ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Implementación y evaluación de un detector masivo de Web Spam"

INFORME DE MATERIA DE GRADUACIÓN

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN ESPECIALIZACIÓN SISTEMAS DE INFORMACIÓN

Presentada por:

WASHINGTON RAÚL BASTIDAS SANTOS

JESUS HUMBERTO GONZÁLEZ VERA

Guayaquil - Ecuador

2009

AGRADECIMIENTO

A Dios autor de mi existencia y responsable directo de mis grandes oportunidades a Él sea siempre todo la honra.

A mis padres por su sacrificio incondicional y desmedido, a mis hermanos por su apoyo y ánimo. A todos ellos por su amor irremplazable que ha servido de aliciente en toda etapa de mi vida.

AGRADECIMIENTO

Le agradezco a Dios ante todo por darme el regalo más grande que es la vida y colmado de bendiciones.

Le agradezco a mis padres, si no fuera por ellos no podría haber llegado hasta estas instancias de mi vida, a mis hermanos que gracias a su cariño y motivación me respaldaron siempre durante todo este proceso.

A la Ing. Cristina Abad, que gracias a su impartirnos sus conocimientos y darnos su guía pudimos terminar este proyecto.

A mi compañero Jesús González, que gracias a su perspicacia, inteligencia e interés pudimos concluir un proyecto del cual sentirnos satisfecho.

TRIBUNAL DE SUSTENTACIÓN

MSc. Cristina Abad
PROFESOR DE LA MATERIA

Ing. Federico Raue

PROFESOR DELEGADO DEL DECANO

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Trabajo de Graduación, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la "**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**".

_____ _____

Washington Raúl Bastidas Jesús Humberto González

ÍNDICE GENERAL

ÍNDICE GRÁFICOS

ÍNDICE TABLAS

INTRODUCCIÓN

En el mundo actualmente globalizado el acceso, recuperación y reutilización de la información es una prioridad fundamental en el diario vivir de las personas. El uso de las máquinas de búsqueda para realizar dichas tareas han sido esenciales debido al crecimiento de la información existente en la Web; pero, a causa de las falencias existentes en las máquinas de búsqueda [1], tales como darle prioridad a palabras que tienen mayor valor monetario o problemas con la facilidad de manipular los algoritmos de búsqueda como el PageRank [9]; así como de los servicios gratuitos de gestión de información, algunos internautas han logrado beneficiarse de los medios existentes, manipulando información maliciosamente.

El mecanismo comúnmente usado para realizar esta forma ilícita de manipulación es el Web spam [4], que es simplemente la asignación injustificable de relevancia a una o varias páginas produciendo resultados inesperados en las máquinas de búsqueda, deteriorando así la calidad de las consultas y su veracidad.

Dado que existe un gran tráfico de información en la Web a través de los buscadores y el alto valor monetario que éste acarrea, es normal que algunos operadores de sitios en la Web traten de influir en el posicionamiento dentro de las páginas de resultados que nos ofrece

los buscadores [2]. La forma en la que nosotros atacamos esta problemática es a través del uso de técnicas de máquinas de aprendizaje, específicamente *máquinas de vectores de apoyo* (SVM, por sus siglas en inglés) [3], las cuales permiten una optima clasificación para la de posibles página con contenido Web spam.

Los principales buscadores en los últimos años han comenzado una fiera batalla contra el Web spam. Se han logrado muchas mejoras a través de este tiempo pero, aún no existe un método lo suficientemente eficiente como para acabar con este tipo de problema. Como agravante a esta situación, tenemos que cada vez que se encuentra una solución parcial al problema de Web spam, los spammers [8] se encargan de buscar otra forma de eludir este mecanismo, lo que hace de esto una lucha interminable. Aunque no es muy difícil para una persona experta detectar un Web spam, la meta es poder suplir este tipo de métodos debido a la gran cantidad de páginas existentes y tener una solución automatizada que aprenda rápidamente de los nuevos métodos de ataque.

CAPÍTULO 1.

1. DESCRIPCIÓN DEL PROBLEMA

El presente trabajo documenta un módulo que clasifica las páginas HTML en buenas o malas, entendiendo por malas a aquellas que han sido detectadas como Web spam, a través de una máquina de vectores de apoyo.

Los principales aspectos del proyecto son:

- Obtener las características previamente establecidas de cada página para poder llenar los vectores con los que trabajamos en la máquina de aprendizaje.
- Utilizar algoritmos de aprendizaje y métodos para clasificar las páginas Web, analizando tanto el contenido como los hipervínculos de las páginas HTML.
- Realizar el proceso de evaluación de características (predicción) de nuevas páginas detectando y asignando un valor como Spam o no Spam.

1.1 Web Spam

El término *Web spamming* [4] es definido como cualquier acción humana deliberada que permite dar a lugar una relevancia injustificable de valor e importancia de algunas páginas.

La Web contiene muchas personas que buscan la forma de beneficiarse de los millones de usuarios en la red a un bajo costo. Por lo tanto hay un incentivo económico para manipular los motores de búsqueda mediante la creación de páginas que hacen que otras páginas destaquen y tenga un mejor ranking dentro de los resultados de los buscadores, independiente de su real mérito.

Una larga fracción de visitantes de Websites se origina de las máquinas de búsqueda, y se ha comprobado que los usuarios en su mayoría solo observan las primeras páginas mostradas por el buscador [5]. Estos argumentos han sido suficientes para originar una lucha interminable entre quienes buscan optimizar las consultas a través de las máquinas de búsqueda y entre los ya mencionados spammers [6]. Existen básicamente dos tipos de Web spam, de los cuales se hablará un poco a continuación.

1.2 Spam basado en contenidos

La primera generación de buscadores tenía básicamente mucha confianza en el clásico modelo de vectores espaciales del cual obtenían información. Así los primeros Web spammers manipulaban el contenido de las páginas repitiendo los keywords muchas veces. Muchas páginas generadas de esta manera aún existen en la actualidad. Este tipo de resultados pueden caracterizarse como anómalos a través de un análisis estadístico [7], centrándose en la naturaleza de la estructura con la que cuenta la página. Además, éstas estructuras no solo cuentan con las palabras más usadas sino también con palabras escritas erróneamente tales como “googel”, “accomodation” que suelen ser comúnmente escritas. Ntoulas [2] nos describe qué el tipo de características que se debería tomar en cuenta, son como el número de palabras, el tamaño del título, así como la fracción de palabras populares con las que cuenta la página.

Se ha demostrado, que en este tipo de Web spam tiene bastante éxito debido a que las máquinas de búsqueda no realizan filtros de spam en las consultas más populares y mejor pagadas [8]. Esta lista de consultas fácilmente puede ser encontrada en el servicio de pago por click de Google AdWords (http://adwords.google.com).

1.3 Spam basado en enlaces

Google a través de su algoritmo PageRank realiza el ranking de las páginas en la búsqueda, marcó la pauta para que los otros buscadores comenzaran a usar este tipo de algoritmos. A partir de este momento también nació una gran cantidad problemas con el spam, debido a la facilidad que otorga este algoritmo para realizar este tipo de acciones.

Una de las principales maneras de realizar este tipo de spam es creando granjas de enlaces [6], que son un conjunto de páginas densamente conectadas, creadas explícitamente con el fin de engañar a los algoritmos basados en el ranking de los enlaces (ver ejemplo en Figura 1.1). Esto también es definido como *colusión* [10] que es “la manipulación de la estructura de los enlaces por un grupo de usuarios que intentan mejorar el ranking de uno o más usuarios en el grupo”.

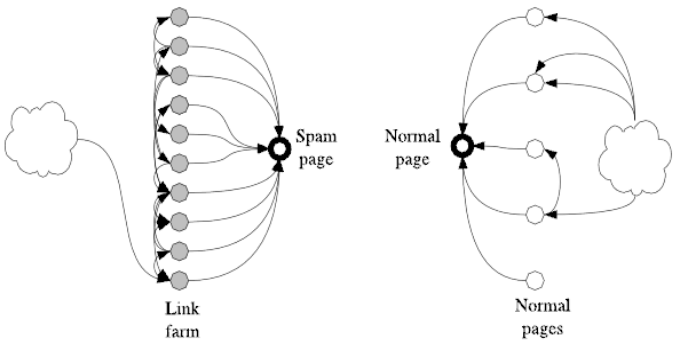


Figura 1.1 Esquema de los vecinos de una página participando en una granja de enlaces [6]

En la Figura 1.1 se ilustra la diferencia entre una página a la que se está tratando de elevar el ranking con otra página normal. Por lo general, las páginas que son spam no tienen mucha relación con la granja de enlaces, para evitar ser detectadas.

CAPÍTULO 2.

2. METODOLOGÍA

2.1 Aprendizaje automático

El aprendizaje automático es el estudio de métodos para permitir a las computadoras aprender [11]. El área de máquinas de aprendizaje tiene que lidiar con el diseño de programas que puede aprender de los datos, adaptarse a los cambios y mejorar el desempeño con la experiencia. En la actualidad tiene varias aplicaciones entre las que tenemos:

- Clasificación de proteínas en base a secuencias de ADN
- Detección de spam
- Biometría computacional
- Reconocimiento de patrones
- Minería de datos

El *aprendizaje o entrenamiento* es el mejoramiento en base a la experiencia de alguna tarea. En general, para tener un problema de aprendizaje bien definido [21], debemos identificar 3 características:

a. El tipo de tarea.
b. La medida del desempeño a ser mejorada.

c. La fuente de experiencia.

El presente trabajo utiliza técnicas de aprendizaje supervisado y hablaremos más al respecto a continuación.

2.2 Aprendizaje Supervisado y Clasificación

El Aprendizaje automático supervisado es la búsqueda de algoritmos que razonen a partir de ejemplos provistos externamente para producir hipótesis generales, que entonces hagan predicciones acerca de ejemplos futuros. En otras palabras la meta del aprendizaje supervisado es construir un modelo conciso de distribución de las etiquetas de clases en términos de un predictor de características [29].

Entre los tipos de aprendizaje supervisado se encuentra el de clasificación, el cual dado un conjunto de datos donde cada dato pertenece a una clase, construye un modelo que permite predecir la clase de un nuevo dato, en general el modelo es mejor si el error de predicción es menor. En nuestro proyecto se utiliza este algoritmo para clasificar entre las clases spam y no spam los datos no etiquetados.

2.3 Máquina de vectores de apoyo

Para resolver el problema de detección de Web Spam, utilizamos la alta capacidad de clasificación de las máquinas de vectores de apoyo, las mismas que fueron desarrolladas por Vapnik [3] y están basadas

en la teoría de aprendizaje estadístico.

Considerando el caso más básico de clasificación, en el que hay que decidir entre dos clases diferenciables en el sentido de que se pueden separar mediante un hiperplano se lo conoce como *clasificación binaria con datos linealmente separables,* y consiste en:

Dados los vectores de entrenamiento $\boldsymbol{x_i} \in \boldsymbol{R^d}, \boldsymbol{i} = \mathbf{1}, \dots, \boldsymbol{l}$, etiquetados en dos clases en **Y** tal que $\boldsymbol{y_i} \in \{-\mathbf{1}, \mathbf{1}\}$**:**

$$\{x_i, y_i\};\ i = 1, \dots, l;\ y_i \in \{-1,1\};\ x_i \in R^d$$

Además un hiperplano, que divide a las dos clases en **Y {-1,1}**, descrito como:

$$w \cdot x + b = 0 \quad (1)$$

Donde w es un vector normal al plano, ver Figura. 2.1.

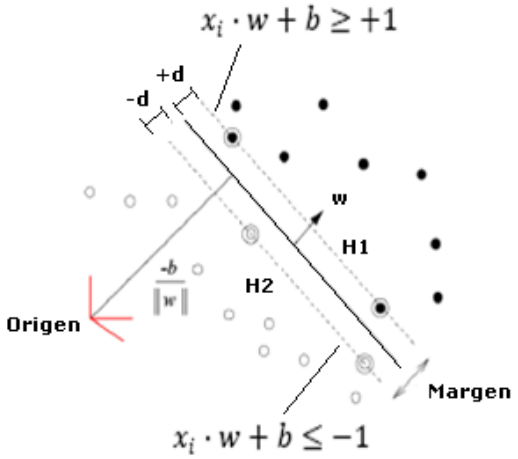


Figura 2.1 Muestra del hiperplano que separa las clases

Sean $\frac{-b}{\| w \|}$ la distancia que existe desde el plano al origen y **|| w ||** la norma euclidiana de **w**, siendo ésta el producto escalar $\sqrt{\boldsymbol{w} \times \boldsymbol{w}}$**.**

Sean **+d** y **–d** las distancias que existe entre el hiperplano al punto más cercano de ambas clases.

Satisfaciendo las siguientes condiciones:

$$x_i \cdot w + b \geq +1 \; para \; y_i = +1 \qquad (2)$$

$$x_i \cdot w + b \leq -1 \; para \; y_i = -1 \qquad (3)$$

Teniendo así una sola ecuación:

$$y_i(x_i \cdot w + b) - 1 \geq 0 \; \forall i \qquad (4)$$

Cabe señalar que H1 y H2 son hiperplanos paralelos y no contienen vectores entre ellos.

Dado que la suma de **+d** y **–d** es $\frac{1}{\| w \|^2}$ se pueden obtener los hiperplanos que maximicen el margen mediante la minimización de $\| w \|^2$ sujeto a las restricciones en (2 y 3).

Estos vectores que satisfacen las restricciones en (2 y 3), es decir que están en el borde de los hiperplanos, se conocen como *vectores de soporte* de la máquina.

Consideremos ahora el problema primal para la minimización:

$$\min_{w,b,\xi,p} \frac{1}{2} w^T w - vp + \frac{1}{l} \sum_{l}^{1} \xi_i$$

$$\text{Sujeto a} \quad y_i(w^T\phi(x_i + b) \geq p - \xi_i),$$

$$\xi_i \geq 0, i = 1, \dots, l, p \geq 0.$$

Y el dual:

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha$$

$$\text{Sujeto a} \quad 0 \leq \alpha^T \leq \frac{1}{l}, \quad i = 1, \dots, l,$$

$$e^T\alpha \geq v,$$

$$y^T\alpha = 0$$

$$\text{Donde} \quad Q_{ij} \equiv y_i y_j K(x_i, x_i)$$

Entonces la función de decisión es:

$$F(x) = sgn(\textstyle\sum_{i=1}^{l} y_i\alpha_i(k(x_i, x) + b)). \quad (5)$$

Los valores de α_i son los multiplicadores de LaGrange de la ecuación, $\boldsymbol{k(x_i, x)}$ es la función Kernel utilizada y ***b*** la variable independiente.

La ecuación 5, que está definida por la función *sgn(x),* sirve para la clasificación de futuros vectores.

Como argumentos positivos encontrados en base a la práctica se tiene que las máquinas de vectores de apoyo muestran un excelente desempeño tanto en escenarios donde existan vectores con pocas características [16] como en donde se necesiten significativamente menos cantidad de datos para el entrenamiento [17], asunto que es diferencial al momento de comparar este con otros métodos de clasificación.

2.4 Función Kernel

Visto anteriormente el problema de clasificación entre dos clases (+1,-1) se torna muy manejable cuando los datos son linealmente separables, es decir que pueden ser divididos por un hiperplano, pero un gran problema surge cuando aparecen escenarios en donde los datos no son linealmente separables. Para estos casos Vapnik presenta una solución al problema, mapeando la dimensión de los vectores de entrada a otra dimensión más alta, con el fin de tratar los vectores mapeados como un caso lineal separable. Dichos mapeos se los realizan a través de funciones denominadas *Kernels* [3].

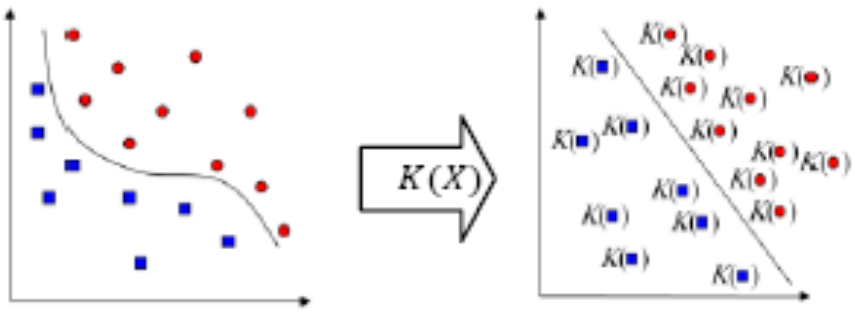


Figura 2.2. Ilustración del proceso de una función Kernel, llevando un espacio de una dimensión a otro de una dimensión más alta convirtiéndose en un caso linealmente separable

A continuación se muestran varias de las funciones Kernel comúnmente utilizadas:

- *Polinomial :* $k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j})^d$

- *Función de base radial:* $k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2)$*, γ > 0*
- *Función de base radial gaussiana:*

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}\right)$$

- *Tangente Hiperbólica:* $k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa\mathbf{x_i} \cdot \mathbf{x_j} + c)$*, para algún (no todos) valor κ > 0 and c < 0*

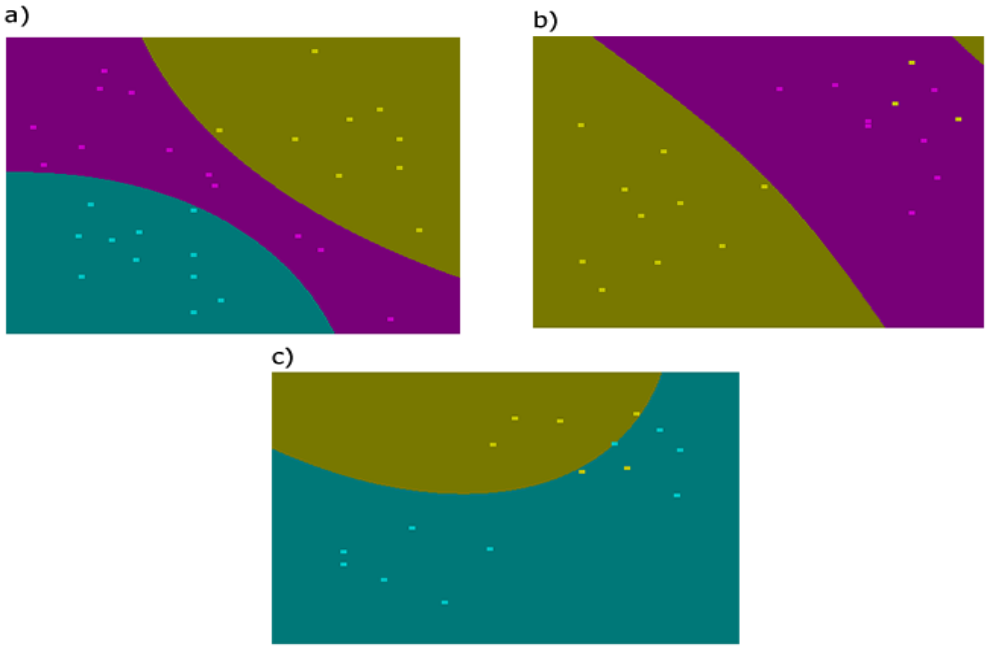


Figura 2.3. Ejemplos del desempeño de las máquinas de vectores de apoyo con mapeos a través de las funciones Kernel, generados con LibSVM [18]

Como muestran la Figura 2.3 la distribución de los vectores en este caso no tiene un comportamiento lineal, es muy dispersa, debido a esto el uso de Kernels que lleva a los vectores de entrada del espacio

a otro espacio de una mayor dimensión facilitando y permitiendo así la clasificación.

2.5 SVM en cascada

Las máquinas de vectores de apoyo a pesar de ser una excelente alternativa para los problemas de clasificación, han mostrado en la práctica que, a medida que se incrementan los datos de entrenamiento muestran un consumo de recursos demasiado elevado (memoria, procesamiento, etc.…), agravando así el problema de escalabilidad de la solución [20].

Tal falencia de rendimiento con gran cantidad de datos, exige un mecanismo de paralelización que acelere los tiempos de procesamiento o entrenamiento del algoritmo. Este mecanismo es encontrado en [20] y presenta un modelo de varios niveles denominado SVM en cascada. (Ver Figura 2.4).

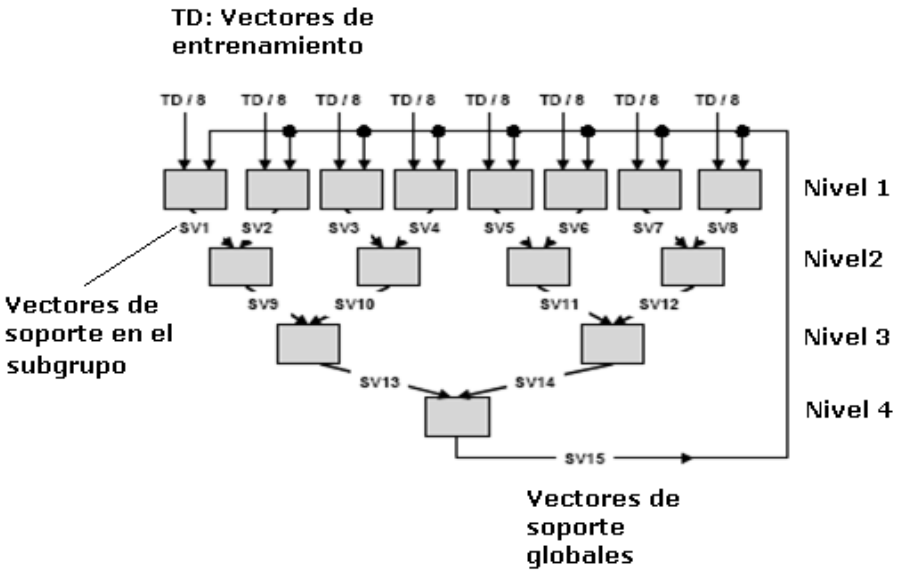


Figura 2.4. Modelo en cascada de SVM [20], los datos son divididos en subgrupos de datos donde cada grupo es evaluado de forma independiente

En el modelo en cascada mostrado en la Figura 2.4 inicialmente se dividen los vectores de entrenamiento en subgrupos de vectores con un tamaño definido, y se seleccionan los vectores de apoyo de cada subgrupo. Luego en la siguiente capa o nivel se combinan los vectores de soporte de varios subgrupos, pasando así a ser, una nueva población y se repite el proceso; hasta que al final se encuentran los vectores de soporte globales (De todo el dataset de vectores de entrenamiento).

2.6 Vectores de características

Los vectores de características son vectores los cuales están constituidos por datos numéricos, cada uno de estos datos

representan las características que se extraen de las páginas web, mediante expresiones regulares, que se encuentran en nuestro dataset más una etiqueta que indica si es spam o no. Un pilar fundamental previo al entrenamiento de la máquina, es la apropiada selección del tipo de características a evaluar en las páginas, optimizando de esta manera el proceso de clasificación.

En este trabajo se utilizaron características basadas en los contenidos basándonos en el trabajo de Ntoulas [2]. A continuación se presenta la lista de las características que se han considerado para llenar nuestro vector:

- *Número de palabras en la página* (f1): Para esta característica nosotros contamos la cantidad de palabras existentes en la página sin tomar en cuenta el contenido de los tags de las páginas.

- *Número de palabras en el título* (f2): Se tome en consideración las palabras que se encuentran incluidas en tag "*TITLE",* y se cuenta el número de palabras de este.

- *Promedio de palabras* (f3): Se encuentra un promedio del tamaño de las palabras dentro de la página.

- *Fracción del texto anclado* (f4).- Para esta característica se hace una relación entre la cantidad de palabras que hay dentro de texto

de anclas, palabras que se encuentra entre el tag “*a*”, dividido para la cantidad de palabras visibles dentro de la página.

- *Porcentaje de texto oculto* (f5).- Se hace una relación entre la cantidad de palabras de texto oculto y la cantidad total de palabras. Las palabras de texto oculto están localizadas dentro del contenido de *alt* que es una propiedad del tag *“img”* así como el texto dentro del tag *“input”* cuando este es del tipo *hidden*.

2.7 Modelo de programación MapReduce

Es un modelo de programación para procesamiento de grandes cantidades de datos implementado en ambientes con arquitecturas distribuidas, que permite computar los datos masivamente en forma paralela.

Como vemos en [30], la computación paralela con MapReduce se lo realiza tomando un conjunto de datos de entrada representados como duplas *clave/valor* y produciendo un conjunto de datos de salida también expresados como duplas clave/valor. El usuario del modelo MapReduce expresa la computación como la simple ejecución de dos funciones: *Map* y *Reduce*.

Un mapper es el que realiza un proceso Map, el cual está escrito por

un usuario. Es el que toma las duplas de entrada y produce una dupla clave/valor intermedia. La librería MapReduce agrupa todos los valores intermedios asociados con la misma clave intermedia y los pasa a la función Reducer.

La función Reducer, también escrita por el usuario, acepta una clave intermedia y un set de valores para la clave. Este junta los valores para formar un posible set de valores más pequeño. Típicamente cero y un valor de salida es producida por una invocación reducer. Los valores intermedios son proveídos al reducer del programa a través de un iterador. Esto permite manejar una lista de valores que son muy grandes para ser manejados por la memoria. La Figura 2.5 muestra de forma más clara como trabaja este modelo.

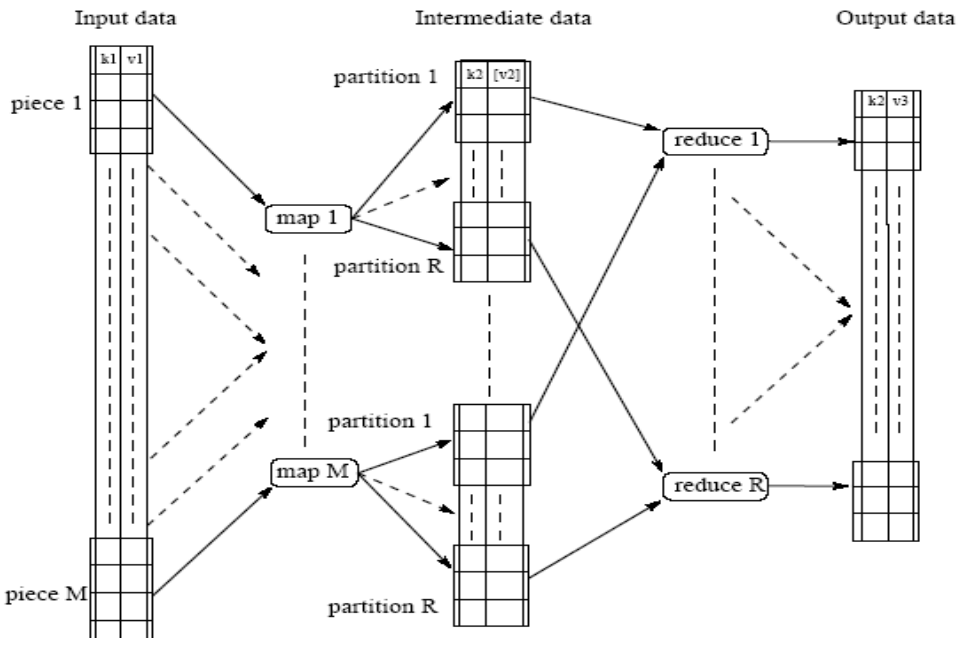


Figura 2.5.Modelo de trabajo MapReduce

2.8 Amazon Web Services S3

El Amazon Simple Storage Service (S3) [24] es un repositorio para Internet. Fue diseñada para hacer la computación web escalable más fácil para los desarrolladores.

Amazon S3 provee una interfaz de servicios web que puede ser usada para almacenar y recuperar cualquier cantidad de información, en cualquier momento, desde cualquier lugar de la web. Este da al desarrollador acceso a la misma alta escalabilidad, confiabilidad, rapidez, infraestructura de almacenamiento barata que Amazon usa para su propia red global de sitios web. El servicio permite maximizar los beneficios de escala y pasar esto a los desarrolladores. La Figura 2.6 demuestra cómo trabaja el S3 normalmente.

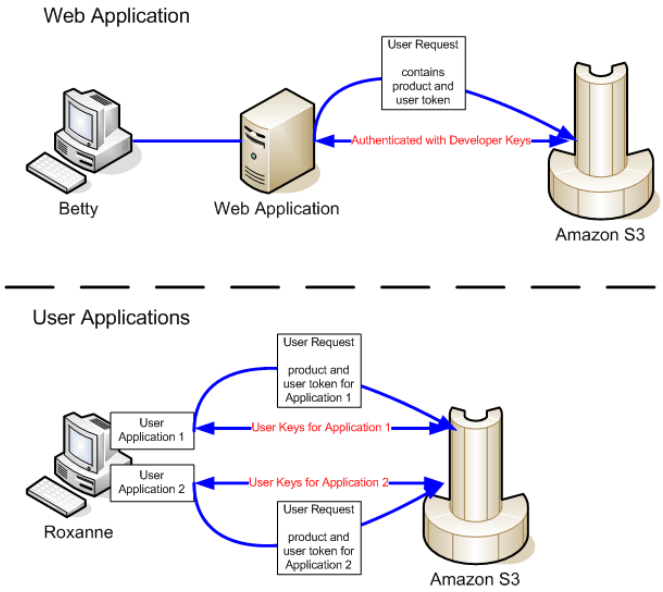


Figura 2.6. Representación de la interacción de usuarios y el Amazon S3 [25]

2.9 Amazon Web Services EC2

El Amazon Elastic Compute Cloud (EC2) [26] es un servicio web que provee capacidad computacional reajustable en la nube. Está diseñado para hacer la computación web escalable más fácil para los desarrolladores.

Amazon EC2 es una interfaz de servicios web simple que te permite la capacidad de obtención y configuración con un mínimo desgaste. Te provee un completo control de tus recursos computacionales y los permite correr en un ambiente computacional probado de Amazon. Amazon EC2 reduce el tiempo requerido para obtener y arrancar nuevos instancias de servidores al minuto, permitiendo así subir y bajar la capacidad escalable rápidamente. Amazon EC2 cambia la economía de la computación permitiéndote pagar solo por la capacidad que se usa. Amazon EC2 provee a los desarrolladores las herramientas para crear aplicaciones flexibles para errores y aisladas de posibles escenarios de error.

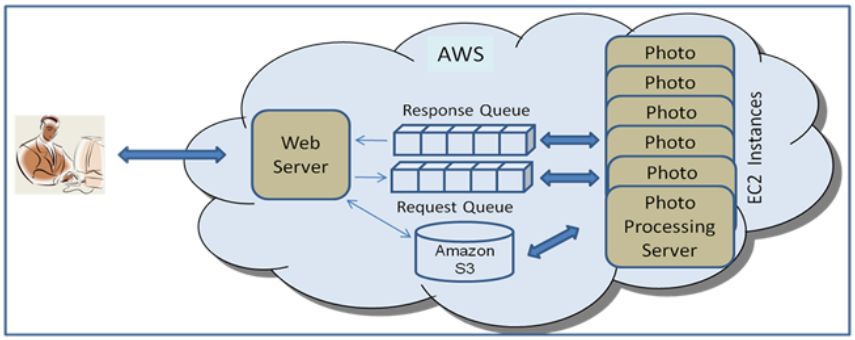


Figura 2.7. Ejemplo de EC2 y Amazon S3 trabajando juntos [28]

2.10 Modelo General

Para resolver la problemática de detección, se utilizaron procesos de tipo MapReduce, basando así el diseño en dos subprocesos principales: extracción, donde se excava la información en las páginas y entrenamiento, donde se generan los vectores de apoyo útiles para la clasificación.

En el subproceso de “extraer y cuantificar” se realizan los siguientes pasos:

1. Se inicia una tarea MapReduce para la extracción y se almacena el dataset en el sistema de archivos distribuido (HDFS, por sus siglas en inglés) [22]
2. Los mapers reciben las páginas almacenadas en el HDFS (Contenido HTML) como datos de entrada.
3. Los mapers excavan y cuantifican la información en las páginas
4. Los mapers arman los vectores de características y los distribuyen a los reducers.
5. Los reducers escriben los vectores en el HDFS

En el subproceso de “Entrenamiento” se realizan los siguientes pasos:

1. Se inicia una tarea MapReduce para el entrenamiento y se asigna la ubicación en el HDFS, de donde se sacan los datos de entrada para

la tarea (la ubicación debe ser el directorio de salida del subproceso de extracción).

2. Los mapers reciben los vectores armados desde el HDFS.
3. Los mapers distribuyen los vectores a los reducers, formando así grupos de vectores.
4. Los reducers reciben los grupos vectores (un grupo por reducer) y se eligen los vectores de apoyo para dichos grupos.

Cabe mencionar que el proceso de entrenamiento se realiza de forma cíclica y en cascada, siendo cada ciclo un nivel del subproceso. En cada nivel del subproceso de generan vectores de apoyo que sirven de entrada para el próximo nivel de manera que al final del subproceso se obtengan los vectores de apoyo globales, es decir, como si se hubiera aplicado el entrenamiento a todo el almacén de datos inicial. El gráfico de la Figura 2.8 nos da una visión más clara de cómo se hizo todo el proceso en este programa.

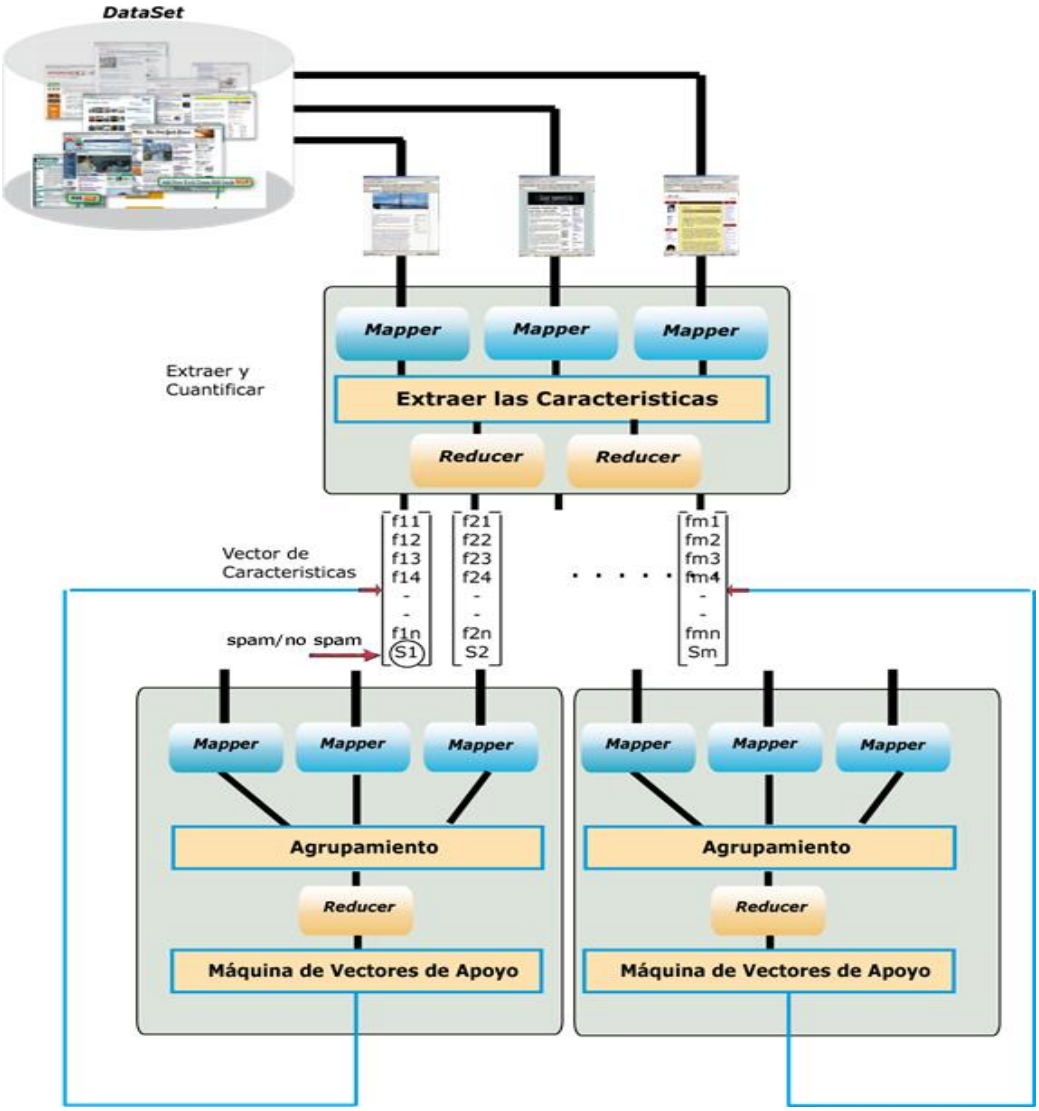


Figura 2.8. Modelo general de procesos

CAPÍTULO 3.

3. IMPLEMENTACIÓN

3.1 Dataset

Para el entrenamiento de la máquina de vectores de apoyo de nuestro proyecto utilizamos el dataset WEBSPAM-UK2006 [12] el cual ha sido llenado con páginas de dominio ".uk". Este almacén de datos fue colectado en Mayo del 2006 por el grupo de investigación del laboratorio de "Universita degli Studi di Milano".

El almacén de datos fue obtenido usando el software UbiCrowler [13] que utilizó el método de búsqueda primero a lo ancho. El proceso de llenado comenzó desde un gran conjunto de páginas semillas enlistadas en el Proyecto de directorio abierto (ODP, por sus siglas en inglés) [15]. Esta semilla contenía 190000 URL's en cerca de 150000 hosts. Como resultado 77,9 millones de páginas fueron guardadas. Para cada página el contenido y el link fue obtenido, las páginas obtenidas fueron guardadas en el formato WARC 0.9 [14].

Para el presente trabajo se utilizó un total de aproximadamente 120,000 páginas, que representa solo un volumen de los 8 obtenidos Cada volumen tiene un tamaño aproximado de 1.7 GB.

Cabe mencionar que como una de las ventajas del uso de máquinas de vectores de apoyo, no es necesario trabajar con un almacén de datos extremadamente grande [17]. Debido a su excelente capacidad de clasificación, el conjunto de datos que utilizamos proporciona un buen comportamiento en el proceso de clasificación.

Este dataset sirve a dos subprocesos principales que son:

- Extracción, donde a través de expresiones regulares, sobre cada página web etiquetada, se obtienen los datos que permitirán llenar los vectores de características. Los vectores en nuestro caso están compuestos por 5 características más la etiqueta, representados en un archivo de texto con la siguiente estructura.

  Dirección_página_web1 **\t** *f11 ; f12 ; f13 ; f14 ; f15 ; etiqueta1*

  Dirección_página_web2 **\t** *f21 ; f22 ; f23 ; f24 ; f25 ; etiqueta2*

  Dirección_página_web: Es la dirección de la página web de la cual estamos extrayendo las características.

  \t*:* Es la representación de un carácter de tabulación dentro de una línea.

  fn1 ; fn2 ; fn3 ; fn4 ; fn5: Representan las 5 características de una página n cualquiera.

Etiqueta: Es el valor previamente asignado de la pagina y tiene valores de ***-1*** spam, ***0*** no decidido y ***1*** no spam.

- Entrenamiento, este subproceso recibe como datos de entradas los archivos con los vectores anteriormente mencionados.

3.2 Librería y herramientas usadas

Para la ejecución de procesos de tipo ***MapReduce*** se utilizó la versión de ***Hadoop 1.8***, que implementa dicho paradigma y es un marco de trabajo para procesamiento en arquitecturas distribuidas Open Source.

Para trabajar con los procesos MapReduce implementados con Hadoop, utilizamos la distribución de ***Cloudera***, que es una distribución Linux específicamente para procesamiento distribuido.

Para utilizar la distribución Linux de Cloudera sobre Windows utilizamos el software ***VMware Player***, que permite la virtualización de sistemas operativos.

Para interactuar con las instancias de Amazon EC2 utilizamos el ***Ec2 Api tools 1.3.41620,*** que permite levantar clústeres, manipular el HDFS y ejecutar procesos MapReduce en los servidores EC2.

En el entrenamiento y clasificación de vectores se utilizó ***LibSVM 2.89***, [18] que es una implementación del algoritmo de máquina de vectores de apoyo en Java.

Para gestionar los datos almacenados en S3 se utilizó el pluggin de Fiefox ***"S3 Organizer"*.**

Para gestionar las instancias dentro de EC2 se utilizó el pluggin de Firefox ***"ElasticFox"*.**

CAPÍTULO 4.

4. EVALUACIÓN Y RESULTADOS

4.1 Análisis de Kernel

A continuación se muestra la relación que existe entre los vectores iniciales de entrenamiento del almacén de datos utilizado y los vectores de soporte de la máquina de aprendizaje, utilizando diferentes funciones kernel:

Vectores Iniciales	Vectores de Soporte		
	Kernel Gaussiano	Kernel Lineal	Kernel Polinomial
500	156	500	156
7600	3300	7600	3300
55000	20370	55000	20370
100000	37037	100000	37037

Tabla 4.1. Relación entre vectores iniciales y de soporte

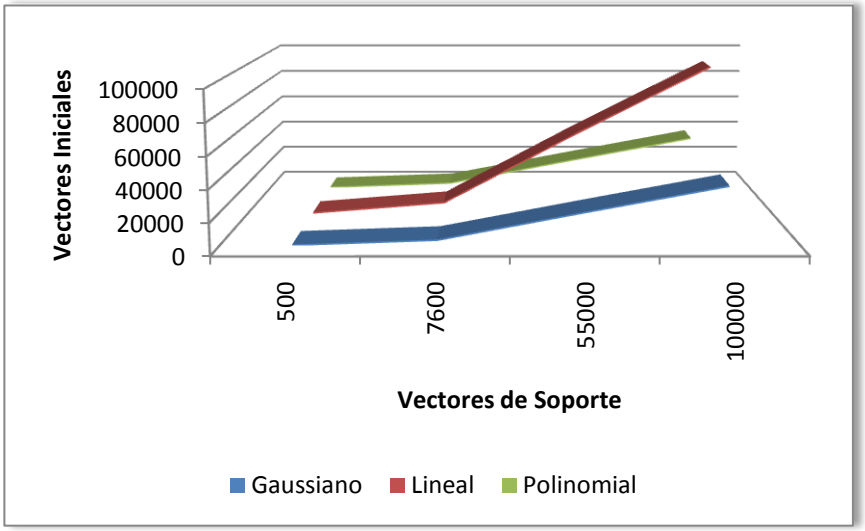


Figura 4.1. Relación entre vectores iniciales y de soporte

Como podemos observar en el gráfico tanto el Kernel Gaussiano como el Polinomial tienen un comportamiento muy parecido, estos nos dan como vectores de soporte casi la mitad del vector inicial de datos, lo que es muy diferente al Kernel Lineal, el cual tiene un comportamiento lineal por esta razón el numero de vectores de soporte es igual al número de vectores iniciales de entrenamiento, esto se debe básicamente a que los vectores está muy distanciados entre sí, lo que hace difícil que este tipo de kernel sea más efectivo.

4.2 Análisis de características

Tipo	Numero De Palabras
No Spam	444,79
Spam	583,3

Tabla 4.2. Número de palabras Spam y No Spam

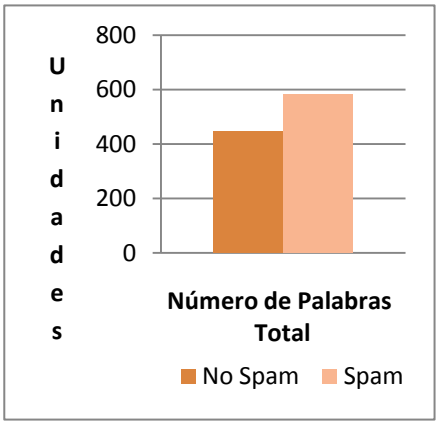
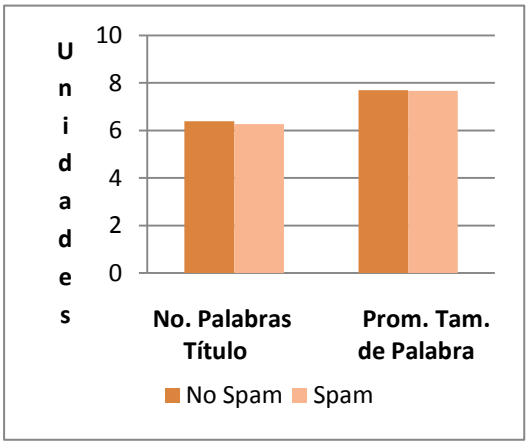


Figura 4.2. Número de palabras Spam y No Spam



	Características	
Tipo	Numero de Palabras Título	Promedio Tamaño de Palabra
No Spam	6,39	7,69
Spam	6,27	7,67

Tabla 4.3. Número de Palabras Título y Prom. de Tamaño de Palabra Spam y No Spam

Figura 4.3. Número de Palabras Título y Prom. de Tamaño de Palabra Spam y No Spam

	Características	
Tipo	Porcentaje Texto Ancla	Porcentaje Texto Visible
No Spam	34%	76%
Spam	37%	87%

Tabla 4.4. Porcentaje Texto Ancla y Texto Visible Spam y No Spam

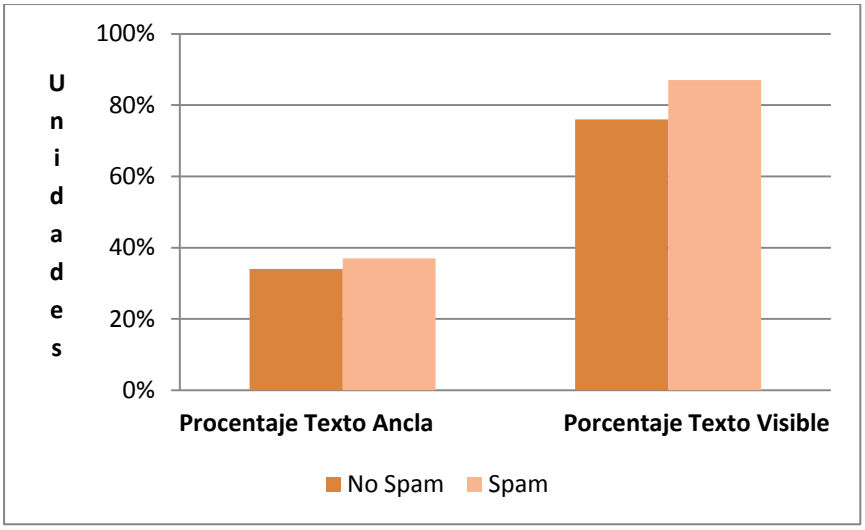


Figura 4.4. Porcentaje Texto Ancla y Texto Visible Spam y No Spam

Estos gráficos demuestran el promedio de las características de las clases spam y no spam, estas son las características que fueron establecidas anteriormente. Como vemos la diferencia entre la una clase y la otra no es tan grande para este dataset, lo que hace que la predicción sea una tarea más complicada, por ende la máquina de aprendizaje no nos da resultados efectivos. Para solucionar este problema debería utilizarse mayor cantidad de características y que nos permitan ver una diferencia mayor entre una y otra clase.

4.3 Matriz de confusión

		Clases Predecidas	
		1	-1
Clases Conocidas	1	89,70%	0
	-1	10,30%	0

Tabla 4.5. Matriz de Confusión

Total Analizado: 1000 vectores.
Tipo de Kernel: RBF.
Precision = 89,70/(89,70+10,00) = 0.90.
Recall = 89,70/(89,70+0) = 1.

Analizando la matriz de confusión los resultados muestran un buen desempeño en el detector teniendo en cuenta lo siguiente:

- El dataset de entrenamiento está compuesto en su mayoría por páginas etiquetadas como no spam.
- Por lo expuesto en el punto anterior, el detector puede predecir con certeza cuando una página no es web spam, más no lo contrario.

4.4 Mediciones en EC2

Nodos EC2	Extracción Tiempo (seg.)	Entrenamiento Tiempo (seg.)
3	83.127	302
8	50.54	250

Tabla 4.6. Mediciones con 3 y 8 nodos en EC2

Kernel: RBF (Radial Basis Function)

Número de vectores: 67,577

Número de vectores de apoyo: 20,338

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. Hadoop es una herramienta muy poderosa y de gran utilidad en la actualidad por ser una excelente alternativa para el procesamiento distribuido, que facilita el trabajo con grandes cantidades de información en las medianas y grandes empresas.
2. El uso de los servicios Web de Amazon permiten ahorrar costos de infraestructura, y es de gran ayuda para empresas que recién comienzan.
3. SVM es una muy capaz herramienta de clasificación, adaptable a cualquier problema. Para el presente trabajo mostró resultados muy aceptables.
4. Los problemas de rendimiento generados por el consumo de recursos de SVM como tal, pueden ser solucionados con técnicas de paralelismo como con el paradigma MapReduce, trabajado en este proyecto.
5. Las características utilizadas en nuestro proyecto han sido bastante estudiadas los últimos años, lo que permite a los spammers evitar ser detectados.

Recomendaciones

1. En nuestro proyecto utilizamos SVM en cascada, es posible implementar otro tipo de paralelismo como la solución de "Sub-problemas cuadráticos".
2. Extender la dimensión de los vectores de características, para asegurar una mejor clasificación. Tales características aumentadas que incrementan la dimensión de los vectores, podrían ser analizadas y

cuantificadas de las relaciones de una página con sus vecinos más cercanos.

3. Se pueden utilizar mecanismos de validación cruzada para el ajuste de parámetros en el modelo de la máquina de vectores de apoyo (SVM)

REFERENCIAS BIBLIGRAFICAS

1. C. Castillo, D. Donato, L. Becchetti, P. Boldi, M. Santini, and S. Vigna. A reference collection for web spam detection.Technical report, DELIS, September 2006.

2. A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In Proceedings of the World Wide Web conference, pages 83–92, Edinburgh, Scotland, May 2006.

3. V. Vapnik. The Nature of Statistical Learning Theory. Springer, N.Y., 1995. ISBN 0-387-94559-8.

4. Z. Gyongyi and H. Garcia-Molina. Web spam taxonomy. In AIRWeb, 2005.

5. B. Jansen and A. Spink. An Analysis of Web Documents Retrieved and Viewed. In International Conference on Internet Computing, June 2003.

6. Becchetti and Carlos Castillo and Debora Donato and Stefano Leonardi and Ricardo Baeza-Yates. Link-Based Characterization and Detection of Web Spam, In AIRWeb 2006

7. Dennis Fetterly and Mark Manasse and Marc Najork. Spam, Damn Spam, and Statistics: Using statistical analysis to locate spam web pages. In Proceedings of WebDB 2004

8. Carlos Castillo and Debora Donato and Vanessa Murdock and Fabrizio Silvestri. Know your neighbors: Web spam detection using the web topology. In Proceedings of SIGIR 2007.

9. L. Page, S. Brin, R. Motwani and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Stanford Digital Library Technologies Project, 1998.

10. H. Zhang, A. Goel, R. Govindan, K. Mason, and B. Van Roy. Making eigenvector-based reputation systems robust tocollusion. In Proceedings of the third Workshop on Web Graphs (WAW), volume 3243 of Lecture Notes in Computer Science, pages 92–104, Rome, Italy, October 2004. Springer.

11. Machine Learning, Thomas G. Dietterich

12. C. Castillo, D. Donato, L. Becchetti, P. Boldi, M. Santini, and S. Vigna. A reference collection for web spam detection. Technical report. DELIS, September 2006

13. P. Boldi, B. Codenotti, M. Santini, and S. Vigna. Ubicrawler: a scalable fully distributed web crawler. Software, Practice and Experience, 34(8):711–726, 2004

14. http://www.niso.org/

15. http://www.dmoz.org/

16. Thorsten Joachims , Fachbereich Informatik , Fachbereich Informatik, Fachbereich Informatik , Fachbereich Informatik, Lehrstuhl Viii. Text Categorization with Support Vector Machines: Learning with Many Relevant Features.

17. Zhenchun Lei, Yingchun Yang, Zhaohui Wu .Ensemble of Support Vector Machine for Text-Independent Speaker Recognition. IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.5A, May 2006

18. Chih-Chung Chang and Chih-Jen Lin. LIBSVM, a library for support vector machines. 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

19. Máquinas de soporte vectorial en el reconocimiento automático del habla

20. Hans Peter Graf and Eric Cosatto and Leon Bottou and Igor Durdanovic and Vladimir Vapnik. Parallel support vector machines: The cascade svm. In Advances in Neural Information Processing Systems, pages 521--528, 2005

21. Tom Mitchell, Machine Learning, 1997, Mc-Graw Hill

22. Doug Cutting, Hadoop the definitive Guide, published by O'Railly, 2009

23. Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI 04, 6th Symposium on Operating Systems Design and Implementation, Sponsored by USENIX, in cooperation with ACM SIGOPS, pages 137 – 150, 2004

24. http://aws.amazon.com/s3/

25. http://docs.amazonwebservices.com/AmazonS3/latest/images/devpay_installations.gif

26. http://aws.amazon.com/ec2/

27. http://www.yr-bcn.es/webspam/datasets/

28. http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1464

29. S. B. Kotsiantis, Supervised Machine Learning: A Review of Classification Techniques, Department of Computer Science and Technology University of Peloponnese, Greece

30. Ralf Lämmel, Google's MapReduce programming model, Published by Elsevier North-Holland, Inc, 2007

ANEXO 1

PROCESOS DEL PROGRAMA

Utilizar SVM (Support Vectors Machine) como mecanismo de aprendizaje de clasificación utilizando un modelo de cascada

Proceso para poner modificar obtener el dataset deseado

- Del dataset se toma solo las páginas que tienen etiqueta y se crea un nuevo dataset.
- El nuevo dataset de páginas HTML se graba en el S3, de este el 70% se utilizará para el entrenamiento y el 30% para predicción.

Proceso de extraer y cuantificar

- Asignar el dataset de entrenamiento a un proceso MapReduce.
- El proceso contara con su propio *HTMLRecordReader* el cual indica la manera en que se llenan los datos del *HTMLInputFormat*, este último es usado como datos de entrada para el Map; cada página es asignada a un *HTMLInputFormat*.
- Se realiza un proceso Map en donde se extraen las características de las páginas.
- El resultado es un conjunto de vectores de características (VCs), en el cual cada vector representa a una página.

Proceso de Entrenamiento

WHILE (true) {

- Asignar los archivos con VCs al proceso MapReduce.
- Setear parámetros de entrada para el proceso.
- Proceso Map:
    a. Lee los datos de entrada y crea los arreglos de características de tipo FeaturesArrayWritable.
    b. Distribuye los arreglos a los reducers, aplicando función de asignación de claves.
- Proceso Reducer:
    a. Recibe los arreglos de características y obtiene los vectores de apoyo de entre esos vectores.
- Llena directorio en HDFS con archivos de vectores de apoyo generados.

}