y características.

A partir de estas ideas Tuevo Kohonen presentó en 1982 un sistema con un comportamiento semejante, se trataba de un modelo de red neuronal con capacidad para formar mapas de características de manera similar a como ocurre en el cerebro; el objetivo de Kohonen era demostrar que un estímulo externo (información de entrada) por sí solo, suponiendo una estructura propia y una descripción funcional del comportamiento de la red, era suficiente para forzar la formación de los mapas.

Este modelo tiene dos variantes denominadas Learning Vector Quantization (LVQ) y Topology Preserving Map (TPM) o Self Organizing Map (SOM), ambas se basan en el principio de formación de mapas topológicos para establecer características comunes entre las informaciones (vectores) de entrada a la red, aunque difieren en las dimensiones de éstos, siendo de una sola dimensión en el caso de LVQ y bidimensional o tridimensional en la red SOM.

El aprendizaje en el modelo de Kohonen es de tipo Off-line, por lo que se distingue una etapa de aprendizaje y otra de funcionamiento. En la etapa de aprendizaje se fijan los valores de las conexiones (feedforward) entre la capa de entrada y la salida. Esta red utiliza un aprendizaje no supervisado de tipo competitivo, las neuronas de la capa de salida compiten por activarse y sólo una de ellas permanece activa ante una determinada información de entrada a la red, los pesos de las conexiones se ajustan en función de la neurona que haya resultado vencedora.

Durante la etapa de entrenamiento, se presenta a la red un conjunto de informaciones de entrada (vectores de entrenamiento) para que ésta establezca en función de la semejanza entre los datos las diferentes categorías (una por neurona de salida), que servirían durante la fase de funcionamiento para realizar clasificaciones de nuevos datos que se presenten a la red. Los valores finales de los pesos de las conexiones entre cada neurona de la capa de salida con las de entrada se corresponderán con los valores de los componentes del vector de aprendizaje que consigue activar la neurona correspondiente. En el

caso de existir más patrones de entrenamiento que neuronas de salida, más de uno deberá asociarse con la misma neurona, es decir pertenecerán a la misma clase.

En este modelo el aprendizaje no concluye después de presentarle una vez todos los patrones de entrada, sino que habrá que repetir el proceso varías veces para refinar el mapa topológico de salida, de tal forma que cuantas más veces se presenten los datos, tanto más se reducirán las zonas de neuronas que se deben activar ante entradas parecidas, consiguiendo que la red pueda realizar una clasificación más selectiva [9].

Clustering

El clustering o agrupamiento es partir el grupo de entrada en pequeños grupos, para este particionamiento se usan distancias. Tiene en cuenta todas las características de los datos. Parte de patrones (observaciones) sin etiquetar y un número de prototipos definidos (por el diseñador del clasificador).

- El espacio de entrada se divide en k clases y k prototipos.
- Mueve los prototipos una vez realizado el aprendizaje con todas las observaciones.

El objetivo de este algoritmo es intentar situar los prototipos de forma tal que aquellos patrones cercanos (distancia euclídea) sean similares entre sí.

Las distancias más utilizadas son:

- Euclídea, la más común.
- Manhattan.
- Hamming.

A través del cálculo de distancias se agrupan los elementos de acuerdo a los más cercanos según el cálculo. El clustering es utilizado en la recuperación y extracción de información para realizar agrupamientos de conceptos cercanos de manera que tratando el grupo, el resultado sea similar.

Ante un nuevo dato el sistema simplemente lo comparará con los prototipos y lo clasificará según la clase definida para el más cercano. Entrenamiento potencialmente lento y clasificación rápida [5].

Self Organizing Map

Se cree que algunos sistemas biológicos realizan sus operaciones siguiendo un método de trabajo que algunos investigadores han llamado, on-center/off-surround; este término describe un patrón de conexión entre neuronas, cada neurona se refuerza a ella misma (center) mientras inhibe a todas las neuronas a su alrededor (surround). En las redes competitivas biológicas, lo que sucede realmente es que cuando una neurona se refuerza a ella misma, refuerza también las neuronas que están cerca; la transición entre reforzar las neuronas "vecinas" o inhibirlas, se realiza suavemente a medida que la distancia entre las neuronas aumenta.

Tratando de emular la actividad biológica, sin tener que implementar conexiones on-center/off-surround; de realimentación no lineal, Kohonen diseñó la red conocida como Self Organizing Map. Esta red determina primero la neurona ganadora i^* usando el mismo procedimiento que las redes competitivas, luego los vectores de pesos de todas las neuronas que se encuentren en una región cercana "vecindario", serán actualizados mediante la regla de Kohonen.

$$iw(q) = iw(q-1) + \alpha(p(q) - iw(q-1))para i \in N_{i*}(d) \tag{1.1}$$

Donde el vecindario N_i^* contiene el índice para todas las neuronas que se encuentren a un radio "d" de la neurona ganadora i^*.

Cuando un vector p es presentado, los pesos de la neurona ganadora y de sus vecinas tenderán hacia p, el resultado es que después de muchas presentaciones las neuronas vecinas habrán aprendido vectores similares que cada una de las otras.

El vecindario puede determinarse en diferentes formas; Kohonen, por ejemplo ha sugerido vecindarios rectangulares o hexagonales para lograr alta eficiencia; es importante destacar que el rendimiento de la red no es realmente sensitivo a la forma exacta del vecindario [9].

Características del mapa

- Para que el mapa clasifique su tamaño debe ser reducido.

- El número de clases suele ser inferior al tamaño del mapa ya que existen neuronas que no son ganadoras para ningún elemento de la data.
- Pueden detectarse observaciones aisladas (outliers). Éstas son atraídas en forma aislada por una neurona e inclusive puede pertenecer a una neurona se parada de las restantes en el mapa.
- Establece proximidad entre las clases y en consecuencia es posible unificar clases cercanas en el caso de que el número de clases sea excesivo. Los prototipos se desplazan según se van procesando los ejemplos (al contrario que en k-medias).
- El aprendizaje es más rápido y eficaz.
- No requiere aprendizaje en el procesamiento de datos nuevos.

Learning Vector Quantization

Esta red es un híbrido que emplea tanto aprendizaje no supervisado, como aprendizaje supervisado para clasificación de patrones [9].

El algoritmo LVQ es un método de clasificación basado en el aprendizaje competitivo neuronal, que permite reforzar positivamente (premiando) o negativamente (castigando) los pesos de las conexiones, dependiendo de que la clasificación haya sido realizada correcta o incorrectamente. LVQ usa un aprendizaje supervisado para definir regiones de clases en el espacio de los datos de entrada. Con este propósito, un subconjunto de vectores de similitud etiquetados de forma similar forman una región de una clase [10].

La arquitectura de la red LVQ presenta dos capas con N neuronas de entrada y M de salida. Cada una de las N neuronas de entrada se conecta a las M de salida hacia delante (feedforward). Entre las neuronas de la capa de salida, existen conexiones laterales de inhibición (peso negativo). El valor que se asigne a los pesos de las conexiones feedforward entre las capas de entrada y salida durante el proceso de aprendizaje de la red, va a depender de esta interacción lateral. El aprendizaje es de tipo competitivo. Las neuronas de la salida compiten por activarse y sólo una de ellas permanece activa ante una determinada información de entrada a la red. Esta neurona se denomina prevalente o vencedora, y en función de ella se ajustan los pesos de las conexiones [8].

Dada una secuencia de datos de entrada, se selecciona un grupo de vectores de referencia W_k (codebook). En cada iteración, se selecciona un vector de entrada X_i y se actualiza el vector W, para ajustar X_i de la mejor manera. El algoritmo LVQ trabaja de la siguiente manera:

A cada clase, k, se le asocia un vector de peso W_k. En cada iteración, el algoritmo selecciona un vector de entrada, X_i, y lo compara con cada vector de pesos, w_k, usando la distancia euclídea $||X_i - W_k||$; el vector W_c será el ganador si es el más cercano a X_i, por lo que c será la clase asignada:

$$||X_i - W_c|| = \min_k \{||X_i - W_k||\}$$

Las clases compiten entre ellas para encontrar el vector de entrada más parecido, para que el ganador sea el que menor distancia euclídea tenga respecto al vector de entrada. Sólo la clase ganadora podrá modificar el vector de pesos usando un algoritmo de aprendizaje reforzado, o positivo o negativo, dependiendo de que la clasificación sea correcta o no. De este modo, si la clase ganadora pertenece a la misma clase que el vector de entrada (la clasificación ha sido correcta), se incrementará el peso, acercándose ligeramente al vector de entrada (premio). Por el contrario, si la clase ganadora es diferente a la clase del vector de entrada (la clasificación no ha sido correcta), se decrementará el peso, alejándose ligeramente del vector de entrada (castigo).

Suponiendo que d es la clase a la que pertenece X_i, la regla de aprendizaje utilizada por el algoritmo LVQ es la siguiente:

$$W_c = \begin{cases} W_c + \alpha(t) \cdot (X_i - W_c) & \text{si } c = d \\ W_c - \alpha(t) \cdot (X_i - W_c) & \text{si } c \neq d \end{cases}$$

Sólo se ajustan los pesos de la unidad ganadora, permaneciendo sin modificación el resto de vectores prototipo. Si la clase ganadora c de la competición es correcta, es decir, coincide con la clase del vector de entrada d, los pesos de W_c se modifican acercándose al vector de entrada (se premia el acierto en la clasificación). Si por el contrario, la clasificación es incorrecta, se modifican alejándolo del vector de entrada (se castiga el error en la clasificación).

$\alpha(t)$ es el ratio de aprendizaje, siendo $0 < \alpha(t) < 1$, una función monótona decreciente del tiempo. La inicialización de $\alpha(t)$ se hace con un valor cercano a cero, y decrece linealmente de manera que al final del proceso de aprendizaje su valor es prácticamente nulo [11].

Características

Las características más relevantes de este método son la sencillez de las heurísticas empleadas y la rapidez de cálculo.

Aplicaciones

La red basada en el algoritmo de aprendizaje LVQ ha sido utilizada con éxito en muchas aplicaciones, fundamentalmente en análisis de imágenes, reconocimiento de patrones, problemas de clasificación, etc [11].

1.5. Algoritmo del vecino más cercano (Nearest Neighbour, NN) y variantes

El paradigma del vecino más cercano ha sido objeto de numerosos estudios en esta última década. Su criterio de aprendizaje se basa en que los miembros de una población suelen convivir (y al límite, coexistir) rodeados de individuos similares, con propiedades parecidas. Bajo esta hipótesis toda información descriptiva (adicional o desconocida) que quiera extraerse de un individuo puede observarse en otras instancias cercanas, en sus vecinos más cercanos. Consecuencia directa es la aplicación de este método como técnica de clasificación: del valor de la etiqueta de los miembros de una población se induce el valor desconocido de la etiqueta de un nuevo individuo.

Conocidos también como algoritmos de aprendizaje basados en instancias (IBL Algorithms o basados en ejemplos), se engloban dentro del Lazy-Learning al aplazar el proceso de inducción o generalización hasta que se completa la clasificación.

En el Lazy-Learning, rama del Aprendizaje Automático (IA), se implementa un algoritmo que se ejercita para terminar aprendiendo a realizar correctamente una determinada tarea, entrenándose previamente

con aquellas situaciones o casos posibles con los que puede encontrarse, de los cuales, selecciona los más representativos del concepto sobre el cual se quiere extraer conocimiento (Aprendizaje Supervisado).

Los elementos que forman el conjunto de entrada se denominan ejemplos o instancias y se componen de un vector y una o varias etiquetas. Cada dimensión del vector representa una característica o atributo del conjunto de entrada. El dominio de cada uno de los atributos puede ser discreto o continuo. La etiqueta se denomina clase y clasifica categóricamente a cada ejemplo del conjunto de entrada.

Basándose en *similitudes* entre los datos del conjunto de entrada o *entrenamiento* T se almacena únicamente un subconjunto *editado* S con lo que se reduce considerablemente el espacio de búsqueda y se acelera el proceso de aprendizaje. A partir de este nuevo subconjunto se generan hipótesis para describir, clasificar o predecir el comportamiento de nuevos ejemplos cuya clase se desconoce. Estos últimos conforman el *test* del sistema de aprendizaje.

A diferencia de otros métodos predictivos, el conjunto de entrenamiento no es procesado para crear el modelo, ya que éste es el modelo en sí mismo. Cuando un nuevo ejemplo se presenta, el algoritmo encuentra el subconjunto de ejemplos que son más similares a él y los utiliza para predecir su etiqueta [12].

Conviene aclarar que el paradigma k-NN es un tanto atípico si se compara con el resto de los paradigmas clasificatorios. Mientras que en el resto de los paradigmas la clasificación de un nuevo caso se lleva a cabo a partir de dos tareas, como son la *inducción* del modelo clasificatorio y la posterior *deducción* (o aplicación) sobre el nuevo caso, en el paradigma k-NN al no existir modelo explícito, las dos tareas anteriores se encuentran colapsadas en lo que se acostumbra a denominar *transinducción*.

En caso de que se produzca un *empate* entre dos o más clases, conviene tener una *regla heurística* para su ruptura. Ejemplos de reglas heurísticas para la ruptura de empates pueden ser: seleccionar la clase que contenta al vecino más próximo, seleccionar la clase con distancia media menor, etc [13].

Regla del vecino más próximo

Considerando un espacio de representación, el caso a ser clasificado tomará la clase que esté más cerca dentro de ese espacio.

Regla de los K vecinos más próximos (k-NN)

El nuevo caso a ser clasificado se ubicará en la clase con más votos en el contexto de los K vecinos más cerca del conjunto de entrenamiento.

Variantes del algoritmo k-NN

- *Regla k-NN con rechazo*. Se toma en cuenta un nivel fijado con anterioridad que sirve como referencia para que cuando una clase tiene un mayor número de votos que ese nivel, entonces la clase podrá ser asignada. Ese nivel puede tomar un valor entre $\frac{K}{M}$ y K, donde K es el número de vecinos más próximos y M es el total de clases.

  La idea subyacente al k-NN con rechazo es que para poder clasificar un caso se deben tener ciertas garantías. Pues puede ocurrir que un caso quede sin clasificar si no existen ciertas garantías de que la clase a asignar sea la correcta [13].

  Dos ejemplos utilizados para llevar a cabo clasificaciones con garantías son los siguientes:

  - El número de votos obtenidos por la clase deberá superar un *umbral prefijado*. Si se supone que trabajamos con $K = 10$, y $m = 2$, dicho umbral puede establecerse en 6.
  - Establecimiento de algún tipo de *mayoría absoluta* para la clase a asignar. Así si se supone que $K = 20$, $m = 4$, podemos convenir en que la asignación del nuevo caso a una clase sólo se llevará a cabo en el caso de que la diferencia entre las frecuencias mayor y segunda mayor supere a 3.

- *Regla k-NN por distancia media*. Un caso es clasificado en una clase si el valor de la distancia media es el menor con respecto al de las otras clases.

- *Regla k-NN por distancia mínima*. En el k-NN con distancia mínima se comienza seleccionando un caso por clase, normalmente el caso más cercano al baricentro de todos los elementos de dicha clase. En este paso se reduce la dimensión del fichero de casos a almacenar de N a m. A continuación se asigna el nuevo caso a la clase cuyo representante esté más cercano.

El procedimiento anterior puede verse como un 1-NN aplicado a un conjunto de m casos (uno por cada clase). El coste computacional de este procedimiento es inferior al k-NN genérico, si bien su efectividad está condicionada a la homogeneidad dentro de las clases (cuanto mayor sea dicha homogeneidad, se espera que el procedimiento sea más efectivo).

- *Regla k-NN con pesado de casos seleccionados*. La idea del k-NN con el pesado de los casos seleccionados es que los K casos seleccionados no se contabilicen de igual forma, sino que se tenga en cuenta la distancia de cada caso seleccionado al nuevo caso que se pretende seleccionar.

- *Regla k-NN con pesado de variables*. En todas las aproximaciones presentadas hasta el momento, la distancia entre el nuevo caso que se pretende clasificar, x, y cada uno de los casos $x_r, r = 1, \ldots, N$ ya clasificados pertenecientes al fichero de casos D, da el mismo peso a todas y cada una de las n variables, $X_1, \ldots, X_n$. Es decir, la distancia $d(x, x_r)$ entre x y x_r se calcula por ejemplo por medio de la distancia euclídea, de la siguiente manera:

$$d(x, x_r) = \sum_{j=1}^{n} (x_j, x_{rj})^2 \tag{1.2}$$

Esta manera de calcular la distancia, es decir, otorgando la misma importancia a todas las variables, puede resultar peligrosa para el paradigma k-NN en el caso de que algunas de las variables sean irrelevantes para la variable clase C. Este es el motivo por el que resulta interesante el utilizar distancias entre casos que ponderen cada variable de una manera adecuada. Es decir, la distancia entre x y x_r se calcularía:

$$d(x, x_r) = \sum_{j=1}^{n} w_j (x_j, x_{rj})^2 \tag{1.3}$$

con lo cual la variable X_j tiene asociado un peso w_j que habrá que determinar [13].

Características

La dimensión de los ejemplos afecta la velocidad de clasificación. Cada nuevo patrón hay que realizar un aprendizaje completo, y es muy costoso. Es sensible a la elección de k y a que la capacidad explicativa de los atributos sea distinta. Existe la posibilidad de ponderar las probabilidades de pertenencia a una

clase u otra en función de la distancia de cada vecino al patrón. El método es aplicable para salida continua (normalmente medias o medias ponderadas).

Aplicaciones

El poder del aprendizaje por vecindad ha sido demostrado en numerosos problemas reales, tales como la correcta pronunciación de palabras, diagnósticos para enfermos de tiroides, reconocimiento de voz, predicciones de negocio, simuladores de vuelo y pilotos automáticos, detectores de fraudes bancarios, expertos en juegos o la tan esperada secuenciación del ADN y ARN [12].

1.6. Support Vector Machines

Los SVM resultan ser sistemas de aprendizaje universal que, en su forma más simple, son capaces de encontrar una función lineal discriminante que separe el espacio de representación en regiones correspondientes a cada una de las clases consideradas. Como en todo aprendizaje supervisado, la entrada al sistema es un conjunto de ejemplos de entrenamiento. Si el conjunto de entrenamiento es linealmente separable, el SVM encuentra un hiperplano que maximiza la distancia euclídea a los ejemplos de entrenamiento más cercanos.

En términos geométricos, SVM puede ser visto como el intento de encontrar una superficie (σ_i) que separe a los ejemplos positivos de los negativos por el margen más amplio posible.

La búsqueda de σ_i que cumple que la distancia mínima entre él y un ejemplo de entrenamiento sea máxima, se realiza a través de todas las superficies $\sigma_1, \sigma_2, \ldots$ en el espacio $|A|$-dimensional que separan a los ejemplos positivos de los negativos en el conjunto de entrenamiento (conocidas como *superficies de decisión*).

Cabe resaltar que la mejor superficie de decisión es determinada únicamente por un conjunto pequeño de ejemplos de entrenamiento, llamados *vectores de soporte*.

En el caso de que el conjunto no pueda ser linealmente separable, se mide el error en el entrenamiento. Así, el cálculo de este hiperplano se reduce a un problema de optimización de errores en el que se aplican

restricciones. Las restricciones fuerzan a que todos los ejemplos de entrenamiento sean clasificados correctamente por encima de un error mínimo.

Una ventaja importante de SVM es que permite construir clasificadores no lineales, i.e., el algoritmo representa datos de entrenamiento no lineales en un espacio de alta dimensionalidad (llamado el espacio de características), y construye el hiperplano que tiene el margen máximo. Además, debido al uso de una función kernel para realizar el mapeo, es posible calcular el hiperplano sin representar explícitamente el espacio de características [4].

Características

Una característica fundamental de estos clasificadores es que son independientes de la dimensión del espacio de características. Así, la medida de la complejidad no depende del número de rasgos, sino de cómo puedan separarse los datos. Esto significa que siempre que los datos sean separables, podría generalizarse la presencia de muchos rasgos en una determinada categoría.

Aplicaciones

Las máquinas de vectores de soporte han mostrado conseguir buen desempeño de generalización sobre una amplia variedad de problemas de clasificación, destacando recientemente en problemas de clasificación de textos, donde se aprecia que SVM tiende a minimizar el error de generalización.

1.7. Evaluación de la clasificación

La efectividad es considerada usualmente el criterio de evaluación más importante gracias a su confiabilidad para comparar diferentes metodologías. A diferencia de la eficiencia, que se deja en segundo término debido a su dependencia de parámetros volátiles, e.g. diferentes plataformas software y/o hardware. La forma usual de medir la efectividad de un clasificador es por medio de su *exactitud* (α), i.e. el porcentaje de decisiones correctas. Un cálculo para la exactitud sobre la clase c_i puede darse en términos de su tabla de contingencia de la siguiente manera:

$$\alpha_i = \frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i} \tag{1.4}$$

Categoría c_i		Juicio del experto	
		SI	NO
Juicio del clasificador	SI	TP_i	FP_i
	NO	FN_i	TN_i

Cuadro 1.1: Contingencia para la clase c_i

donde FP_i (*falsos positivos*) es el número de instancias de prueba incorrectamente clasificados bajo c_i, TN_i (*verdaderos negativos*) es el número de instancias de prueba correctamente clasificados bajo $\bar{c}_i$, TP_i (*verdaderos positivos*) y FN_i (*falsos negativos*) son definidos consecuentemente (Ver Cuadro 1.1).

Durante la experimentación es usual dividir Ω en los tres conjuntos disjuntos Tr (el *conjunto de entrenamiento*), Va (el *conjunto de validación*) y Te (el *conjunto de prueba*). El conjunto de entrenamiento es el conjunto de instancias observadas cuando el aprendiz construye el clasificador. El conjunto de validación es el conjunto de instancias utilizadas para optimizar parámetros en los clasificadores, o para seleccionar uno en particular. Entonces, el conjunto de prueba es el conjunto de instancias sobre las cuales se evalúa finalmente la efectividad del clasificador. Sin embargo, la partición anterior no es adecuada cuando la cantidad de datos en Ω es limitada; en tal caso, el camino estándar para calcular la exactitud de una técnica de aprendizaje es usar una *validación cruzada con 10 pliegues* (*10FCV*, por sus siglas en inglés). Esta técnica consiste en dividir aleatoriamente Ω en diez partes, conservando en cada partición la proporción original de las clases, posteriormente cada parte es mantenida una vez y el esquema de aprendizaje entrena sobre las nueve partes restantes, entonces la exactitud es calculada sobre la parte conservada fuera del proceso de entrenamiento. Así, el proceso es ejecutado un total de diez veces sobre diferentes conjuntos de entrenamiento. Finalmente, los diez estimados de exactitud son promediados para producir una completa estimación de la misma [4].

1.8. Herramientas

Para el desarrollo del presente trabajo se utilizaron algunas herramientas para facilitar la implementación de un prototipo funcional, así como la evaluación de la efectividad del mismo a partir de los resultados arrojados en las pruebas realizadas.

Las ventajas principales de un IDE radica en la automatización de algunas tareas asociadas al desarrollo de aplicaciones y al poseer algunas funcionalidades que facilitan el desarrollo de software disminuyendo así el tiempo de construcción. La selección de un IDE para el desarrollo de aplicaciones en C++ en el sistema operativo GNU/Linux se hace hasta cierto punto difícil, ya que ninguno de los IDEs existentes completa código, lo cual es una de las características que más agiliza el desarrollo de software, al no tener que memorizar y teclear todas las funciones existentes en otras clases. Finalmente se encontró uno que si hace esta tarea y es el caso de Eclipse con un pluging llamado CDT, el cual tiene las siguientes características:

- Editor de C/C++ (Funcionalidades básicas, resaltamiento de sintaxi, completamiento de código, etc)
- Depurador C/C++ (Usando GDB)
- Lanzador C/C++ (Lanzadores y aplicaciones externas)
- Parser18
- Herramienta de búsqueda
- Proveedor de asistencia de contenido
- Generador de Makefile

Visual Paradigm es una herramienta que sirve para realizar modelado UML siguiendo el estándar UML 2.1. Esta herramienta tiene unas características graficas muy cómodas que facilitan la realización de los diagramas de modelado que sigue el estándar de UML que son:

- Diagramas de clase, Casos de Uso, Comunicación, Secuencia, Estado, Actividad, Componentes, etc.

Weka es una extensa colección de algoritmos de Máquinas de conocimiento útiles para ser aplicados sobre datos mediante los interfaces que ofrece, o para embeberlos dentro de cualquier aplicación. Además Weka contiene las herramientas necesarias para realizar transformaciones sobre los datos, tareas de clasificación, regresión, clustering, asociación y visualización. Está diseñado como una herramienta orientada a la extensibilidad por lo que añadir nuevas funcionalidades es una tarea sencilla.

Sin embargo, y pese a todas las cualidades que Weka posee, tiene una escasa documentación orientada al usuario, lo que la hace una herramienta difícil de comprender y manejar sin información adicional.

La licencia de Weka es GPL, lo que significa que este programa es de libre distribución y difusión. Además, ya que Weka está programado en Java, es independiente de la arquitectura, ya que funciona en cualquier plataforma sobre la que haya una máquina virtual Java disponible [14].

Es justo mencionar algunas herramientas auxiliares utilizadas para la confección del presente documento que, aunque no formaron parte del desarrollo del prototipo funcional, si contribuyeron a disminuir el tiempo y aumentar la calidad del trabajo; ellas son:

- LATEX: Es un sistema tipográfico para generar documentos científicos de alta calidad, lo cual permitió centrarse en el documento dejando a este la tarea de generar un documento que cumpliera con la calidad requerida para una tesis de grado.
- Kile: Es un IDE para LATEX el cual posibilitó un desarrollo fluido del documento.

1.9. Conclusiones

Luego de analizar los softwares de categorización de contenidos en los ámbitos internacional y nacional se evidencia la necesidad de diseñar e implementar uno propio, ya que los existentes, en el ámbito internacional, no cumplen con las características requeridas por FILPACON y en el ámbito nacional no se tienen referencias de la existencia de alguno. Mediante el estudio de las técnicas de IA, herramientas y tecnologías a utilizar, se han sentado las bases para el desarrollo de un prototipo funcional capaz de procesar de manera automática los documentos HTML a partir de las respuestas entregadas por los restantes módulos. A continuación se presentan algunos aspectos importantes definidos en el presente Capítulo.

Tecnologías

Se adoptaron las siguientes tecnologías para el desarrollo del trabajo.

- como lenguaje de programación: C++

- como bibliotecas: OpenCV y FANN
- como IDE de desarrollo: Eclipse con el pluging CDT
- como herramienta para la evaluación de la efectividad: Weka

Inteligencia Artificial

Se estudiaron los algoritmos que se utilizan para la categorización de documentos HTML seleccionando el siguiente, en el cual se basará nuestro trabajo.

- como algoritmos de entrenamiento y categorización: k-NN y MLP

Capítulo 2

Diseño del Módulo Decisor

2.1. Introducción

Para guiar el diseño del Módulo Decisor es necesario comprender su contexto e identificar las condiciones o capacidades que debe cumplir. En este capítulo se muestra su funcionamiento integrado a los restantes componentes de MOCIC, y se explica como se realiza el proceso de clasificación, con las propuestas de los algoritmos seleccionados y las pruebas realizadas al prototipo funcional.

2.2. Integración con MOCIC

Para que MOCIC clasifique correctamente precisa de mecanismos de decisión eficaces, que estarán incluidos en el Módulo Decisor, este será el encargado de categorizar los contenidos a partir de la respuesta de los módulos de clasificación y formará parte de la arquitectura de MOCIC, que se describe a continuación.

2.2.1. Arquitectura

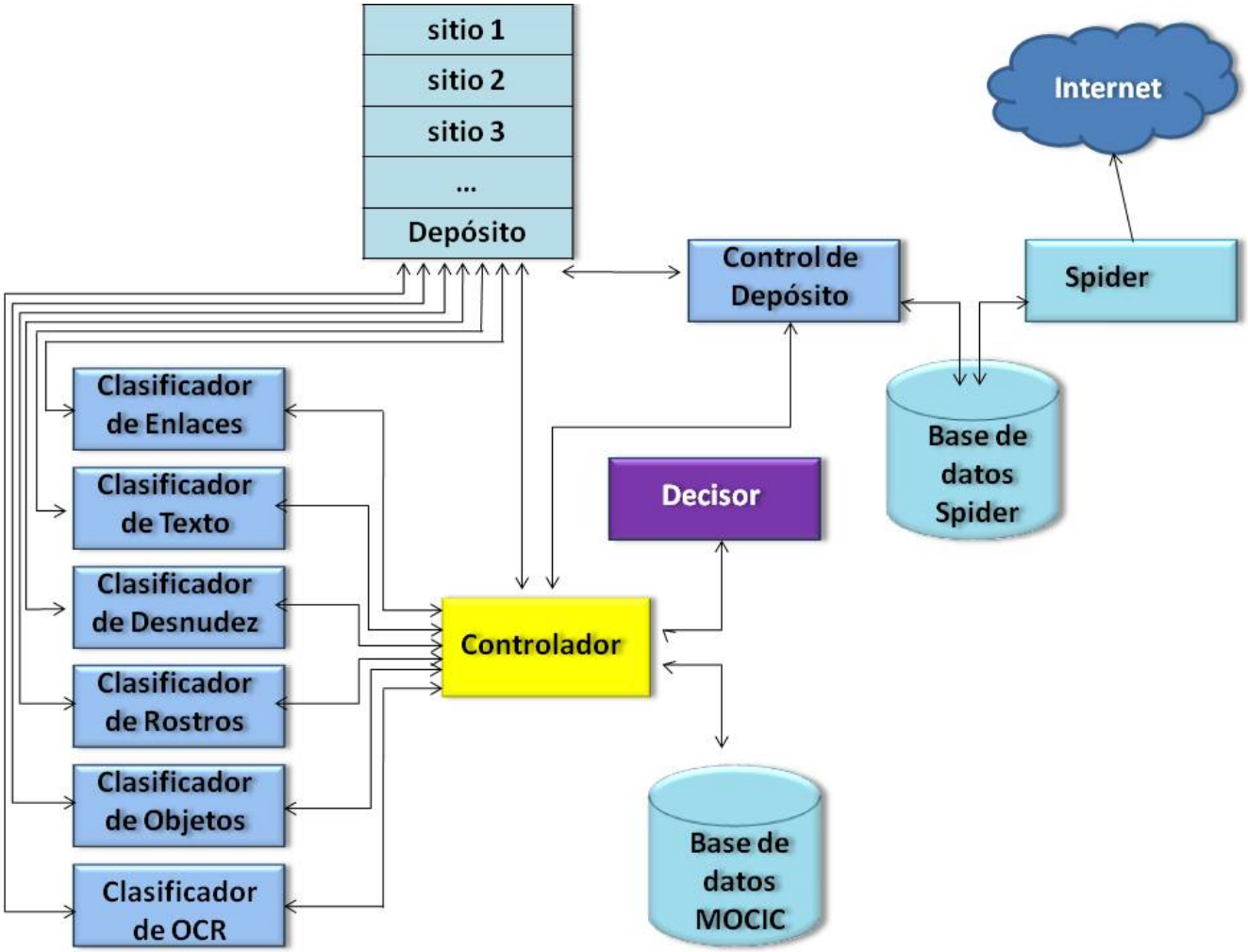

Figura 2.1: Arquitectura MOCIC

El Motor de Categorización Inteligente por Contenidos tiene como funcionalidad principal, recibir información digitalizada para su clasificación en categorías preestablecidas.

Para una mejor comprensión en cuanto a su funcionamiento y debido al gran impacto que en estos momentos tiene, se tomará como base en la explicación que el sistema clasificará la información contenida en la Web. Sin embargo, cabe destacar que la arquitectura fue pensada para que pueda ser utilizado en diferentes aplicaciones como: Filtrado de Contenidos, Organización de Documentos, Seguimiento de Noticias, Procesamiento Inteligente de Encuestas, Seguimiento de objetos en vídeos, entre otros.

Modos de Operación

El motor funcionará de dos formas: ON-LINE y OFF-LINE

ON-LINE: El sistema funcionará con los bloques imprescindibles y con los algoritmos más ligeros posibles. Este mecanismo evidentemente busca lograr una relación de compromiso entre eficacia y tiempo de retardo, tratando de que este último sea el mínimo posible. En caso de que alguno de los módulos emita una clasificación con una alta probabilidad de ser definitiva, con respecto a una determinada página Web, se enviarán señales de control por parte del Módulo Controlador, para que todos los módulos que se encuentren clasificando algún contenido referente a esa página sean reseteados, quedando listos para realizar la nueva clasificación.

Las URLs clasificadas en este modo serán marcadas para luego de forma OFF-LINE reclasificarlas.

OFF-LINE: El sistema funcionará con los bloques establecidos y con los algoritmos más eficaces posibles. También se busca una relación de compromiso entre eficacia y tiempo de retardo, pero para este caso se busca que el primero sea lo máximo posible.

A continuación se explicarán los diferentes módulos y bloques que conformarán el MOCIC.

Módulo-Clasificador de Texto

Este módulo tiene como función recibir a través de la consulta de un fichero de solicitudes, un mensaje proveniente del MOD-Controlador el cual indica la clasificación del texto de un documento HTML. Para clasificarlo accederá al directorio de localización del documento HTML, cargará el fichero .html correspondiente, le realizará un pre-procesamiento para identificar el idioma al que pertenece y seguidamente determinará las categorías de contenido previamente definidas (Ciencias, Computadoras, Deporte, Juegos, Pornografía, Violencia, Sustancias dañinas, Salud) asociadas al documento HTML, elaborando un mensaje que colocará en un fichero de respuestas. Este mensaje será posteriormente utilizado por el Módulo Decisor, para decidir a que categoría finalmente pertenece el documento HTML.

Módulo-Clasificador de Rostros

Este módulo tiene como función recibir a través de la consulta de un fichero de solicitudes, un mensaje proveniente del MOD-Controlador el cual indica la clasificación de las imágenes de un documento HTML. Para clasificarlo accederá al directorio de localización del documento HTML, cargará el directorio de sus imágenes y le realizará un procesamiento para determinar el número de rostros de personas encontrados en cada imagen.

Seguidamente se elabora un mensaje que se colocará en un fichero de respuestas. Este mensaje será posteriormente utilizado por el Módulo Decisor, para decidir a que categoría finalmente pertenece el documento HTML.

Módulo-Clasificador de Desnudez

Este módulo tiene como función recibir a través de la consulta de un fichero de solicitudes, un mensaje proveniente del MOD-Controlador el cual indica la clasificación de las imágenes de un documento HTML. Para clasificarlo accederá al directorio de localización del documento HTML, cargará el directorio de sus imágenes y le realizará un procesamiento para determinar por cada imagen la existencia o no de desnudez (presencia de piel humana). Seguidamente se elabora un mensaje que se colocará en un fichero de respuestas. Este mensaje será posteriormente utilizado por el Módulo Decisor, para decidir a que categoría finalmente pertenece el documento HTML.

Módulo-Clasificador de Objetos

Este módulo tiene como función recibir a través de la consulta de un fichero de solicitudes, un mensaje proveniente del MOD-Controlador el cual indica la clasificación de las imágenes de un documento HTML. Para clasificarlo accederá al directorio de localización del documento HTML, cargará el directorio de sus imágenes y le realizará un procesamiento para determinar por cada imagen el número de símbolos encontrados de cada categoría (Ciencias, Computadoras, Deporte, Juegos, Pornografía, Violencia, Sustancias dañinas, Salud). Seguidamente se elabora un mensaje que se colocará en un fichero de respuestas. Este mensaje será posteriormente utilizado por el Módulo Decisor, para decidir a que categoría finalmente pertenece el documento HTML.

Módulo-Reconocimiento Óptico de Caracteres

Este módulo tiene como función recibir a través de la consulta de un fichero de solicitudes, un mensaje proveniente del MOD-Controlador el cual indica la clasificación de las imágenes de un documento HTML. Para clasificarlo accederá al directorio de localización del documento HTML, cargará el directorio de sus imágenes y le realizará un procesamiento para extraer por cada imagen todo el texto asociado y elaborar un único fichero de texto a partir del cual, y utilizando los mismos algoritmos del clasificador de textos determinará las categorías de contenido (Ciencias, Computadoras, Deporte, Juegos, Pornografía, Violencia, Sustancias dañinas, Salud) asociadas al documento HTML, elaborando un mensaje que colocará en un fichero de respuestas. Este mensaje será posteriormente utilizado por el Módulo Decisor, para decidir a que categoría finalmente pertenece el documento HTML.

Módulo-Clasificador de Enlaces

Este módulo tiene como función recibir a través de la consulta de un fichero de solicitudes, un mensaje proveniente del MOD-Controlador el cual indica la clasificación de los enlaces relacionados a un documento HTML. Para dicha clasificación accederá al directorio de localización del documento HTML, cargará el fichero donde están depositados todos los enlaces correspondientes, le realizará un procesamiento para determinará las categorías de contenido presentes (Ciencias, Computadoras, Deporte, Juegos, Pornografía, Violencia, Sustancias dañinas, Salud) asociadas al documento HTML, elaborando un mensaje que colocará en un fichero de respuestas. Este mensaje será posteriormente utilizado por el Módulo Decisor, para decidir a que categoría finalmente pertenece el documento HTML.

Módulo-Decisor

Este módulo tiene como función recibir por parte del MOD-Controlador, toda la información proveniente de los módulos clasificadores y devolverle la categoría más probable a la que pertenece el documento HTML, en caso que este se haya podido clasificar y además si se categorizó faltándole elementos o si no se pudo categorizar.

Módulo-Almacenamiento (Base de Datos MOCIC)

Este módulo tiene como función recibir a través de la consulta de un fichero de solicitudes, un mensaje proveniente del MOD-Controlador, el cual indica el almacenamiento en la Base de Datos MOCIC de la

clasificación final de un documento HTML (generada por el MOD-Decisor) y asociada a una o varias categorías (Ciencias, Computadoras, Deporte, Juegos, Pornografía, Violencia, Sustancias dañinas, Salud). Almacenará además para el documento HTML todos aquellos datos necesarios para una verificación o seguimiento de la clasificación realizada. Finalmente elabora un mensaje que colocará en un fichero de respuestas. Este mensaje será posteriormente utilizado por el MOD-Controlador, para renombrar a Clasificado o Pendiente el directorio del documento HTML en el depósito.

Depósito

Bloque que identifica al lugar donde es almacenado el contenido que va a ser clasificado. En este caso, no es más que un directorio con una determinada estructura, que funcionará como una cola del tipo First In First Out (FIFO), en relación al mecanismo de prioridad del análisis. Contendrá por cada URL un directorio y este a su vez incluirá el fichero .html, el directorio de las imágenes y un fichero con los enlaces asociados, indicando en la primera línea del fichero, el nombre de la propia URL.

Suministrador de depósito

Se encargará de consultar de manera recurrente la base de datos del Spider para el llenado del depósito de URLs cumpliendo con la estructura de directorios y ficheros anteriormente descritos.

Spider

Este bloque no forma parte del MOCIC, pero sí interactúa con él. Es el encargado, para el caso del análisis de contenido Web, de recuperar información y reestructurarla en una base de datos propia, utilizada para añadir las URLs al Depósito para su posterior análisis.

Módulo-Controlador

Controla y sincroniza todo el funcionamiento del motor. Posee un fichero de configuración central en el cual se puede:

1. Especificar la forma de trabajo (OFF-LINE/ON-LINE).
2. Activar o desactivar Módulos.

3. Configuración general de los módulos (forma de comunicación con los restantes módulos, localización de ficheros de configuración específico, localización de directorio DEPÓSITO)

4. Posee una interfaz Web para la configuración de los módulos activos y para el monitoreo del funcionamiento de los mismos.

2.2.2. Modos de Respuesta

La respuesta enviada por cada uno de los módulos clasificadores al MOD-Controlador tienen la estructura siguiente:

Clasificador de Texto

-R id xx xx xx xx xx xx xx xx

Id son 2 bits que corresponden al idioma al que pertenece el documento HTML.

Cada xx equivale a una de las categorías definidas de manera ordenada. Estas categorías se muestran en el Anexo A

- Cuando xx = 00 significa que el texto NO pertenece a la categoría.
- Cuando xx = 11 significa que el texto SI pertenece a la categoría.
- Cuando xx = 99 no se pudo clasificar este texto.

Clasificador de Rostros

-R xx xx xx xx xx xx xx xx ... xx

Las imágenes estarán ordenadas por nombre y cada xx corresponde a una imagen y el valor de xx representa la cantidad de rostros encontrados en la imagen.

Valores válidos para xx: [00 – 90]

- Cuando xx = 99 no se pudo clasificar esta imagen.

Clasificador de Desnudez

-R xx xx xx xx xx xx xx xx ... xx

Las imágenes estarán ordenadas por nombre y cada xx corresponde a una imagen.

- Cuando xx = 00 NO se detectó desnudez en esta imagen.
- Cuando xx = 11 SI se detectó desnudez en esta imagen.
- Cuando xx = 99 no se pudo clasificar esta imagen.

Clasificador de Objetos

-R A B xx xx xx xx xx xx xx xx ... xx

Las imágenes estarán ordenadas por nombre y cada xx corresponde a una imagen.

Se conforma una matriz de A filas y B columnas, que se debe linealizar. El valor de A representa la cantidad de imágenes y B el número de categorías analizadas (están en orden según las categorías de MOCIC). El valor de xx representa la cantidad de símbolos encontrados en la imagen para una categoría determinada.

Valores válidos para xx: [00 – 90]

- Cuando xx = 99 no se pudo clasificar esta imagen, esto implica que debe aparecer xx = 99 B veces consecutivas.

Aunque los módulos Clasificador de Enlaces y Reconocimiento Óptico de Caracteres figuran en la arquitectura de MOCIC estos aún no están implementados y por tanto no estarán comprendidos en la investigación.

2.2.3. Conformación del vector de características

- Del vector que devuelve el *Clasificador de Textos* se identifica a que categoría pertenece este texto observando el bit del vector que se activa (11) en ese momento Este vector estará compuesto por 8 bits correspondiente a cada una de las categorías predefinidas y para su mejor manipulación se expresará en decimal, tomando valores de 1 a 256. En caso que el documento no se haya podido clasificar se tomará con valor 0.
- Del vector que devuelve el *Clasificador de Desnudez* se obtiene la cantidad total de imágenes, determinando el porciento de imágenes donde se detectó desnudez a partir de la fórmula siguiente:

$$X = \frac{100 \cdot N}{(Total\_img - Total\_img\_no\_clasif)}$$

Donde:

- **X**: % de imágenes donde se detectó desnudez.
- **N**: número de imágenes con desnudez que hay en el vector.
- **Total_img**: total de imágenes que contiene el vector.
- **Total_img_no_clasif**: total de imágenes que contiene el vector y que no se pudieron clasificar.

- El vector que devuelve el *Clasificador de Rostros* aporta la cantidad total de rostros encontrados en una imagen, determinando el promedio de rostros por imagen el documento HTML a partir de la fórmula siguiente:

$$X = \frac{\text{Total\_rostros}}{\text{Total\_imag} - \text{Total\_imag\_no\_clasif}}$$

Restándole al total de imágenes que componen el vector las que el *Clasificador de Rostros* no pudo identificar.

Donde:

$$\text{Total\_rostros} = (RXX_1 + RXX_2 + RXX_3 + \ldots + RXX_n)$$

$$\text{Total\_imag} = (XX_1 + XX_2 + XX_3 + \ldots + XX_n)$$

- **Total_rostros**: cantidad total de rostros identificados en cada una de las imágenes del documento HTML.
- **Total_imag**: cantidad total de imágenes que conforman el vector.
- **Total_img_no_clasif**: cantidad total de imágenes que contiene el vector pero que no se pudo identificar si había rostros en ellas.

- Del vector que devuelve el *Clasificador de Objetos* se obtiene que porciento representa la cantidad de objetos que le corresponde a cada una de las categorías existentes en el vector, del total de objetos identificados en el documento HTML, sumándose la cantidad de objetos que existen por categoría y la cantidad total de objetos identificados en el documento HTML, aplicándose el criterio de que la categoría que posea el mayor por ciento es la predominante, aunque es muy poco probable encontrar más de una categoría en un documento HTML se tomarán, si existen, las dos con por cientos mayores. En este análisis no se contemplan las imágenes que no pudieron ser clasificadas.

$$X = \frac{100 \cdot N}{Total\_obj}$$

Donde:

- **X**: % representa la cantidad de objetos de un mismo tipo identificados dentro de una categoría, respecto al total de objetos identificados en el documento HTML.
- **N**: sumatoria de la cantidad de objetos identificados por cada categoría
- **Total_obj**: cantidad total de objetos identificados en las imágenes que contiene el documento HTML.

Finalmente el vector de características queda conformado de la siguiente manera:

[texto, desnudez, rostro, objeto1, objeto2]

2.2.4. Normalización de los datos

Una vez seleccionados los datos significativos, estos deben ser normalizados antes de ingresar en la red neuronal. Para normalizar el vector simplemente se dividirá cada componente entre el valor máximo que este puede tomar, llevando los valores a un rango entre 0 y 1, manteniendo la posición relativa de cada uno de los valores del vector, de esta manera cada campo obtiene un valor entre 0 y 1, siendo homogéneo y correcto para su ingreso en la red neuronal. A continuación se describe como se realizó el proceso de normalización en el vector de características:

El vector que se obtiene tiene la forma V_i con $i \in R[1, n]$, donde n es la cantidad de campos que posee el vector.

$$V_i =< V_1, V_2, V_3, \ldots, V_n > \tag{2.1}$$

El vector normalizado quedará de la siguiente manera:

$$V_{ni} = \frac{V_i}{X_i} \tag{2.2}$$

donde X_i es el valor máximo que puede obtener cada campo.

$$V_{ni} =< V_{n1}, V_{n2}, V_{n3}, \ldots, V_{nn} > \tag{2.3}$$

1er campo

$$V_{n1} = \frac{V_1}{X_1} \tag{2.4}$$

V_1 representa la categoría a la que pertenece el texto.

X_1 es el valor máximo que puede tomar este campo, debido a que el vector que devueve el clasificador de texto tiene 8 bits correspondiente a cada una de las categorías predefinidas para expresarlo en decimal realizamos la siguiente operación:

$$X_1 = 2^n \tag{2.5}$$

donde n es el número de categorías, en este caso 8, por tanto, el valor máximo que puede tomar X_1 es 256.

2do campo

$$V_{n2} = \frac{V_2}{X_2} \tag{2.6}$$

V_2 representa el por ciento de imágenes donde se detectó desnudez .

X_2 es el valor máximo que puede tomar este campo y como el vector que devueve el clasificador de desnudez está representado en por ciento se dividirá por 100.

3er campo

$$V_{n3} = \frac{V_3}{X_3} \tag{2.7}$$

V_3 representa el promedio de rostros por imagen que contiene el documento HTML.

X_3 equivale a la multiplicación del total de imágenes encontradas en el documento HTML por 90, que es la cantidad máxima de rostros que se reconocerán por cada imagen.

4to campo

$$V_{n4} = \frac{V_4}{X_4} \tag{2.8}$$

V_4 representa el por ciento de objetos perteneciente a la categoría predominante en el documento HTML.

X_4 es el valor máximo que puede tomar este campo y como el vector que devueve el clasificador de objetos está representado en por ciento se dividirá por 100.

5to campo

$$V_{n5} = \frac{V_5}{X_5} \tag{2.9}$$

V_5 representa el por ciento de objetos perteneciente a la segunda categoría más representativa en el documento HTML.

X_5 es el valor máximo que puede tomar este campo y como el vector que devueve el clasificador de objetos está representado en por ciento se dividirá por 100.

2.2.5. Propuesta de los algoritmos de clasificación

Como se estudió en el Capítulo 1 existen varios algoritmos de clasificación automática, como el MLP, Modelo de Kohonen, SOM, LVQ, k-NN y SVM; que por la efectividad que los caracteriza en la solución de problemas de clasificación son muy utilizados.

En este trabajo de diploma se proponen dos algoritmos para la clasificación de documentos HTML, con el fin de que MOCIC sea lo más flexible y adaptable posible y que el usuario pueda seleccionar con cual desea categorizar; para esto debe tener en cuenta el modo en que se esté trabajando (On-Line/Off-Line).

De los algoritmos estudiados se seleccionaron el MLP y el k-NN. Para esto se tuvo en cuenta la efectividad y rapidez de los algoritmos, por ser los factores más importantes para MOCIC, estos se enmarcan dentro del aprendizaje supervisado ya que se tiene un previo conocimiento de las categorías, como muestra el Anexo A. Cuando este trabaje de manera On-Line se precisa que lo haga de forma rápida, aunque pierda en efectividad, los documentos que sean clasificados de modo On-Line serán almacenados y reclasificados posteriormente de la misma manera que se hace con los documentos que se clasificarán de modo Off-Line, donde la rapidez no es tan importante como la efectividad.

- k-NN: Tiene un proceso de aprendizaje rápido, ya que guarda en un fichero de entrenamiento la clase que se quiere categorizar, esto es una característica muy importante, que permite, en caso de necesitar cambiar la estructura del vector (adicionar o eliminar posiciones), realizar un nuevo entrenamiento en un periodo breve.
- MLP: Aunque su entrenamiento no sea tan rápido como en otros algoritmos, como toda red neuronal es muy potente y tiene gran efectividad en los resultados que se obtienen siempre y cuando sea entrenada correctamente, esto está directamente relacionado con la arquitectura, ya que no existe un patrón predefinido para su diseño.

2.2.6. Entrenamiento y clasificación de los algoritmos

k *Nearest Neighbour*

Para aplicar el algoritmo k-NN primeramente se definen las clases, cada una correspondiente a las categorías establecidas por MOCIC.

Los ejemplos de entrenamiento son vectores en un espacio característico multidimensional descritos en términos de p atributos, considerando q clases para la clasificación.

La fase de entrenamiento del algoritmo consiste en almacenar los vectores característicos y las etiquetas de las clases de los ejemplos de entrenamiento en un fichero.

En la fase de clasificación, la evaluación del ejemplo (del que no se conoce su clase) es representada por un vector en el espacio característico.

Se calcula la distancia entre los vectores almacenados y el nuevo vector, y se seleccionan los k ejemplos más cercanos. El nuevo ejemplo es clasificado con la clase que más se repite en los vectores seleccionados. Generalmente se usa la distancia euclideana.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p}(x_{ir} - x_{jr})^2} \tag{2.10}$$

Se define un umbral para la distancia entre el vector característico y las clases, si la distancia euclideana calculada sobrepasa este umbral el documento HTML no se encuentra dentro de las categorías predefinidas y deberá ser reclasificado.

Multilayer Perceptron

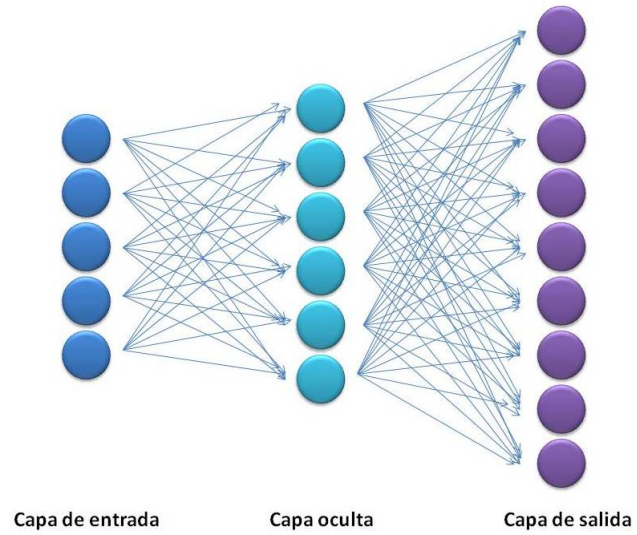

Figura 2.2: Modelo del MLP

Para determinar la arquitectura de la red primeramente se determinaron el conjunto de entrada y el de salida.

- En la capa de entrada (CE) se fijaron 5 neuronas, correspondiente a cada uno de los valores del vector característico.
- Aunque las redes MLP admiten tanto en la capa de entrada como en la de salida valores racionales, tienen un mejor funcionamiento cuando estos son binarios. Teniendo en cuenta que los valores de

la capa de entrada son racionales se debe garantizar una capa de salida (CS) con valores binarios. Para esto se definieron 9 neuronas en la capa de salida, la primera de ellas corresponde a la clasificación de la “No Categoría”, que es asignada cuando el documento HTML no pertenece a ninguna de las categorías definidas, de obtenerse esta salida el documento debe ser reclasificado. Las restantes neuronas representan las categorías definidas por MOCIC (Anexo A), de esta forma se activará la neurona correspondiente a la categoría a la que pertenece el documento HTML. Este diseño de la capa de salida permite agregar más categorías (neuronas), sin afectar la ubicación de la “No Categoría”.

Para constituir la capa oculta existen criterios propuestos por diferentes autores, fundamentados en los resultados obtenidos.

- La primera propuesta es determinar la capa oculta al aplicar la fórmula

$$CO = \frac{CE}{2} + \frac{CS}{2} \tag{2.11}$$

- Y la segunda plantea que la capa oculta debe tener más neuronas que la mayor de la capa de entrada o salida.

A partir de esto la capa oculta va a variar su arquitectura hasta obtener resultados favorables.

En este caso la capa oculta inicialmente solo poseerá 6 neuronas (como se muestra en la figura 2.2), tomando como referencia el primer criterio.

2.2.7. Implementación de un prototipo funcional

La construcción de un Módulo Decisor fundamentado en aprendizaje se basa en diversos elementos tomados de los campos de la Recuperación de Información y del Aprendizaje Automático. Esencialmente, el proceso consiste en la representación de un conjunto de documentos (en nuestro caso, páginas Web – documentos en formato HTML) manualmente clasificados (llamado colección de entrenamiento) como vectores de pesos de términos, a los que se aplica un algoritmo de aprendizaje que construye un modelo de clasificación o clasificador. Los documentos a clasificar se representan de igual forma, de modo que el clasificador es capaz de asignar una categoría (Ciencias, Computadoras, Deportes, Sustancias Dañinas,

Juegos, Pornografía, Salud, Violencia) a los mismos. Este proceso será explicado mediante la siguiente figura:

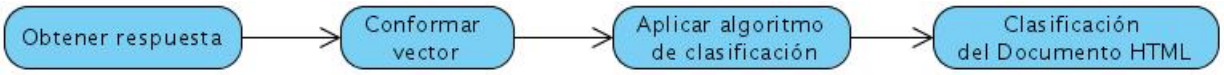

Figura 2.3: Proceso de clasificación

Primeramente el Decisor obtiene las respuestas devueltas por los módulos clasificadores a través del Controlador y se conforma el vector a partir de las características extraídas de las mismas. A este vector se le aplica un algoritmo de categorización, y se obtiene como resultado la clasificación final del documento HTML, que es devuelta al Controlador.

Este proceso de categorización lo realiza el Decisor a través de dos mecanismos de decisión: ON-LINE y OFF-LINE, que son imprescindibles para que MOCIC pueda clasificar de manera automática e "inteligente".

El Módulo Decisor para comenzar a categorizar se encuentra en espera de las peticiones realizadas por el Módulo Controlador, estos mensajes llegan con la siguiente estructura:

<Sello de tiempo> <Nombre Filtro> <Mensaje Parametrizado> <C.Fin>

Donde

Sello de tiempo: Indica el día y la hora que fue generada la respuesta o solicitud (AA/MM/DDhh:mm:ss).

Nombre Filtro: Indica el nombre del módulo que envió la respuesta.

- CTEX: Clasificador de Texto.
- CENL: Clasificador de Enlaces.
- DROS: Detector de Rostros.
- DPDE: Detector de Personas Desnudas.
- ROBJ: Reconocimiento de Objetos.
- ROCA: Reconocimiento Óptico de Caracteres.

- IPBD : Intérprete de Base de Datos.
- CDEP :Control depósito.
- DECI : Decisor.

Parámetros

- -i (Indica localización del directorio de entrada). El Módulo Controlador especifica la localización de la URL a procesar, esta localización se refiere al servidor de almacenamiento (depósito).
- -p (Indica el ID del proceso). Es un número único generado en la primera creación del proceso.
- -u (Indica la URL). Se especifica la dirección URL.
- -c (Indica mensaje de control). Es aplicable a todos los Filtros + Decisor. Se utilizarán los mensajes:
  - ABORT: Abortar solo el proceso referenciado por su ID. Se emite cuando: TTL (proceso) expira o cuando el decisor da la respuesta del proceso.
  - RESET: Reiniciar el filtro completo. Esto ocurrirá cuando el número de ABORT exceda un valor predeterminado.
- -R (Indica la respuesta del filtro).

Del mensaje recibido el Decisor solo necesita el nombre del módulo que lo envió, la dirección de la URL, la respuesta del módulo y el sello de tiempo. Estos datos, excepto el último, son almacenados en una matriz de A filas y B columnas con la siguiente estructura:

	Texto	**Desnudez**	**Rostro**	**Objeto**	**ID_Proceso**
URL_1	R	R			0000000001
URL_2			R	R	0000000002
URL_3	R	R	R	R	0000000005
...	R				
...		R			
URL_N					

Cuadro 2.1: Matriz

Cada columna corresponde a los módulos de filtrado que se hayan configurado para funcionar en ese momento, por el momento solo se contemplan los módulos de filtrado de Texto, Desnudez, Rostros y Objetos. Las filas corresponden a las URLs que van a ser clasificadas. En los campos de la matriz se sitúan los modos de respuesta de los módulos a medida que van llegando.

Los mensajes están situados en un fichero y cuando son leídos por el Decisor y ubicados en la matriz son eliminados; así va recopilando cada URL con las respuestas de los diferentes módulos de filtrado que van arribando en un tiempo progresivo.

Con el sello de tiempo se crea un vector que almacena este valor en correspondencia con la ubicación de cada una de las URLs de la matriz, pero la fecha y hora que se guarda es con la que se insertó la URL en la matriz a partir de la primera respuesta recibida de los módulos de filtrado, este valor no se modifica con la entrada de las respuestas restantes. Esto se hace con el fin de conocer que tiempo lleva la URL en procesamiento para clasificarse, y determinar si excede los tiempos establecidos para cada modo.

ON-LINE

En modo ON-LINE es muy importante el factor rapidez como ya se ha mencionado en la sección 2.2.4, es por esto que el Módulo Decisor puede concederle una categoría a un documento HTML sin tener todas las respuestas de los módulos de filtrado pertenecientes a una URL específica.

Esto depende de los módulos de filtrado que falten por enviar su respuesta, pues todos no tienen la misma importancia, en este caso se le concede mayor valor al Módulo Texto y al Módulo Desnudez. Si la respuesta de alguno de estos módulos no llega en un tiempo prudencial[1] se le envía una petición al Módulo Controlador para que reclasifique el documento HTML. Sin estas respuestas no se puede clasificar ningún documento HTML.

[1]Este tiempo varía en dependencia del mecanismo que se esté utilizando, en el modo OFF-LINE el tiempo de espera es mayor que cuando se encuentra ON-LINE.

Por el contrario, si no se tienen las respuestas de los restantes módulos se toma un valor aleatorio de la colección de entrenamiento para completar el vector y poder clasificar el documento HTML, esta se almacena en la Base de Datos MOCIC para una posterior reclasificación.

OFF-LINE

En modo OFF-LINE lo más importante es la efectividad, por eso se esperará por todas las respuestas de los módulos de filtrado durante un tiempo prudencial. Si en este tiempo no llegan todas las respuestas se le envía una petición al Módulo Controlador para que vuelva a reclasificar el documento HTML.

Si solo se tienen las respuestas del Módulo Texto y Módulo Desnudez se procederá de la misma manera que si estuviera ON-LINE, pero si las respuestas obtenidas son de los restantes módulos se le envía una petición al Módulo Controlador para que vuelva a reclasificar el documento HTML.

El Decisor enviará una secuencia binaria de 4 + B elementos. Los dos primeros corresponden al idioma al que pertenece el documento HTML. El tercer bit indica si el documento se categorizó con todos los elementos o no. El bit 4 pertenece a la No Categoría y los últimos 8 bits a las categorías MOCIC, ordenadas de izquierda a derecha, cuando un bit está en 1 significa que el documento HTML pertenece esa categoría, cuando está en 0, significa que no pertenece.

- -R id 0 0 x x x x x x x x: cuando los bits 3 y 4 están en 0 significa que la URL no está pendiente a clasificación porque se categorizó con todas las respuestas de los módulos de filtrado y no pertenece a la No Categoría.
- -R id 0 1 x x x x x x x x: cuando el bit 3 está en 0 significa que la URL no está pendiente a clasificación porque se categorizó con todas las respuestas de los módulos de filtrado y pertenece a la No Categoría como indica el bit 4 (activado en 1), en este caso los restantes bits estarán en 0.
- -R id 1 0 x x x x x x x x: cuando el bit 3 esta en 1 significa que la URL está pendiente a clasificación, en este caso porque no se categorizó con todas las respuestas de los módulos de filtrado y no pertenece a la No Categoría como indica el bit 4.
- -R id 1 1 x x x x x x x x: los bits 3 y 4 están activados en 1 cuando la URL está pendiente a clasificación porque no se categorizó con todas las respuestas de los módulos de filtrado y pertenece a

la No Categoría o porque se verificaron los tiempos de cada URL y alguno excedió el establecido para ser clasificada.

Esta respuesta forma parte del mensaje que enviará el Decisor al categorizar un documento HTML, el mensaje tendrá la siguiente estructura:

<Sello de tiempo> <DECI> <Mensaje Parametrizado> <C.Fin>

Modelo de dominio

El Modelo de Dominio (o Modelo Conceptual) es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real y no de los componentes de software [15].

Identificar muchos objetos o conceptos forma parte de una investigación del problema. El lenguaje UML contiene la notación en diagramas de estructura estática que explican gráficamente los Modelos de Dominio. El paso esencial de un análisis orientado a objetos es descomponer el problema en conceptos individuales; ya que es una representación de conceptos en un dominio del problema se ilustra con un grupo de diagramas de estructura estática donde no se define ninguna operación. Puede mostrarnos:

- conceptos
- asociaciones de conceptos
- atributos de conceptos

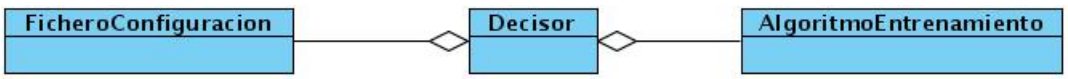

Figura 2.4: Modelo de Dominio

Teniendo en cuenta el modelo de dominio surge el siguiente diagrama de clases.

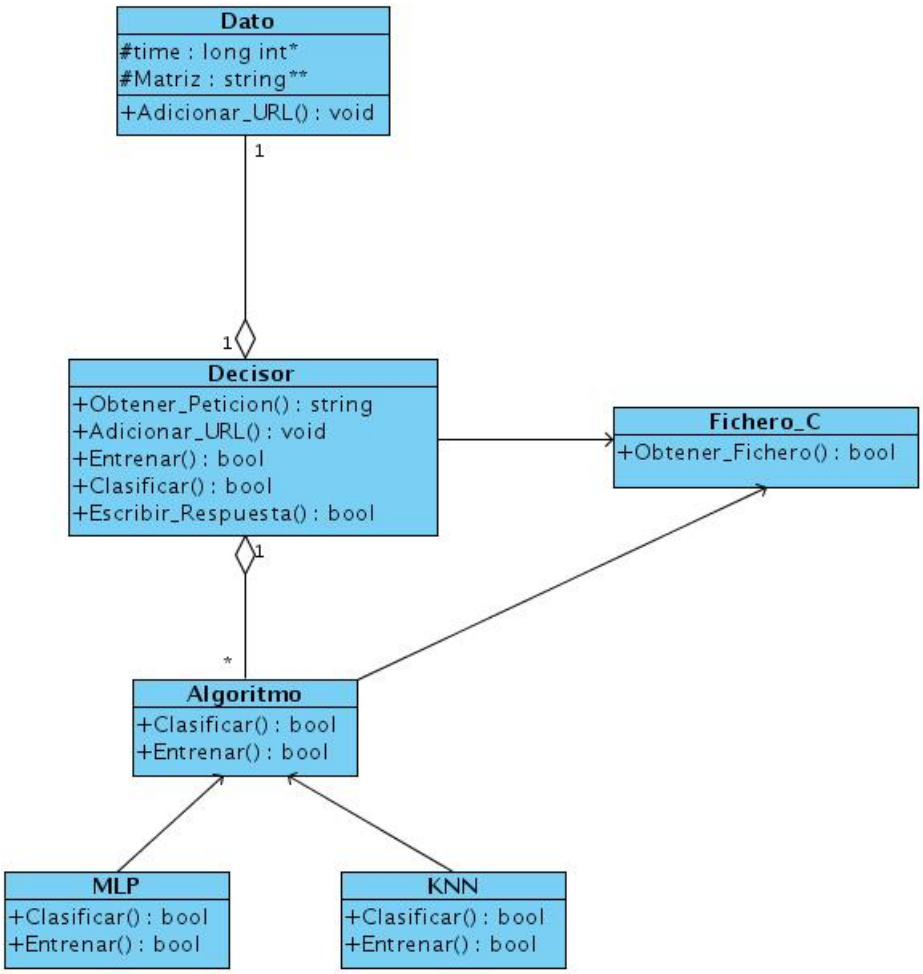

Figura 2.5: Diagrama de clases

El diagrama de clases muestra los principales atributos y métodos definidos para la implementación del prototipo funcional (Decisor V0.1), esto solo es una versión inicial y a medida que se vaya mejorando el producto e incorporándose funcionalidades se actualizará el diagrama. A continuación se hace una explicación detallada de sus funcionalidades.

Decisor

- **Obtener_Petición**: obtiene a través del fichero de configuración la ruta al fichero de peticiones que puede o no contener peticiones. Si está vacío, se esperan 5 segundos para verificar si contiene alguna petición, sino extrae la primera, y la descompone en los parámetros que conforman el mensaje. Luego evoca al método **Adicionar_URL** y le pasa los parámetros extraídos anteriormente,

exceptuando el sello de tiempo.

- **Adicionar_URL**: primeramente verifica si la URL contenida en el mensaje se encuentra en la matriz, de encontrarse se inserta en la columna correspondiente la respuesta asociada al módulo que la envió, sino adiciona una nueva fila en la matriz con el nombre de la URL, la respuesta asociada al módulo que la envió y el ID del proceso, luego almacena en el vector time (timestamp) el tiempo en que se adicionó la URL.

  - **Entrenar**: obtiene a través del fichero de configuración la ruta al fichero de entrenamiento y crea con sus vectores una matriz con una estructura de 6 columnas y N filas como se muestra a continuación:

Texto	Desnudez	Rostro	Objeto1	Objeto2	Categoría
R	R	R	R	R	N
R			R	R	I
R	R	R	R	R	A
R	R	R	R	R	I
R	R	R	R	R	I
R	R	R	R	R	A

Cuadro 2.2: Matriz de entrenamiento

Donde cada columna corresponde a los valores del vector característico y a la categoría del vector y N es la cantidad de vectores que contiene el fichero. Para la colección de entrenamiento no se establece un número específico de vectores pero mientras más posea, más fiable será el entrenamiento. Luego le pasa por parámetro la matriz al método **Entrenar** de la clase **Algoritmo**.

  - **Clasificar**: Recorre la matriz y determina si alguna de las URLs almacenadas se puede categorizar, de ser así crea el vector característico con la respuesta de los módulos. Luego le pasa el vector como parámetro al método **Clasificar** de la clase **Algoritmo**. Conforma el mensaje

de respuesta con la categoría asignada e invoca el método **Escribir_Respuesta** pasándole este como parámetro. Se crea el mensaje de respuesta y finalmente la URL clasificada es eliminada de la matriz.

- **Escribir_Respuesta**: A través del fichero de configuración obtiene la ruta del fichero de respuesta donde será almacenado el mensaje de respuesta creado por el método **Clasificar** de la clase **Decisor**.

Fichero_Config

- **Obtener_fichero**: es un método estático que devuelve una instancia de la clase **Fichero**. Para esta función se utiliza el patrón Singleton. Este permite crear una única instancia de un objeto para toda la aplicación. Este patrón es implementado en la clase **Fichero**, permitiendo que el fichero sea el mismo en todo el software.

Algoritmo

Esta es una clase abstracta y por tanto sus métodos serán redefinidos en las hijas.

- **Clasificar**: Lee del fichero de configuración que algoritmo se esta utilizando en ese momento y evoca al método **Clasificar** de dicha clase.
- **Entrenar**: Lee del fichero de configuración que algoritmo se esta utilizando en ese momento y evoca al método **Entrenar** de dicha clase.

k-NN

- Clasificar: es utilizado para realizar la clasificación con este algoritmo.
- Entrenar: es utilizado para realizar el entrenamiento a este algoritmo.

MLP

- Clasificar: es utilizado para realizar la clasificación con este algoritmo.
- Entrenar: es utilizado para realizar el entrenamiento a este algoritmo.

Dato

- **Adicionar_URL**: es utilizado para seleccionar del mensaje recibido solo los elementos que necesita para adicionar cada URL a la matriz.

Esta clase además tiene como atributos la matriz donde se adicionan todas las URLs y el vector que especifica cuando se inició el proceso de clasificación para cada URL (timestamp).

2.2.8. Componentes

Entrenamiento

El módulo Decisor para realizar el proceso de entrenamiento depende de la biblioteca FANN.so y accede al fichero de Configuracion.conf para conocer que algoritmo será utilizado para el entrenamiento y la localización del fichero Entrenamiento.txt predefinida para ese algoritmo, luego realiza el entrenamiento y lo guarda un fichero Entrenamiento.net en la dirección indicada por el fichero de Configuracion.conf.

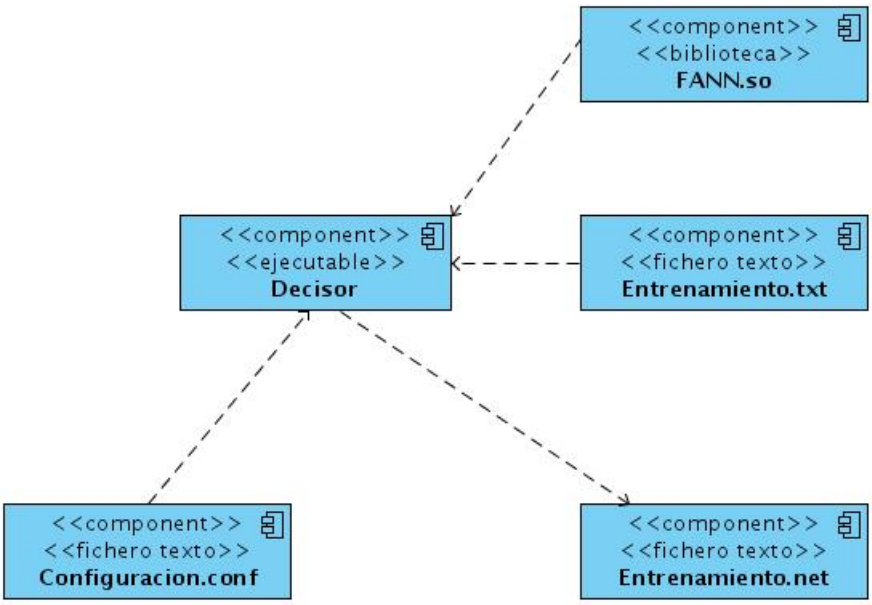

Figura 2.6: Diagrama de componentes: Entrenar

Clasificación

El módulo Decisor para realizar el proceso de clasificación depende de la biblioteca FANN.so y accede al fichero de Configuracion.conf para conocer mediante que algoritmo deberá clasificar, la ruta del entrenamiento realizado para este algoritmo (fichero Entrenamiento.net) , así como la localización del fichero Peticiones.txt donde están almacenadas todas las peticiones llegadas hasta el momento, luego realiza la clasificación del documento HTML y genera una respuesta que guarda en el fichero Respuestas.txt con la estructura predefinida para este tipo de mensaje, la localización de este fichero se la indica el fichero Configuración.conf.

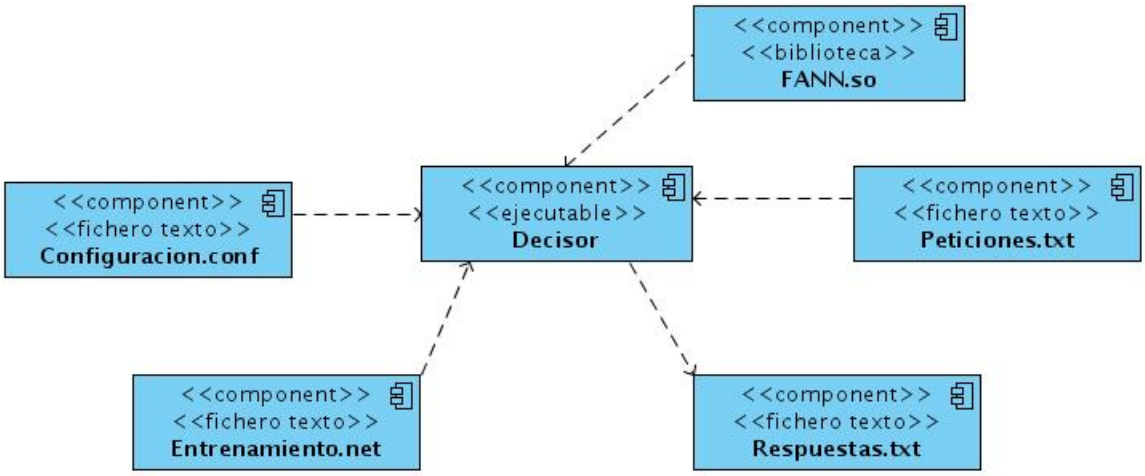

Figura 2.7: Diagrama de componentes: Clasificar

2.3. Selección de la colección de entrenamiento

Este proceso se centra en seleccionar un conjunto de documentos HTML previamente clasificados para entrenar el algoritmo y que pueda, por si solo, clasificar nuevos documentos. A continuación se mencionan los aspectos que se deben tener en cuenta para la selección de la colección de entrenamiento:

- La colección de entrenamiento debe ser una muestra representativa de documentos HTML, previamente clasificados en las 8 categorías definidas por MOCIC.
- Se toman solamente documentos HTML con el texto en español.
- Los símbolos contenidos en la colección de entrenamiento deben coincidir con los seleccionados por el Clasificador de Objetos.

2.4. Conclusiones

En este capítulo se definió el diseño para el Módulo Decisor, luego del profundo estudio realizado en el capítulo 1 de los algoritmos de aprendizaje automático para la solución de problemas de clasificación. Se establecen los algoritmos a utilizar para la clasificación y las técnicas para realizar las pruebas al prototipo funcional.

Conclusiones

Con la culminación del presente trabajo de diploma se cumple con el objetivo trazado en el mismo, mediante el diseño del Módulo Decisor, que permita a MOCIC categorizar de forma automática e ”inteligente”los documentos HTML a partir de la respuesta obtenida de los restantes módulos. Se cumplió además con los siguientes aspectos:

- Se estudió el mecanismo de decisión del sistema de filtrado POESIA, utilizándolo como guía a la solución propuesta.
- Se eligieron como algoritmos de entrenamiento y categorización el Multilayer Perceptron y el k Nearest Neighbour, que cumplen con las características requeridas para los modos de configuración.
- Se implementó un prototipo funcional cumpliendo con el diseño del Módulo Decisor y se integró al sistema.
- El Módulo Decisor solo se entrenó con 50 vectores por categoría, siendo esta una muestra poco significativa.

Recomendaciones

- Entrenar el sistema con una colección de entrenamiento que contenga al menos 1000 vectores por categoría.
- Incorporar nuevos algoritmos para que el mecanismo de filtrado sea más flexible y adaptable a las necesidades del usuario final.
- Implementar una nueva versión del Módulo Decisor incorporándole más descriptores al vector característico para una mejor clasificación.

Referencias

[1] Juventud Rebelde. Internet es vital para el desarrollo de cuba [en línea]. 2002. Disponible en: http://www.juventudrebelde.cu/cuba/2009-02-06/internet-es-vital-para-el-desarrollo-de-cuba/ [visitada 5 de mayo del 2009].

[2] *DM Filtering Components Software*. POESIA, 2003.

[3] *Representación Autocontenida de Documentos HTML: una propuesta basada en Combinaciones Heurísticas de Criterios*. PhD thesis, Escuela Superior de Ciencias Experimentales y Tecnología – Universidad Rey de España – España, Departamento de Ingeniería Telemática y Tecnología Electrónica.

[4] Extracción de información con algoritmos de clasificación. Maestro en ciencias computacionales, Instituto Nacional de Astrofísica, Óptica y Electrónica, 2005.

[5] Extracción y recuperación de infornación no supervisada. 2008.

[6] *Redes Neuronales Difusas para modelado via agrupanmiento en-línea: Aplicación a un condensador de aspiración*. PhD thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional – Distrito Federal – México, Departamento de Control Automático.

[7] *Algoritmos de búsqueda de vecinos más próximos en espacios métricos*. PhD thesis, Universidad Politécnica de Valencia – España, Departamento de Sistemas Informáticos y Computación.

[8] Comparación de topologías mlp y lvq de redes neuronales para la detección de arritmias ventriculares. 2000.

[9] Ing. Camilo A. Zuluaga M. Ing. María Isabel Acosta B., Ing. Harold Salazar I. Tutorial de redes neuronales [en línea]. 2000. Disponible en: http://ohm.utp.edu.co/neuronales/ [visitada 2 de febrero del 2009].

[10] Aplicación de redes neuronales y redes bayesianas en la detección de multipalabras para tareas ir. 2002.

[11] Resolución de la ambigüedad mediante redes neuronales. 2002.

[12] Técnicas de reducción en bases de datos. Technical report, Universidad de Sevilla, 2000. arXiv: LSI-2000-09.

[13] Clasificadores k-nn.

[14] *Manual de Weka*.

[15] Modelo de dominio [en línea]. Disponible en: http://iie.fing.edu.uy/ense/asign/desasoft/practico/hoja8/ejemplos_clase2.pdf [visitada 20 de mayo del 2009].

Bibliografía

- Abdelmalik Moujahid, Iñaki Inza, Pedro Larrañaga. Clasificadores k-nn.
- Alfonso Ballesteros. Tutorial introducción a las redes neuronales [en línea]. Disponible en: http://www.redes-neuronales.netfirms.com/tutorial-redes-neuronales/tutorial-redes.htm [visitada 2 de febrero del 2009].
- Álvaro González Fernández. Extracción y recuperación de la información II: clasificación supervisada [en línea]. 2007. Disponible en: http://supervisadaextraccionrecuperacioninformacion.iespana.es/ [visitada 2 de febrero del 2009].
- Carolina Fuentes T. y Carlos Ibañez D. Inteligencia artificial [en línea]. Disponible en: http://www.comenius.usach.cl/gvillarr/cursoia/alumnos/fuentesibanez/index.html [visitada 2 de febrero del 2009].
- Carolina Fuentes T. y Carlos Ibañez D. Aprendizaje automático [en línea]. Disponible en: http://www.comenius.usach.cl/gvillarr/cursoia/alumnos/fuentesibanez/links%20rellacionado/tema.html [visitada 2 de febrero del 2009].
- DM Filtering Components Software. POESIA, 2003.
- Dr. Diego Andina de la Fuente. Redes neuronales artificiales [en línea]. Disponible en: http://www.gc.ssr.upm.es/inves/neural/ann2/anntutor.htm [visitada 2 de febrero del 2009].
- Extracción y recuperación de infornación no supervisada. 2008.
- Galo Rodríguez Lalangui Eras. Paradigmas de aprendizaje automático [en línea]. Disponible en: http://galopriva.wordpress.com/2008/05/01/paradigmas-de-aprendizaje-automatico/ [visitada 5 de febrero del 2009].

- George Gratzer. Math into latex [en línea]. Disponible en: http://books.google.com.cu/books?id=k1NccNnxcUsC\&dq=math+into+latex\&pg=PP1\&ots=m5XrPpobde\&sig=_qn6yQ4lZ-SF5RTZnNWb2kdcB9U&hl=es\&sa=X\&oi=book_result\&resnum=1\&ct=result [visitada 15 de marzo del 2009].
- Ing. Camilo A. Zuluaga M., Ing. María Isabel Acosta B., Ing. Harold Salazar I. Tutorial de redes neuronales [en línea]. 2000. Disponible en: http://ohm.utp.edu.co/neuronales/ [visitada 2 de febrero del 2009].
- Ing. Sergio R. Richter. Redes neuronales - perceptron multicapa [en línea]. Disponible en: http://www.webelectronica.com.ar/news21/nota09.htm [visitada 2 de febrero del 2009].
- Irene Hyna y Elisabeth Schlegl Tobias Oetiker, Hubert Partl. The not so short introduction to latex [en línea]. Disponible en: http://tobi.oetiker.ch/lshort/lshort.pdf [visitada 15 de marzo del 2009].
- J. J. Merelo Guervós. Mapa autoorganizativo de kohonen [en línea]. Disponible en: http://geneura.ugr.es/~jmerelo/tutoriales/bioinfo/Kohonen-2005.html [visitada 2 de febrero del 2009].
- Joaquín Ataz López. Guía casi completa de bibtex [en línea]. Disponible en: http://www.ctan.org/tex-archive/info/spanish/guia-bibtex/guia-bibtex.pdf [visitada 15 de marzo del 2009].
- Propuesta de guía para la presentación del trabajo de diploma.
- Raúl Mata Botana. Tablas en latex [en línea]. Disponible en: http://www.lug.fi.uba.ar/documentos/tablas/tablas.pdf [visitada 15 de marzo del 2009].
- Resolución de la ambiguedad mediante redes neuronales. 2002.
- Tobias Oetiker. An acronym environment for latex [en línea]. Disponible en: http://www.ctan.org/tex-archive/macros/latex/contrib/acronym/acronym.pdf [visitada 15 de marzo del 2009].
- TREC Soluciones. Redes neuronales artificiales [en línea]. 2001. Disponible en: http://electronica.com.mx/neural/ [visitada 2 de febrero del 2009].

Anexo A

Categorías de MOCIC

Categorías/descripción	I	N	A
Ciencias: sitios de carácter científico (Física, Geografía, Matemáticas, Medio ambiente, Química, Tecnología, Agricultura, Astronomía, Biología, Ciencia alternativa, Ciencia tecnología y sociedad, Ciencias de la Tierra, Ciencias Sociales).			X
Computadoras: sitios relacionados con computadoras; hardware (fabricantes, componentes, precios), software (S.O, programas, temas relacionado al software libre y privativo, computación paralela, Inteligencia Artificial, Juegos, algoritmos), temas de telecomunicaciones, robótica, realidad virtual, seguridad informática.			X
Deportes: sitios relacionados con el deporte; clubes, equipos, federaciones deportivas, resultados deportivos, eventos como: Juegos Olímpicos, campeonatos mundiales.			X
Sustancias Dañinas: sitios donde se promueve el uso de drogas.(LSD, heroína, cocaína, XTC, pot, amphetamines, hemp, éxtasis). Sitios que promueven el consumo de alcohol y el tabaquismo.	X		
Juegos: incluye sitios Web de juegos de computadoras, zonas de Internet de juegos on-line, sitios de vendedores. Sitios que promueven los juegos de azar, casino y agencias de apuestas.			X
Continúa en la próxima página			

Pornografía: incluye sitios Web que contienen actividad sexual explícita, pornografía infantil, la Pedofilia, la Pederastia.	X		
Salud: sitios que promueven la salud; medicinas, hospitales, hábitos dietéticos, enfermedades, primeros auxilios, sexología, salud mental, Publicaciones, Eventos.			X
Violencia: Sitios que hagan apología o inciten a la violencia, a la guerra, al racismo, a la desigualdad entre el hombre y la mujer, a la violencia familiar. Además sitios de fabricación de explosivos, venenos, manuales para fabricación de bombas, instrucciones para asesinar personas. (Falta búsqueda relacionado con bombas y explosivos).	X		

Leyenda

- **I**: Inadecuado
- **N**: Nocivo
- **A**: Admisible

Glosario de términos

ADN Ácido Desoxirribonucleico

ARN Ácido Ribonucleico

BDUC Base de Datos de URLs Categorizadas

CDT C/C++ *Development Tools*:
Es un pluging para Eclipse el cual permite la programación en el lenguaje C/C++

CE capa de entrada

CS capa de salida

CO capa oculta

FANN FANNFast Artificial Neural Network

FIFO First In First Out

FILPACON Filtrado de Paquetes por Contenido:
Es un sistema que pretende ser flexible y fiable para regular los contenidos nocivos de Internet.

GNU *GNU is Not Unix*:
Sistema operativo cuyo nombre es un acrónimo recursivo que significa “GNU No es Unix”

GPL *General Public License*:
Licencia Pública General. Es una licencia creada por la Free Software Foundation (FSF) y orientada principalmente a los términos de distribución, modificación y uso del software.

HTML *HyperText Mark Language*:
Lenguaje de Marcas de Hipertexto. Es el lenguaje de marcado predominante para la construcción de páginas Web.

IA Inteligencia Artificial:
Rama de la informática que desarrolla procesos que imitan a la inteligencia de los seres vivos.

IBL Instance-Based Learning

IDE Integrate Development Enviroment:
Entorno de desarrollo integrado. Herramienta que se usa para facilitar el desarrollo de software.

JavaScript Lenguaje de Script para páginas Web, basado en la sintaxis de Java.

k-NN k *Nearest Neighbour*:
Proviene del idioma ingles y es traducido como los k vecinos más cercanos.

Linux Es el núcleo (kernel) del sistema operativo libre GNU

LMS Least-Mean-Square

LVQ Learning Vector Quantization

MLP Multilayer Perceptron:
Perceptron multicapa. Topología de Red Neuronal definida por Rosenblatt en el año 1957.

MOCIC Motor de Categorización Inteligente por Contenidos

OpenCV Open Computer Vision

OPTENET *Optimal Internet*:
Sistema de filtrado con más fuerza en Europa

OSRI Oficina de Seguridad para las Redes Informáticas

PICS Platform for Internet Content Selection

POESIA Public Open-source Enviroment for a Safar Internet Access:
Es un sistema de filtrado de código abierto bajo la licencia GPL

RNA Redes Neuronales Artificiales

SOM Self Organizing Map

SVM Support Vector Machines

TPM Topology Preserving Map

UCI Universidad de las Ciencias Informáticas

UML *Unified Modeling Language*:
Lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

URL Uniform Resource Locator:
Se refiere al texto que identifica a una página Web.

Web Red:
La traducción literal de esta palabra inglesa es tela de araña, pero en términos informáticos significa mucho más que eso.

WWW World Wide Web:
Telaraña o malla mundial. Sistema de información distribuido con mecanismos de hipertexto. Es el universo de servidores HTTP, que permiten mezclar texto, gráficos y archivos de sonido juntos.