# Técnicas de detección de spams

Pedro Fernández Manteca
Cédric Prieels

Marzo 2020

## Introducción

En la sociedad actual, las personas tienden a estar cada vez más en contacto y por lo tanto, a enviar cada vez más correos electrónicos; una manera fácil y cómoda de contactar una o varias personas a la vez. En el año 2020, se estima que más de 4000 millones de personas usan este método de comunicación [1] y, al día de hoy, son más de 306.000 millones los correos electrónicos que se envían cada año en el mundo.
Con el objetivo de sacar rentabilidad a estos datos de comunicación, numerosas compañías gastaron en total más de 350 millones de dólares en 2019 en anuncios enviados por email, que se suelen considerar como correos electrónicos no deseados o spams por parte de la mayoría de las personas. De esta manera, se estima que el 50% de todos los correos electrónicos enviados son spams (Bill Gates recibe él sólo 4 millones de spams cada día), y es por lo tanto imprescindible desarrollar métodos que permiten filtrar los correos electrónicos recibidos para aumentar la productividad de los trabajadores o la propia comodidad del usuario. Además, los spams no solamente se pueden encontrar en los correos electrónicos, ya que muchas empresas que venden productos en línea dependen de las valoraciones y críticas proporcionadas por los usuarios para vender sus productos, y resulta por tanto muy tentador para dichas firmas crear críticas falsas sobre los productos con el fin de convencer a los compradores de que consuman el mismo.

Dado la cantidad de dinero que estos mensajes de spam mueven diariamente es fácil darse cuenta de lo imprescindible que es el desarrollo de métodos de detección de spams, que permiten filtrar estos correos electrónicos no deseados. Estos métodos deben ser lo más automáticos posible, dado el volumen de correos electrónicos y de críticas dejadas cada día en plataformas como Amazon o Booking. Muchos de estos métodos se nutren por lo tanto de los últimos desarrollos en dos campos principales, ambos muy activos hoy en día en I+D+i: la semántica y la inteligencia artificial.
El uso de métodos de semántica es imprescindible, ya que permiten “traducir” un texto escrito por un usuario en cualquier idioma a un lenguaje que un ordenador pueda entender: el Natural language processing (NLP).
Una vez traducidos, se usan ejemplos de críticas o correos electrónicos ya clasificados por un humano para entrenar algoritmos predictivos para que sean capaces de distinguir y filtrar un spam de un correo electrónico o crítica verdaderos.

## Natural language processing

El primer paso para la detección de spam, o de manera más general de cualquier análisis de text mining, consiste en la transformación de los textos a un formato manejable desde el punto de vista computacional: es lo que se denomina el Natural Language Processing (NLP). Este etapa es imprescindible en cualquier análisis de text mining, ya sea para traducir un texto, indexar un documento, recuperar información de un documento, o filtrar spams. Este procesado es por lo tanto

muy útil en muchos campos, pero también es bastante complejo, ya que el lenguaje humano es por definición complicado y tiene muchas sutilezas (como el sarcasmo), muy difíciles de entender por una máquina y por métodos de machine learning. Además, el texto que se analiza suele tener formatos muy distintos e incluso estar escrito en diferentes idiomas que no comparten siempre las mismas reglas semánticas, lo que complica todavía más el trabajo del processing NLP.

El NLP suele poder dividirse en dos categorías distintas: una parte relacionada con la sintaxis del documento, y otra relacionada con un análisis de tipo semántico.
El análisis sintáctico es bastante sencillo de implementar en un algoritmo ya que responde a reglas predefinidas (por ejemplo, una frase suele empezar por una mayúscula y acabar por un punto).
Por otra parte, las técnicas de análisis semántico más contemporáneas suelen precisar de la implementación de algoritmos de machine learning más complejos como máquinas vector soporte o redes neuronales.

# Análisis sintáctico

Como se ha descrito en la sección anterior, tratar de enseñar a una máquina a “entender” no solamente las palabras escritas sino también el sentido de cada frase y el sentido global del documento analizado, no es ni mucho menos una tarea sencilla y requiere el uso de diferentes etapas, la primera de ellas se corresponde con la preparación del dataset que se quiere analizar.

## Preparación del dataset

Antes de empezar con el análisis de las palabras de un documento, un paso previo de limpieza del documento suele ser necesario. Este proceso, conocido como *text normalization*, generalmente consta de las siguientes fases:

- En primer lugar cabe destacar que un algoritmo de normalización de texto estándar tiende a normalizar el uso de mayúsculas y minúsculas del documento. Sin embargo, para la detección de spams, este etapa no suele ser necesaria, ya que típicamente una de las características más significativas de un electrónico electrónico no deseado es precisamente el uso de letras en mayúscula para atraer la atención del usuario que lo lee.
- El documento suele ser limpiado también eliminando información inútil como los signos de puntuación (aunque en el filtrado de spams esta etapa tampoco se suele aplicar, ya que dichos spams suelen contener numerosos caracteres de este tipo). Por otra parte, algunos signos de puntuación pueden también tener especial importancia, como el apóstrofo inglés que implica que las palabras it’s y its serían iguales, por lo que esta etapa debe ser tratada con especial cuidado.
- Una vez el documento limpio, se pueden quitar las palabras no relevantes como los *stop words*, pequeñas palabras como (*la, el, de, así, a)* o (*the*) en inglés, que no suelen aportar información semántica.
- El *stemming - lemmatization* es probablemente la etapa más compleja pero también muy importante en la preparación del documento, y consiste en adjuntar las palabras que tengan la misma base. Por ejemplo, todos los verbos conjugados suelen ser agrupados, ya que el sentido detrás de los mismos se puede considerar equivalente. De la misma manera, los sinónimos se suelen agrupar también mediante el uso de diccionarios de palabras.

  Cabe destacar que mientras el proceso de stemming implica la simple truncación de palabras para intentar encontrar prefijos comunes, el proceso de lemmatization es más complejo y se basa en un análisis morfológico de las palabras para encontrar la forma más básica de las mismas, denominada lemma.

El algoritmo de stemming más famoso fue desarrollado por Porter en al año 1980 [2], y permite suprimir el sufijo de palabras en inglés suponiendo que no tengamos acceso a un diccionario (por ejemplo, queremos ser capaces de agrupar las palabras CONNECT, CONNECTED, CONNECTING, CONNECTION y CONNECTIONS, ya que suelen aportar la misma información semántica). Este proceso, representado en la Figura 1, se basa en 5 etapas secuenciales y, a pesar de su complejidad, sigue siendo utilizado al día de hoy con una versión un ligeramente mejorada.
Básicamente, lo que hace este algoritmo es en primer lugar poner todas las palabras a su forma singular y guardar solamente la base de los verbos (quitando las terminaciones -(e)d y -ing de los verbos en inglés). Posteriormente, el algoritmo intenta encontrar el nombre relacionado con los adjetivos encontrados en el documento (y que pueden tener formas muy distintas, dependiendo del nombre al que se refiere), teniendo en cuenta las consonantes dobles que suelen aparecer en el inglés. De esta manera, por ejemplo, la palabra GENERALIZATIONS quedaría reducida a su *stem* GENER. Algoritmos similares existen hoy en día para cualquier idioma, como en el caso del español [3].

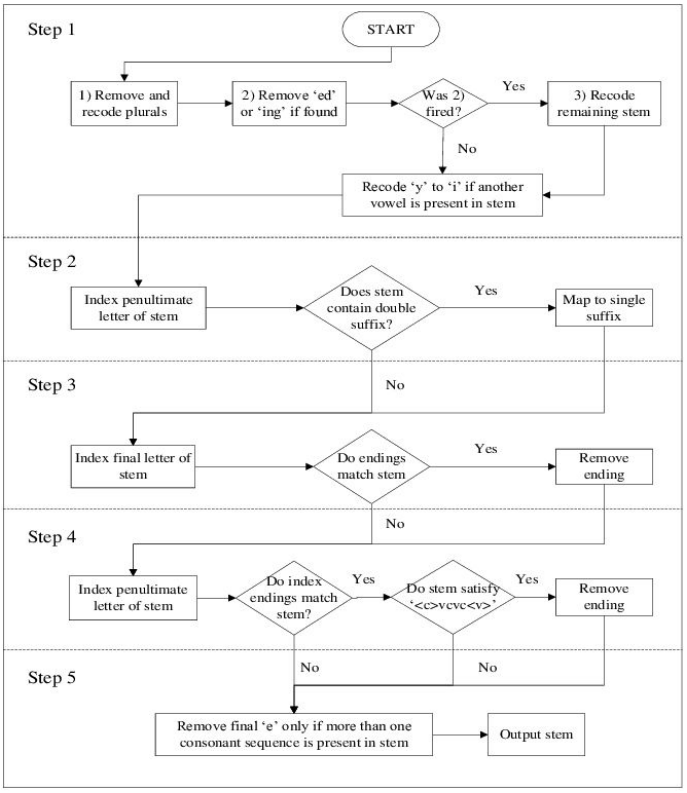

Figura 1: Las 5 etapas del algoritmo de stemming de Porter (1980, [2])

- Por último, se suele **separar la unidad de texto a analizar** (palabras por separado o frases) del cuerpo de texto. Esto permite simplificar el análisis anterior, usando bloques de datos más pequeños para los cuales es más fácil extraer información.

Todas estas etapas son en general fácilmente implementables en la mayoría de los lenguajes de programación, como Python y su Natural Language Toolkit [4], que permite limpiar el texto mediante el uso de distintos métodos.

Una correcta preparación del dataset nos permitirá extraer información semántica del documento de una forma más sencilla. Para llevar a cabo esta tarea es frecuente el uso de métodos de vectorización de palabras y métodos avanzados de machine learning que se describirán a continuación.

## Tokenización y vectorización

Un ordenador no suele ser capaz de entender directamente el sentido general transmitido por un texto que se le proporciona. Para poder extraer información de un dataset, los documentos tienen que someterse primero un proceso de tratamiento semántico que permita transformar el texto a representaciones numéricas, usando modelos matemáticos relativamente complejos para aprender las dependencias entre cada palabra, frase y documento.

La etapa siguiente a la preparación del dataset consiste por lo tanto en la *tokenizar* el documento (típicamente, separar cada palabra del documento en entidades individuales) y en la creación de matrices de incidencia, proceso también denominado como vectorización, que nos permita pasar de un texto a un espacio de fases vectorial donde se puedan usar métodos y modelos matemáticos (como la adición o la medida de la distancia y de la similitud entre palabras), para así extraer el sentido de un conjunto de palabras concreto. Por ejemplo, en este espacio de fases, la distancia entre dos palabras nos dará una idea de la similitud semántica que existe entre ellas.

Este proceso permite en primera instancia extraer información de las palabras mismas, pero además hace posible analizar el texto a una escala más grande, analizando frases completas, párrafos, e incluso documentos en su totalidad (considerados en este caso como un vector de vectores de palabras del cuál un algoritmo concreto puede extraer información semántica a gran escala), de manera que se puedan comparar documentos entre sí.

## Análisis de los vectores de palabras

Una vez el documento vectorizado, una primera información que se puede extraer es la frecuencia de aparición de cada palabra en el documento, basándose en la hipótesis según la cual palabras usadas en los mismos contextos tienden a tener el mismo sentido.

El método de vectorización tf-idf (term frequency - inverse document frequency) es uno de los algoritmos más frecuentemente utilizados para tener en cuenta la importancia de cada palabra en un documento debido a su sencillez y eficacia.
Este método consiste en calcular un estadístico para cada término en un documento de la manera siguiente:

$$tf - idf_{t,\,d} = (1 + log_{10}(tf_{t,d})) \,.\, log_{10}(N/tf_{t,d})$$

Donde *N* hace referencia al número total de documentos, $df_t$ al número de documentos *d* donde aparece la palabra *t,* y $tf_{t,d}$ al el número de veces que la palabra *t* aparece en el documento *d*.

Esta magnitud nos permite definir el espacio de fases de las palabras que necesitamos para al análisis del documento mediante el uso de distintos algoritmos predictivos que se describirán a continuación.

El cálculo de este estadístico es también es bastante sencillo en Python, ya que está incluido en paquetes tan famosos y comunes como Scikit learn [5] (método TfidfVectorizer).

# Análisis semántico

Los métodos de NLP y los primeros métodos de análisis semántico para sacar el sentido global de un documento fueron introducidos en los años 50. En los siguientes años se utilizaban conjuntos de reglas escritas a mano y pre-definidas. Dado el enorme volumen de datos del que disponemos hoy en día, las técnicas de filtrado de spam han derivado a métodos de machine learning capaces de extraer información semántica de grandes datasets de entrenamiento.
Estos métodos son mucho más cómodos de usar, ya que son capaces de analizar grandes conjuntos de datos en altas dimensiones (ideales en este caso, ya que el espacio de fases de las palabras tiende a ser de alta dimensionalidad) y que no necesitamos escribir de manera previa reglas que permitan filtrar spams como en el pasado. Diferentes métodos de Machine Learning se pueden usar en la detección de spams.

## Clasificadores Bayesianos:

Uno de los métodos que históricamente se han utilizado para el filtrado Spam / no-Spam son los clasificadores Bayesianos [6]. En esta sección se describirán los fundamentos de este procedimiento.

Cabe destacar que en la actualidad casi todos los filtros antispam utilizan clasificadores ingenuos de Bayes. La principales ventajas que ofrecen respecto a otros algoritmos es una precisión en la predicción altamente efectiva, rápida capacidad de entrenamiento, y alta escalabilidad (funcionan bien con datos de alta dimensión). De esta manera, como muestra el famoso trabajo de Paul Graham titulado "Un plan para el spam" [7], los clasificadores bayesianos son una buena elección para este problema de clasificación de spam, proporcionando una alta precisión en la clasificación (del orden del 99.9% para la mayoría de datasets) y una baja tasa de falsos positivos.

En cuanto a su funcionamiento, el clasificador Bayesiano asigna una probabilidad de que un nuevo mensaje pertenezca a una clase o a otra (en este caso Spam o no-Spam), a partir de tanto las palabras que están en el mensaje a clasificar como las que no. El cálculo de la probabilidad de pertenecer a cada categoría se basa en el teorema de Bayes.
Dada una palabra W, la probabilidad de que esa palabra pertenezca a la categoría Spam viene dada por la ecuación siguiente, asumiendo a priori que P(Spam) = P(no-Spam)= 0.5:

$$P(Spam \mid W) = \frac{P(W \mid Spam)\,P(Spam)}{P(W)}$$

Si tenemos varias palabras W1, W2, ¬W3:

$$P(Spam \mid W1 \cap W2 \cap \neg W3) = \frac{P(W1 \cap W2 \cap \neg W3 \mid Spam)\,P(Spam)}{P(W1 \cap W2 \cap \neg W3)}$$

Asumiendo que las palabras son independientes entre sí, obtenemos:

$$P(Spam \mid W1 \cap W2 \cap \neg W3) = \frac{P(W1 \mid Spam)\,P(W2 \mid Spam)\,P(\neg W3 \mid Spam)\,P(Spam)}{P(W1)\,P(W2)\,P(\neg W3)}$$

De esta manera, es posible calcular las probabilidades marginales a partir de un dataset de entrenamiento, cuya categoría (Spam / no-Spam) de cada mensaje es conocida de antemano. Tras haber entrenado el modelo, este se utiliza para clasificar nuevos mensajes (usando un dataset de

testeo). Al igual que en la fase de preparación del dataset, existen multitud de paquetes en Python para entrenamiento y testeo de clasificadores Bayesianos, siendo uno de los más famosos e1071 [8].

Los métodos bayesianos pueden ser mejorados usando leyes de probabilidades distintas para, por ejemplo, tener en cuenta el hecho de que una palabra haya sido considerada muy pocas veces en el entrenamiento no debe tener un peso grande a la hora de clasificar el documento:

$$P'(Spam \mid W) = \frac{s\,P(Spam) + n\,P(Spam \mid W)}{s+n}$$

Donde el parámetro *n* viene definido como el número de veces que una palabra *W* haya aparecido durante la fase de entrenamiento, y el peso *s* tiene en cuenta la fuerza dada a la información de fondo ya conocida y global, comparado al valor de confianza obtenido considerando el documento en particular.

La gran ventaja de estos tipos de filtros es que se adaptan a los diferentes usuarios, lo que es muy útil en este caso ya que los spams recibidos suelen estar conectados a la actividad en línea de cada individuo particular.
Pero este método también tiene sus problemas, ya que la efectividad de estos filtros puede ser afectada usando por ejemplo métodos de *bayesian poisoning*, que consiste en poner en los spams una gran cantidad de texto que proviene de libros, con el objetivo de esconder el spam detrás de este texto y conseguir así que el filtro no sea capaz de identificarlo.

Cambiar el texto por imágenes es otra estrategia que suele ser utilizada para confundir a estos filtros, ya que estos no son capaces de analizar este tipo de información (pero las imágenes suelen pesar más que la misma información en texto, por lo que cuesta más enviar en masa este tipo de correos electrónicos). Por su parte, Gmail desarrolló en los últimos años un método de reconocimiento de caracteres en imágenes por esta razón.

## Support vector machines (SVMs):

El segundo método más utilizado históricamente en técnicas de filtrado de spam son las máquinas de vector soporte o métodos Kernel.

La idea principal de las máquinas de vector soporte es construir una función de kernel no lineal φ(x) que mapea los datos del espacio de entrada en un espacio de características normalmente de alta dimensión (ver Figura 2), para luego obtener el hiperplano óptimo que maximiza el margen entre las dos clases o categorías (en este caso Spam / no-Spam). De esta manera, la separación lineal se hace en el hiperespacio transformado, pero no en el de entrada.

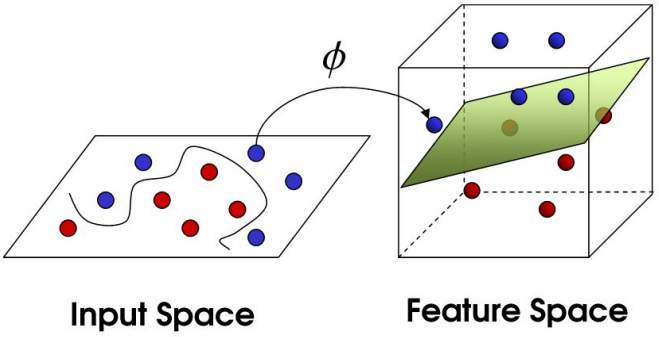


Figura 2: Representación gráfica de la transformación φ(x) del espacio de entrada al espacio transformado y la separación lineal en el espacio transformado. Imagen tomada de [9].

En el espacio transformado la función de decisión es lineal:

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$$

Pero en el espacio de entrada dicha función es no lineal y se expresa en función del denominado Kernel k:

$$\begin{aligned} f(\mathbf{x}) = & \underbrace{\left( \sum_i \alpha_i y_i \Phi(\mathbf{x}_i) \right)^T}_{\mathbf{w}^T} \Phi(\mathbf{x}) + b \\ = & \sum_i \alpha_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + b = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \end{aligned}$$

Lo que es más interesante de este método es que la transformación φ(x) es en principio desconocida e innecesaria para resolver el problema de clasificación; basta simplemente con conocer las matrices de kernel de productos escalares (artificio conocido comúnmente como kernel trick), que dan una idea de la similitud entre los posibles patrones. Esto simplifica mucho el problema, y ahí reside la mayor ventaja de este tipo de métodos.

Para la elección de la matriz de Kernel, es suficiente con que esta sea positiva semidefinida para el conjunto de datos de entrenamiento, siendo algunos de los kernels más utilizados el lineal, el polinómico o el gaussiano.

Una vez elegido el kernel, el último paso es resolver el problema de minimización:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \tfrac{1}{2} \boldsymbol{\alpha}^T \mathbf{YKY} \boldsymbol{\alpha} - \mathbf{1}^T \boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha}^T \mathbf{y} = 0, \\ & 0 \leq \boldsymbol{\alpha} \leq C \end{aligned}$$

Donde C hace referencia al parámetro de regularización, que establece un compromiso entre el error de entrenamiento y la complejidad del modelo.

La gran ventaja de estos métodos es que permiten hacer clasificación de los documentos en diferentes categorías, no se usan solamente para hacer una distinción spam/no spam. Además, son métodos no-supervisados en el sentido de que son capaces de determinar ellos mismos la mejor clasificación posible, sin que un usuario tenga que hacerlo a mano.

Hoy en día existen métodos SVM online bastante sencillos [10], capaces de aprender en el momento mismo con cada electrónico recibido.

## Redes neuronales y otras técnicas:

Las redes neuronales artificiales son grupos de unidades de procesamiento simples que están interconectadas y se comunican entre sí por medio de un número considerable de conexiones ponderadas. Cada una de las unidades acepta la entrada de las unidades en la capa anterior y calcula la salida que se transmite al siguiente nivel.

Las redes neuronales son un algoritmo potente para resolver cualquier problema de aprendizaje automático que requiere clasificación. Debido a su cada vez más frecuente utilización en distintos campos de la ciencia, están evolucionando como una herramienta muy relevante [11] en la detección de correo electrónico no deseado, aunque por el momento sigue siendo el clasificador bayesiano el método más utilizado como se comentó en la sección anterior.

En el caso de redes neuronales, lo que mejora su alta precisión es la gran cantidad de componentes (o grados de libertad) de procesamiento (neuronas) interconectados que trabajan conjuntamente para proporcionar una solución al problema de clasificación. Por ejemplo, Google documentó el aumento en la precisión de los filtros de spam de Gmail del 99.5% al 99.9% después de incorporar redes neuronales a sus algoritmos de identificación. Esto sugiere que las redes neuronales podrían ser útiles para mejorar el rendimiento de los filtros de spam, particularmente cuando se hibridan con la clasificación bayesiana y otras técnicas.
Por otro lado, aún se necesita hacer mucha investigación sobre la aplicación de redes neuronales para la detección de spam, y casi todo el trabajo actual está enfocado en el estudio de la optimización de la estructura de red.

Por otra parte, los modelos generativos profundos han mostrado resultados prometedores para el aprendizaje semi-supervisado. Específicamente, las Redes Adversarias Generativas (GAN), que tienen la capacidad de generar muestras muy cercanas a datos reales, han logrado muy buenos resultados en este área en el caso de datasets de entrenamiento con estadística limitada.
Dado que actualmente la mayoría de las aplicaciones de las GANs son para datasets de imágenes (valores continuos) y no para datos de texto (valores discretos), su estudio en el contexto de spam-detection para los casos en los que el número de textos de entrenamiento son limitados puede ser de gran interés científico.

Las GAN operan entrenando dos redes neuronales que operan de la siguiente manera: el discriminador D intenta discriminar las muestras de entrenamiento reales de las falsas, mientras que el generador G intenta generar muestras de entrenamiento falsas para engañar al discriminador. De esta manera, el clasificador Spam-no Spam se puede nutrir de datos reales y de datos generador por G mitigando así el problema falta de estadística para realizar el entrenamiento.
En la Figura 3 se representa un esquema gráfico del funcionamiento de esta técnica.

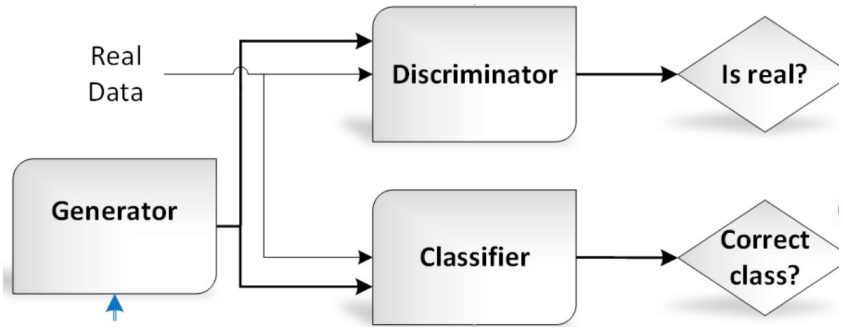


Figura 3: Esquema del funcionamiento de GANs. Imagen tomada de [12]

Se ha demostrado que este tipo de algoritmos proporcionan un accuracy en la clasificación de aproximadamente el 80% usando los datos en abierto de TripAdvisor [12]

# Conclusiones

El filtrado de mensajes de spam es algo muy relevante hoy en día, especialmente dado el alto volumen de información intercambiada entre ordenadores y por el hecho de que el 80% de todos los correos electrónicos enviados cada día son spams. Por evidentes motivos de productividad,, el uso de un método de filtro automático eficiente es imprescindible en la mayoría de las empresas.

Estos métodos de detección de spam, como casi cualquier análisis de text mining, requiere dos etapas fundamentales:
En primer lugar, una fase previa al análisis de la sintaxis del documento analizado, con el objetivo principal de simplificar el documento para que un ordenador pueda analizarlo y extraer la información necesaria. Esto se hace en diferentes etapas descritas en este trabajo, y pasando las palabras del documento a un espacio vectorial donde se pueden definir distancias y otros parámetros matemáticos que permitan el análisis posterior del documento.
Una vez finalizada esta etapa, el análisis semántico en sí puede empezar, típicamente mediante el uso de distintos métodos de aprendizaje automático como redes bayesianas, SVMs o neural networks, que proporcionan en cada caso distintas ventajas y fortalezas a la hora de analizar los correos electrónicos de los usuarios.
Estos métodos se aplican típicamente de forma online, siendo capaces de aprender en el momento mismo con cada electrónico recibido.

Mediante el uso combinado de estos métodos de detección, los mejores filtros de spam en la actualidad son capaces de alcanzar una precisión de detección superior al 99%, consiguiendo una tasa de falsos positivos (correos electrónicos de verdad identificados como spam) muy baja, y permitiendo de esta manera optimizar cada vez más la productividad de los empleados de todas las compañías alrededor del mundo.

# Bibliografía


[1]. Statista, “Number of e-mail users worldwide from 2017 to 2023” [consultado en línea, Marzo 2020] https://www.statista.com/statistics/255080/number-of-e-mail-users-worldwide/.

[2]. M.F. Porter, “An algorithm for suffix stripping”, 1980 [consultado en línea, Marzo 2020] https://tartarus.org/martin/PorterStemmer/def.txt.

[3]. A. Honrado et al., “A word stemming algorithm for the Spanish Language” [consultado en línea, Marzo 2020] https://ieeexplore.ieee.org/document/878189.

[4]. Python Natural Language Toolkit, https://www.nltk.org/ [consultado en línea, Marzo 2020].

[5]. Scikit learn, TfidfVectorizer method, https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html [consultado en línea, Marzo 2020].

[6]. Maron, M. E. "Automatic Indexing: An Experimental Inquiry". Journal of the ACM. 8 (3): 404–417, 1961.

[7]. Paul Graham. “A Plan for Spam” August 2002. http://paulgraham.com/spam.html [consultado en línea, Marzo 2020].

[8]. David Meyer et al, “Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien”, versión 1.7-3, 2019.

[9]. N.J. Nalini, S. Palanivel. “Music emotion recognition: The combined evidence of MFCC and residual phase”. Egyptian Informatics Journal 17(1). September 2015.

[10]. K.W. Lau, Q.H. Wu, “Online training of support vector machine”. Pattern Recognition, Volume 36, Issue 8, August 2003, Pages 1913-1920.

[11]. David Ndumiyana et al, “Spam Detection using a Neural Network Classifier”, ISSN 2315-5027; Volume 2, Issue 2, pp. 28-37; April, 2013. Online Journal of Physical and Environmental Science Research.

[12]. Gray Stanton,  Athirai A. Irissappane, “GANs for Semi-Supervised Opinion Spam Detection”, arXiv:1903.08289.