Detección de Correos Spam utilizando Técnicas de Inteligencia Artificial

Angeles Lliuya Paola Viviana
Escuela de Ingeniería de sistemas y computación
Universidad Peruana de Ciencias Aplicadas
Lima, Perú
u20191a919@upc.edu.pe

Chuco Michel Angel Gustavo
Escuela de Ingeniería de sistemas y computación
Universidad Peruana de Ciencias Aplicadas
Lima, Perú
u201613694@upc.edu.pe

Mosqueira Cesar
Escuela de Ingeniería de sistemas y computación
Universidad Peruana de Ciencias Aplicadas
Lima, Perú
u202910750@upc.edu.pe

Abstract— En el siguiente documento se tiene como meta implementar un administrador de correos que permita detectar correos spam, utilizando técnicas de la inteligencia arficial como redes neuronales y bayesianos ingénuos. Se mostrará el modelo implementado y los resultados obtenidos.

Keywords: redes neuronales, naive bayes, correos spam, inteligencia artificial.

I. INTRODUCCIÓN

En la actualidad, los e-mails son frecuentemente utilizados como medio de comunicación.

Lamentablemente, este medio ha sido utilizado, también por personas y empresas con malas intenciones, que usan este servicio para mandar promociones o, en algunos casos, correos con intenciones de robar información. Agregado a ello, no solamente realizaban el envío de un correo esporádico sino de manera masiva.

Es por ello, que se han desarrollado diversas formas de evitar este tipo de mensajes, por ejemplo: la creación de un clasificador de correos basura en base al contenido del mensaje, un servicio conectado a un modelo de redes neuronales, o utilizando filtraciones por clasificaciones de usuarios, que corrobora la información brindada de otros usuarios con respecto a los dominios, para clasificar el correo utilizando un modelo bayesiano. Se han planteado diversas soluciones al pasar del tiempo, cada uno con sus pros y contras. En este proyecto se abordará y realizará una solución con base del algoritmo Naive Bayes, con el cual podremos clasificar los diferentes tipos de mensajes para separar los mensajes importantes de los correos no deseados. Para ello se pretende realizar la implementación de la solución utilizando el lenguaje Python y herramientas como NLTK y el entorno Anaconda.

II. ESTADO DEL ARTE

Desde el siglo XVIII aproximadamente, el correo se ha vuelto un método eficaz por el cual se puede enviar información a uno o más destinatarios. En la actualidad, se siguen utilizando de manera digital, por esta razón han surgido nuevas dificultades en su uso. Uno de estos problemas es que existen mucho spam, es decir, correos no deseados enviados de manera indiscriminada con el fin de publicitar, dañar o molestar a los receptores.
El Deep Learning fue fundamental en los estudios de clasificación de mensajes y ha producido un gran

avance en la clasificación de estos. Por ejemplo, se llegó a crear una red neuronal basada en memoria de corto plazo y largo plazo para poder administrar el correo no deseado en las bandejas de entrada. Sin embargo, para hacer uso del Deep Learning se requieren diversos datos etiquetados.

Una solución a este problema, es el uso del método de aprendizaje activo, que se utiliza para disminuir el costo de etiquetado y aumentar la adaptabilidad del modelo. En nuestro proyecto, usaremos el nuevo modelo que es superior a las redes neuronales convolucionales y recurrentes en la clasificación de correos.

En efecto, este modelo cuenta con dos partes, una para el mensaje y otra enfocada al asunto. Previo al procesamiento de data, se asegura que en el la cantidad de letras no sea mayor a veinticinco. En este, se utiliza la técnica pooling para poder obtener las palabras claves y se pueda mapear en su contenido. Para la parte del contenido del mensaje se usa una R.N. (Red Neuronal) con una arquitectura LSTM (Long short-term memory). En síntesis, el mensaje de entrada cuenta con dos partes (asunto y cuerpo) y la salida es el resultado la clasificación del correo electrónico.

Asimismo, se han propuesto soluciones para eliminar todos los correos no deseados en un grupo grande de mensajes. Este filtro, puede reducir los problemas de este tipo, siempre y cuando actúen de manera predecible. También, se pueden ver todos los e-mails que fueron marcados como spam y volverlos a la bandeja de entrada en caso de que haya ocurrido un error. Primero, se extrae el grupo de e-mails que se tienen que analizar con los filtros ya definidos anteriormente.

Segundo, cada mensaje extraído de cada filtro se reporta en la parte de clasificación, aquí se utilizan softwares como el SpamAssasin entre otros. Finalmente, se supone que, gracias al margen de error registrado, el filtro aprenderá las características de cada emisor de spam, red de noticias y los usuarios maliciosos.

Otras investigaciones implementan una solución del filtro de correos spam con un enfoque colaborativo que aprende a filtrar spam mediante la generalización de correos etiquetados como spam/no spam aprendiendo desarrollando modelos locales por cada usuario (para limitar el efecto de los votos contrarios); los votos se comparten solo entre usuarios y correos electrónicos que son similares entre sí, definiendo similitudes usuario-usuario y email-e-mailing utilizando funciones resistentes al spam que son extremadamente difíciles de encasillar para los spammers.

Se realiza todo este cambio para hacer referencia de transponer mayoritariamente los problemas que causan los spammers cuando ajustan rutinariamente sus correos electrónicos para evitar los filtros de spam.

Un impacto marginal útil de esta metodología es que el número de parámetros que deben estimarse es muy enano, lo que ayuda a la hora de elegir algoritmos de adiestramiento en serie para lograr una buena explicación mientras se evita el sobre entrenamiento de datos.

Además, la tecnología de filtrado colaborativo personalizado ha sido completamente adoptado por plataformas de ecommerce, redes sociales y motores de batida, demostrando su efectividad para priorizar correos spam de los importantes, con el fin de arreglar estos problemas o implementar sistemas de spam más perfeccionados.

Los algoritmos de consejo de filtrado colaborativo son basados en programa o usuario, en primer lugar, en la red frecuente, el programa es el correo, esto se debe a que la cantidad de correos es muy gigante, a representación de los usuarios, que resulta ser relativamente prudente. Por lo tanto, es mejor amparar un filtrado colaborativo en la red de mailing. Cuando el usuario recibe un correo electrónico, el sistema le dará una división del transporte percibido basado en la evaluación de sus usuarios vecinos.

Siendo estos vecinos los modelos Bayesianos creados para cada usuario de cada correo. Si el universalismo de los usuarios vecinos piensa que un mailing es spam, es muy verosímil que la mensajería asimismo sea acreditada por tu norma como spam. también, esta dialéctica no se ocupa del contenido del e-mailing, por lo que no consiste en un quebranto de la privacidad de los usuarios.

Otro enfoque sugiere la implementación de un medicamento electrónico antispam basado en contenido. Utilizando el cálculo Ripper para encasillar los correos electrónicos y Teorema de Bayes para filtrar correos spam, demostrando en los resultados que el encuadre de filtrado basado en Bayes resulta solemne basado en definiciones clave en ganancia.

Otro intento utilizando una operación Boosting determinó que los resultados son superiores comparados con los algoritmos de Bayes, árboles de decisión. después se comparó los 4 algoritmos, Bayes, árboles de alternativa, redes neuronales, Boosting y se llegó a la relación que las redes neuronales son más efectivas a la hora de encasillar correos spam.

En la implementación de esta posibilidad se presenta un Learning vector quantization, una categoría de red neuronal que supera al basado en BP, debido a categorizamos los correos electrónicos no deseados en varias subclases de contenido para que las definiciones destacadas de cada subclase de email de repuesto sean más relacionados y más cercanos, así como las características de cada subclase de correos electrónicos de repuesto son más fáciles de identificar.

En cada caso mencionado, se hace énfasis a la importancia en los tipos de filtros, el estudio de estos es importante pues nos brinda la oportunidad de poder aprender sobre la interacción del usuario y su confianza con los sistemas autónomos y adaptativos. En efecto, los usuarios, por ejemplo, deben confiar en la efectividad de un filtro de spam, ya que corren el riesgo de perder la información que es relevante para ellos. El desarrollo de filtros entrenables plantean un torneo adicional para ganar la calma del usuario, visto que confían en la enmienda de errores de la medicina. El análisis de los filtros adaptativos se ha centrado principalmente en sistemas de consejo colaborativa, una leyenda ascendiente de la acción de filtrado debe ser apto para el usuario. Cómo el sistema aprende y puede ser diseñado debe quedar aguachento en un grado necesario, para eludir los problemas con la pareja terminada.

Sin embargo, la clasificación de spam no fue lo único que ha tenido desarrollo en nuestra era, pues también existen problemas como el phishing, que se puede definir como el robo de información mediante direcciones URL falsas. Estas son utilizadas frecuentemente pro hackers o usuarios con malas intenciones para poder robar información a otros usuarios. Por este y otros motivos se han implementado diversas soluciones basadas en filtros de spam para poder administrar el contenido textual del correo electrónico y analizar los URL's que tienen en ellos disminuyendo el tiempo de detección de phishing.

El autoencoder en un método de aprendizaje no supervisado, que aprende características de la entrada original. Una red neuronal que está compuesta de 3 partes: una capa de entrada una de salida un como mínimo, una capa oculta. El correo va a un extractor de contenido, y este obtiene toda la información así como el URL que se encuentra en el mensaje, para poder pasar por un PV-DM Paragraph Vector-Distributed Memory) para detectar posibles palabras claves que sean muy utilizadas en la mayoría de casos de phishing.

Asimismo, el autoencoder fue propuesto para evitar una nueva característica en el robo de datos o phishing, creando un punto de incertidumbre para finalmente pasar por el clasificador, la cual se encarga de diferenciar correos phishing según el nivel de cambio que exista en el Autoencoder y el DAE.

Existen diversas herramientas, las cuales se basan principalmente en conceptos como Bag of Word y Term frequency, Con el fin de lograr una investigación estilo métrico y semántico. Gracias a los anteriores mencionados, se puede obtener los datos y heurísticas necesarias para el cambio de un tópico listo de encasillar de manera exitosa crónicas de spam. Con el fin de ganar esto, sobre todo se hace uso del Machine Learning y sus diversas formas de entrenamiento en un arquetipo encauzado a ejecutar un conflicto. Entre las principales técnicas se encuentran la educación supervisada, no supervisado y semi supervisado; cada uno de ellos con sus principales ventajas y desventajas. En riesgo de un problema de segmentación el patrón limitante en el desarrollo de entrenamiento es la cantidad de datos, por lo que la técnica de adiestramiento debe evaluarse en afinidad a ello.

Nearest Neighbor Algorithms es una propuesta para Machine Learning, como K-d Tree, K Nearest Neighbor (KNN)y Weighted KNN y, en conjunto con algoritmos de extracción de datos y técnicas de Procesamiento de Lenguaje Natural. El proceso propuesto consta de 4 fases: recolección de datos, preprocesamiento, extracción de features y clasificación. La recolección de datos inicia escogiendo los datos ideales con los cuales se puede entrenar el modelo, en esta situación ejemplos de correos de spam y phishing,

Una vez se realice la recolección de datos el preprocesamiento se encargará de normalizar el texto y eliminar mayúsculas y cosas no relevantes. Consecuentemente, se obtendrán los principales Feature que se tienen que analizar mediante alguno de los algoritmos de extracción ya mencionados. Finalmente, se realiza su clasificación mediante un modelo NN. Esto finaliza de una manera increíble, pues la eficiencia del modelo puede ser diferente debido al algoritmo de extracción de los datos elegidos o las heurísticas usadas.

Diversos Algoritmos encargados en la clasificación de datos son usados en machine learning, pues estos se basan en la selección de grupos. En el caso de la clasificación de correo no deseado y hams los principales features son base a la estilometria, la red asociada y la estructura de correo. Los anteriores mencionados, en gran medida determinan la eficiencia de la clasificación cuando se utilizan algoritmos de aprendizaje supervisado. Además de ello cabe resaltar la fase de preprocesamiento de datos, ya que es en esta en donde se extraen los

features que se usarán para el proceso de clasificación.
Ejemplo de tipos de features son: basado en contenido, léxico, usuario, sentimiento y semántico. Una de las técnicas más comunes y efectivas para la clasificación de datos, es el algoritmo Naive Bayes, el cual se basa en el Teorema de Bayes. Este método de clasificación estadístico permite obtener la probabilidad de que un determinado dato pertenezca a una categoría u otra.

En el caso de la clasificación de spam, este proceso se realiza calculando la probabilidad de que un email sea spam o no, en base a features procedentes de las palabras que contiene. Primero se normalizan y extraen las palabras claves del email, de forma que solo se analicen las palabras clave del mismo.

Una vez realizada la extracción de datos, se calcula las probabilidades con respecto a las palabras del conjunto de emails de ser spam o no. Finalmente se utilizan los datos entrenados anteriormente para clasificar los mensajes. Naive Bayes provee una gran efectividad, además de ser uno de los algoritmos más simples y livianos.

III. MODELO DE ARQUITECTURA

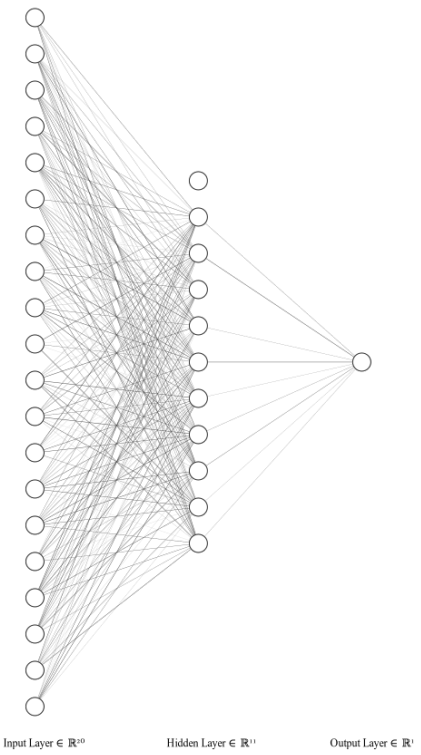
Con respecto al diseño de la aplicación en sí, es una aplicación web que estará desplegada en la plataforma Heroku, la cual está conformada por 2 procesos.

Red Neuronal

Con el objetivo de clasificar los mensajes como spam o ham, se hizo uso de una red neuronal compuesta de 3 capas, en donde el resultado es un único valor (1 spam - 0 ham). Esta red hace uso de los valores asociados a cada peso así como de la función sigmoid para poder clasificar cada tipo de mensaje, luego de un proceso de entrenamiento en base a datos previamente clasificados. El modelo fue entrenado en base a un dataset compuesto por 5575 mensaje, tanto de spam como ham, del cual solo el 80% fue usado como datos de entrenamiento.

La red neuronal utilizada y mencionada con anterioridad, está compuesta por 3 capas (input, hidden y output). Cada capa posee un tamaño de N, (N+1)/2 y 1 respectivamente, donde N son las cantidad de palabras únicas en el corpus.

Inicialmente todos los pesos y bias fueron asignados aleatoriamente en un intervalo de -1 a 1, además el ratio de aprendizaje fue por defecto 0,1. La función de activación utilizada a lo largo de la red fue la función sigmoid, en su variación logarítmica, con la que se buscó clasificar los inputs en 2 clases.



N: Cantidad de palabras únicas en el corpus
Input = N
Hidden = (N/2) +1
Output = 1

Dentro del proceso de entrenamiento, se extrajeron los features màs frecuentes en todo el dataset, en este caso las 760 palabras únicas más usadas. Estas palabras sirven como modelo de input para cada mensaje que queramos clasificar. Una vez calculados todos los inputs del set de entrenamiento, se entrenó, de forma supervisada, la red durante un número determinado de épocas.

A lo largo del proceso de entrenamiento, el modelo redujo su valor de error mediante sus gradientes y logró aproximarse al valor de 0. El output de la red correspondió a un valor entre 0 y 1, el cual indico si el mensaje fue ham o spam respectivamente.

Modelo Naive Bayes

El modelo toma como datos de entrenamiento un dataset que contiene 2 columnas: el tipo de clase al

que pertenece el email, y contenido del email. Se crea un para guardar todos los mensajes, además de 2 arreglos adicionales, donde se guardan los mensajes clasificados como “ham” y “spam” respectivamente. Además se calculan el valor porcentual de prioridad de cada clase con respecto al total de elementos del dataset mediante la fórmula:

$$\hat{P}(C) = \frac{N_c}{N}$$

Nc : cantidad de registros pertenecientes a la clase c

N: cantidad de registros en el dataset

Y finalmente, se guardan todas las palabras encontradas en el dataset en otro arreglo.

Para predecir la clase a la que pertenece un mensaje, el mensaje entrante se descompone en palabras, se crean 2 diccionarios con las palabras encontradas en el mensaje entrante, uno para cada tipo de clase, a continuación se realiza una sumatoria de repeticiones de palabra, colocando el número de repeticiones su llave correspondiente.

A continuación se aplica a cada valor de las palabras en el diccionario la fórmula de Naive Bayes que se expresa como:

$$\hat{P}(w|c) = \frac{count(w,c) + 1}{count(c) + |V|}$$

Donde:

- count(w,c): cantidad de repeticiones de la palabra en el arreglo de la clase c
- count(c): cantidad de palabras en el arreglo de la clase c
- |V|: cantidad de palabras únicas en el dataset.

Una vez calculado los pesos individuales, se halla el peso de cada clase con respecto al mensaje, multiplicando todos los valores de cada diccionario, adicionando el valor porcentual de prioridad anteriormente calculado. De estos pesos, se entiende que el peso superior es la clase a la que pertenece el mensaje de entrada.

Por último es necesario mencionar, que el desarrollo de esta aplicación garantiza que este servicio será utilizado de acuerdo con normas éticas, y en general, se asegura un correcto servicio. Dichas normas en mención aplicada a nuestro proyecto se detalla en el artículo científico realizado por Villalba en el 2020 (pp. 11-12).

IV. DISEÑO DE LA APLICACIÓN

El diseño de la solución consta de dos aplicaciones web realizadas en python con el framework Flask, de la cuál como evidencia se encuentra una desplegada en la plataforma de Heroku. Como ya se mencionó, se utilizó Flask como framework para el back-end de la aplicación, por otro lado, el apartado del front-end fue realizado utilizando jinja2 con Material Design for Bootstrap (MDB) como motor de plantilla y framework de CSS respectivamente.

El desarrollo de los algoritmos fue realizado usando las siguientes las herramientas:

- Jupyter Notebook: Entorno de ejecución de Python
- Numpy: Manejo y cálculos de los arreglos
- Pandas: Lectura del dataset
- Regex: Manejo de strings
- Math: Operaciones matemáticas de primer orden (pow, sqrt).
- NLTK: Procesamiento de lenguaje natural. Ejemplos: tokenización de oraciones,

generalización de verbos y eliminación de “Stop Words”

- Matplotlib: Visualización de datos y métricas a través de gráficos.

V. VALIDACIÓN DE RESULTADOS Y DISCUSIÓN

Para el caso de la validación de resultados, se utilizó el mismo dataset en las aplicaciones de red neuronal y la red bayesiana:

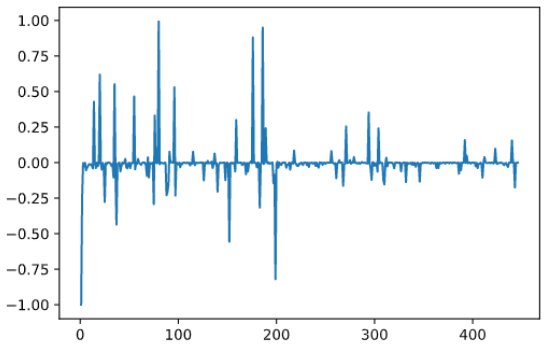
- *Tamaño del dataset: 5575 mensajes*
- *Spams: 748 mensajes*
- *Hams: 4238 mensajes*

Red Neuronal

Para conseguir la validación de mensajes de spam en base a la red neuronal, fue necesaria la realizacion en base al input de un arreglo de palabras que tienen origen en el procesamiento y tokenización de un mensaje. Dando como resultado un número entre el 0 y 1, cuyo valor denotaba si era ham o spam, siendo 0 para ham y 1 para spam.

Métricas de la Red Neuronal

Errores con respectos al tiempo de entrenamiento (20 epocas)



En base al dataset de pruebas (20% del dataset original), se obtuvieron las siguientes métricas.

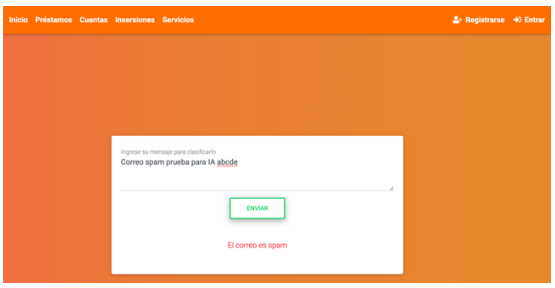
- *Accuracy: 0.9829443447037702*
- *Precision: 0.9774436090225563*
- *Recall: 0.8904109589041096*

Métricas de Naive Bayes

En base al dataset de pruebas (20% del dataset original), se obtuvieron las siguientes métricas.

- *Accuracy : 0.9238754325259516*
- *Precision : 0.8714285714285714*
- *Recall : 0.968253968253968*

Interfaz de la aplicación desarrollada:



CONCLUSIONES

Respetando las métricas, la red neuronal que fue adiestrada para este ejercicio, consiguió un puntaje más alto en los apartados de accuraccy y precisión, a contrates con el puntaje de recall, cuyo puntaje fue menor, esto basado en el algoritmo de Naive Bayes.

Esto se interpreta como, en un conjunto, la red neuronal ha clasificado de manera correcta una cantidad mayor de mensajes de spam en relación al total de mensajes de pruebas. También, este consiguió mayores resultados en relación a cuántos correos de spam clasificó de manera correcta cuando predijo que eran spam. Por consecuente, el algoritmo de Naives Bayes logro un valor mayor el número de veces que predijo correctamente cuando un mensaje fue spam. A futuro se estima que todos los servicios de correo incorporen una detección automática y mejorada de spam que evite que la vulnerabilidad de la informacion de los usuarios.

Aplicación desplegada: tf-izispam.herokuapp.com
Repositorio proyecto: github.com/vivieall/TF-IA

REFERENCIAS

Chen, Z., Tao, R., Zhimin, W. & Xiao, L. (2019). Active learning for spam classification. Recuperado de https://doi.org/10.1145/3377713.3377789

Chuan, Z., Xiangliang, L., Mengshu, H. & Zhou, X. (2005). A LVQ-based neural network anti-spam email approach. Recuperado de https://doi.org/10.1145/1044552.1044555

Crawford, M., Khoshgoftaar, T., Prusa, J.D. et al. (2015). Survey of review spam detection using machine learning techniques. Journal of Big Data 2, 23 (2015). Recuperado de https://doi.org/10.1186/s40537-015-0029-9

Dasgupta, A. & Gurevich, K. (2011). Enhanced email spam filtering through combining similarity graph. Recuperado de https://doi.org/10.1145/1935826.1935929

Douzi, S., Meryem, A. & Bouabid, E. (2017). Advanced phishing filter using autoencoder and denoising autoencoder. Recuperado de https://doi.org/10.1145/3175684.3175690

Gordon V. & Cormack, R. (2007). Online Supervised Spam filter evaluation. Recuperado de https://doi.org/10.1145/1247715.1247717

Henriette, S., Cramer, V. & Maarten W. (2009). Awareness, training and trust in interaction with adaptive spam filters. Recuperado de https://doi.org/10.1145/1518701.1518839

Hnini, G., Riffi, J., Mahraz, M., Yahyaouy, A., & Tairi, H. (2021). Spam filtering system based on nearest neighbor algorithms. Recuperado de https://doi.org/10.1007/978-3-030-53970-2_4

Pinandito, A., Perdana, R., Saputra, M. & Az-Zahra, H. (2017). Spam detection framework for android twitter application using naive bayes and K-nearest neighbor classifiers. Paper presented at the ACM International Conference Proceeding Series, 77-82. Recuperado de https://doi.org/10.1145/3056662.3056704

Richter, A. & Najeda, H. (2015). Survery of review spam detection using machine learning techniques. Journal of Physics: Conference Series. Recuperado de https://doi.org/10.1088/1742-6596/1575/1/012054

Trivedi, S. K. (2016). A study of machine learning classifiers for spam detection. Paper presented at the 2016 4th International Symposium on Computational and Business Intelligence, ISCBI 2016, 176-180. Recuperado de https://doi.org/10.1109/ISCBI.2016.7743279

Villalva, J. (2020). Algor-ética: La ética en la Inteligencia Artificial. Recuperado de https://www.investigacionyciencia.es/revistas/investigacion-y-ciencia/el-multiverso-cuntico-711/tica-en-la-inteligencia-artificial-15492

Wei, Q. (2018). Understanding of the naive bayes classifier in spam filtering. Paper presented at the AIP Conference Proceedings. Recuperado de https://doi.org/10.1063/1.5038979

Yang, T., Qian, K., Lo, D., Al Nasr, K., & Qian, Y. (2016). Spam filtering using association rules and naïve bayes classifier. Proceedings of 2015 IEEE International Conference on Progress in Informatics and Computing. pp 638-642. Recuperado de https://doi.org/10.1109/PIC.2015.7489926

Yongchan, W., Yuyan, C. & Lifeng, H. (2018). A study of neighbor users selection in email networks for spam filtering. Recuperado de https://doi.org/10.1145/3290480.3290500

Zamir, A., Khan, H., Mehmood, W., Iqbal, T., & Akram, A. (2020). A feature-centric spam email detection model using diverse supervised machine learning algorithms. Electronic Library, pp 633-657. Recuperado de https://doi.org/10.1108/EL-07-2019-0181