Clasificador de E-mails Anti-Spam utilizando un Perceptrón Multicapa

W. Fuertes, M. Almache y J. Ruiz
Departamento de Ciencias de la Computación, Escuela Politécnica del Ejército, Sangolquí, Ecuador
{wfuertesd, malmache, jaruiz} @espe.edu.ec

RESUMEN: La grandes amenazas del correo electrónico (e-mail) son los virus, gusanos, troyanos, ataques de denegación de servicio y el flujo de mensajes no solicitados conocidos como Spam. Ante este problema, el presente artículo propone la implementación de un filtro para reducir el Spam, basado en un clasificador automático, mediante el entrenamiento de una red neuronal tipo Perceptrón multicapa. Se ha seguido un procedimiento incremental para la determinación del modelo clasificatorio y la elección del método de aprendizaje. En consecuencia, se han implementado algoritmos, tanto para la validación y extracción de la matriz de clasificación, como para el pre-procesamiento de los datos en la red de entrenamiento, normalizando las entradas y salidas en intervalos definidos; asimismo, se ha recurrido al algoritmo de entrenamiento Backpropagation, inherente a los Perceptrones multicapa. Los resultados experimentales muestran que, la aplicación de las redes neuronales, es una opción viable para detectar Spam; sin embargo, para obtener un aprendizaje neuronal óptimo, los ejemplos de entrenamiento deben ser apropiadamente seleccionados y lo suficientemente refinados.

ABSTRACT: The major threats in the usage of electronic mail (e-mail) are viruses, worms, trojans, denial of service attacks, and the unsolicited bulk messages flow known as Spam. Faced with this problem, this paper proposes the implementation of a filter to reduce Spam based on an automatic classifier by means of a type of neural networks denoted multilayer Perceptron. It follows an incremental method for determining the classificatory model and the choice of learning method. In this work, algorithms have been implemented, both for validation and extraction of the classification matrix and for the network training pre-processing, as well as standardizing inputs and objectives in defined intervals. In addition, it has been used to backpropagation training algorithm which is inherent to the multilayer Perceptron. Experimental results show that the application of neural networks is a viable option to detect Spam. However, to get an optimal learning, training examples must be properly selected and refined more.

1 INTRODUCCIÓN

La grandes amenazas del correo electrónico (e-mail) son los virus, gusanos, troyanos, ataques de denegación de servicio y el flujo de mensajes no solicitados conocidos como Spam. El Spam está siendo utilizado para anunciar productos y servicios de dudosa calidad. Técnicamente el Spam se refiere a las acciones destinadas a engañar a los motores de búsqueda, llevando a una degradación de los resultados de búsqueda y a la saturación de servidores de e-mail [1]. En un estudio de America Online, Inc. (AOL) y Microsoft Network (MSN) se reportó un total diario de 2,4 millones de Spams en e-mails entrantes, cifra que correspondía a un incremento del 80% del tráfico Spam [2]. Estos datos son alarmantes si se considera que su proliferación aún no tiene una solución definitiva.

El Spam consiste normalmente en cadenas de mensajes con esquemas piramidales, avisos de servicios de envío indiscriminado de e-mails publicitarios, software ilegal, productos milagrosos, ofertas de muy dudoso origen, etc. Todo esto desencadena un sinnúmero de problemas como fraudes, desplazamiento de e-mails genuinos, interrupción de servicio, incremento de costos incurridos en la transmisión de cada e-mail, consumo de procesador y memoria debido al incremento del volumen de información, etc.

En los últimos años, la comunidad científica ha mostrado un creciente interés en investigar nuevos enfoques para prevenir el Spam [3]. Entre ellos se puede citar: ***i)*** "Detección y prevención del Spam en los blogs", que es la forma automática de promocionar servicios comerciales en los blogs, libros de visitas u otros medios de acceso público sin permiso alguno; ***ii)*** "Filtrado de Spam basado en el contenido", que bloquea o permite un patrón de búsqueda. El mas popular de este tipo es un filtro estadístico que utiliza el Clasificador Bayesiano de Naive; ***iii)*** "Identificación de sitios de Spam", que es un enfoque que se basa en clasificar páginas o sitios como Spam. El inconveniente de este enfoque es que el Spam no es característica esencial de páginas o sitios Web, sino de mensajes instantáneos; y ***iv)*** Detección y prevención de Spam a través de algoritmos genéticos, que son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y que están basados en el proceso genético de los organismos vivos.

Sin embargo, la mayor parte del Spam que se recibe es especialmente difícil de detectar ya que los spammers (individuos o empresas que envían Spam) activamente cambian la naturaleza de sus mensajes para eludir los filtros anti-Spam.

Frente a este escenario, como contribución, el presente artículo se basa en la definición de una técnica complementaria de filtrado de Spam basado en contenido. Para llevarlo a cabo, se propone la implementación de un filtro para reducir el Spam, basado en un clasificador automático, mediante el entrenamiento de una red neuronal tipo Perceptrón multicapa [4]. Para ello, se ha seguido un procedimiento incremental para la determinación del modelo clasificatorio y la elección del método de aprendizaje. De esta forma, dado que las redes neuronales se basan en el reconocimiento de patrones[5], la premisa de nuestro aporte es que cada mensaje se puede cuantificar en función de un patrón. Para identificar estos patrones, la red neuronal debe ser "entrenada". Este entrenamiento incluye un análisis computacional del contenido completo de los mensajes utilizando muestras representativas de Spam y No Spam. En particular, nuestra meta consiste en comprobar que el Spam puede ser detectado entrenando una red neuronal y verificar el porcentaje de precisión (% de aciertos) de dicho aprendizaje.

El resto del artículo ha sido organizado como sigue: La sección 2 describe los fundamentos del aprendizaje basado en redes neuronales. La sección 3 detalla el diseño e implementación del algoritmo anti-Spam. En la sección 4 se muestran los resultados experimentales. En la sección 5, se analizan algunos trabajos relacionados. Finalmente, en la sección 6, se presentan las conclusiones y líneas de trabajo futuro sobre la base de los resultados obtenidos.

2 APRENDIZAJE BASADO EN REDES NEURONALES

2.1 Método de clasificación supervisada

El aprendizaje automático es considerado un pilar muy importante de investigaciones para la Inteligencia Artificial, ya que, presenta la posibilidad de implementar aplicaciones informáticas de alto desempeño, debido a las consecuencias autonómicas del aprendizaje.

La naturaleza del aprendizaje "supervisado" hace referencia al hecho de que, debe existir un ente externo (llamado supervisor o maestro) encargado de proporcionar los ejemplos necesarios para que el aprendizaje se ejecute de forma efectiva en el receptor. De forma más específica, el supervisor proporciona tanto las entradas como las respuestas de tales ejemplos, al tiempo que mide el desempeño del receptor. De esta manera, el aprendizaje supervisado, viabiliza la posibilidad de que las máquinas puedan realizar actividades de clasificación con los ejemplos proporcionados por el maestro.

2.2 Aprendizaje automático mediante redes neuronales

El modelo de aprendizaje conexionista (basado en "redes neuronales"), está inspirado en una simplificación del sistema neuronal biológico (cerebral). Una red neuronal artificial está conformada por elementos de cálculo aritmético (llamadas neuronas), conectadas entre sí.

Existen diversas formas de clasificar a las redes neuronales, dependiendo de aspectos tales como: la topología (pe. cascada, recurrentes), la naturaleza de la información que procesan (digitales, analógicas), el algoritmo de entrenamiento (determinista, estocástico), el tipo de aprendizaje (pe. supervisado, no supervisado), temporalidad del entrenamiento (off line, on Line), etc. No obstante, lo realmente relevante, en cualquier configuración neuronal, es el hecho de que, si la red ha sido diseñada para ejecutar alguna tarea especial, puede "aprender" a ejecutarla. Los algoritmos de aprendizaje han sido motivo de mucha investigación en el ámbito de las neurociencias; desde un punto de vista simplista, tales algoritmos, realizan una tarea de "búsqueda" de los pesos sinápticos que posibiliten un desempeño satisfactorio de las redes neuronales, en tareas como: el reconocimiento de patrones, la clasificación y, la interpretación de datos, principalmente.

2.3 Red neuronal Perceptrón Multicapa.

Es una red neuronal en cascada (FeedForward); esto quiere decir que cada una de las neuronas ubicadas en un nivel N (o capa N), solamente puede estar conectada con neuronas de capas de nivel M (N<M). Las capas del Perceptrón, le permiten solucionar problemas en los que, los datos no son linealmente separables (deficiencia del Perceptrón simple). Sin embargo, la principal dificultad del Perceptrón multicapa radica en el hecho de que, puede alcanzar prematuramente un "mínimo local" en la función de error, definida en base a los pesos sinápticos de la red. Sin embargo, a pesar de todas las dificultades, el Perceptrón multicapa tiene desempeños satisfactorios en problemas relacionados a asociación de patrones, segmentación de imágenes, compresión de datos, etc. Para efectos prácticos, es suficiente una topología de tres capas para un Perceptrón que solvente problemas de separabilidad lineal.

2.4 Algoritmo de entrenamiento Backpropagation

Backpropagation (retro-propagación) es un algoritmo de aprendizaje de naturaleza determinista, desarrollado para el entrenamiento de redes neuronales en cascada (Perceptrones multicapa). Se lo conoce como retro-propagación ya que, los cambios de pesos calculados, se propagan desde la capa de salida hacia las capas de atrás. La variación de un peso en particular, es proporcional al "error" detectado en la neurona asociada al referido peso.

Por lo tanto, el algoritmo procura minimizar una "función de error" (comúnmente cuadrático) de los pesos sinápticos, por medio de gradiente descendiente (o descenso de colina); así, la parte esencial del algoritmo, consiste en calcular las derivadas parciales de dicho error con respecto a los parámetros de la red neuronal.

La función de transferencia que se utiliza en este algoritmo es conocida como función sigmoidal, y está definida por la ecuación (1):

$$\varsigma(\mu) = \frac{1}{1+\exp(-\mu)} \tag{1}$$

Siendo μ el "valor de incidencia externa" calculado en cada neurona de la red. Esta función, además de ser diferenciable, tiene la particularidad de que su derivada se puede expresar en términos de sí misma (ver ecuación (2)):

$$\frac{d\varsigma(\mu)}{d\mu} = \varsigma(\mu)(1-\varsigma(\mu)) \tag{2}$$

Adicionalmente, permite realizar un "descenso suave" por la función de error, posibilitando la simplificación de los cálculos en el algoritmo de aprendizaje descrito a continuación:

i) *Tomar un parámetro de rapidez* (r)*, que representa la "velocidad" de aprendizaje;*

ii) *Mientras el desempeño de la RN no sea satisfactorio*

- *Para cada muestra de entrada*

  ** Calcule la salida observada* (Or) *en cada neurona de la capa de salida* z *;*

  ** Calcule el error de salida* (Ez) *para cada neurona de la capa de salida* z

  $(Ez) = (Dz) - (Or)$*; //*(Dz) *es la salida deseada en la neurona de la capa z;*

  ** Calcule el error* E_j *para todos los demás nodos, así:*

  $E_j = \sum_k W_{jk} Or(1 - Or) E_k$ *; //*E_k *error de neurona en capa* $_k$

  //W_{jk} *peso o enlace entre neuronas* $_j$ *y* $_k$ *;*

  ** Calcule los incrementos de peso* (ΔW_{ij})*, para todos los pesos*

  $(\Delta W_{ij}) = E_j r O_i O_j ((1 - O_j) E_j$ *;*

- *Sume los cambios en los pesos de todas las muestras (cambie los pesos).*

En esencia, la medida del desempeño satisfactorio, estará en función del mínimo margen de error que se le permitirá a la red neuronal. En general, a mayor precisión de la red, se requiere más tiempo de entrenamiento (mayor número de épocas o períodos procesados)

3 IMPLEMENTACION

En esta sección se describe el método utilizado para el entrenamiento de la red neuronal y la implementación del filtro anti-Spam. La Figura 1 muestra los componentes y procedimiento incremental que fueron utilizados y que serán analizados a continuación, de manera secuencial:

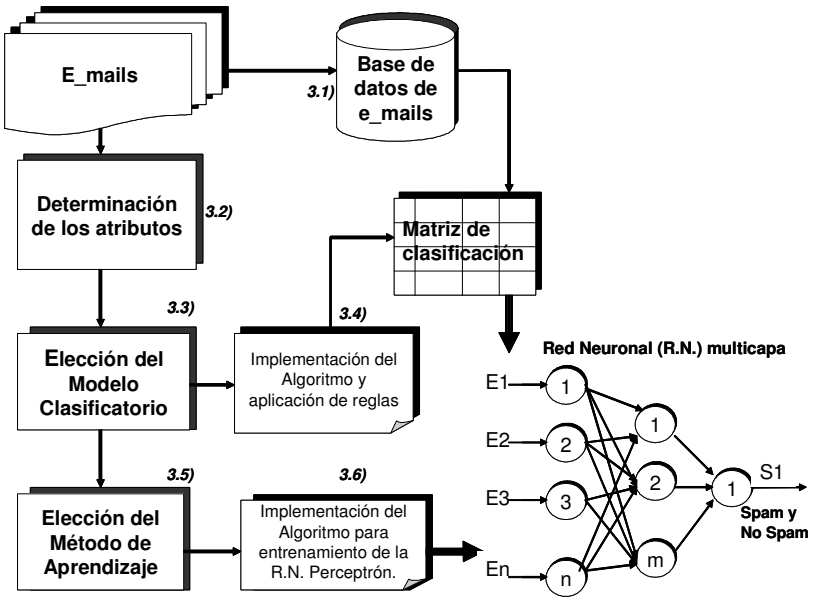

Figura No. 1 "Componentes y procedimiento incremental para la implementación del anti-Spam"

3.1 Creación de la base de datos de e-mails

Para proceder a la implementación, se extrajo 2000 e-mails que fueron copiados en formato tipo texto plano y que luego fueron concatenados en un solo fichero general denominado Base de datos de e-mails. Los e-mails fueron extraídos de un servidor de e-mail institucional, que se ejecuta bajo Linux, de la cuenta de uno de los autores de este artículo.

3.2 Determinación de los Atributos

Dado que esta técnica utiliza palabras clave a ser comparadas, fue necesario identificar cada uno de los atributos de cada e-mail y seleccionarlos en base a la información que podían aportar. Estos atributos fueron: fecha; remitente; destinatario; asunto; idioma; archivo adjunto.

3.3 Elección del Modelo Clasificatorio

Sea un conjunto de N e-mails, cada uno caracterizado por n + 1 variables. Las primeras n variables, $\chi_1, \chi_2, \ldots, \chi_n$, serán denominadas variables predictoras, y la variable de índice $n+1$, identificada como c, será denominada clase. Estos datos pueden representarse en forma tabular utilizando la siguiente notación (ver Tabla 1):

Tabla 1. "Elementos en una clasificación supervisada".

	χ_1	χ_2	.....	χ_i	.....	χ_n	c
1	χ_1^1	χ_2^1	.....	χ_i^1	.....	χ_n^1	c^1
.	.	.	.....	.	.....	..	.
.	.	.	.....	.	.....	..	.
j	χ_1^j	χ_2^j	.....	χ_i^j		χ_n^j	c^j
.	.	.	.....	.	.....	..	.
.	.	.	.....	.	.....	..	.
N	χ_1^N	χ_2^N	.....	χ_i^N	.....	χ_n^N	c^N

Donde:

χ_i^j será el valor que en el j-ésimo hecho toma la variable predictora i-ésima;
$i = 1;.....;n$ que representan a las instancias o atributos;

$j = 1;.....;N$ que representan a cada uno de los e-mails;

c^j será la clase a la que pertenece el j-ésimo correo, y que en este caso será Spam o No Spam.

El objetivo es conseguir un modelo clasificatorio que posibilite la clasificación del valor de la variable c ante la llegada de un nuevo caso, formado por n variables predictoras y del que se desconoce el valor de c. Es decir, clasificar correctamente un nuevo e-mail basándose en las evidencias anteriores.

3.4 Implementación del algoritmo para disponer de una Matriz de clasificación.

Para ello, fue necesario disponer de una base de conocimiento amplia y de información confiable. "*Amplia*", ya que cuanto más número de casos se disponga sobre el problema, mejores serán las aproximaciones obtenidas; y de "*información confiable*", ya que la presencia de casos repetidos, ruido, errores en los datos, variables irrelevantes (que aporten poca información), puede hacer que el modelo no se ajuste correctamente a la realidad, siendo sus predicciones erróneas. Con este fin se implementaron los siguientes algoritmos en MatLab [6]:

a) Análisis_de_contenido.m: Con el fin de normalizar la información mediante la aplicación de reglas y tipos de filtros anti-Spam. Además se utilizaron artificios de programación para cambiar el texto a mayúsculas, ignorar palabras sin vocales, expresiones cifradas y palabras que tengan caracteres especiales.

b) Crea_Matriz_de_Clasificación.m: Que emplee una búsqueda lineal que cuantifique el histograma de cada atributo en cada correo. Una vez preparada la matriz se determinó la clase de correo, registrándose 1 -1 para Spam, -1 1, (simulador bipolar de variables objetivo) no Spam. Este método fue utilizado por su naturaleza de ser clasificador supervisado y por su facilidad de aproximación. La matriz resultante fue similar a la Tabla 1.

3.5 Elección del método de aprendizaje

Se requirió de un método basado en reconocimiento de patrones, que sea un aproximador matemático, que resuelva el problema OR exclusivo (XOR) y sirva para predicción o clasificación. En razón de que todo problema relacionado a estas características, puede resolverse mediante Inteligencia Artificial, y por su potencial matemático, se decidió entrenar una red neuronal Perceptron Multicapa. Por esta característica se ha aplicado el algoritmo de entrenamiento conocido como *Backpropagation* o regla generalizada delta de errores, que es un algoritmo de inclinación de la pendiente para minimizar el error total al cuadrado en la salida computada de la red, tal como se explicó en la sección 2.

3.6 Implementación del Algoritmo para el entrenamiento de una red neuronal

El cual inició desde la recuperación de la matriz de clasificación, la implementación en MatLab del algoritmo de aprendizaje descrito en la sub-sección 2.4 y la obtención gráfica de resultados.

4 RESULTADOS EXPERIMENTALES

4.1 La matriz de correspondencia:

Como resultado del procesamiento del algoritmo de análisis de contenido, se obtuvo una serie de palabras claves de Spam, que constituyó la matriz de correspondencia obtenida y que fue el insumo del aprendizaje de la red neuronal.

4.2 Entrenamiento y Validación:

Con el fin de contrastar el algoritmo de entrenamiento, se realizaron varias pruebas de validación regulando parámetros de entrenamiento como el factor de rapidez, el error permisible, etc. Sin embargo, ya para efectuar el análisis de resultados, el entrenamiento se ajustó a 10 neuronas ocultas, dos neuronas de salida y el Back propagation. La meta (Goal) de entrenamiento fue 0.002. Para el cumplimiento de la meta, se incrementó el número de periodos (epochs), manteniendo constantes los otros parámetros de entrenamiento. Una vez ejecutado el entrenamiento, se verificó el porcentaje de aciertos, comparándolo con los objetivos de validación. Los resultados obtenidos son los siguientes (ver Figuras 2 y 3):

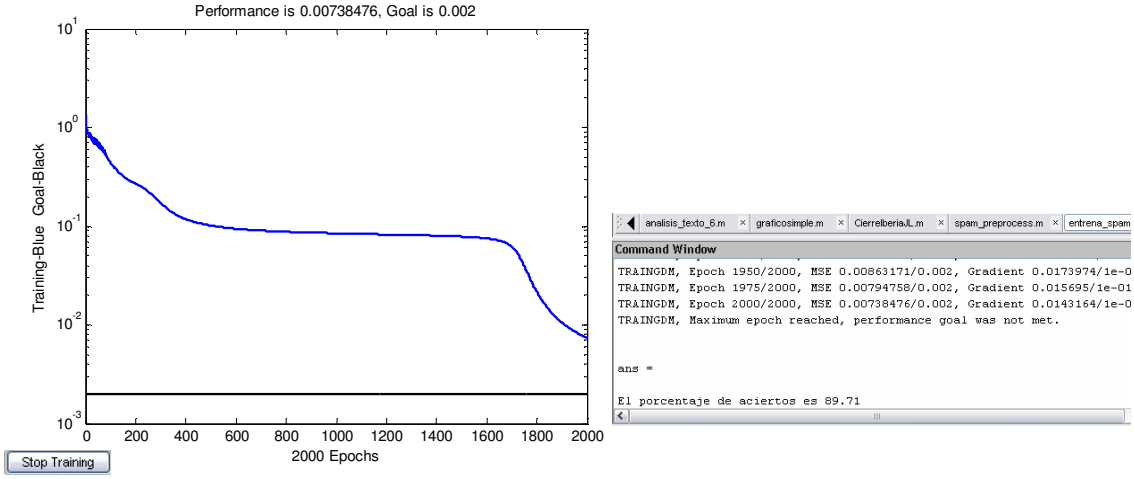

Figura No. 2 “Entrenamiento con 2000 períodos (izquierda), y porcentaje de aciertos (derecha)”

La Figura 2 muestra que el porcentaje de aciertos equivale al 89,71% sin embargo la red no ha convergido, pues no se cumplió la meta en el entrenamiento con 2000 periodos

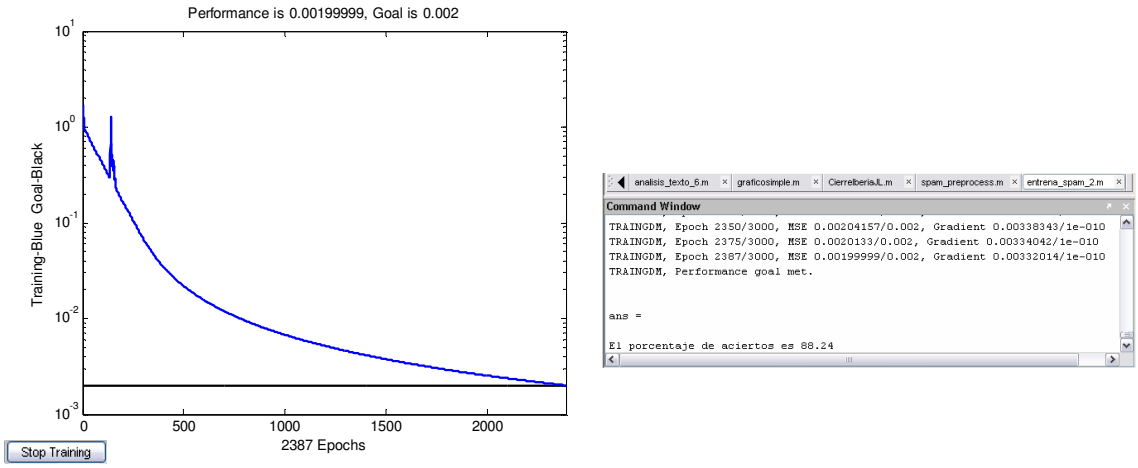

Figura No. 3 "Entrenamiento con 3000 períodos (izquierda), y porcentaje de aciertos (derecha)"

La Figura 3 muestra que el entrenamiento con 3000 períodos ha tenido un menor porcentaje de acierto (-1.47%) que cuando fue aplicado al de 2000 periodos de entrenamiento. Sin embargo la red convergió cumpliendo la meta de rendimiento (88.24%), apenas en los 2387 períodos. Esto obedece a que todo entrenamiento de red neuronal padece de un grado de incertidumbre. Por ello, es requerido combinar coherentemente el factor de aprendizaje, el factor de rapidez, el error permisible, la cantidad de ejemplos, etc. Por lo tanto, se puede inferir que para obtener un aprendizaje neuronal óptimo, los ejemplos de entrenamiento deben ser apropiadamente seleccionados, suficientemente refinados, con factores de aprendizaje bien combinados.

5 TRABAJOS REALACIONADOS

Aunque exista una diversidad de trabajos relacionados, en esta sección se han incluido los más relevantes, que se han encontrado durante la investigación:

En lo que se refiere a filtros anti-Spam basados en Algoritmos Genéticos, el trabajo presentado en [7] describe la aplicación de un filtro anti-Spam basado en los patrones de las cabeceras de los e-mails, de tal forma que devuelve información cifrada que evita de cualquier manera que se continúe enviando e-mails. Los autores también muestran como los Algoritmos Genéticos desarrollan mejores niveles de eficiencia que los filtros de Redes Bayesianas de Naive. Sin embargo el uso de estas técnicas requiere de una gran potencia de análisis.

En lo que se refiere a filtros anti-Spam en las redes sociales (Facebook, MySpace, etc.), el trabajo propuesto por Goldbeck y Hendler en [8], presentan un algoritmo que infiere una puntuación de confianza para un determinado emisor sobre la base de las relaciones sociales, pero comprometen la privacidad de los usuarios, al exigir que se publiquen sus cuentas.

En relación a la aplicación de filtros Bayesianos, el trabajo propuesto por Carpinter y Hunt [9], proponen filtros que clasifican e-mails sin la interacción humana. Por los resultados obtenidos, se puede deducir que, como el clasificador de Bayes utiliza el mismo patrón de reconocimiento en base a la información en las cabeceras de los e-mails, podría no ser muy efectivo, en razón de que los spammers saben como eludir el reconocimiento de patrones. En este mismo contexto, en [10] se describe los experimentos utilizando dos eventos de modelado, el primero le corresponde a Bernoulli, el segundo es un Modelo Multinomial, cuyos resultados demuestran la mejoría en la precisión del modelo.

Respecto a investigaciones basadas en redes neuronales, el trabajo propuesto en [11] presenta un sistema para la automatización de e-mail clasificando en carpetas y filtrado anti-spam. Los experimentos muestran la aplicación de diversas características de selección, ponderación y normalización de métodos, así como la portabilidad del filtro anti-spam a través de los diferentes usuarios. Comparado con nuestro trabajo, se ha implementado el filtrado anti-Spam basado en el contenido, mediante la detección de un conjunto de palabras clave existentes

en un e-mail o en los archivos adjuntos; con la particularidad que fue basado en un clasificador automático, mediante la técnica de aprendizaje de una red neuronal Perceptrón Multicapa. Además, comparándolo con los clasificadores Bayesianos, con matices de rigurosidad y complejidad matemática, el enfoque neuronal tiene la ventaja de ser una propuesta sencilla y eficaz de clasificación, y no requiere de un conocimiento previo del problema a resolver.

Finalmente, en relación a otro tipo de técnicas, el trabajo presentado por Cook et al. [12], se enfocó a aplicar filtros anti-Spam antes que éste llegue a su destino, utilizando técnicas de detección en tiempo real sobre los servidores de e-mail, a través de un software que puede identificar la dirección IP del Spammer para bloquear los e-mails. Dada la complejidad de las redes de comunicación, aún se necesita mayor investigación futura para su correcta aplicación.

6 CONCLUSIONES Y TRABAJO FUTURO

En esta investigación se ha comprobado que las redes neuronales son una opción viable y eficaz para detectar Spam. Sin embargo, para obtener mejores resultados, es necesario tener una cantidad suficiente de análisis de datos, tanto en la matriz de clasificación como en la tabla de correspondencia. Para el entrenamiento de la red, se requiere que los datos sean debidamente refinados en una matriz de clasificación, que contenga los elementos de una clasificación supervisada. Según los parámetros de aprendizaje, a mayor entrenamiento, mejor aprende la red. En la fase de validación del aprendizaje, se determinó un 90% de aciertos en la detección de Spam inmerso en correos reales; en consecuencia, el algoritmo es efectivo y confiable. No obstante, el desempeño de la red puede ser mejorado, regulando parámetros de entrenamiento como: el factor de rapidez, el error permisible, la cantidad de ejemplos, etc. Una restricción de nuestra solución es que el algoritmo debe revisar completamente cada e-mail para detectar las palabras clave obtenidas en la matriz de correspondencia. Así mismo, se ha detectado que cada atributo de los e-mails no aporta la misma calidad de información, por lo que el aprendizaje de la red, según va incrementando el volumen de información se hace más lento.

Como trabajo futuro se aplicarán técnicas de selección o reducción de variables, tipo PCA (Análisis de Componentes Principales) para resolver las restricciones de nuestro algoritmo.

REFERENCIAS

[1] Z. Gyongyi and H. Garcia-Molina. “Web Spam taxonomy”. In Proc of First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), 2005. Held in conjunction with the 14th International World Wide Web Conference, Japan.10 May 2005.

[2] L. Cazita, C. Almeida, J. Almeida, V. Almeida and W. Meira, “Characterizing a Spam traffic”, In Proc. of the 4th ACM SIGCOMM conference on Internet measurement, 2004, Pp(s): 356-369, Sicily, Italy.

[3] G. Mishne, D. Carmel and R. Lempel, “Blocking Blog Spam with Language Model Disagreement”. In Proc of Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web AIRWeb 2005. Held in conjunction with the 14th International World Wide Web Conference, 10 May 2005, Japan.

[4] R. Duda, E. Hart and G. Stork. “Pattern Classification”. Wiley, 2000. ISBN 0471056693

[5] C. Miller. “Neural Network-based Antispam Heuristics”. Symantec Enterprise Security. White paper. 2006.

[6] User guide and technical referece of Toolboxs, MatLab 7.2.0

[7] F. Garcia, J. Hoepman, J. Van Nieuwenhuizen. “Spam filter análisis”. Edited by Kluwer Academic Publishers of CiteSeerX - Scientific Literature Digital Library and Search Engine. 2008

[8] J. Golbeck, J. Hendler “Spam Filter Analysis”. In Proc. of International Conference on Collaborative Computing”. USA, July, 2005.

[9] J. Carpinter and R. Hunt “Tightening the net: A review of current and next generation spam filtering tools”. Computers & Security. Volume 2 5. 2006. Pp(s): 566 – 578.

[10] K. M. Schneider, “A Comparison of Event Models for Naïve Bayes Anti-Spam E-Mail Filtering”, Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics. Budapest, Hungary, Volume 1, Pp(s): 307–314. 2003.

[11] J. Clark, I. Koprinska, J. Poon. “A Neural Network Based Approach to Automated E-mail Classification”, In Proc. of the 2003 IEEE/WIC International Conference on Web Intelligence, USA, Pp(s): 702 – 705

[12] D. Cook and J. Hartnett, K. Manderson and J. Scanlan. “Catching spam before it arrives: domain specific dynamic blacklists**”.** In Proc. of the 2006 Australasian workshops on Grid computing and e-research – Australian Computer Society, Inc. Volume 54. Pp(s): 193-202.