UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

TESIS DE GRADO

"SISTEMA INTELIGENTE PARA LA DETECCION DE CONVERSACIONES CON POSIBLE CONTENIDO PEDOFILICO, BASADO EN REDES NEURONALES"

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

POSTULANTE: EYNAR DAVID TORREZ TORREZ
TUTOR METODOLÓGICO: LIC. FREDDY MIGUEL TOLEDO PAZ
ASESOR: M. Sc. MOISES MARTIN SILVA CHOQUE

LA PAZ– BOLIVIA

2018

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

DEDICATORIA

A mi mama María Torrez, por darme la vida, quererme mucho, creer en mí y porque siempre me apoyaste.

A mi hermano Adrián Murguía, por estar conmigo y apoyarme siempre, te quiero mucho.

A Mercedes Quiñajo, por siempre apoyarme y estar conmigo en todo momento.

AGRADECIMIENTOS

A mi familia por el apoyo incondicional a lo largo de mi carrera universitaria.

A mi tutor Lic. Freddy Miguel Toledo Paz, por sus sugerencias, sus observaciones y por guiarme para la elaboración de la presente tesis de Grado.

A mi asesor M.Sc. Moisés Martin Silva Choque, por su orientación, su tiempo, y sus sugerencias y colaboración en la presente tesis de grado.

A Mercedes Quiñajo, por haberme ayudado a realizar este trabajo.

RESUMEN

El creciente uso de internet lleva con sigo el aumento de casos de pedofilia, en donde personas adultas encontraron el ambiente propicio en el uso del chat para poder contactarse y aprovecharse de niños. En esta tesis se presenta un modelo de sistema inteligente que sea capaz de clasificar conversaciones con contenido pedófilo usando Redes Neuronales.

Se explica de manera detallada los métodos y técnicas aplicadas sobre minería de texto y modelos de recuperación de procesamiento del lenguaje natural, los cuales son combinados con algoritmos de Redes Neuronales, para desarrollar el modelo que logre detectar las conversaciones pedófilas. El método de minería de textos hace uso de un aprendizaje supervisado, lo que implica que se tuvo que recolectar las conversaciones con contenido pedófilo, necesarias para que la red neuronal pueda aprender.

Luego de determinadas pruebas realizadas se logró definir un modelo final que se evaluó en base a medidas de Precisión y Exhaustividad, en donde los resultados obtenidos son prometedores.

Palabras clave: Redes Neuronales, Minería de Textos, Clasificación automática de textos, Pedofilia en internet.

ABSTRACT

The growing use of the internet has led to an increase in cases of pedophilia, where adults found the right environment in the use of chat to be able to contact and take advantage of children. This thesis presents an intelligent system model that is capable of classifying conversations with pedophile content using Neural Networks.

It is explained in detail the methods and techniques applied to text mining and natural language processing recovery models, which are combined with Neural Networks algorithms, to develop the model that manages to detect pedophile conversations. The method of text mining makes use of supervised learning, which implies that had to be collected conversations with pedophile content, necessary for the neural network to learn.

After certain tests carried out, it was possible to define a final model, that was evaluated based on Precision and Recall measurements, where the results obtained are promising.

Key words: Neural Networks, Text Mining, Automatic text classification, Pedophilia on the internet.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

CAPITULO I

MARCO INTRODUCTORIO

1.1. Introducción

La evolución de internet, ha marcado profundamente la vida de las personas, al mismo tiempo aparecen nuevos riesgos y nuevas amenazas. El flujo de información se ha incrementado, permite acceder a lo último en tecnología incorporando aplicaciones de todo tipo, editores de fotos, juegos, chats con las cuales se puede interactuar en tiempo real y con mejor experiencia, Internet ha popularizado el contacto con usuarios de cualquier parte del mundo. Las plataformas virtuales cuentan con perfiles personalizables donde los usuarios tienen la posibilidad de crear nuevas relaciones interpersonales, laborales o amorosas.

Una característica de la comunicación es a través del chat[1], en donde un usuario usa su nombre real o uno ficticio, esto en muchas ocasiones permite que este, se desinhiba verbalmente y poder dar rumbo a la conversación en el sentido que él quiera orientarlo, son muchos los usuarios mayores de edad, los que encuentran propicio el ambiente para contactar menores de edad y sostener diálogos de contenido sexual. (Beltrán, Ordoñez, 2014). Los niños son usuarios activos en internet, siendo la población más vulnerable, convirtiéndose en víctimas del acoso en todas sus categorías o la pornografía infantil, los cuales se ven potenciadas en la red mediante las múltiples plataformas. Así es que algunas de las lacras de la cibersociedad son las conductas de pedofilias[2] en internet.

En este sentido se propone el desarrollo de un modelo de sistema inteligente para identificar las conversaciones con contenido pedófilo que pueda presentarse en un chat, de este modo presentar una nueva alternativa eficiente de clasificación automática de textos, fundamentado en técnicas de minería de texto, de procesamiento del lenguaje natural y Redes Neuronales con algoritmos de aprendizaje supervisados destinada al ámbito de detección y clasificación, para el presente estudio se analizaran únicamente conversaciones en el idioma inglés.

1 chat: propiamente 'charla'. Intercambio de mensajes electrónicos a través de internet que permite establecer una conversación entre dos o varias personas.

2 Pedofilia: atracción erótica o sexual que una persona adulta siente hacia niños o adolescentes.

Se hicieron avances en la detección automática de depredadores sexuales, empresas como Facebook han impulsado este tipo de investigaciones, sin embargo aún no se han podido lograr efectivamente la detección e identificación, algo que perjudica este tipo de investigaciones es la escasez de material para el análisis ya que podría por un lado ser una posible violación a la privacidad, y también otro tipo de situaciones, como el hecho de que existen términos obscenos, que no siempre usan los depredadores sexuales, también el estudio de los morfemas (segmento lingüístico mínimo dotado de significado gramatical y léxico) pueden ser escritos total o parcialmente, con errores ortográficos o creación por parte de los mismos usuarios.

1.2. Estado del Arte

1.2.1. Investigaciones Internacionales

La investigacion realizada el año 2007 por Pendar, denominada " *Toward spotting the pedophile telling victim from predator in text chats*[3].",es un estudio piloto "sobre el uso de técnicas de clasificación automática de textos para identificar depredadores sexuales en línea", este utiliza un *corpus*[4] obtenido del sitio *Perverted justice*[5] que se presenta como el inicio para la detección del *grooming attack*[6], en la investigación se identifican los textos de los depredadores sexuales y los textos de las victimas (Clasificación en dos clases),utilizando Maquinas de soporte Vectorial (SVM) y k-vecinos (KNN) se hicieron varias pruebas con diferentes características de 5000 a 10000, obteniendo mejores resultados con un conjunto de 10000 características con K-NN (k=30 depredaros). (Pendar, 2007)

3 Hacia la detección de pedófilos, diciendo víctima de depredador en chats de texto

4 Conjunto cerrado de textos o de datos destinado a la investigación científica.

5 HTTP://www.perverted-justice.com organización estado anídense, que investiga, identifica y divulga la conducta de los adultos que se hacen pasar por menores para solicitar conversaciones sexuales en línea con menores reales.

6 Proceso de comunicación por el cual un autor aplica estrategias de búsqueda de afinidad, mientras que simultáneamente adquiere información sobre sus víctimas con el fin de desarrollar las relaciones que resulten en cumplimiento de su necesidad, por ejemplo, acoso sexual físico.

La "CLEF PAN Lab on Uncovering Plagiarism, Autoriship, and Social software Misuse[7]*"* es el congreso donde se abordan temas relacionados con plagio de textos, y vandalismo informático; a partir del año 2012 se incluyó a sus atribuciones la subtarea de "*Sexual predator Identification*[8]*".* En el congreso ya se realizaron investigaciones con la temática de identificación de depredadores sexuales en internet.

Con la investigación realizada el año 2012, se logró reducir la dimensión del corpus, en donde no se realiza ningún procesamiento, simplemente un filtrado para quitar aquellas con versaciones muy cortas, con caracteres confusos, o donde los actores intervienen muy poco, donde se obtuvo un *F-score*[9] alto en la subtarea de identificación de depredadores, ponderando la precisión y exhaustividad. (Villatoro, Juarez, Escalante, Montes, Pineda, 2012)

En México se realizó la siguiente investigación; Detección de depredadores sexuales utilizando un sistema de detección y clasificación supervisada, la investigación fue realizada por 4 estudiantes, en la Facultad de Ciencias de la Computación BUAP[10] en puebla México. (Aleman, Vilariño, Tobar, 2014)

El Sistema inteligente para la detección de diálogos con posibles contenidos pedófilos llevada a cabo por los grupos de investigación GITIS[11] y GESDATOS[12] de la Universidad Distrital Francisco José de Caldas, se publicó en la Revista virtual de la Universidad Católica del Norte el año 2014, nuevamente en esta tesis se confirma que el algoritmo de Maquina de Soporte de Vectores es el método con mejor precisión para la clasificación de textos con alta dimensionalidad y tolerante a variaciones léxicas de los textos. (Revista virtual Universidad Católica del Norte, 2014)

7 http://pan.webis.de/ PAN es una serie de eventos científicos y tareas compartidas sobre análisis forense de textos digitales

8 Identificación de depredadores sexuales

9 En estadística es la medida de precisión que tiene un test, se usa en la fase de prueba de algoritmos de búsqueda y recuperación de información y clasificación de documentos.

10 Benemérita Universidad autónoma de Puebla

11 Grupo de Investigación en Telecomunicaciones e Ingeniería de Software.

12 Grupo de Gestión Integral en Protección de Datos.

El año 2013 se revoluciona la tecnología con esta nueva herramienta Negobot: Agente conversacional basado en teoría de juegos para la detección de conductas pedófilas, una investigación realizada en la universidad de Deusto, realizada por Carlos Laorden Gómez, Patxi Galán García, Igor Santos Grueiro, Pablo García Bringas, Borja Sanz Urquijo y José Maria Gomez Hidalgo, este sistema observa la conversación como una competición en la que el objetivo es obtener la máxima información, para poder decir si el sujeto con el que negobot está hablando tiene tendencias pedófilas o no. Negobot adopta el comportamiento de un niño o de una niña que es el gancho para los depredadores de internet, haciendo uso de 7 bots[13] de chat con distintos comportamientos que permiten actuar de forma distinta según el estado de la conversación. (Laorden, Galan, Santos, Garcia, Sanzs, Gomez, 2013)

Un trabajo más reciente es el Diseño de la herramienta para detectar usuarios depredadores en Facebook, llevado a cabo por Paola Andrea Valens Orejuela, en la Universidad de San Buenaventura Cali Colombia en Ingeniería multimedia el año 2016, con la colaboración de Facebook y el patrocinio de Mark Zuckerberg. Este proyecto investigativo ha sido aplicado en la red social llegando a identificar depredadores en tiempo real y al instante, ya que realiza el análisis inmediatamente que una persona conversa con otra, el sistema clasifica las palabras haciendo un filtrado y comparándolo con su corpus diseñado solo para palabras en inglés. Actualmente Facebook continúa trabajando en este proyecto. (Valens, 2016)

1.2.2. Investigaciones Nacionales

A nivel nacional, Bolivia no existe alguna Investigación en el campo de detección de conversaciones pedófilas o clasificación de documentos.

1.3. Planteamiento del problema

Los riesgos y las vulnerabilidades se han incrementado considerablemente esta última década, las actividades de pedofilia no son la excepción, por lo que en estos últimos años la Pedofilia ha aumentado a la par de la tecnología, con medios donde se pueden compartir y

13 Bot: Programa informático que efectúa automáticamente tareas repetitivas a través de internet, aféresis de robot.

comentar contenidos, por esta facilidad de relacionarse tanto con menores de edad como con otros pedófilos a nivel mundial, han existido varias iniciativas para investigar este fenómeno. En la gráfica 1 se muestran las estadísticas de usuarios menores de edad en internet año 2017. Es aquí en donde la investigación, propuesta de algoritmos y desarrollo del prototipo de tecnologías como el procesamiento del lenguaje natural, agentes o sistemas inteligentes, dirigidas a combatir delitos como la Pedofilia a través de internet son necesarios ya que este tipo de delincuentes se apropia cada vez más del saber tecnológico y cibernético para lograr sus objetivos.

1.3.1. Problema General[14]

¿Cómo identificar de manera eficiente conversaciones con contenido pedófilo que pueda presentarse en un chat usando Redes Neuronales?

1.3.2. Problemas Específicos

- No se cuenta con un corpus de conversaciones con contenido pedófilo que permita entrenar el algoritmo.
- Los archivos de conversaciones textuales no cuentan con un formato ni una estructura definida, lo que produce dificultad para el análisis de los contenidos de las conversaciones.
- No se cuenta con un diseño definido de una Red neuronal Artificial que clasifique las conversaciones, por lo cual no se puede distinguir entre conversaciones positivas y conversaciones negativas.

1.4. Hipótesis

Un sistema inteligente basado en redes neuronales puede detectar conversaciones con contenido pedófilo.

[14] Ver diseño del Árbol de problemas en el Anexo B.

1.5. Objetivos

1.5.1. Objetivo General[15]

Desarrollar un modelo usando Redes Neuronales que sea capaz de reconocer conversaciones de chat con contenido pedófilo.

1.5.2. Objetivos Específicos

- Recopilar las conversaciones necesarias con contenido pedófilo para que la Red Neuronal pueda aprender de ellas.
- Realizar el Pre procesamiento de las conversaciones extraídas, para el análisis de contenido.
- Diseñar una Red Neuronal Artificial de clasificación, para distinguir conversaciones positivas y conversaciones negativas.

1.6. Justificación

1.6.1. Justificación Social

Un paso importante para la detección de individuos que se dedican a actividades pedófilas en internet, es la de un sistema inteligente que permita detectar conversaciones con posible contenido pedófilo, Basado en Redes Neuronales.

1.6.2. Justificación Técnica

El sistema Inteligente usa herramientas de código abierto para el desarrollo del modelo, el uso de potentes servidores es necesario para el procesamiento en la fase de entrenamiento de la Red Neuronal y la conexión a internet que se requiere para la obtención y transferencia de los datos.

[15] Ver diseño del Árbol de objetivos en el Anexo C

1.6.3. Justificación Científica

La investigación emplea técnicas de minería de texto, procesamiento del lenguaje natural y uso del perceptrón multicapa como técnica de aprendizaje de máquina para la clasificación de textos, que facilita la manipulación de la información, proponiendo así una alternativa al algoritmo de Máquinas de Vectores de Soporte, K-NN y Naive Bayes.

1.6.4. Justificación Económica

El desarrollo del modelo del sistema inteligente implica una inversión de un aproximado de $180 donde se contemplan los costos de servidor para el entrenamiento de la Red Neuronal.

1.7. Límites y Alcances - Delimitación de la tesis

1.7.1. Delimitación temática

Detección de posibles conversaciones pedófilas mediante el reconocimiento y clasificación de textos. Las conversaciones recolectadas son en su totalidad en idioma inglés.

1.7.2. Delimitación espacial

La presente Tesis de investigación se llevará a cabo en la carrera de Informática de la Universidad Mayor de San Andrés.

1.7.3. Delimitación temporal

El desarrollo de la investigación se efectúa a partir del año 2017. Los datos de estudio se refieren esencialmente al ámbito de detección de pedófilos en internet, desde el año 2000 hasta los días actuales.

1.8. Metodología de Investigación

La tesis pretende detectar conversaciones con posible contenido pedófilo usando redes Neuronales, por lo tanto, es necesario realizar diferentes pruebas controladas que permitan el buen diseño de la Red Neuronal y el modelado ideal en el pre-procesado de los textos, es así que la investigación responde a una investigación experimental y se realiza una investigación aplicada para buscar una Red Neuronal que aprenda a reconocer los patrones de las conversaciones pedófilas y pueda clasificarlas.

Investigación Experimental

El método experimental trata de determinar la presencia de una causa y un efecto definidos. Esto implica que una vez que se usa este método puede emitirse un juicio acerca de que si A causa que B suceda o A no causa que B suceda. Otros métodos, como los modelos histórico y descriptivo, no brindan tales beneficios. Aunque esos métodos pueden servir para descubrir relaciones entre variables, no hay forma de establecer una relación causal. Debido a que el método experimental, permite controlar las posibles fuentes de las diferencias (o la varianza), se puede afirmar lo siguiente: un factor está relacionado con otro de tal manera que los cambios en ese factor tienen una relación causal con los cambios en el otro. (Salkind, 1999)

Investigación Aplicada

La investigación aplicada recibe el nombre de "investigación práctica o empírica", que se caracteriza porque busca la aplicación o utilización de los conocimientos adquiridos, a la vez que se adquieren otros, después de implementar y sistematizar la práctica basada en investigación. El uso del conocimiento y los resultados de investigación que da como resultado una forma rigurosa, organizada y sistemática de conocer la realidad. (Murillo, 2008)

Metodología de diseño

"Para la construcción del modelo de clasificación se siguió técnicas en minería de texto para la clasificación de textos" (Vandana & Namrata, 2012). Las técnicas de clasificación se aplicaron teniendo en cuenta algunas variaciones; por ejemplo, el incluir técnicas de Stemming y TF-IDF. También se usó de técnicas para la creación y desarrollo de Redes

Neuronales (Gestal, 2010) al momento de diseñar el algoritmo de la RNA. Con el fin de evaluar el modelo de clasificación se emplearon las siguientes medidas, el índice de precisión y exhaustividad.

1.9. Aporte

La presente tesis de investigación, propone un modelo de clasificación de documentos que usa técnicas y metodologías de Minería de Textos conjunto con técnicas de Procesamiento del Lenguaje Natural y Redes Neuronales Artificiales, las cuales son combinadas para obtener un modelo diferente de clasificación de documentos que podrá ser utilizada para futuras investigaciones.

CAPITULO II

MARCO TEÓRICO

2.1. Minería de textos

Se denomina minería de textos al proceso de descubrimiento de conocimientos no explícitos y no triviales en corpus textuales escritos en lenguaje natural. A diferencia de la minería de datos, que se encarga de extraer información de fuentes estructuradas, normalmente bases de datos, la minería de textos utiliza fuentes textuales cuyo contenido puede estar semi-estructurado en el mejor de los casos o carecer de cualquier tipo de estructura. Esta diferencia añade un grado de dificultad extra a la minería de textos frente a la minería de datos, ya que implica la utilización de técnicas y métodos que permitan estructurar la información textual antes de realizar cualquier proceso de minería de datos. (Feldman, 2006)

2.1.1 Importancia de la minería de textos

Una consecuencia del uso generalizado de las computadoras es que la mayoría de los datos se originan en forma digital. Si se intercambia una acción o se escribe un libro o se compra un producto en línea, estos eventos evolucionan electrónicamente. Dado que las transacciones en papel de los hombres ahora están en forma digital sin papel, muchos datos "grandes" están disponibles para un análisis posterior.

El concepto de minería de datos, la búsqueda de patrones valiosos en los datos, es una respuesta obvia a la recopilación y el almacenamiento de grandes volúmenes de datos. La minería de datos ya no es una tecnología emergente que espera un mayor desarrollo. Aunque su aplicación está lejos de ser universal, las técnicas de extracción de datos están muy desarrolladas y algunas formas de análisis están entrando en una fase madura.

Se ambiciona poder decir "Danos datos y encontraremos los patrones" desafortunadamente, los métodos de minería de datos esperan un formato de datos altamente estructurado, lo que requiere una extensa preparación de datos. O bien que transformar los datos originales, o los datos se proporcionan en un formato altamente estructurado.

Los métodos de extracción de datos aprenden de experiencias pasadas. Especialistas en minería de datos predictivos, usan datos serán en forma numérica. Estas personas son los

"chicos de los números". Los "mineros del texto" no esperan una serie ordenada de números. Están felices de mirar colecciones de documentos, donde los contenidos son legibles y su significado es obvio. Esta es la primera distinción entre minería de datos y texto: números vs. texto. Son conceptos distintos. Ambos se basan en muestras de ejemplos pasadas. La composición de los ejemplos es muy diferente, sin embargo, muchos de los métodos de aprendizaje son similares. Eso se debe a que el texto se procesará y se transformará en una representación numérica. (Feldman, 2006)

2.1.2. Diferencia entre texto y número

Las presentaciones de datos para la minería de datos clásica y la minería de textos son diferentes. Mientras que los métodos de minería de datos les gusta ver los datos en formato de hoja de cálculo, los métodos de minería de texto les gusta ver un formato de documento y la presentación estándar de aprendizaje es una variante del formato denominado XML utilizado en el mundo de documentos.

Se espera que el texto sea bastante diferente de los números. Aun así, los métodos son similares a los usados para la extracción de datos. Estos métodos han demostrado ser notablemente exitosos sin comprender las propiedades específicas del texto como los conceptos de gramática o el significado de las palabras. Se utiliza información de frecuencia estrictamente de bajo nivel, como el número de veces que aparece una palabra en un documento, y luego se aplican métodos bien conocidos de aprendizaje automático. Uno de los temas principales que respaldan la minería de textos es la transformación del texto en datos numéricos, por lo que, aunque la presentación inicial es diferente, en una etapa intermedia, los datos pasan a la codificación de minería de datos clásica. Los datos no estructurados se estructuran. (Weiss, Indurkhya y Zhang, 2005)

2.1.3. Tipos de problemas que se pueden resolver

Un foco primario de atención es la clasificación y la predicción. Estos son algunos de los métodos y aplicaciones más ampliamente estudiados y aplicados de minería de datos, el objetivo es encontrar las respuestas correctas para nuevos ejemplos. Se considerarán esos

tipos de problemas, como la categorización de texto, que son aplicaciones claras para métodos predictivos. (Weiss, Indurkhya y Zhang, 2005)

2.1.4. Clasificación de documentos

La categorización de texto es el nombre ampliamente utilizado, pero opresivo, para la clasificación de documentos. Es la encarnación más pura del modelo de hoja de cálculo con respuestas etiquetadas.

Una vez que los datos se transforman al formato de hoja de cálculo numérico habitual, se aplican los métodos estándar de extracción de datos. Los documentos están organizados en carpetas, una carpeta para cada tema. La figura 2.1. ilustra la aplicación de clasificación de documentos.

Fig. 2.1. Clasificación de documentos
Fuente (Weiss, Indurkhya y Zhang, 2005)

Se presenta un documento nuevo, y el objetivo es colocar este documento en las carpetas apropiadas. Por ejemplo, se podría tener una carpeta para documentos domésticos o financieros y se quiere agregar nuevos documentos a la carpeta correcta. La aplicación es casi siempre una clasificación binaria porque un documento generalmente puede aparecer en varias carpetas.

Originalmente, este tipo de problema se consideraba una forma de indexación, al igual que el índice delgado de un libro. A medida que hay más documentos disponibles en línea, la aplicabilidad de esta tarea se ha ampliado. Algunas de las tareas más obvias se relacionan con el correo electrónico: por ejemplo, el reenvío automático de correo electrónico al departamento correspondiente o la detección de correo no deseado. El modelo de hoja de cálculo con una columna correspondiente a la respuesta correcta es el modelo de clasificación

universal para datos, y los datos de texto transformados se pueden combinar fácilmente con datos numéricos estándar minería de datos. Como ejemplo, puede pensar que puede predecir el movimiento de existencias futuras en función de la experiencia previa. Recopila noticias que aparecieron antes del aumento de todos los precios de las acciones junto con los datos financieros de la compañía. Las etiquetas serían binarias, 1 para arriba y 0 para abajo. (Weiss, Indurkhya y Zhang, 2005)

2.1.5. Recuperación de la información

La recuperación de información es el tema más comúnmente asociado con los documentos en línea. Se obtiene una colección de documentos, y se dan pistas sobre los documentos que se desea recuperar de la colección, y luego los documentos que coincidan con las pistas se presentan como respuesta a la consulta.

Las pistas son palabras que ayudan a identificar los documentos almacenados relevantes. El proceso se puede generalizar en una matriz de documentos, donde en lugar de unas pocas palabras, un documento completo se presenta como un conjunto de pistas. El documento de entrada luego se combina con todos los documentos almacenados, recuperando los mejores documentos coincidentes. (Weiss, Indurkhya y Zhang, 2005)

El nuevo documento es equivalente a una nueva fila. La nueva fila se compara con todas las otras filas, y las filas más similares y sus documentos asociados son las respuestas.

En la figura 2.2. se muestra el proceso de recuperación de la información.

Fig. 2.2. Recuperando Documentos Coincidentes
Fuente: (Weiss, Indurkhya y Zhang, 2005)

2.1.6. Extracción de información

La representación de datos analiza la información en términos de palabras. En comparación con las representaciones numéricas clásicas de minería de datos, estas mediciones son muy superficiales. Solo se está midiendo la presencia o ausencia de palabras. Una representación de minería de datos puede tener el mismo aspecto, pero las medidas en sí mismas serán mucho más amplias y complejas. Pueden ser variables de valor real, como el volumen de ventas o un código, como un código de la industria como "industria automotriz". Alguien es responsable de definir estos atributos y depositarlos en una base de datos.

Una forma alternativa de ver estas distinciones es que la minería de datos espera datos altamente estructurados, y el texto está naturalmente desestructurado. Para estructurar el texto, se emplea una representación muy superficial que mide la ocurrencia simple de las palabras. La extracción de información es un subcampo de minería de textos que intenta impulsar la minería de textos en igualdad de condiciones con el mundo estructurado de la minería de datos. El objetivo es tomar un documento no estructurado y completar automáticamente los valores de una hoja de cálculo.

En términos del modelo de hoja de cálculo de datos, el objetivo es completar las celdas. El papel de los ejemplos no ha cambiado; son documentos. Las columnas no son solo palabras clave, sino que pueden ser conceptos de nivel superior que se encuentran en el proceso de extracción de información. Este proceso examina documentos y rellena las celdas, un proceso que es equivalente a rellenar una tabla de base de datos. Una vez que se completa el proceso, los métodos usuales de aprendizaje se pueden aplicar a la hoja de cálculo. (Weiss, Indurkhya y Zhang, 2005)

En la figura 2.3 se ilustra la tarea de extracción de información

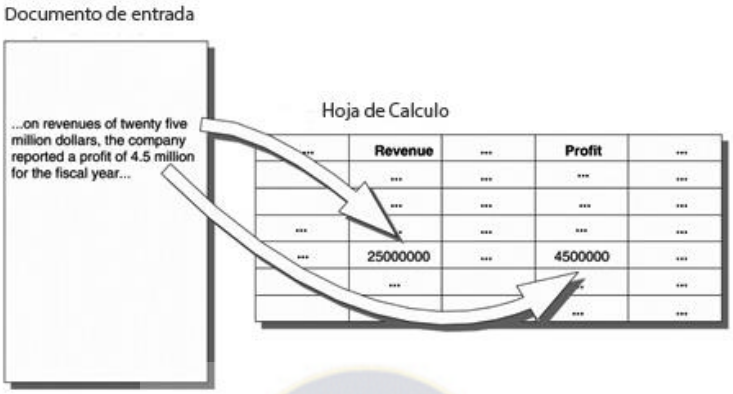


Fig. 2.3. Extraer información de un documento
Fuente: (Weiss, Indurkhya y Zhang, 2005)

2.2. De Información textual a Vectores Numéricos

Para extraer el texto, primero se debe procesar en una forma que los procedimientos de extracción de los datos (*Data Mining*) se pueda usar, esto implica generar características en un formato de hoja calculo. La minería de datos (Data mining) clásica analiza datos altamente estructurados. El modelo de hoja de cálculo es la encarnación de una representación que apoya el modelado predictivo. A continuación, obtener los tipos de características comúnmente generadas a partir del texto.

2.2.1. Recolección de documentos

El primer paso en la minería de textos es recopilar los datos (es decir, los documentos relevantes). En muchos escenarios de minería de textos, es posible que ya se hayan entregado los documentos relevantes o que formen parte de la descripción del problema. Por ejemplo, una aplicación de recuperación de página web para una intranet, especifica implícitamente que los documentos relevantes son las páginas web en la intranet. Si los documentos se identifican fácilmente, se pueden obtener, y el problema principal es limpiar las muestras y garantizar que sean de alta calidad. Al igual que con los datos no textuales, la intervención humana puede comprometer la integridad del proceso de recopilación de documentos y, por lo tanto, se debe tener extremo cuidado.

A veces, los documentos se pueden obtener de almacenes de documentos o bases de datos. En estos escenarios, es razonable esperar que la limpieza de los datos se haya realizado antes del depósito y se pueda confiar en la calidad de los documentos.

En algunas aplicaciones, uno puede necesitar tener un proceso de recopilación de datos. Por ejemplo, para una aplicación web que comprende varios sitios web autónomos, uno puede implementar una herramienta de software, como un rastreador web que recopila los documentos. En otras aplicaciones, se puede tener un proceso de registro conectado a una secuencia de datos de entrada por un período de tiempo.

Para la investigación y el desarrollo de técnicas de minería de textos, pueden ser necesarios más datos genéricos. Esto generalmente se llama un corpus.

En los primeros días del procesamiento de textos (años 1950 y 1960), un millón de palabras se consideraba una colección muy grande. Este fue el tamaño de una de las primeras colecciones ampliamente disponibles, el corpus Brown, que consta de 500 muestras de alrededor de 2000 palabras cada una de los textos en inglés de diversos géneros. Algunos corpus ampliamente utilizados son el Penn Tree Bank, una colección de oraciones del Wall Street Journal analizadas manualmente; las colecciones TREC (Text Retrieval and Evaluation Conferences), que constan de selecciones del Wall Street Journal, el New York Times, las publicaciones de Ziff-Davis, el Federal Register y otros; las actas del Parlamento canadiense en traducciones paralelas inglés-francés, ampliamente utilizadas en la investigación estadística de traducción automática; y el Proyecto Gutenberg, una gran colección de textos literarios y de otro tipo puestos en forma de lectura mecánica a medida que el material sale de los derechos de autor. Una colección de noticias de Reuters llamada Reuters-21578 Distribution 1.0 ha sido ampliamente utilizada en el estudio de métodos para la categorización de texto. (Weiss, Indurkhya y Zhang, 2005)

Otro recurso a considerar es la propia World Wide Web[16]. Los rastreadores web pueden crear colecciones de páginas de un sitio en particular, como Yahoo, o sobre un tema en particular. Dado el tamaño de la Web, las colecciones creadas de esta manera pueden ser enormes. El

16 En informática WWW es un sistema de distribución de documentos de hipertexto interconectados y accesibles desde internet.

principal problema con este enfoque para la recolección de documentos es que los datos pueden ser de dudosa calidad y requieren una limpieza exhaustiva antes de su uso.

2.2.2. Estandarización de documentos

Una vez que se recopilan los documentos, no es raro encontrarlos en una variedad de formatos diferentes, dependiendo de cómo se generaron los documentos. Por ejemplo, algunos documentos pueden haber sido generados por un procesador de textos con su propio formato propietario; otros pueden haber sido generados usando un editor de texto simple y guardados como texto ASCII; y algunos pueden haber sido escaneados y almacenados como imágenes. Evidentemente al procesar todos los documentos, es útil convertirlos a un formato estándar.

La industria de la informática en su conjunto, incluida la mayoría de la comunidad de procesamiento de textos, ha adoptado XML (Lenguaje de marcado extensible) como su formato estándar de intercambio, y este es el estándar que también se adopta para las colecciones de documentos en esta tesis. XML es una forma estándar de insertar etiquetas en un texto para identificar sus partes. Aunque las etiquetas se pueden anidar dentro de otras etiquetas a una profundidad arbitraria, se usa esa capacidad solo moderadamente. Cada documento está separado de los otros documentos en el corpus al tener una etiqueta distintiva al principio, como <DOC>. Según la convención XML, las etiquetas vienen en pares inicial y final. Están encerrados en corchetes angulares, y la etiqueta final tiene una barra invertida inmediatamente después del corchete angular de apertura. Dentro de un documento, puede haber muchas otras etiquetas para marcar secciones del documento. Las secciones comunes son <DATE>, <SUBJECT>, <TOPIC> y <TEXT>. el enfoque principalmente es en <SUBJECT>, <TOPIC> y <TEXT>. Los nombres son arbitrarios. Podrían ser <HEADLINE> y <BODY>. Un ejemplo de documento XML se muestra en la figura 2.4, donde el documento tiene una etiqueta distintiva de <DOC>. (Weiss, Indurkhya y Zhang, 2005)

```
<DOC>
<TEXT>
<TITLE>
Solving Regression Problems with Rule-based Classifiers
</TITLE>
<AUTHORS>
<AUTHOR>
Nitin Indurkhya
</AUTHOR>
<AUTHOR>
Sholom M. Weiss
</AUTHOR>
</AUTHORS>
<ABSTRACT>
We describe a lightweight learning method that induces an ensemble
of decision-rule solutions for regression problems. Instead of
direct prediction of a continuous output variable, the method
discretizes the variable by k-means clustering and solves the
resultant classification problem. Predictions on new examples are
made by averaging the mean values of classes with votes that are
close in number to the most likely class. We provide experimental
evidence that this indirect approach can often yield strong
results for many applications, generally outperforming direct
approaches such as regression trees and rivaling bagged regression
trees.
</ABSTRACT>
</TEXT>
</DOC>
```

Fig. 2.4. Un documento XML
Fuente: (Weiss, Indurkhya y Zhang, 2005)

La razón principal para identificar las piezas de un documento, es permitir la selección de aquellas partes que se usarán para generar características. Casi siempre se quiere usar la parte delimitada como <TEXTO>, pero también se puede incluir partes marcadas como <SUBJECT>, <HEADLINE> o similares. Además, para la clasificación de texto o la agrupación, se desea generar características desde una sección o tema si hay uno. Las partes del documento seleccionado pueden concatenarse en una sola cadena de caracteres o pueden mantenerse separadas si se quiere distinguir las características generadas del título, de las generadas a partir del cuerpo del documento, y quizás ponderarlas de manera diferente.

La principal ventaja de estandarizar los datos es que las herramientas de minería se pueden aplicar sin tener que considerar el origen del documento. Para recolectar información de un

documento, es irrelevante qué editor se usó para crearlo o cuál era el formato original. Las herramientas de software necesitan leer datos solo en un formato, y no en los muchos formatos diferentes en los que vinieron originalmente.

2.2.3. Tokenización

Partiendo de la colección de documentos en formato XML y que se encuentra preparado para examinar el texto no estructurado para identificar características útiles. El **primer paso** para manejar el texto es dividir el flujo de caracteres en palabras o, más precisamente, fichas. Esto es fundamental para un análisis posterior. (Weiss, Indurkhya y Zhang, 2005)

Sin identificar los tokens, es difícil imaginar extraer información de mayor nivel del documento. Cada token es una instancia de un tipo, por lo que la cantidad de tokens es mucho mayor que la cantidad de tipos. Como ejemplo, en la oración anterior hay dos tokens deletreados "the". Estos son dos ejemplos de un tipo "the", que aparece dos veces en la oración. Correctamente hablando, siempre debe referirse a la frecuencia de ocurrencia de un tipo, pero el uso suelto también habla de la frecuencia de un token. Romper una secuencia de caracteres en tokens es trivial para una persona familiarizada con la estructura del lenguaje. Sin embargo, un programa de computadora, siendo lingüísticamente desafiado, encontraría la tarea más complicada. La razón es que ciertos caracteres son a veces delimitadores de tokens y otras no, dependiendo de la aplicación. Los caracteres espacio, tabulación y nueva línea que se supone son siempre delimitadores y no se cuentan como tokens.

A menudo se les llama colectivamente espacio en blanco. Los signos () <>! ? " son siempre delimitadores y también pueden ser tokens. Los caracteres .,: - ' pueden ser o no delimitadores, dependiendo de su entorno. Un punto, una coma o dos puntos entre los números normalmente no se consideraría un delimitador, sino que formaría parte del número. Cualquier otra coma o dos puntos es un delimitador y puede ser un token. Un punto puede ser parte de una abreviatura (por ejemplo, si tiene una letra mayúscula en ambos lados). También puede ser parte de una abreviatura cuando está seguido de un espacio (por ejemplo, Dr.). Sin embargo, algunos de estos son realmente los extremos de las oraciones. A los

efectos de la tokenización, probablemente sea mejor tratar cualquier período ambiguo como un delimitador de palabras y también como un token.

El apóstrofe también tiene varios usos. Cuando lo preceden y lo siguen los no delimitadores, debe tratarse como parte del token actual (por ejemplo, it's o D'angelo). Cuando va seguido de un terminador inequívoco, puede ser una cita interna de cierre o indicar un posesivo (por ejemplo, Tess '). Un apóstrofe precedido por un terminador es inequívocamente el comienzo de una cita interna, por lo que es posible distinguir los dos casos haciendo un seguimiento de la apertura y el cierre de las comillas internas(Weiss, Indurkhya y Zhang, 2005)

Un guion es un terminador y un token si está precedido o seguido por otro guion. Un guion entre dos números puede ser un símbolo de resta o un separador (por ejemplo, 555-1212 como número de teléfono). Probablemente sea mejor tratar un guión no adyacente a otro guion como un terminador y un token, pero en algunas aplicaciones podría ser mejor tratar el guion, excepto en el caso del doble guion, simplemente como un carácter.

Para obtener las mejores características posibles, siempre se debe personalizar el tokenizador para el texto disponible; de lo contrario, es posible que se requiera un trabajo adicional después de obtener los tokens. El proceso de tokenización depende del idioma. La tesis, se enfoca en documentos en inglés. Aunque para otros idiomas, los principios generales serán los mismos, los detalles serán diferentes. Un ejemplo de pseudocodigo para tokenización se muestra en la Fig. 2.5.

```
Initialize:
   Set Stream to the input text string
   Set currentPosition to 0 and internalQuoteFlag to false
   Set delimiterSet to ' , . ; : ! ? ( ) <> + " \n \t space
   Set whiteSpace to \t \n space
Procedure getNextToken:
   L1: cursor := currentPosition; ch := charAt(cursor);
      If ch = endOfStream then return null; endif
   L2: while ch is not endOfStream nor instanceOf(delimiterSet) do
         increment cursor by 1; ch := charAt(cursor);
      endwhile
      If ch = endOfStream then
         If cursor = currentPosition then return null; endif
      endif
      If ch is whiteSpace then
         If currentPosition = cursor then
            increment currentPosition by 1 and goto L1;
         else
            Token := substring(Stream,currentPosition,cursor-1);
            currentPosition := cursor+1; return Token;
         endif
      endif
      If ch = ' then
         If charAt(cursor-1) = instanceOf(delimiterSet) then
            internalQuoteFlag := true; increment currentPosition by 1; goto L1;
         endif
         If charAt(cursor+1) != instanceof(delimiterSet) then
            increment cursor by 1; ch := charAt(cursor); goto L2;
         elseif internalQuoteFlag = true then
            Token := substring(Stream,currentPosition,cursor-1);
            internalQuoteFlag := false;
         else
            Token := substring(Stream,currentPosition,cursor);
         endif
         currentPosition := cursor+1; return Token;
      endif
      If cursor = currentPosition then
         Token := ch; currentPosition := cursor+1;
      else
         Token := substring(Stream,currentPosition,cursor-1);
         currentPosition := cursor;
      endif
      return Token;
endprocedure
```

Fig. 2.5. Algoritmo de Tokenizacion
Fuente: (Weiss, Indurkhya y Zhang, 2005)

2.2.4. Lematización o stemming

Una vez que una secuencia de caracteres se ha segmentado en una secuencia de tokens, el siguiente paso es convertir cada uno de los tokens en una forma estándar, un proceso que generalmente se conoce como derivación (*stemming*) o lematización. Para fines de clasificación de documentos, la derivación puede proporcionar un pequeño beneficio positivo en algunos casos. Tenga en cuenta que uno de los efectos de la derivación es reducir el número de **tipos** distintos en un cuerpo de texto y aumentar la frecuencia de aparición de algunos **tipos** individuales. Por ejemplo, en la oración anterior, las dos instancias de "tipos" se reducirían en "tipo" raíz y se contarían como instancias de ese tipo, junto con las instancias de los tokens "tipo" y "tipografía". Para algoritmos de clasificación que tienen en cuenta la frecuencia, esto a veces puede hacer una diferencia. En otros escenarios, el procesamiento adicional puede no proporcionar ganancias significativas. (Weiss, Indurkhya y Zhang, 2005)

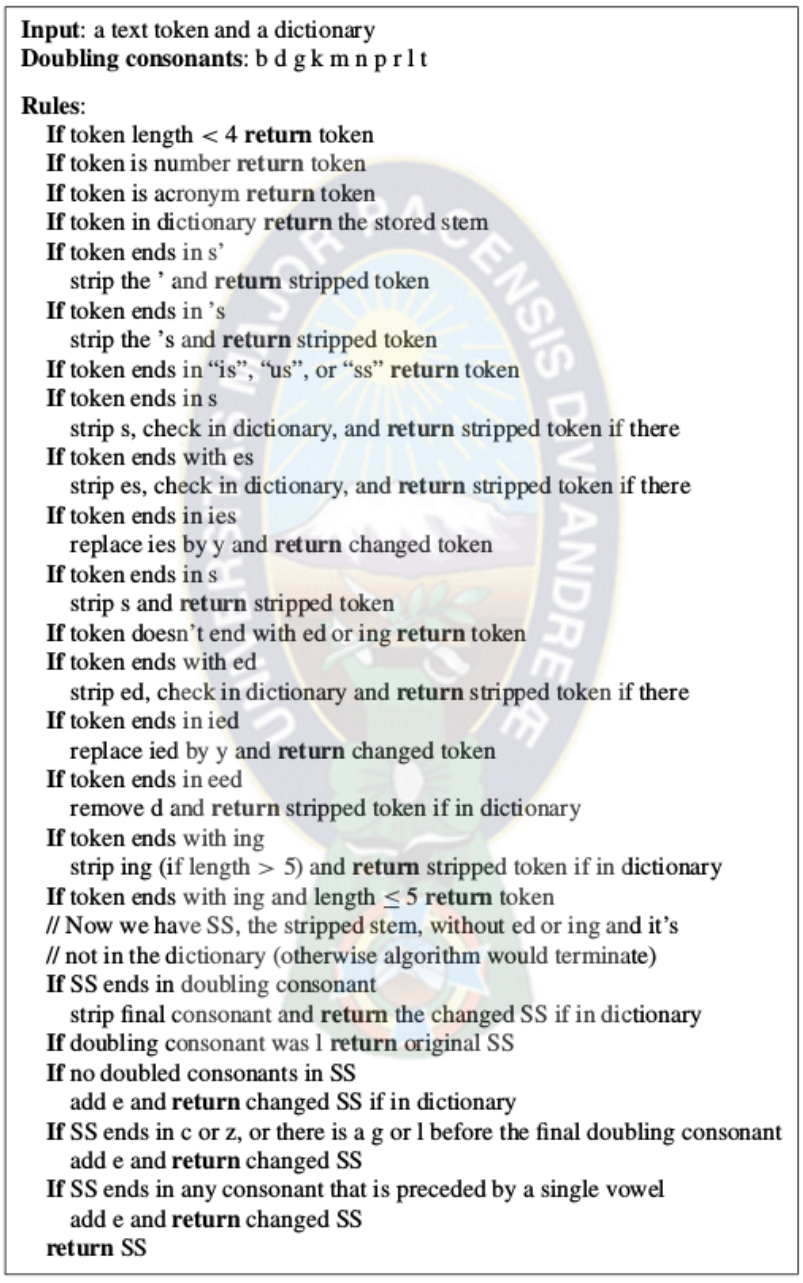
a) **Stemming flexible**

En inglés, como en muchos otros idiomas, las palabras aparecen en el texto en más de una forma. Cualquier hablante nativo de inglés aceptará que los nombres "book" y "books" son dos formas de la misma palabra. A menudo, pero no siempre, es ventajoso eliminar este tipo de variación antes del procesamiento adicional (es decir, para normalizar ambas palabras en el único formulario "book"). Cuando la normalización se limita a regularizar variantes gramaticales como singular, plural, presente y pasado, el proceso se denomina "derivación flexible". En terminología lingüística, esto se denomina "análisis morfológico". Para un idioma como el inglés, con muchas formas de palabras irregulares y ortografía no intuitiva, es más difícil. No hay una regla simple, por ejemplo, para unir "seek" y "sought". De manera similar, la raíz para "rebelled" es "rebel", pero la raíz para "belled" es "bell". En otros idiomas, inflexiones puede tomar la forma de infijo, como, en el alemán "angeben" (declarar), para el cual el participio pasado es "angegeben". (Weiss, Indurkhya y Zhang, 2005)

En inglés, un algoritmo para la derivación flexible debe basarse en reglas y en diccionarios. Cualquier algoritmo de derivación para inglés que opere solo en tokens, sin más información gramatical como parte de la oración, cometerá algunos errores debido a la ambigüedad. Por ejemplo, ¿es "bored" el adjetivo como "he is bored" o es el tiempo pasado del verbo "bore"?

Además, ¿el verbo "bore" es una instancia del verbo "bore a hole", o es el tiempo pasado del verbo "bear"? En ausencia de un proceso de desambiguación a menudo complicado, un algoritmo de derivación probablemente debería elegir la opción más frecuente.

El pseudocódigo para un derivador flexible simplificado para el inglés se da en la figura 2.6.



Input: a text token and a dictionary
Doubling consonants: b d g k m n p r l t

Rules:
If token length < 4 **return** token
If token is number **return** token
If token is acronym **return** token
If token in dictionary **return** the stored stem
If token ends in s'
strip the ' and **return** stripped token
If token ends in 's
strip the 's and **return** stripped token
If token ends in "is", "us", or "ss" **return** token
If token ends in s
strip s, check in dictionary, and **return** stripped token if there
If token ends with es
strip es, check in dictionary, and **return** stripped token if there
If token ends in ies
replace ies by y and **return** changed token
If token ends in s
strip s and **return** stripped token
If token doesn't end with ed or ing **return** token
If token ends with ed
strip ed, check in dictionary and **return** stripped token if there
If token ends in ied
replace ied by y and **return** changed token
If token ends in eed
remove d and **return** stripped token if in dictionary
If token ends with ing
strip ing (if length > 5) and **return** stripped token if in dictionary
If token ends with ing and length ≤ 5 **return** token
// Now we have SS, the stripped stem, without ed or ing and it's
// not in the dictionary (otherwise algorithm would terminate)
If SS ends in doubling consonant
strip final consonant and **return** the changed SS if in dictionary
If doubling consonant was l **return** original SS
If no doubled consonants in SS
add e and **return** changed SS if in dictionary
If SS ends in c or z, or there is a g or l before the final doubling consonant
add e and **return** changed SS
If SS ends in any consonant that is preceded by a single vowel
add e and **return** changed SS
return SS

Fig. 2.6. Algoritmo de Stemming flexible
Fuente (Weiss, Indurkhya y Zhang, 2005)

El algoritmo consiste en reglas que se aplican en secuencia hasta que una de ellas se cumpla, también las referencias frecuentes a un diccionario, generalmente referido como un diccionario raíz.

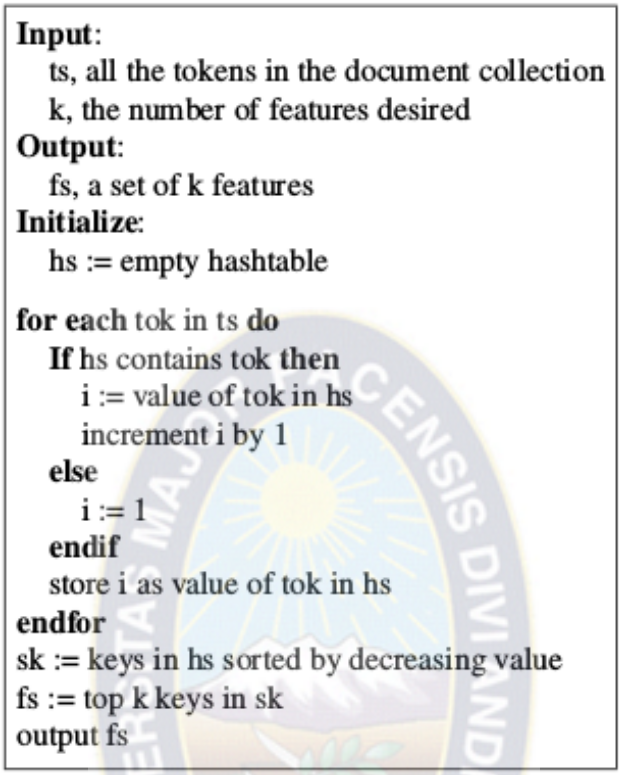
b) Derivación (Stemming) a una Raíz

La normalización agresiva es ventajosa al menos para algunas aplicaciones de procesamiento de texto. La intención de estos stemmers es llegar a una forma de raíz sin prefijos ni sufijos de derivación o flexión. Por ejemplo, la "desnormalización" se reduce a la "norma". El resultado final de tal ramificación agresiva es reducir el número de tipos en una colección de textos de manera muy drástica, lo que hace que las estadísticas de distribución sean más confiables. Además, las palabras con el mismo significado básico se fusionan, de modo que un concepto como "apply" tiene solo un tallo, aunque en el texto puede haber "reapplied", "applications". La utilidad de la derivación depende mucho de la aplicación. (Weiss, Indurkhya y Zhang, 2005)

2.2.5. Generación vectorial para predicción

Las características de los documentos son los tokens o palabras que contienen, sin un análisis profundo del contenido lingüístico de los documentos, se puede elegir describir cada documento por características que representan los tokens más frecuentes. Al conjunto colectivo de características se suele llamar diccionario. Los tokens o palabras en el diccionario forman la base para crear una hoja de cálculo de datos numéricos correspondientes a la colección de documentos. Cada fila es un documento, y cada columna representa una característica. Por lo tanto, una celda en la hoja de cálculo es una medida de una característica (correspondiente a la columna) para un documento (que corresponde a una fila). Existen métodos predictivos que aprenden de dichos datos. Pero se explorará primero los diversos matices de este modelo de datos y cómo podría influir en los métodos de aprendizaje. En el modelo básico de dichos datos, simplemente se verifica la presencia o ausencia de palabras, y las entradas de celda son entradas binarias que corresponden a un documento y una palabra. El diccionario de palabras cubre todas las posibilidades y corresponde al número de columnas en la hoja de cálculo. Todas las celdas tendrán ceros o unos, dependiendo de, si las palabras se encontraron en el documento.

La figura 2.7 describe este proceso de categorización de documentos



```
Input:
   ts, all the tokens in the document collection
   k, the number of features desired
Output:
   fs, a set of k features
Initialize:
   hs := empty hashtable

for each tok in ts do
   If hs contains tok then
      i := value of tok in hs
      increment i by 1
   else
      i := 1
   endif
   store i as value of tok in hs
endfor
sk := keys in hs sorted by decreasing value
fs := top k keys in sk
output fs
```

Fig. 2.7. Funciones de Generación de tokens
Fuente (Weiss, Indurkhya y Zhang, 2005)

Un método de aprendizaje puede manejar las altas dimensiones de un diccionario global, el modelo simple de datos puede ser muy efectivo. Buscar palabras es simple porque no se revisa cada palabra en el diccionario. Se construye una tabla hash[17] de las palabras del diccionario y se ve si las palabras del documento están en la tabla hash. Grandes muestras de documentos digitales están disponibles. Esto da la confianza de que muchas variaciones y combinaciones de palabras aparecerán en la muestra. Esta expectativa argumenta que para pasar menos tiempo computacional preparando los datos para buscar palabras similares o eliminar palabras débiles, se debe dejar que la velocidad de la computadora encuentre su propio camino durante el proceso de aprendizaje. Debido a las circunstancias, es posible que

17 Hash: Es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con longitud fija, independiente de la longitud de los datos de entrada el valor hash de salida tendrá siempre la misma longitud.

se desee trabajar con un diccionario reducido, la muestra puede ser relativamente pequeña, o puede ser un diccionario grande difícil de manejar, en tales casos se podría intentar reducir el tamaño del diccionario mediante varias transformaciones, dependiendo del método de aprendizaje, estas transformaciones pueden mejorar el rendimiento predictivo.

La Tabla 2.1 Algunas de las transformaciones que se pueden realizar.

Local dictionary	Diccionario Local
Stoprwords	Palabras vacias
Frequent words	Palabras frecuentes
Feature Selection	Selección de caracteres
Token Reduction: stemming, synonyms	Reduccion en tokens: Derivacion, sinonimos

Tabla 2.1. Técnicas de Reducción de diccionario
Fuente (Weiss, Indurkhya y Zhang, 2005)

Si la meta es la predicción, se necesita una columna adicional para la respuesta correcta (o clase) para cada documento. Al preparar los datos para un método de aprendizaje, esta información estará disponible en las etiquetas de los documentos. Las etiquetas son generalmente binarias, y la clase más pequeña es casi siempre la de interés. En lugar de generar un diccionario global para ambas clases, se puede considerar solo palabras encontradas en la clase que se está tratando de predecir. Si esta clase es mucho más pequeña que la clase negativa, lo que es típico, dicho diccionario local será más reducido que el diccionario global. Otra reducción en el tamaño del diccionario es compilar una lista de palabras vacías (Stopwords) y eliminarlas del diccionario. Estas son palabras que casi nunca tienen ninguna capacidad de predicción, como los artículos "a" y "el" y pronombres tales como él y ellos. Estas palabras comunes pueden descartarse antes del proceso de generación de características, pero es efectivo generar primero las características, aplicar todas las demás transformaciones y, en la última etapa, rechazar las que corresponden a los stopwords.

La información de frecuencia sobre el recuento de palabras es bastante útil para reducir el tamaño del diccionario y, a veces, puede mejorar el rendimiento predictivo de algunos métodos. Las palabras frecuentes suelen ser palabras vacías y se pueden eliminar. Las palabras que se utilizan en menor frecuencia suelen ser las palabras importantes que deben

permanecer en un diccionario local. Las palabras muy raras son a menudo errores tipográficos y también pueden descartarse. Para algunos métodos de aprendizaje, un diccionario local de las palabras más frecuentes, quizás menos de 200, puede ser sorprendentemente efectivo. (Weiss, Indurkhya y Zhang, 2005)

Una alternativa a la generación de diccionarios locales es generar un diccionario global a partir de todos los documentos de la colección. Las rutinas de selección de características especiales intentarán seleccionar un subconjunto de palabras que parezcan tener el mayor potencial de predicción. Estos métodos de selección a menudo son complicados e independientes del método de predicción. Se puede probar cualquiera de los métodos de selección de características que se han utilizado en configuraciones alternativas de estadística o de aprendizaje automático. Muchos de estos han sido desarrollados para variables reales y sin énfasis en atributos discretos o binarios. Muchos de los métodos de predicción se han ajustado para que el texto trate con diccionarios más grandes en lugar de generar diccionarios más pequeños repetidamente. Si se determinan muchas clases, se debe repetir la generación de un diccionario más pequeño para cada problema de predicción. Por ejemplo, si se tiene 100 temas para categorizar, entonces existe 100 problemas de predicción binarios para resolver. Las opciones son 100 diccionarios pequeños o uno grande. Normalmente, los vectores implicados en un modelo de hoja de cálculo también se regenerarán para que correspondan al pequeño diccionario. (Weiss, Indurkhya y Zhang, 2005)

En general, un diccionario pequeño, tiene mayor inteligencia en su composición siendo necesario para capturar la mayor cantidad de palabras y la mejor de las palabras. El uso de tokens y stemming son ejemplos de procedimientos útiles para componer diccionarios pequeños. Todos estos pasos sirven para una mejor manejabilidad del aprendizaje y mayor precisión, el aprendizaje puede avanzar más rápidamente con diccionarios más pequeños.

Una vez que se ha determinado el conjunto de características, la colección de documentos se puede convertir a formato de hoja de cálculo. Cada columna en la hoja de cálculo corresponde a una función. Para la interpretabilidad, se necesita mantener la lista de características para traducir del número de columna al nombre de la función. Se necesita la colección de documentos para referirse a los documentos originales de las filas.

La Figura 2.8 un ejemplo de cómo se puede convertir a formato de hoja de cálculo para las características binarias.

```
Input:
   fs, a set of k features
   dc, a collection of n documents
Output: ss, a spreadsheet with n rows and k columns
Initialize: i := 1

for each document d in dc, do
   j := 1
   for each feature f in fs, do
      m := number of occurrences of f in d
      if (m > 0) then ss(row=i, col=j) := 1;
      else ss(row=i, col=j) := 0;
      endif
      increment j by 1
   endfor
   increment i by 1
endfor
output ss
```

Fig2.8. Conversión de Documentos a una hoja de calculo
Fuente (Weiss, Indurkhya y Zhang, 2005)

Para lograr la mejor precisión predictiva, se consideran transformaciones adicionales a partir de esta representación. La Tabla 2.2 enumera tres transformaciones diferentes que pueden mejorar las capacidades predictivas.

	Conversión de documentos a una hoja de
Word pairs, collocations	calculo
Frequencies	frecuencias
TF-IDF	TF_IDF

Tabla 2.2. Transformación de características del diccionario
Fuente (Weiss, Indurkhya y Zhang, 2005)

En lugar de ceros o unos como entradas en las celdas de la hoja de cálculo, se podría usar la frecuencia real de ocurrencia de la palabra. Si una palabra aparece diez veces en un documento, este conteo se ingresará en la celda. Se tiene toda la información de una representación binaria, y a la vez se tiene información adicional para contrastar con otros

documentos. Para algunos métodos de aprendizaje, el recuento da un resultado ligeramente mejor. También puede conducir a soluciones más compactas porque incluye el mismo espacio de solución que el modelo de datos binarios, aunque la información de frecuencia adicional puede ofrecer una solución más simple. Esto es especialmente cierto en algunos métodos de aprendizaje cuyas soluciones usan solo un pequeño subconjunto de las palabras del diccionario. En general, las frecuencias son útiles en la predicción, pero agregan complejidad a las soluciones propuestas. Una tarea que funciona bastante bien es tener un sistema de tres valores para las entradas de celda: uno o cero como en la representación binaria, con la posibilidad adicional de un 2. Tal esquema captura gran parte del valor agregado de la información de frecuencia sin agregar mucha complejidad al modelo. Otra variante implica valores de puesta a cero por debajo de un cierto umbral en los fundamentos plausibles de que los tokens deben tener una frecuencia mínima antes de ser considerados de algún uso. Esto puede reducir la complejidad de la hoja de cálculo de manera significativa y una necesidad para algunos algoritmos de minería de datos. Además de los umbrales simples, hay una variedad de métodos más sofisticados para reducir la complejidad de la hoja de cálculo, como el uso de chi-cuadrado, información mutua, odds ratio y otros. La información mutua puede ser útil al considerar las funciones de palabras múltiples.

La tabla 2.3 enumera las tres posibilidades, donde se asignan todas las ocurrencias de más de dos veces en un valor máximo de 2.

0- word did not occur	0- La palabra no ocurrió
1- word ocurred once	1- La palabra ocurrió una vez
2- word ocurred 2 or more times	2- Palabra ocurrida dos o más veces

Tabla 2.3. Las frecuencias de Umbral a tres valores
Fuente (Weiss, Indurkhya y Zhang, 2005)

El siguiente paso después de contar la frecuencia de una palabra en un documento es modificar el conteo por la importancia percibida de esa palabra. La conocida formulación ***TF-IDF*** se ha utilizado para calcular ponderaciones o puntajes para palabras, los valores serán números positivos para capturar la presencia o ausencia de la palabra en un documento. En el (2.1), se ve que el peso **TF-IDF** asignado a la palabra "j" es el término frecuencia (es

decir, el conteo de palabras) modificado por un factor de escala para la importancia de la palabra. El factor de escala se *denomina frecuencia inversa del documento*, que se da en (2.2). Simplemente verifica la cantidad de documentos que contienen la palabra "j" (es decir, df (j)) y revierte la escala. Cuando aparece una palabra en muchos documentos, se considera que no es importante y la escala se reduce, cerca de cero. Cuando la palabra es relativamente única y aparece en pocos documentos, el factor de escala se amplía hacia arriba debido a su importancia. (Weiss, Indurkhya y Zhang, 2005)

$$tf - idf = tf(j) * idf(j). \quad (2.1)$$

$$idf(j) = \log(\frac{N}{df(j)}) \quad (2.2)$$

Existen alternativas de la formula básica **TF-IDF**, pero la noción general es la misma. El resultado neto de este proceso es un puntaje positivo que reemplaza la frecuencia simple o la entrada binaria verdadera o falsa en la celda de la hoja de cálculo. Si el puntaje es grande, el valor de puntaje esperado tiene mayor importancia para el método de aprendizaje, aunque esta transformación es solo una pequeña modificación del modelo binario original, pierde la claridad y la simplicidad de la presentación anterior.

Otra variante es ponderar los tokens de diferentes partes del documento de forma diferente. Por ejemplo, las palabras de un documento podrían recibir un peso adicional. Una variante efectiva es generar conjuntos separados de características para las categorías (para cada categoría, el conjunto de características se deriva solo de los tokens de documentos de esa categoría) y luego agrupar todos los conjuntos de características.

Todos estos modelos de datos son variaciones del modelo binario básico para la presencia o ausencia de palabras. No existe una transformación de datos que se considere sea la mejor, la experiencia ha demostrado que la mejor precisión de predicción depende de que una de estas variaciones coincida con un método de aprendizaje específico.

Se transforma el modelo de datos de palabras en una presentación críptica, los datos siguen siendo entradas en las celdas de la hoja de cálculo, pero su valor es inteligible. Algunas de estas transformaciones son técnicas para reducir la duplicación y las dimensiones. Otros se

basan en una cuidadosa experimentación empírica que respalda su valor en el aumento de las capacidades predictivas. (Weiss, Indurkhya y Zhang, 2005)

Los datos completan una hoja de cálculo, y se espera que la mayoría de las celdas sean cero. La mayoría de los documentos contienen un pequeño subconjunto de las palabras del diccionario. En el caso de la clasificación de texto, un corpus de texto puede tener miles de tipos de palabras. Sin embargo, cada documento individual tiene solo unos cientos de tokens únicos. Por lo tanto, en la en la hoja de cálculo, casi todas las entradas para ese documento serán cero. En lugar de almacenar todos los ceros, es mejor representar la hoja de cálculo como un conjunto de vectores dispersos, donde una fila se representa por una lista de pares, un elemento del par es un número de columna y el otro elemento es la característica distinta de valor cero. Al no almacenar los ceros, los ahorros en la memoria pueden ser inmensos. Los programas de procesamiento se pueden adaptar fácilmente para manejar este formato.

La Figura 2.9 proporciona un ejemplo simple de cómo una hoja de cálculo se transforma en vectores dispersos.

Spreadsheet				Sparse Vectors
0	15	0	3	(2,15) (4,3)
12	0	0	0	(1,12)
8	0	5	2	(1,8) (3,5) (4,2)

Fig. 2.9. Hoja de cálculo a vectores dispersos
Fuente (Weiss, Indurkhya y Zhang, 2005)

a) Etiquetas para las respuestas correctas

Para la predicción, se debe agregar una columna adicional a la hoja de cálculo. Esta última columna de la hoja de cálculo, que contiene la etiqueta, no es diferente de las demás. Es uno o cero que indica que la respuesta correcta es verdadera o falsa. Cualquier respuesta que se pueda medir como verdadera o falsa es aceptable. Podría tratarse de un tema o categoría, o un artículo. Siempre que las respuestas estén etiquetadas correctamente en relación con el concepto, el formato es aceptable. Eso no significa que el problema puede resolverse fácilmente. En el formato de vector disperso, las etiquetas se anexan a cada vector por separado como uno (clase positiva) o cero (clase negativa). Fuente (Feldman, 2006)

2.3. Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural (NLP) es parte de Inteligencia Artificial y Lingüística, dedicada a hacer que las computadoras entiendan las declaraciones o palabras escritas en los lenguajes humanos. El procesamiento del lenguaje natural surgió para facilitar el trabajo del usuario y satisfacer el deseo de comunicarse con la computadora en lenguaje natural. Dado que todos los usuarios pueden no estar bien instruidos en el lenguaje específico de la máquina, **NLP** brinda servicios a aquellos usuarios que no tienen suficiente tiempo para aprender nuevos idiomas u obtener perfección en él. (Kao y Poteet, 2007)

Un lenguaje se puede definir como un conjunto de reglas o un conjunto de símbolos. Los símbolos se combinan y utilizan, para transmitir información. El procesamiento del lenguaje natural básicamente se puede clasificar en dos partes, la comprensión del lenguaje natural y la generación del lenguaje natural, que desarrolla la tarea de comprender y generar el texto.

Para hablar de procesamiento del lenguaje natural, se debe hablar en primer lugar de lenguaje natural. El lenguaje natural se entiende como el lenguaje hablado y escrito con el propósito de comunicación entre una o varias personas, es más directo para expresar lo que se quiere comunicar, el lenguaje natural es menos susceptible a malas interpretaciones por el empleo de términos con un solo significado. La comunicación es importante en el lenguaje natural debido a que este proceso involucra la transmisión y recepción de información.

El lenguaje natural es complejo, pero se demostró que las expresiones del lenguaje humano están organizadas a través de un conjunto de reglas. Todas las expresiones tienen una clara organización: las palabras en una oración se asocian para describir objetos y acciones, posiblemente complejas. El objetivo de un analizador sintáctico es precisamente descubrir estas asociaciones entre palabras, lo que se conoce como estructura sintáctica. Un analizador sintáctico es un programa que toma como entrada una oración y trata de descubrir la estructura sintáctica que explica las relaciones entre las palabras de esa oración. Los analizadores buscan la estructura correcta dentro de un conjunto de análisis posibles, este conjunto esta usualmente definido por una gramática. El modelo de lenguaje en el cual se

basa el analizador sintáctico decide cuáles son los componentes sintácticos de las oraciones y como éstos están relacionados. (Kao y Poteet, 2007)

El primer avance obtenido en el PLN se dio en el área del acceso a las bases de datos con el sistema lunar (1973) construidos en la NASA por William Woods.

2.4. El PLN[18] y la Recuperación de la Información

PLN es el lenguaje entre hombre y máquina, con el objetivo de que la maquina le responda satisfactoriamente a la necesidad manifestada, existen diversos modelos asociados a esta recuperación de información que constituyen una herramienta que permite diferenciar una consulta previa y una seria de respuestas para dicha consulta.

En el funcionamiento de estos modelos se han implementado múltiples tipos de búsquedas con la razón fundamental, que los motores de búsqueda o la maquina pueda modelar las preguntas a lenguaje de máquina.

Los modelos de recuperación de información tienen una alta importancia debido a los meta-buscadores que existen en Internet, por esta razón es necesario entender cómo se estructuran internamente estos modelos de recuperación de información. (Kao, Poteet, 2007)

A continuación, los diferentes modelos de recuperación de información:

2.4.1. Modelo de Recuperación Booleano

Constituye el primer modelo teórico, empleado para establecer el subconjunto de documentos relevantes, en relación a una consulta específica, de entre todos los que configuran la colección (ya se trate del fondo de una biblioteca o de todas las páginas disponibles en la web). Al mismo tiempo es, uno de los modelos más sencillos tanto desde un punto de vista teórico como práctico, al basarse en la teoría de conjuntos y en el álgebra de Boole, al ser fácil de diseñar e implementar en la práctica. (Kao, Poteet, 2007)

18 PLN: Procesamiento del Lenguaje Natural

El procesamiento automatizado de un documento textual comienza con la extracción de los términos de indización[19] (indexación), es decir, los términos que van a ser utilizados para describir el contenido del documento. Consiste en considerar todas las palabras aisladas que aparecen en el texto como los términos de indización. También se eliminan las denominadas palabras vacías, entre las que suelen figurar los números, las preposiciones, conjunciones, verbos ser, haber y estar, aunque la consideración o no de estos procesos añadidos (como la inclusión de una lista de palabras vacías) no influyen en absoluto sobre los principios teóricos del modelo. Una vez extraídas las palabras del texto, se ordenan por orden alfabético y se guardan en un denominado fichero inverso (fichero diccionario), junto con la referencia del documento de donde proceden (normalmente un número de documento asignado previamente por el sistema). Si se repite este proceso con todos los documentos de la colección, finalmente se obtiene un fichero inverso que almacena los siguientes datos: (Martínez, 2008)

- En primer lugar, los términos de indización (las palabras) que aparecen en toda la colección (ya sean los propios textos, los resúmenes de los textos del fondo y/o los títulos).
- En segundo lugar, cada uno de dichos términos (palabras) incorpora una lista con los números de los documentos en los que aparece.

En este proceso no se menciona la frecuencia de aparición de cada termino en cada documento. De ahí que el modelo booleano clásico sea denominado modelo binario, pues de la consulta del fichero inverso únicamente puedo saber si un determinado término de indización está presente (en cuyo caso se simbolizará por el número 1) o está ausente (en tal caso se simbolizará por el número 0) en cada uno de los documentos de la colección.
De manera que el fichero inverso (en concreto el fichero diccionario) puede representarse por una tabla cuyos datos básicos son los siguientes: a continuación, la tabla 2.4

19 Indización: Ordenación de una serie de datos o informaciones de acuerdo con un criterio común a todos ellos, para facilitar su consulta y análisis.

	D1	D2	…..	Dn
T1	1	0	…..	1
T2	0	1	…..	0
…..	…..	…..	…..	…..
Tt	0	1	…..	1

Tabla 2.4. Datos mínimos del fichero inverso
Fuente (Martinez, 2008)

Donde T1, T2, …, Tn son los términos de indización empleados en la colección; D1, D2, …, Dn son los documentos que componen la colección; y donde el "1" significa que el término correspondiente aparece en ese documento concreto, mientras que el "0" significa que el término no aparece en dicho documento. Ello implica que no se tiene en cuenta la frecuencia de aparición de los términos en los documentos: tanto si aparece veinte veces como si aparece una sola vez, en todos los casos ese término en dicho documento se representará mediante un "1", reservándose el "0" para cuando no aparezca. En el modelo Booleano únicamente se juega con dos posibilidades: la aparición y la no aparición de los descriptores en los documentos.

Al observar la tabla, se puede deducir de ella las dos representaciones empleadas al manejar el modelo binario, cada término de indización se representa por la lista de documentos en los que aparece, lo que implica la observación de la tabla por filas:

$$T1 = \{D1, \dots, Dn\}$$
$$T2 = \{D2, \dots\}$$
$$\dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots$$
$$Tt = \{D2, \dots, Dn\}$$

Por otra parte, cada documento se representa por la lista de ceros y unos correspondientes a los términos de indización que contiene, lo que implica la observación de la tabla por columnas:

$$D1 = \{1,0, \dots .,0\}$$
$$D2 = \{0,1, \dots .,1\}$$
$$D_n = \{1,0, \dots .,1\}$$

En el modelo binario todo documento se representa mediante una serie ordenada de ceros y unos, tantos como términos de descripción se empleen en la colección: el primer número

corresponderá siempre a T1, el segundo dígito corresponderá a T2, y así sucesivamente hasta llegar a T_t, siendo t el número de descriptores distintos que representan el contenido de esa colección. (Martínez, 2008)

2.4.2. Modelo de Recuperación Vectorial

TF-IDF (Frecuencia de termino - frecuencia Inversa del documento) es un método de ponderación utilizado en la recuperación de información y principalmente en la minería de texto. Esta medida estadística hace posible evaluar la importancia de un término en un documento, en relación con una colección o corpus. La frecuencia bruta de un término es el número de apariciones de este término en un documento específico, esta frecuencia también se denomina frecuencia de término. (Salton y Yang, 1973)

La frecuencia inversa del documento es una medida de importancia de una colección completa. En el modelo **TF-IDF**, aumenta el peso a menos términos repetidos, considerados menos discriminatorios. Consiste en calcular la base-10 logaritmo de la proporción de documentos en el corpus que contiene el término.

TF-IDF (Term Frequency – Inverse Document Frequency)

Uno de los métodos empleados para la creación de listas de palabras clave para los sistemas de búsqueda de información es **TF-IDF** (Term Frequency – Inverse Document Frequency), el cual es la unión de dos métricas, la primera de ellas es la de frecuencia del término, term frequency o **TF** y la segunda que es la frecuencia inversa de los documentos, inverse document frequency o **IDF**.

El método de **TF-IDF** genera listas de palabras clave con una calificación o peso que indica qué tan relevante es la palabra con respecto al documento seleccionado y al corpus en general. Estas listas permiten calificar a los documentos del corpus con base en estas palabras clave, es decir, si las palabras claves tienen un gran peso, entonces el documento está relacionado con ellas que uno con las mismas palabras clave, pero con menor peso. Cuando se ingrese una consulta, los documentos que tengan las palabras de esa consulta con mayor peso serán los que muestre el modelo de búsqueda.

TF Term Frequency

La primera medida, "**TF,** es un sistema de pesos basado en la idea de que construcciones (palabras, frases, grupos de palabras) que frecuentemente ocurren en el texto de documentos tienen alguna relación con el contenido de los textos" (Salton y McGill, 1986). El cálculo de estos pesos se lleva a cabo calculando la frecuencia relativa[20] de cada una de las construcciones (llamados términos) en cada uno de los documentos a analizar; esto se puede representar por medio de la siguiente fórmula: (Salton y Yang, 1973)

$$TF_{ij} = f_i^j$$

Donde f es el número de ocurrencias del termino i en el documento j.

El termino frecuencia, es una medida de cuantas veces los términos están presentes en el diccionario. Se define el termino frecuencia como una función de conteo, a continuación, se muestra la fórmula empleada:

$$tf(t,d) = \sum_{x \in d} fr(x,t)$$

Donde $fr(x,t)$ es una función simple definida como:

$$fr(x,t) = \begin{cases} 1, & si\ x = t \\ 0, & en\ otros\ casos \end{cases}$$

Lo que $tf(t,d)$ devuelve es cuantas veces el termino está presente en el documento d.

A continuación, se crea el vector de documento que está representado de la siguiente manera:

$$\overrightarrow{V_{d_n}} = (tf(t_1,d_n), tf(t_2,d_n), tf(t_3,d_n), \dots, tf(t_n,d_n))$$

Cada dimensión del vector de documento está representada por el término del vocabulario, donde $tf(t_1,d_2)$ representa la frecuencia de término del término 1 o (t_1) en el documento (d_2).

A continuación, la representación matricial de la siguiente manera $|D| * Fform$, donde $|D|$ es la cardinalidad del espacio del documento y F la cantidad de características, un ejemplo de matriz, conocida como matrices dispersas, es el siguiente:

20 La frecuencia relativa es el número de ocurrencias de un caso en determinado evento. De diferencia de la frecuencia absoluta que es la frecuencia relativa entre la suma de las ocurrencias de todos los casos en el evento.

$$M_{|D|*F} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 0 \end{bmatrix}$$

IDF Inverse Document Frequency

La segunda métrica llamada Inverse Document Frequency o **IDF**, cuenta el número de documentos de la colección en donde se busca, que contienen o están indizadas por el término en cuestión [Robertson, 2004], en otras palabras, es saber el número de documentos de un corpus en donde las construcciones de palabras se encuentran. El nombre inicial de **IDF** fue term specificity y su fórmula fue la siguiente (Spärck-Jones, 1972):

$$term\ specificity = f(N) - f(n) + 1$$

Donde la función $f(x) = y\ tal\ que\ 2^{y-1} < x \leq 2^y$ en la figura 2.10. se puede observar los valores que podría tomar y; N es el número de documentos que existen en el corpús y n es el número de documentos donde el termino analizado se encuentra.

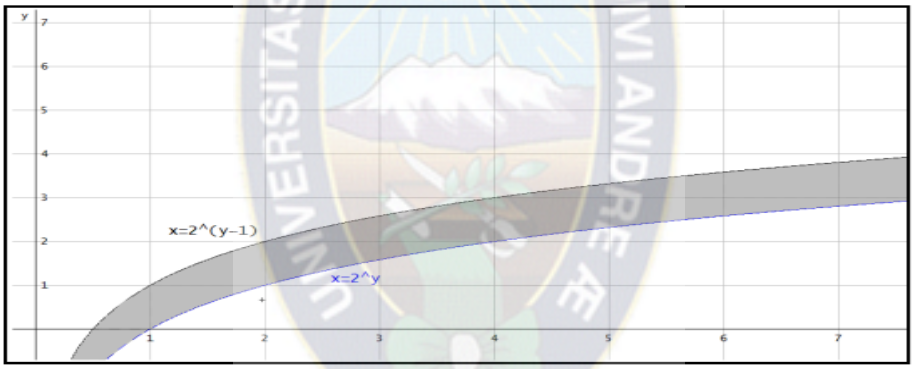


Fig. 2.10. Grafica de los valores de la función $f(x) = y\ tal\ que\ 2^{y-1} < x \leq 2^y$
Fuente (Spärck-Jones, 1972)

Posteriormente la fórmula de term specificity fue adecuada por Robertson (1972) debido a que la fórmula $f(x)$ se podía aproximar a $f(x) = log_2 x$,esta aproximación es la función inversa de $x = 2^y$, y el 1 que se encontraba en la fórmula original era para evitar valores iguales a 0 en valores de n muy cercanos al valor de N, por lo tanto se podía prescindir de él. De igual forma le fue cambiado el nombre a inverse document frequency **IDF** donde el logaritmo no tenía que ser forzosamente base 2, esto debido a la siguiente propiedad de los logaritmos: $log_b a = \frac{log_c a}{log_c b}$

Donde a es el numero al cual se el calculara el logaritmo; b es la base del logaritmo original y c es la base de un logaritmo distinto a b. En el caso del **IDF**, se cambió el logaritmo de base 2 a base 10 ya que es uno de los más usados.

La fórmula empleada en la actualidad para el cálculo de la métrica Inverse Document Frequency es la siguiente: (Spärck-Jones, 1972)

$$IDF_{(t)} = log \frac{|D|}{1 + |\{d: t \in d\}|}$$

Donde $|\{d: t \in d\}|$es el número de documentos; $t\ el\ termino$ aparece, cuando la función de termino satisface $tf(t, d) \neq 0$ solo se agrega 1 en la fórmula para evitar la división entre cero.

La creación del sistema de pesos basado en **IDF** se fundó en la teoría de Spärk-Jones que indicaba que los términos con alta frecuencia pueden ser útiles para aumentar la cobertura, pero las correspondencias entre la consulta y los términos del documento que ocurren raramente en una colección de documentos deberían ser tomadas como más importantes que aquellas que ocurren frecuentemente (Salton y Yang, 1973).

A partir de estos dos sistemas de pesos, se desarrolló el **TF-IDF**, el cual trabaja determinando la frecuencia relativa de las palabras en un documento específico comparado con la proporción inversa de esa palabra en todo el corpus. Su fórmula es la siguiente:(Salton y Yang, 1973)

$$TF - IDF_{ij} = f_i^j * log \frac{N}{n_i}$$

Donde f_i^j es la frecuencia del termino i en el documento j; N es el número total de documentos que conforman el corpus y n_i es el número de documentos donde se encuentra el termino i.

La fórmula para el tf-idf se puede representar también de la siguiente manera:

$$tf - idf = tf(t, d) \times idf(t)$$

Esta fórmula tiene una consecuencia importante: un alto peso del cálculo de tf-idf que se alcanza cuando se tiene una frecuencia de alto plazo (tf) en el documento dado (parámetro local) y una baja frecuencia del término en toda la colección (parámetro global).

En la Tabla 2.5 se muestra un ejemplo de los pesos calculados por el método de **TF-IDF** para una serie de términos en tres documentos:

Término	Documento 1			Documento 2			Documento 3		
	TF	IDF	TF-IDF	TF	IDF	TF-IDF	TF	IDF	TF-IDF
la	10	$\log\frac{3}{3}=0$	0	12	$\log\frac{3}{3}=0$	0	9	$\log\frac{3}{3}=0$	0
geometría	5	$\log\frac{3}{2}=0.17$	0.85	7	$\log\frac{3}{2}=0.17$	1.19	0	$\log\frac{3}{2}=0.17$	0
analítica	5	$\log\frac{3}{1}=0.47$	2.35	0	$\log\frac{3}{1}=0.47$	0	0	$\log\frac{3}{1}=0.47$	0
descriptiva	0	$\log\frac{3}{1}=0.47$	0	6	$\log\frac{3}{1}=0.47$	2.82	0	$\log\frac{3}{1}=0.47$	0
casa	0	$\log\frac{3}{1}=0.47$	0	0	$\log\frac{3}{1}=0.47$	0	8	$\log\frac{3}{1}=0.47$	3.76
de	11	$\log\frac{3}{3}=0$	0	8	$\log\frac{3}{3}=0$	0	10	$\log\frac{3}{3}=0$	0
José	0	$\log\frac{3}{2}=0.17$	0	2	$\log\frac{3}{2}=0.17$	0.34	5	$\log\frac{3}{2}=0.17$	0.85

Tabla 2.5. Ejemplo de Aplicación de TF-IDF
Fuente (Salton y Yang, 1973)

Como se puede observar, el método de **TF-IDF** permite la discriminación de palabras frecuentes, como lo son las palabras funcionales, estas no suelen quedar en las listas de palabras clave que se emplean en los sistemas de búsqueda de información. Pero además califica con pesos altos las palabras con alto valor de importancia para cada uno de los documentos analizados.

Hasta la fecha se han realizado diversas variaciones al método de **TF-IDF** y se han empleado métodos de normalización. De igual manera el uso de **TF-IDF** ha pasado a otras áreas de PLN.

Normalización de la longitud del documento

Una de las modificaciones que han sido agregadas al método de **TF-IDF** es el empleo de un factor de normalización que permita equilibrar casos en los que se empleen documentos de distintos tamaños, un documento muy extenso tiene una ventaja alta sobre los documentos pequeños a la hora de recuperar información, debido a que el tamaño del documento es un parámetro que afecta el cálculo de los pesos. Esta ventaja se observa cuando el usuario hace una consulta al sistema de recuperación de información, el cual traerá los documentos que contengan los términos de la consulta con mayor peso. (Spärck-Jones, 1972)

Existen dos razones principales por las cuales es necesario emplear la normalización:

Frecuencias de términos más altas: Los documentos grandes usualmente emplean los mismos términos repetidamente. Como resultado, los factores de frecuencia de los términos pueden ser grandes para documentos largos, aumentando la contribución promedio de sus términos a la similitud de la consulta de documentos.

Más términos: Los documentos largos también tienen una gran cantidad de términos diferentes. Esto aumenta el número de coincidencias entre la consulta y un documento largo, aumentando la similitud de la consulta de documentos, y los casos de recuperar documentos largos en vez de documentos cortos.

La normalización, a grandes rasgos, permite tratar de la misma manera a todos los documentos sin importar su longitud.

Existen diversos métodos para llevar a cabo la normalización, algunos de ellos atacan las dos razones principales vistas anteriormente, algunos otros métodos solamente una de ellas.

Normalización de vectores

La normalización de vectores consiste en obtener otro vector unitario de la misma dirección y sentido de un vector ya dado. Se calcula el vector frecuencia de termino $\underset{v_{d_n}}{\longrightarrow}$.

Para realizar la normalización de un vector, se emplea el cálculo de vector unitario del vector y se denotan mediante la notación "hat": $\hat{v}$ la definición del vector unitario $\hat{v}$ de un vector es $\underset{v}{\rightarrow}$ es:

$$\hat{v} = \frac{\vec{v}}{\left\| \vec{v} \right\|_p}$$

El vector unitario cuyo modulo es la unidad y tiene por misión indicar la dirección y sentido de un determinado vector, a dicho vector se le llama también versor.

Donde $\hat{v}$ es el vector unitario, o el vector normalizado, el vector va a ser normalizado y el $\left\| \vec{v} \right\|_p$ es la norma (magnitud, longitud) del vector $\vec{v}$ en el L_P espacio.

El vector unitario en realidad no es más que la normalización del vector, es un vector cuya longitud es 1.

En la figura 2.11. se muestra de manera gráfica el proceso de normalización.

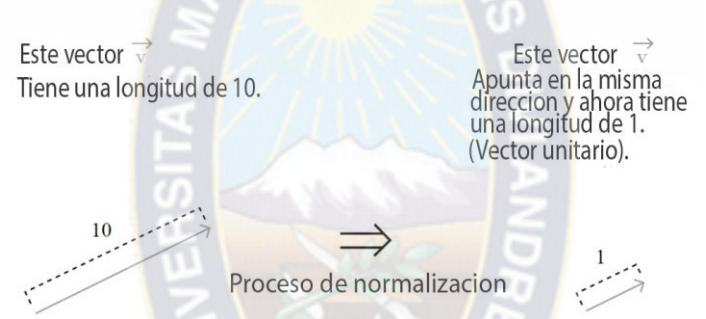


Fig. 2.11. Proceso de normalización
Fuente (Perone, 2015)

Norma Euclidiana

La norma euclidiana define la longitud de un vector desde el punto de vista de la geometría Euclideana.

La norma euclidiana en matemáticas, algebra, geometría y específicamente en análisis real, análisis complejo y geometría analítica, se trata de una función no negativa usada en diversos contextos para calcular la distancia entre dos puntos, primero en el plano y luego en el espacio. También sirve para definir la distancia entre dos puntos en otros tipos de espacios de tres o más dimensiones, y para hallar la longitud de un segmento definido por dos puntos de una recta, del plano o de espacios de mayor dimensión. **(**Bronshtein y Semendiaev, 1973)

Sus bases se encuentran en la aplicación del teorema de Pitágoras sobre triángulos rectángulos, donde la distancia Euclideana viene a ser por lo general la longitud de la hipotenusa del triángulo recto conformado por cada punto y los vectores proyectados sobre los ejes directores al nivel de la hipotenusa. Como se puede ver en la siguiente figura 2.12.

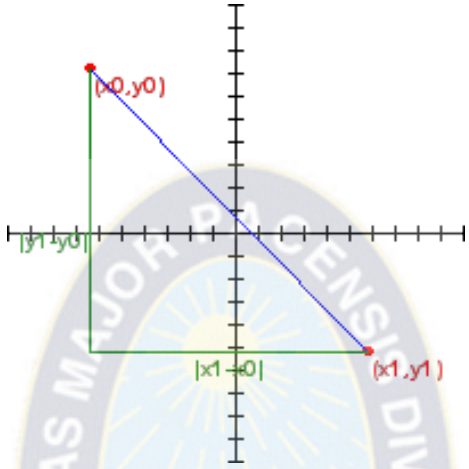


Fig. 2.12 Diagrama de los vectores, sobre los ejes directores a nivel de la hipotenusa Fuente (Bronshtein y Semendiaev, 1973)

Donde la distancia misma es la longitud de la hipotenusa (marcada en Azul) y sus catetos (trazados en verde) que serían las proyecciones sobre los ejes coordenados de dicha recta, trasladados hasta los puntos en cuestión (marcados en rojo), donde el teorema se expresa:

$$|AB|^2 = (x_B - x_A)^2 + (y_B - y_A)^2$$

Que luego queda en la conocida formula de la distancia Euclideana.

Definiciones

En el plano cartesiano sean los puntos $A = (x_A; y_A)$ $B = (x_B; y_B)$ se define la distancia Euclideana entre dichos puntos por:

$$d(A,B) = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

En el espacio, sean los puntos $A = (x_A; y_A; z_A)$ y $B(x_B; y_B; z_B)$ se define la distancia Euclideana mediante la expresión:

$$d(A,B) = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}$$

Y de manera más general en un espacio de N $dimensiones$ la distancia euclideana entre dos puntos $A = (a_1; a_2; \dots; a_n)$ y $B = (b_1; b_2; \dots; b_n)$ se ajusta a :

$$d(A,B) = \sqrt{\sum_{i=1}^{N}(b_1 - a_1)^2} = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + (b_n - a_n)^2}$$

De manera general la métrica Euclideana entre dos puntos se define como la longitud del segmento de recta que une a dichos puntos. (Bronshtein y Semendiaev, 1973)

Propiedades de la norma Euclideana

La distancia o métrica Euclideana (en su caso general de N dimensiones) satisface las condiciones necesarias para ser catalogada como una métrica en términos estrictamente matemáticos que serían:

1. $d(A,B) \geq 0$
2. $d(A,B) = d(B,A) llamada\ propiedad\ simetrica.$
3. $d(A,A) = 0$
4. $d(A,B) \leq d(A,C) + d(C,B)\ llamada\ propiedad\ de\ la\ desigualdad\ triangular.$
5. $Si\ d(A,B) = 0\ entonces\ A = B$

2.5. Redes Neuronales

Las redes neuronales son un modelo computacional basado en un gran conjunto de unidades neuronales simples[21] de forma aproximadamente análoga al comportamiento observado en los axones de las neuronas en los cerebros biológicos. Cada una de las neuronas simples va a tener una forma similar al siguiente diagrama: (Russell y Norving, 2004)

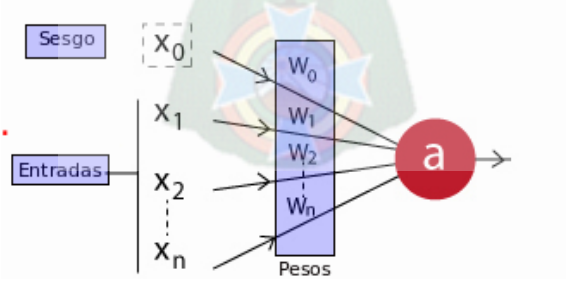


Fig. 2.13. Diseño de Neurona simple
Fuente (Russell y Norving, 2004)

21 Neuronas artificiales.

En donde sus componentes son:

- $x_1, x_2, \dots, x_n$: los datos de entrada en la neurona, los cuales también puede ser que sean producto de la salida de otra neurona de la red.
- x_0: La unidad de sesgo; un valor constante que se le suma a la entrada de la función de activación de la neurona. Tiene valor 1, que permite cambiar la función de activación hacia la derecha o izquierda, otorgándole más flexibilidad para aprender a la neurona.
- $w_0, w_1, w_2, \dots, w_n$: los pesos relativos de cada entrada. Tener en cuenta que incluso la unidad de sesgo tiene un peso.
- a: la salida de la neurona, que va a ser calculada de la siguiente manera:

$$a = f(\sum_{i=0}^{n} w_i \cdot x_i)$$

Aquí $f\ es\ la\ funcion\ de\ activacion$ de la neurona, esta función es la que le otorga tanta flexibilidad a las redes neuronales y le permite estimar complejas relaciones no lineales en los datos. Puede ser tanto una función lineal, como función logística, hiperbólica. Cada unidad neuronal está conectada con muchas otras y los enlaces entre ellas pueden incrementar o inhibir al estado de activación de las neuronas adyacentes. Estos sistemas aprenden y se forman así mismos, en lugar de ser programados de forma explícita y sobresalen en aéreas donde la detección de soluciones o características es difícil de expresar con la programación convencional. (Russell & Norving, 2004)

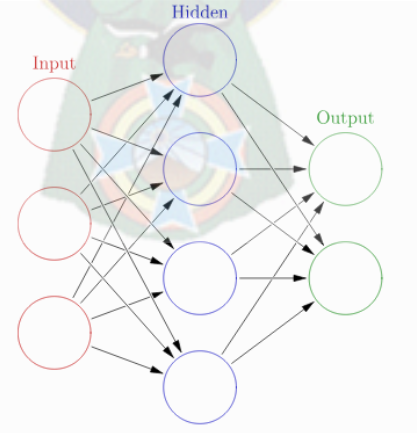


Fig. 2.14 Estructura de la RNA
Fuente (Russell y Norving, 2004)

2.5.1. Propagación hacia atrás

La propagación hacia atrás o backpropagation es un algoritmo que funciona mediante la determinación de la pérdida o error en la salida y luego propagándolo de nuevo hacia atrás en la red. De esta forma los pesos se van actualizando para minimizar el error resultante de cada neurona. Este algoritmo es lo que les permite a las redes neuronales aprender. (Goodfellow, 2016)

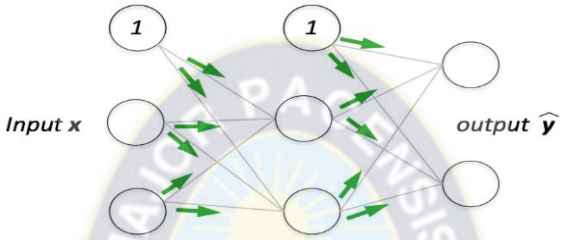


Fig. 2.15. Propagación hacia atrás

Fuente (Goodfellow, 2016)

2.5.2. Aprendizaje supervisado

Con esta técnica de aprendizaje el entrenamiento consiste en presentarle a la red repetitivamente patrones de estímulos de entrada pertenecientes a un juego de ensayo. El juego de ensayo está formado por parejas "patrón de estímulos - respuesta correcta" y debe de ser elegido cuidadosamente. Cada pareja se denomina hecho. En el juego de ensayo debe estar representada equilibradamente toda la información que la red necesite aprender.

Al realizar el entrenamiento la respuesta que da la red a cada patrón se compara con la respuesta correcta ante dicho patrón y, en virtud de esa comparación, se reajustan los pesos sinápticos. El reajuste de los pesos sinápticos está orientado a que, ante el patrón de entrada, la red se acerque cada vez más a la respuesta correcta.

Cuando ante un patrón de entrada la red de neuronas ya responde correctamente, se pasa al siguiente patrón del juego de ensayo y se procede de la misma manera.

Cuando se termina con el último patrón del juego de ensayo, se tiene que volver a empezar con el primero, ya que los pesos se han seguido modificando. (Gestal, 2010)

2.5.3. Funcionamiento de la Red Neuronal Multicapa

La red aprende algo simple en la capa inicial de la jerarquía y luego envía esta información a la siguiente capa. La siguiente capa toma esta información simple, lo combina en algo que es un poco más complejo, y lo pasa a la tercera capa. Este proceso continúa de forma tal que cada capa de la jerarquía construye algo más complejo de la entrada que recibió de la capa anterior. De esta forma, la red irá aprendiendo por medio de la exposición a los datos de ejemplo.

La especificación de lo que cada capa hace a la entrada que recibe es almacenada en los pesos de la capa, que, en esencia, no son más que números. Utilizando terminología más técnica podemos decir que la transformación de datos que se produce en la capa es parametrizada por sus pesos. Para que la red aprenda debemos encontrar los pesos de todas las capas de forma tal que la red realice un mapeo perfecto entre los ejemplos de entrada con sus respectivas salidas objetivo. Pero el problema reside en que una red neuronal puede tener millones de parámetros, por lo que encontrar el valor correcto de todos ellos puede ser una tarea realmente difícil, especialmente si la modificación del valor de uno de ellos afecta a todos los demás. (Russell y Norving, 2004)

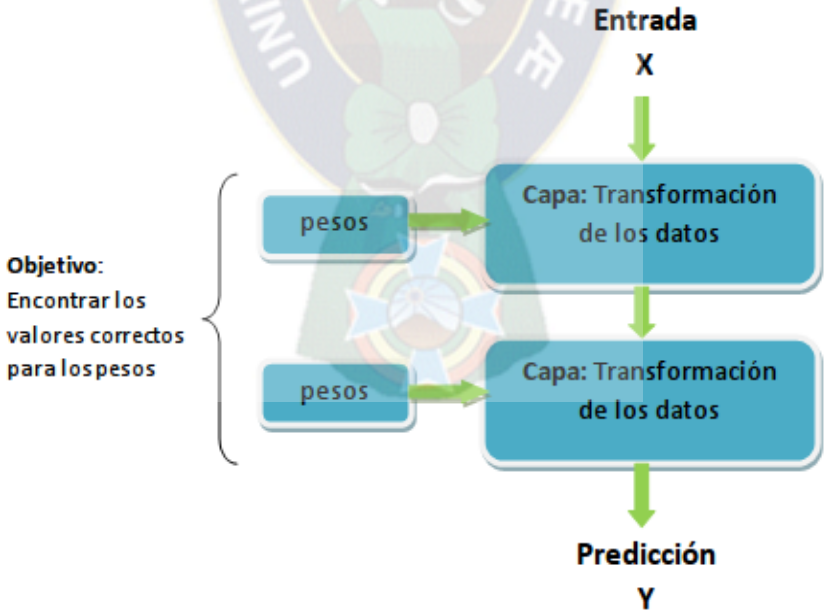


Fig. 2.16. Diseo de funcionamiento de la RNA
Fuente (Russell y Norving, 2004)

Para poder controlar algo, en primer lugar, se debe poder observar. En este sentido, para controlar la salida de la red neuronal, se debería poder medir cuán lejos está la salida que se obtuvo de la que se esperaba obtener. Este es el trabajo de la función de pérdida de la red. Esta función toma las predicciones que realiza el modelo y los valores objetivos (lo que realmente se espera que la red produzca), y calcula cuán lejos se está de ese valor, de esta manera, se puede capturar que tan bien está funcionando el modelo para el ejemplo especificado. El truco fundamental de la red neuronal multicapa es utilizar el valor que nos devuelve esta función de pérdida para retroalimentar la red y ajustar los pesos en la dirección que vayan reduciendo la pérdida del modelo para cada ejemplo. Este ajuste, es el trabajo del optimizador, el cuál implementa la propagación hacia atrás.

Fig. 2.17. Función pedida de la red
Fuente (Russell & Norving, 2004)

2.5.4. Arquitectura de la Red Neuronal

La estructura de datos fundamental de una red neuronal está vagamente inspirada en el cerebro humano. Cada una de las células cerebrales (neuronas) está conectada a muchas otras neuronas por sinapsis. A medida que se experimenta e interactúa con el mundo, el cerebro crea nuevas conexiones, refuerza algunas conexiones y debilita a los demás. De esta forma, en el cerebro se desarrollan ciertas regiones que se especializan en el procesamiento de determinadas entradas. Así se va a tener un área especializada en la visión, otra que se especializa en la audición, otra para el lenguaje. De forma similar, dependiendo del tipo de entradas con las que se trabaje, van a existir distintas arquitecturas de redes neuronales que mejor se adaptan para procesar esa información.

Redes Neuronales Pre alimentadas

Las redes neuronales prealimentadas fueron los primeros que se desarrollaron y son el modelo más sencillo, en estas redes la información se mueve en una sola dirección: hacia adelante. Los principales exponentes de este tipo de arquitectura son el perceptrón y el perceptrón multicapa, se utilizan en problemas de clasificación simples. (Goodfellow, 2016)

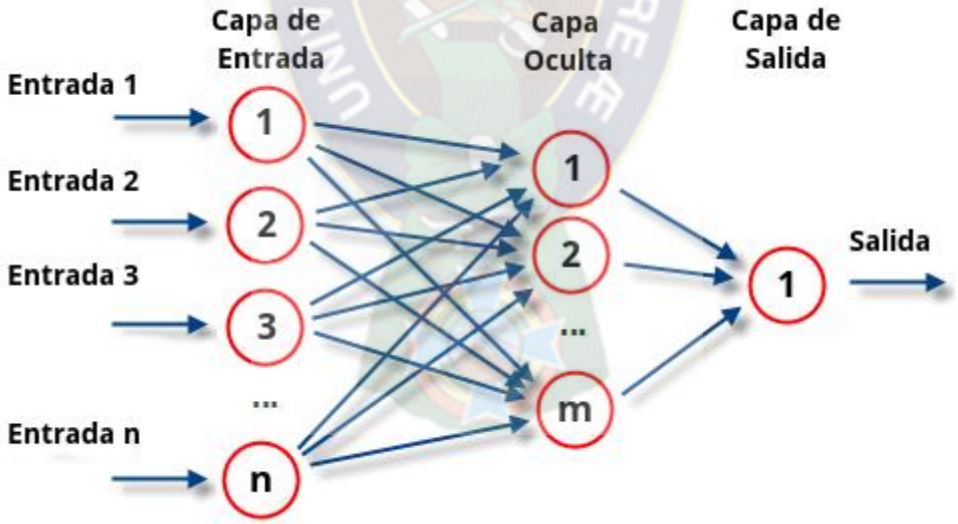


Fig. 2.18. Arquitectura del Perceptrón multicapa
Fuente (Russell y Norving, 2004)

2.6. Tensorflow

TensorFlow es una biblioteca de software de código abierto para el cálculo numérico de alto rendimiento. Su arquitectura flexible permite una fácil implementación de computación en una variedad de plataformas (CPU, GPU) y desde escritorios hasta clústeres de servidores y dispositivos móviles y periféricos. Desarrollado originalmente por investigadores e ingenieros del equipo Google Brain dentro de la organización AI de Google, cuenta con un sólido respaldo para el aprendizaje automático y el aprendizaje en profundidad, y el núcleo de computación numérica flexible se utiliza en muchos otros dominios científicos.

El sistema es flexible y puede utilizarse para expresar una amplia variedad de algoritmos, incluidos algoritmos de entrenamiento e inferencia para modelos de redes neuronales profundas, se utilizó para realizar investigaciones y para implementar sistemas de aprendizaje automático en el área de la ciencia de la computación y otros campos, incluidos el reconocimiento de voz, visión artificial robótica, recuperación de información, procesamiento del lenguaje natural, extracción de información geográfica y descubrimiento de fármacos computacionales. (Tensorflow, 2018)

2.7. Google Cloud Machine Learning Engine

Aprendizaje automático administrado y escalable

Google Cloud Machine Learning Engine es un servicio administrado que te permite generar con facilidad modelos de aprendizaje automático que funcionan con cualquier tipo de datos, sin importar el tamaño.

Se crea el modelo de trabajo con TensorFlow, crea modelos de cualquier tamaño con su infraestructura administrada y escalable. El modelo entrenado con Google Cloud ML Engine está disponible inmediatamente para ser usada con su plataforma de predicción global que admite TB de datos. El servicio está integrado con Google Cloud Dataflow para el procesamiento previo, lo que permite acceder a datos de Google Cloud Storage y Cloud BigQuery. (Google Cloud, 2018)

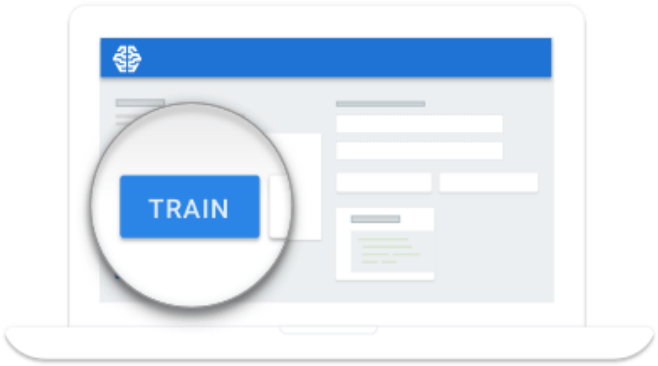


Fig. 2.19. Google Cloud Machine Learning Engine
Fuente (Google cloud, 2018)

2.7.1. Características de Cloud Machine Learning engine

- Integración: los servicios de google están diseñados para operar juntos. Funciona con Cloud Dataflow para el procesamiento de funciones, cloud storage para el almacenamiento de datos y Cloud Datalab para la creación de modelos.
- Hypertune: Genera modelos con rendimiento mejor y más rápido mediante el ajuste automático de hiperparametros, en lugar de dedicar una gran cantidad de horas a detectar manualmente los valores que funcionen con el modelo en cuestión.
- Servicio administrado: este automatiza el aprovisionamiento y la supervisión de todos los recursos.
- Servicio escalable: Genera modelos de datos de cualquier tamaño y tipo con una infraestructura de entrenamiento administrada, distribuida y compatible con CPU. Para acelerar el desarrollo de modelos, se puede entrenar con una gran cantidad de nodos o ejecutar varios experimentos en paralelo.
- Experiencia de cuaderno para el programador: crea y analiza modelos con la experiencia de desarrollo en cuadernos de Jupyter integrada con Cloud Datalab.
- Modelos prácticos: Usa el SDK de TensorFlow de código abierto para entrenar modelos en forma local con conjuntos de datos de muestra y usa Google Cloud Plattform para el entrenamiento a escala, los modelos creados con Cloud ML engine se pueden descargar para ejecutarlos en equipos locales o para la integración en dispositivos móviles.

2.8. Python

Python es un lenguaje de programación eficaz y fácil de aprender. Cuenta con estructuras de datos eficientes, de alto nivel con un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python, su tipado dinámico[22] y multiplataforma, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting[23], desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables. Python es un lenguaje de programación multiplataforma, esto significa que se puede variar estilos, programación orientada a objetos, programación imperativa, y programación funcional. (Alabanese, Merler, Jurman y Visintainer, 2008)

2.9. Scikit-Learn

Scikit-Learn, es un paquete que proporciona versiones eficientes de una gran cantidad de algoritmos comunes. Scikit-Learn se caracteriza por una API limpia, uniforme y optimizada, así como por una documentación en línea muy útil y completa. Un beneficio de esta uniformidad es que una vez que comprende el uso básico y la sintaxis de Scikit-Learn para un tipo de modelo, cambiar a un nuevo modelo o algoritmo es muy sencillo. (Dubois, 2007) Scitit-learn aprovecha el entorno de Python enriquecido para proporcionar implementaciones de última generación de muchos algoritmos de aprendizaje de máquinas conocidos. Esto responde a la creciente necesidad de análisis de datos estadísticos por parte de personas que no son especialistas en el software y las industrias web, así como en campos ajenos a la

22 Tipado dinamico: hace referencia a si una misma variable puede tomar valores de distinto tipo en distintos momentos.

23 En informática, un script, archivo de órdenes, archivo de procesamiento por lotes o, guion, es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Los guiones son casi siempre interpretados, pero no todo programa interpretado es considerado un guion. El uso habitual de los guiones es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario. Por este uso es frecuente que los intérpretes de órdenes sean a la vez intérpretes de este tipo de programas.

informática, como la biología o la física. Scikit-learn difiere de otras herramientas de aprendizaje en Python por varias razones:

i) Se distribuye bajo la licencia de BSD.

ii) Incorpora código compilado para la eficiencia, a diferencia de MDP y pybrain.

iii) depende solo de numpy y scipy para facilitar easdistribution, a diferencia de pymvpa que tiene dependencias opcionales como R y shogun,

iv) Se centra en la programación imperativa, a diferencia de pybrain que usa un flujo de datos marco de referencia. Si bien el paquete está escrito principalmente en Python, incorpora las bibliotecas C ++ LibSVM y LibLinear que proporcionan implementaciones de referencia de SVM y modelos lineales generalizados con licencias compatibles. Los paquetes binarios están disponibles en un amplio conjunto de plataformas, incluidas Windows y las plataformasPOSIX. Además, su licencia liberal, se ha distribuido ampliamente como parte de las principales distribuciones de software libre como Ubuntu, Debian, Mandriva, NetBSD y Mac, y en distribuciones comerciales como la "Enthought Python Distribution".

2.10. NLTK

NLTK es una plataforma líder para construir programas de Python para trabajar con datos de lenguaje humano. Proporciona interfaces fáciles de usar a más de 50 recursos corporales y léxicos como WordNet, junto con un conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico, envoltorios para bibliotecas de PNL de fuerza industrial, y un foro de discusión activo. Posee una colección de paquetes y objetos Python muy adaptados para tareas de PLN.

Gracias a una guía práctica que presenta los fundamentos de programación junto con los temas de lingüística computacional, además de una completa documentación API, NLTK es adecuado para lingüistas, ingenieros, estudiantes, educadores, investigadores y usuarios de la industria por igual. NLTK está disponible para Windows, Mac OS X y Linux. Lo mejor de todo es que NLTK es un proyecto gratuito, de código abierto, impulsado por la comunidad.

NLTK ha sido llamado "una herramienta maravillosa para enseñar y trabajar en lingüística computacional usando Python" y "una biblioteca increíble para jugar con el lenguaje natural". El procesamiento del lenguaje natural con Python proporciona una introducción práctica a la programación para el procesamiento del lenguaje. Escrito por los creadores de NLTK, guía al lector a través de los fundamentos de escribir programas Python, trabajar con corpus, categorizar texto, analizar estructura lingüística y más. (NLTK, 2017)

2.11. Metodología

2.11.1. Clasificación de textos de Vandana Korde y C. Namrata Mahender

Los estudios de minería de textos adquieren importancia debido a la disponibilidad de número creciente de documentos electrónicos de una variedad de fuentes. Que incluye información no estructurada y semi estructurada. El objetivo principal de la minería de textos es permitir a los usuarios extraer información de los recursos textuales y se ocupa de las operaciones como, recuperación y clasificación (supervisada, no supervisada y semi supervisada). (Vandana y Namrata, 2012)

Las técnicas de procesamiento de lenguaje natural (NLP), minería de datos y aprendizaje automático trabajan en conjunto para clasificar y descubrir automáticamente los patrones de los diferentes tipos de documentos.

La clasificación de textos (TC) es una parte importante de la minería de textos, para construir sistemas TC automáticos mediante técnicas de ingeniería del conocimiento, es decir, definir manualmente un conjunto de reglas lógicas que conviertan el conocimiento experto en un conjunto de categorías para la clasificación. La tarea consiste en determinar un modelo de clasificación que pueda asignar la clase correcta a un nuevo documento *D* del dominio. La clasificación de texto tiene dos tipos de etiqueta única y multi-etiqueta. El documento de etiqueta única pertenece a una sola clase y en la etiqueta múltiple el documento puede pertenecer a más de una clase. (Vandana y Namrata, 2012)

Proceso de Clasificación de textos

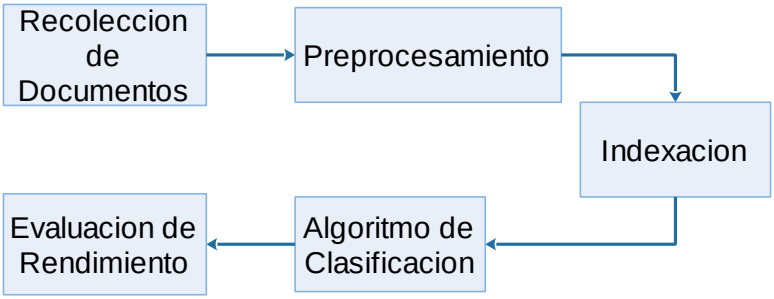


Fig. 2.20. Proceso de la clasificación de textos
Fuente (Vandana y Namrata, 2012)

1) **Recolección de Documentos**

Este es el primer paso del proceso de clasificación en el que se recopila los diferentes tipos (formato) de documentos como html, pdf, doc, xml, csv, contenido web, etc.

2) **Pre procesamiento**

El primer paso del pre-procesamiento se utiliza para presentar los documentos de texto en formato de palabra clara. Los documentos preparados para el siguiente paso en la clasificación de texto están representados por una gran cantidad de características. Comúnmente los pasos tomados son:

Tokenización: un documento se trata como una cadena y luego se divide en una lista de tokens. (Vandana y Namrata, 2012)

Eliminación de palabras vacías: palabras como "the", "a", "and", etc. son frecuentes, por lo que son consideradas palabras insignificantes y deben eliminarse.

Stemming: Aplicación del algoritmo de derivación que convierte formas de palabras diferentes en formas canónicas similares. Este paso es el proceso de combinar tokens con su forma raíz. Ejemplo: conexión para conectarse, computación para computar

3) **Indexación**

La representación de documentos es una de las técnicas de preprocesamiento que se utiliza para reducir la complejidad de los documentos y facilitar su manejo. El documento debe transformarse de la versión de texto completo a un vector de documento. La representación de documento quizás más comúnmente utilizada es llamado modelo de espacio vectorial, los

documentos están representados por vectores de palabras. Por lo general, uno tiene una colección de documentos que está representada por el documento palabra por palabra Matriz. El esquema de representación de BoW / VSM tiene sus propias limitaciones. Algunos de ellos son: alta dimensionalidad de la representación, pérdida de correlación con palabras adyacentes y pérdida de la relación semántica que existe entre los términos de un documento. Para superar estos problemas, los métodos de ponderación de términos se usan para asignar ponderaciones apropiadas al término como se muestra en la matriz. (Vandana y Namrata, 2012)

Donde cada entrada representa la ocurrencia de la palabra en el documento, donde w_{tn} es el peso de la palabra i en el documento n, ya que cada palabra no aparece normalmente en cada documento. Hay varias formas de determinar el peso w_{i1}, como que la ponderación booleana, la ponderación de frecuencia de palabras, tf-idf, entropía, etc. Pero la principal desventaja de este modelo es que da como resultado genera una matriz dispersa enorme, lo que plantea un problema de alta dimensionalidad. Otros métodos diversos se presentan como:

a) Una representación de ontología para un documento para mantener la relación semántica entre los términos en un documento.
b) Una secuencia de símbolos (byte, un carácter o una palabra) llamada N-Grams, que se extraen de una cadena larga en un documento, es muy difícil decidir el número de gramos que se considerarán para la representación efectiva del documento.
c) Términos de palabras múltiples como componentes vectoriales. Pero este método requiere sofisticados algoritmos de extracción automática de términos para extraer los términos automáticamente de un documento
d) Indización semántica latente (LSI) que conserva las características representativas de un documento. El LSI conserva las características más representativas en lugar de características discriminatorias.
e) Locality Preserving Indexing (LPI), descubre la estructura semántica local de un documento. Pero no es eficiente en tiempo y memoria

f) Una nueva representación para modelar los documentos web. Las etiquetas HTML se usan para construir la representación del documento web.

4) Algoritmo de Clasificación

La clasificación automática de documentos en categorías predefinidas se considera de la siguiente manera; los documentos se pueden clasificar de tres formas, métodos no supervisados, supervisados y semi supervisados. Desde hace algunos años, la tarea de la clasificación automática de textos se ha estudiado ampliamente y parece que se avanza rápidamente en esta área, incluidos los enfoques de aprendizaje automático como el clasificador bayesiano, el Árbol de decisiones, el vecino más cercano K (KNN), Máquinas de vectores de soporte (SVM), Redes neuronales, Rocchio's. (Vandana y Namrata, 2012)

5) Evaluación de rendimiento

Esta es la última etapa de la clasificación de texto, en la que las evaluaciones de los clasificadores de texto se realizan generalmente de forma experimental, en lugar de analíticamente. La evaluación experimental de los clasificadores, en lugar de concentrarse en cuestiones de Eficiencia, por lo general trata de evaluar la efectividad de un clasificador, es decir, su capacidad de tomar las decisiones correctas de categorización. Un tema importante de la categorización de texto es cómo medir el rendimiento de los clasificadores. Se utilizan muchas medidas, como Precisión y Exhaustividad; precipitación, error, exactitud, etc. se dan a continuación. La precisión ci (Pri) se define como la probabilidad de que, si un documento aleatorio dx se clasifica en ci, esta decisión es correcta. Análogamente, Exhaustividad ci (Rei) se define como la condición de que, si un documento aleatorio dx debe clasificarse en ci, se tome esta decisión. (Vandana y Namrata, 2012)

$TP_i - Cantidad\ de\ documentyos\ asignados\ correctamente\ a\ esta\ categoria.$

$FN - Numero\ de\ documentos\ asignados\ incorrectamente\ a\ esta\ categoria.$

$FP_i - cantidad\ de\ documentos\ incorrectamente\ rechazados\ asignados\ a\ la\ categoría.$

$TN_i - cantidad\ de\ documentos\ rechazados\ correctamente\ asignados\ a\ esta\ categoría.$

$$Fallout = \frac{FN_i}{FN_i} + TN_i$$

$$Error = \frac{FN_i + FP_i}{TP_i + FN_i + FP_i + TN_i}$$

$$Accuracy = TP_i + TN_i$$

Para obtener estimaciones de precisión y recuperación en relación con todo el conjunto de categorías, se pueden adoptar dos métodos diferentes Micro promediación y Macro promediación. Algunas otras medidas también se usan como Punto de equilibrio, Medición F, Interpolación.

2.11.2. Introducción a las Redes Neuronales Artificiales de Marcos Gestal Pose

Desde la primera mitad del siglo XX se desarrollaron modelos computacionales que intentaron emular el comportamiento del cerebro humano, todos usan una estructura en red en la cual los nodos o neuronas son procesos numéricos que involucran estados de otros nodos según sus uniones. Las Redes Neuronales Artificiales (RNA) se popularizaron debido a su fácil uso e implementación y la habilidad para aproximar cualquier función matemática. El funcionamiento básico de las Redes Neuronales abarca dos grandes etapas. La primera comprende todas las fases de creación y desarrollo de las misma, y posteriormente la fase de funcionamiento real o fase de ejecución, durante la cual la RNA ya es operativa, y tanto su estructura interna como sus valores de los pesos de las conexiones no vuelven a ser modificados, durante esta etapa se usa la red neuronal como cualquier otro programa y es cuando se utiliza de forma efectiva para resolver problemas para los que ha sido diseñada. (Gestal, 2010)

1) **Fase creación y desarrollo:** esta fase comprende todas las fases necesarias para el desarrollo de una RNA, y comprende las siguientes etapas.
    - **Diseño de la arquitectura**, también denominado diseño de la topología, se determina el número de neuronas que tendrá la red, se determina cuantas entradas y salidas tendrá la red.
2) **Entrenamiento de la Red:** en esta etapa se debe proceder a entrenar la red para que aprenda el comportamiento que debe de tener, para que aprenda a dar respuestas adecuada a la configuración de estímulos o patrones de entrada que se le presenten.

Tales sistemas están orientados hacia diferentes operaciones se clasifican en dos tipos: Aprendizaje supervisado y no supervisado.

3) **Fase de Validación o test**: Esta fase también denominada fase de ejecución, es la etapa durante la que se le pedirá a la red que responda a estímulos diferentes a los presentados durante la fase de entrenamiento y por lo tanto gracias a los ejemplos aprendidos del juego d ensayo, la red deberá ser capaz de generalizar y dar respuestas correctas ante patrones de estímulos nuevos. (Gestal, 2010)

CAPITULO III

MARCO APLICATIVO

3.1. Introducción

La construcción del Sistema Inteligente Basado en Redes Neuronales, sigue el proceso para el desarrollo y validación, propia de las metodologías de clasificación de minería de textos y fases del desarrollo de redes neuronales, de esta manera la investigación demuestra la capacidad de la Red Neuronal para clasificar conversaciones con contenido pedófilo. Elaborando los diferentes modelos de RNA[24], se optó por el Perceptrón Multicapa debido a la linealidad del problema.

Con el conjunto de datos obtenidos y con los que se cuenta disponibles se construyó el corpus el cuál se divide en dos, Conversaciones *"Positivas (Es una conversación con contenido Pedófilo)"*, y *"Negativas (Conversaciones sin contenido Pedófilo*)" los que posteriormente son utilizados para el entrenamiento de la RNA de manera automática mediante un modelo de aprendizaje profundo.

El procedimiento para la clasificación de textos (Vandana y Namrata, 2012), es el método empleado para el desarrollo del Sistema inteligente como se ve en la figura 3.1.

3.2. Documentos

3.2.1. Recolección de documentos

La construcción del corpus de conversaciones positivas, así como de negativas fue obtenido de chats de conversaciones reales en ingles de servicios como Skype, Yahoo Messenger, MSN Messenger.

a) Recolección de corpus de conversaciones positivas (con contenido pedófilo)

Para realizar la construcción del corpus de conversaciones positivas se recurrió al sitio Perverted-Justice[25] del cual se extrajeron 2000 conversaciones positivas. Este sitio es el único

24 RNA: Redes Neuronales Artificiales.

25 http://perverted-justice.com/

que publica las conversaciones de Adultos (Pedófilos, pederastas) con Menores de edad, las conversaciones se clasifican entre 1 y 5 dependiendo de la escala de "Adulación Maliciosa"[26]. También se recolectaron documentos de conversaciones positivas del sitio http://pan.webis.de/ en un concurso de identificación de depredadores sexuales que se realizó en 2012 dentro de PAN[27] y otros proporcionados por April Edwards del sitio http://www.chatcoder.com/. Perverted-Justice no proporciona ninguna forma de descarga masiva del dataset, por lo tanto, la recopilación de las conversaciones fue mediante otras dos fuentes, estas dos fuentes son: http://www.chatcoder.com/ y el sitio http://pan.webis.de/ estos corpus tienen el formato XML. Para la elaboración del corpus se incluyeron aquellas conversaciones con una calificación igual o superior a tres, tomando como referencia las puntuaciones vertidas en Perverted-Justice. En la tabla 3.1 se detalla los rangos de adulación maliciosa y conversaciones.

	Nombre del Archivo	Puntuacion (Slimyness Scale)	Sitio de extraccion
1	fleet_captain_jaime_wolfe	4.88	Yahoo
2	DavieWants2	4,86	AOL
3	Vamale_692005	4,84	Yahoo
4	Kalowoodsman	4,84	Yahoo
5	thenewpersin62	4,84	Yahoo
6	jackman_9682	4,76	Yahoo
7	smileman74	4,73	Yahoo
8	shinelfmc2005	4,71	Yahoo
9	can_i_rape_you_anally	4,7	Yahoo
10	REDBD	4,68	AOL

Tabla 3.1. Muestra de las diez conversaciones con contenido pedófilo
Fuente: (Perverted Justice, 2018)

26 Slimyness Scale

27 PAN es una serie de eventos científicos y tareas compartidas sobre análisis forense de textos digitales

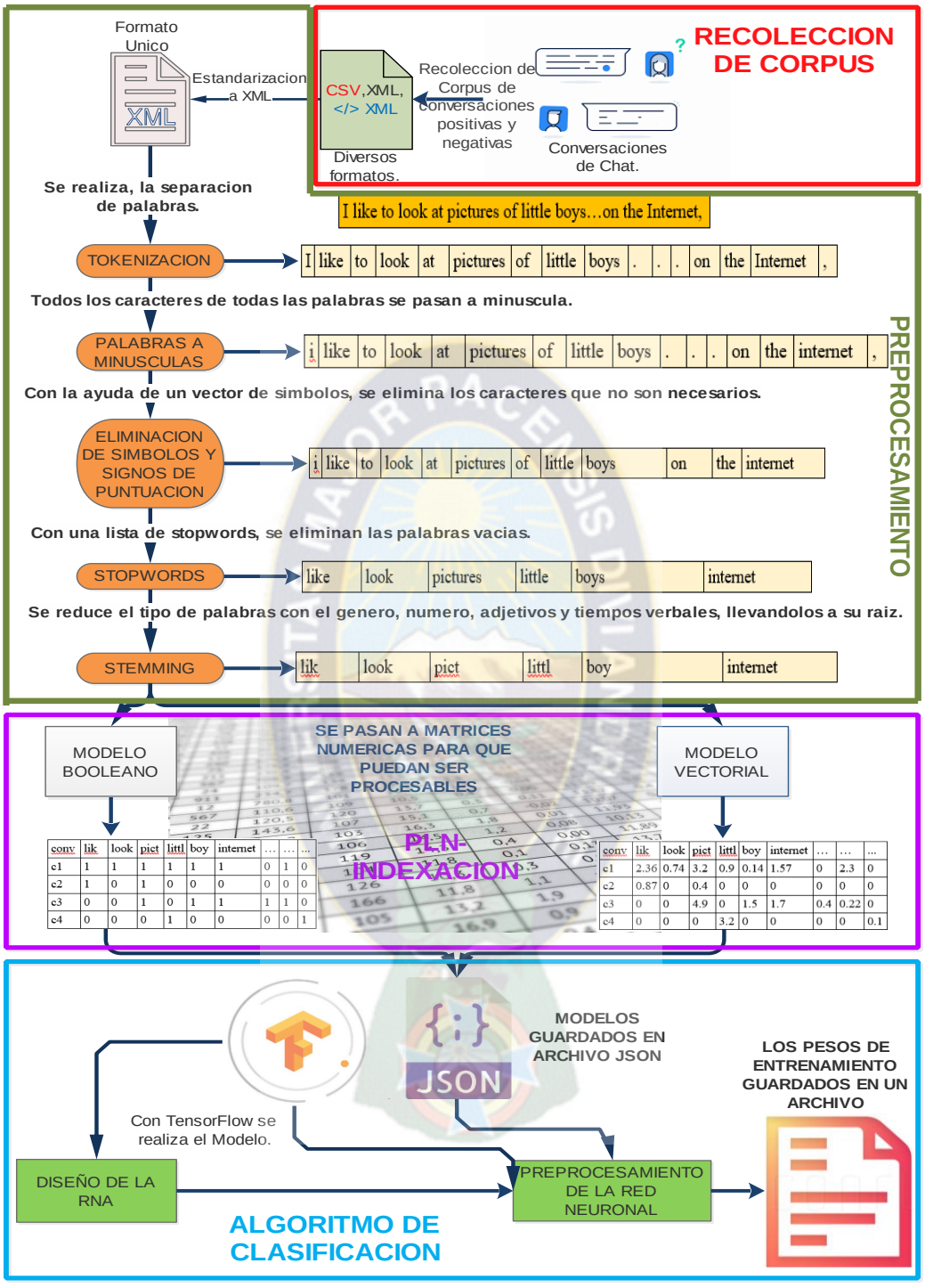


Figura 3.1. Metodología para la clasificación de conversaciones con contenido pedófilo.
Fuente (Elaboración Propia)

En la siguiente tabla: una sección de un chat con contenido pedófilo.

lexi (01/26/08 8:49:44 PM): hey liv2rydu10 (01/26/08 8:49:59 PM): hi lexi (01/26/08 8:50:32 PM): whts up liv2rydu10 (01/26/08 8:51:07 PM): im not yet ;) lexi (01/26/08 8:51:34 PM): lol ur a nutcas lexi (01/26/08 8:51:35 PM): lol liv2rydu10 (01/26/08 9:17:17 PM): you alone ? lexi (01/26/08 9:17:35 PM): moms at th store for a lttl bit why> liv2rydu10 (01/26/08 9:17:52 PM): im naked on my cam lexi (01/26/08 9:18:06 PM): ok lexi (01/26/08 9:18:12 PM): arent u cold like that? lexi (01/26/08 9:19:13 PM): yikes lexi (01/26/08 9:20:30 PM): thats kinda big liv2rydu10 (01/26/08 9:20:48 PM): whats kinda big lexi (01/26/08 9:20:57 PM): your penis liv2rydu10 (01/26/08 9:21:05 PM): you like lexi (01/26/08 9:21:19 PM): sure liv2rydu10 (01/26/08 9:21:32 PM): you hav pics ? lexi (01/26/08 9:21:41 PM): jus the pics on my myspac liv2rydu10 (01/26/08 9:21:45 PM): ok liv2rydu10 (01/26/08 10:23:50 PM): still there lexi (01/26/08 10:23:57 PM): yeah liv2rydu10 (01/26/08 10:24:18 PM): can you view ? lexi (01/26/08 10:24:23 PM): sure liv2rydu10 (01/26/08 10:25:22 PM): ever suck one ?(*he sure wasnt wasting time*) lexi (01/26/08 10:25:56 PM): ya liv2rydu10 (01/26/08 10:26:10 PM): you swallow ? lexi (01/26/08 10:26:16 PM): not realy

Tabla 3.2. Muestra de conversación de una menor de 13 años con un adulto de 54 años Fuente (Perverted Justice, 2018)

b) Recolección de corpus de conversaciones Negativas (Sin contenido pedófilo)

La construcción del corpus de conversaciones negativas ha sido proporcionada por https://www.kaggle.com/, http://krijnhoetmer.nl/irc-logs/, http://www.irclog.org/, y http://omegle.inportb.com/ estos sitios recolectaron corpus de conversaciones regulares grupales y algunas de dos personas. Son pocas las personas que comparten sus conversaciones privadas y estas pocas son de comunidades de Ubuntu. Las conversaciones

negativas forman un total, de 7000 chats. Los formatos de las conversaciones son variados como XML[28], PL[29] y CSV.[30]

En la tabla 3.3 el ejemplo de chat negativo sin contenido pedófilo, el cual fue extraído de Kaggle.

```
<conversations>
 <conversation id="e621da5de598c9321a1d505ea95e6a2d">
  <message line="1">
   <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
   <time>03:20</time>
   <text>Hi.</text>
  </message>
  <message line="2">
   <author>0158d0d6781fc4d493f243d4caa49747</author>
   <time>03:20</time>
   <text>hi.</text>
  </message>
  <message line="3">
   <author>0158d0d6781fc4d493f243d4caa49747</author>
   <time>03:20</time>
   <text>whats up?</text>
  </message>
  <message line="4">
   <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
   <time>03:20</time>
   <text>not a ton.</text>
  </message>
  <message line="5">
   <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
   <time>03:20</time>
   <text>you?</text>
  </message>
  <message line="6">
   <author>0158d0d6781fc4d493f243d4caa49747</author>
   <time>03:20</time>
   <text>same.  being lazy.  M or f?</text>
  </message>
  <message line="7">
```

28 XML es un subconjunto de SGML (Estandar GEneralised Mark-up language) simplificado y adaptado a internet.

29 PL contiene archivos en código y se utiliza para crear programas que analizan texto usando expresiones regulares.

30 CSV del inglés Comma-separated values son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla.

```
    <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
    <time>03:20</time>
    <text>F.</text>
  </message>
  <message line="8">
    <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
    <time>03:21</time>
    <text>Ditto, I&amp;amp;apos;ve done absolutely nothing with my day
besides watching stuff on Hulu.</text>
  </message>
  <message line="9">
    <author>0158d0d6781fc4d493f243d4caa49747</author>
    <time>03:21</time>
    <text>M here.  Just got home from weekend trip.  Tired.</text>
  </message>
  <message line="10">
    <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
    <time>03:21</time>
    <text>Oh, cool. Family thing?</text>
  </message>
  <message line="11">
    <author>0158d0d6781fc4d493f243d4caa49747</author>
    <time>03:22</time>
    <text>yeah.</text>
  </message>
  <message line="12">
    <author>0158d0d6781fc4d493f243d4caa49747</author>
    <time>03:22</time>
    <text>a &amp;amp;amp; l?</text>
  </message>
  <message line="13">
    <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
    <time>03:22</time>
    <text>Gotta love those.</text>
  </message>
```

Tabla 3.3. Ejemplo de una conversación Negativa Extraída de kaggle. Fuente. (Kaggle, 2018)

3.3. Pre-Procesamiento

La etapa del preprocesamiento sirve para descubrir conocimientos en un conjunto de textos, el cual le da al conjunto de textos una forma que le permite ser tratada computacionalmente.

Lo que se hizo en esta etapa es la estandarización de los documentos, mismos que fueron separados en palabras, posteriormente la limpieza y reducción de los textos que no son necesarios para el procesamiento de los mismos, por consiguiente, se pasa por las siguientes etapas:

3.3.1. Estandarización de documentos

Recolectados los documentos, se encuentran con distintos formatos, como ser XML y CSV, así mismo estos tienen su propia estructura. En su mayoría estos documentos se los obtuvo en formato XML, pero de igual manera tienen diferente estructura por la diversidad de fuentes que se tuvo al obtenerlos; en la tabla 3.4, tabla 3.5 y tabla 3.6 se observa la diversa estructura de los documentos. Previamente antes del pre-procesamiento de los documentos es necesario que estén en un mismo formato y con una misma estructura.

folder	dialogueID	date	from	to	Text
3	126125.tsv	2008-04-23T14:55:00.000Z	bad_image		Hello folks, please help me a bit with the following sentence: 'Order here your personal photos or videos.' - I think the only allowed version is 'Order your personal videos or photos here.', but I'm not sure, are you?
3	126125.tsv	2008-04-23T14:56:00.000Z	bad_image		Did I choose a bad channel? I ask because you seem to be dumb like windows user
3	126125.tsv	2008-04-23T14:57:00.000Z	lordleemo	bad_image	the second sentence is better english and we are not dumb
3	64545.tsv	2009-08-01T06:22:00.000Z	mechtech		Sock Puppe?t
3	64545.tsv	2009-08-01T06:22:00.000Z	mechtech		WTF?
3	64545.tsv	2009-08-01T06:22:00.000Z	richardcavell	mechtech	it's a wikipedia term. There is Prodigy, PRDIGY and prgidy and they're all the same guy
3	98758.tsv	2009-05-20T07:58:00.000Z	C-00000100	Severity1	what is bartek trying to do?

folder	dialogueID	date	from	to	Text
3	98758.tsv	2009-05-20T08:01:00.000Z	C-00000100	Severity1	is he trying to play online video?

Tabla 3.4. Conversaciones de grupos de Ubuntu en archivo CSV
Fuente (Elaboración Propia)

```
<?xml version="1.0" encoding="utf-8" ?>
<chatcoderadmin>
  <!-- Table ChatLog -->
    <ChatLog>
       <name>Adamou217</name>
       <CodingVersionID>372</CodingVersionID>
       <lineNum>6</lineNum>
       <category>210</category>
       <userID>{superboy_93}</userID>
       <dateTime>8/1/2006 7:25:31 PM</dateTime>
       <body>cool went sk8ng n swimming</body>
    </ChatLog>
    <ChatLog>
       <name>Adamou217</name>
       <CodingVersionID>372</CodingVersionID>
       <lineNum>9</lineNum>
       <category>200</category>
       <userID>{Adamou217}</userID>
       <dateTime>8/1/2006 7:25:59 PM</dateTime>
       <body>where 'd u go swimming</body>
    </ChatLog>
    <ChatLog>
       <name>Adamou217</name>
       <CodingVersionID>372</CodingVersionID>
       <lineNum>10</lineNum>
       <category>200</category>
       <userID>{Adamou217}</userID>
       <dateTime>8/1/2006 7:26:02 PM</dateTime>
       <body>and skating</body>
    </ChatLog>
          ...
</chatcoderadmin>
```

Tabla 3.5. Conversación de perverted-justice.com en formato XML proporcionada por April Edwards
Fuente (Elaboración Propia)

```
<conversations>
  <conversation id="e621da5de598c9321a1d505ea95e6a2d">
    <message line="1">
      <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
      <time>03:20</time>
      <text>Hi.</text>
    </message>
    <message line="2">
      <author>0158d0d6781fc4d493f243d4caa49747</author>
      <time>03:20</time>
      <text>hi.</text>
    </message>
    <message line="3">
      <author>0158d0d6781fc4d493f243d4caa49747</author>
      <time>03:20</time>
      <text>whats up?</text>
    </message>
    <message line="4">
      <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
      <time>03:20</time>
      <text>not a ton.</text>
    </message>
    <message line="5">
      <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
      <time>03:20</time>
      <text>you?</text>
    </message>
    <message line="6">
      <author>0158d0d6781fc4d493f243d4caa49747</author>
      <time>03:20</time>
      <text>same.  being lazy.  M or f?</text>
    </message>
    <message line="7">
      <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
      <time>03:20</time>
      <text>F.</text>
    </message>
    <message line="8">
      <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
      <time>03:21</time>
      <text>Ditto, I&amp;amp;apos;ve done absolutely nothing with my day besides watching
stuff on Hulu.</text>
    </message>
    <message line="9">
```

```
    <author>0158d0d6781fc4d493f243d4caa49747</author>
    <time>03:21</time>
    <text>M here.  Just got home from weekend trip.  Tired.</text>
   </message>
   <message line="10">
    <author>97964e7a9e8eb9cf78f2e4d7b2ff34c7</author>
    <time>03:21</time>
    <text>Oh, cool. Family thing?</text>
   </message>
</conversations>
```

Tabla 3.6. Conversación de perverted-justice.com en formato XML proporcionada por PAN
Fuente (Elaboración Propia)

En las tablas anteriores, se evidencia que los documentos tienen campos o características distintas, como ser: folder, dialogID, date, from, to text, author, time, name, UserID, body, para el procesamiento no son necesarios todos estos campos.

Lo que se realiza es pasarlos a un solo formato elegido previamente, para el caso de estudio el formato escogido es **XML** debido a que es un formato estándar internacional que permite representar información estructurada con etiquetas para identificar sus partes.

El caso de estudio necesita una parte definida, denominada "cuerpo de la conversación", es decir solo el texto donde se intercambia información, donde se entabla un dialogo y que tipo de conversación representa, ya sea conversación positiva o conversación negativa, se representa con la etiqueta "conversations" este es el contenedor de todas las conversaciones, cada conversación posee la etiqueta "conversation", que a la vez es asignado a una clase llamada "class" el cual indica el tipo de conversación que es, mismo que se puede verificar en la tabla 3.7. A la vez cada línea de una conversación es unida en una sola línea, de este modo pasa a representar toda la conversación, sin hacer diferencia entre usuarios de la conversación, esta unión se realiza debido a que lo que interesa es el contexto de la conversación en conjunto, ya que el procesamiento del texto se lo analizara de manera conjunta y no línea por línea.

```
<conversations>
   <conversation class="negativa">
      Hi. hi. whats up? not a ton. you? same.  being lazy.  M or f? F. Ditto, I've done
absolutely nothing with my day besides watching stuff on Hulu. M here.  Just got home
```

from weekend trip. Tired. Oh, cool. Family thing? yeah. Gotta love those. 17, Hawaii. and yourself? Uh oh. older. 30 Been to Hawaii. whoops xD It's nice, isn't it? Yeah. Always enjoy visiting. Which Island you on Oahu? married? i'm assuming since you went on a family trip :p yeah. Just found this site a few days ago. Yeah, Oahu. Curious to the whole random thing Pretty crazy the individuals you meet, isn't it? It's been eye opening for sure. Yeah, I hear you.

</conversation>

<conversation class="positiva">

where in wi? milw u? like a half hour north of milwaukee kewl a/s? 25/m what town u in im on tha north side 2 suburb hartford u? im in menomonee falls small world lol rad so whats up? not much chillin u? same same bored outta my head lol lol same here so what do u do fer a livin? carpenter neat i go to school lucky me 13/f ugh lol its better than work though if u say so u have a very kewl goatee lol lol thanx i shaved it off for awhile but im gonna start regrowin it very nic epic btw thats kewl 2 u think so? im a dork lol lol but so am i hey kewl we can be dorks 2gether then rad that lil devil is cute lol lol we're talkin about the smiley right lol ur funny u got a bf? no i just got dumped it sucks awww sorry to hear that any thing i can do to make u feel better? aww ur sweet its just nice 2 talk 2 a boy thats not a jerk he was mean there's a lot of them out htere he said that he found another gf at least he was honest lol well thats still shitty of him i spose eh hes dumb lol u mind im 25? no im not mean lol not mean? so what are u into?

</conversation>

<conversation class="positiva">

...

</conversation>

...

</conversations>

Tabla 3.7. Formato y estructura del documento estandarizado
Fuente (Elaboración Propia)

3.3.2. Tokenización

Partiendo de la colección de documentos en formato **XML** unificado y estandarizado se encuentra listo para ser examinado. Lo que se hace en esta etapa es leer el documento XML extrayendo las conversaciones, es decir el texto de cada una de ellas, en las tablas 3.8 y 3.9 se presenta los ejemplos de tokenización, para proceder a dividir el flujo de caracteres en palabras para cada una de las conversaciones, esto se hace para un mejor procesamiento de las mismas, el ejemplo de este proceso se lo presenta en las tablas 3.10 y 3.11.

Hi. hi. whats up? not a ton. you? same. being lazy. M or f? F. Ditto, I've done absolutely nothing with my day besides watching stuff on Hulu. M here. Just got home from weekend trip. Tired. Oh, cool. Family thing? yeah. Gotta love those. 17, Hawaii. and yourself? Uh oh. older. 30 Been to Hawaii. whoops xD It's nice, isn't it? Yeah. Always enjoy visiting. Which Island you on Oahu? married? i'm assuming since you went on a family trip :p yeah. Just found this site a few days ago. Yeah, Oahu. Curious to the whole random thing Pretty crazy the individuals you meet, isn't it? It's been eye opening for sure. Yeah, I hear you.
Trozo de una conversación estandarizada y unida en una sola línea

Tabla 3.8. Fragmento de una Conversación estandarizada
Fuente (Elaboración Propia)

where in wi? milw u? like a half hour north of milwaukee kewl a/s? 25/m what town u in im on tha north side 2 suburb hartford u? im in menomonee falls small world lol rad hi whats up? not much chillin u? same same bored outta my head lol lol same here so what do u do fer a livin? carpenter neat i go to school lucky me 13/f ugh lol its better than work though if u say so u have a very kewl goatee lol lol thanx i shaved it off for awhile but im gonna start regrowin it very nic epic btw thats kewl 2 u think so? im a dork lol lol but so am i hey kewl we can be dorks 2gether then rad that lil devil is cute lol lol we're talkin about the smiley right lol ur funny u got a bf? no i just got dumped it sucks awww sorry to hear that any thing i can do to make u feel better? aww ur sweet its just nice 2 talk 2 a boy thats not a jerk he was mean there's a lot of them out htere he said that he found another gf at least he was honest lol well thats still shitty of him i spose eh hes dumb lol u mind im 25? no im not mean lol not mean? so what are u into?
Trozo de una conversación estandarizada y unida en una sola línea

Tabla 3.9. Fragmento de una Conversación estandarizada ejemplo 2
Fuente (Elaboración Propia)

En las tablas 3.10 y 3.11 se realizó la división del texto en palabras tokens, para la misma tarea se empleó la libreria NLTK que incluye un tokenizador de palabras que acepta caracteres hasta encontrar el final de cada palabra, las secuencias que marcan el final de cada palabra son espacios en blanco, fin de la cadena y signos de puntuación, algunos caracteres se excluye de tokens de palabras iniciales, estos caracteres representados en expresión regular son "[^\(\"\`{\[:;&\#*@\)}\]\-,]" y otros caracteres como "(?:[?!)\";}\]*:@\'\({\[])" no pueden permanecer dentro de las palabras.

División por palabras en un vector:

Hi	.	Hi	.	whats	up	?	not	a	ton
.	you	?	same	.	being	lazy	.	M	or
f	?	F	.	Ditto	I've	done	absolutely	nothing	with

my	day	besides	whatching	stuff	on	Hulu	.	M	here
.	Just	Got	home	from	weekend	trip	.	Tired	.
Oh	,	Cool	.	Family	thing	?	yeah	.	Gotta
love	those	….	…	…	…	…	…	…	..

Tabla 3.10. División por palabras en un vector
Fuente (Elaboración Propia)

Where	In	wi	?	milw	u	?	like	A
half	hour	north	of	milwaukee	kewl	a/s	?	25/m
what	town	u	in	im	on	tha	north	side
2	suburb	Hartford	u	?	im	in	menomonee	falls
small	world	lol	rad	hi	Whats	UP	?	not
much	chillin	u	?	Same	same	bored	outta	my
head	lol	lol	…	…	…	…	…	...

Tabla 3.11. División por palabras en un vector ejemplo 2
Fuente (Elaboración propia)

La línea de texto que contiene la conversación queda separada en palabras creando un diccionario para cada una de las conversaciones utilizando la librería de procesamiento del lenguaje natural llamado **NLTK**[31].

3.3.3. Conversión de Palabras a minúsculas

Tener el conjunto de palabras en minúsculas es útil para una mejor comparación al momento de limpiar las palabras, es decir al excluir las palabras.

A continuación, los ejemplos en las tablas 3.12 y 3.13 son la transformación de las palabras, todas estandarizadas a minúsculas partiendo de las tablas 3.10 y 3.11.

hi	.	Hi	.	whats	up	?	not	a	ton
.	you	?	same	.	being	lazy	.	m	or
f	?	F	.	ditto	i've	done	absolutely	nothing	with
my	day	besides	whatching	stuff	on	hulu	.	m	here
.	just	got	home	from	weekend	trip	.	tired	.
oh	,	cool	.	family	thing	?	yeah	.	gotta
love	those	….	…	…	…	…	…	…	..

Tabla 3.12. Transformada todas las palabras en minúsculas
Fuente (Elaboración propia)

where	In	wi	?	milw	u	?	like	a
half	Hour	north	of	milwaukee	kewl	a/s	?	25/m
what	Town	u	in	im	on	tha	north	side

31 NLTK, es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural (PLN) simbolico y estadísticos para el lenguaje de programación Python

2	Suburb	hartford	u	?	im	in	menomonee	falls
small	World	lol	rad	hi	whats	up	?	not
much	Chillin	u	?	same	same	bored	outta	my
head	Lol	lol	…	…	…	…	…	...

Tabla 3.13. Transformada todas las palabras en minúsculas, ejemplo 2
Fuente (Elaboración propia)

3.3.4. Eliminación de símbolos y signos de puntuación

La eliminación de los signos de puntuación incluyo la construcción de un vector donde se determina los signos que no se consideran importantes, en la tabla 3.14 se encuentra el vector de signos que no son importantes. En las tablas 3.15 y 3.16 se observa las conversaciones sin los signos de la tabla 3.14.

?	<	>	(	)	!	[	]	{	}
:	;	.	$	=	,	/	-	_	

Tabla 3.14. Vector de signos no considerados importantes
Fuente (Elaboración Propia)

En base al vector construido en la etapa de tokenización y transformación a minúsculas, se procede a la eliminación de los signos de puntuación mencionados en la tabla 3.14, el resultado obtenido se presenta en las tablas 3.15 y 3.16.

hi	Hi	whats	up	not	a	ton	you	same	being
lazy	M	or	f	f	ditto	i've	done	absolutely	nothing
with	my	day	besides	whatching	stuff	on	hulu	m	here
just	got	home	from	weekend	trip	tired	oh	cool	family
thing	yeah	gotta	love	those	…	…	…	…	...

Tabla 3.15. Vector de palabras, sin signos.
Fuente (Elaboración Propia)

where	In	wi	milw	u	like	a	half	hour
north	of	milwaukee	kewl	a/s	25/m	what	town	u
in	im	on	tha	north	side	2	suburb	hartford
u	im	in	menomonee	falls	small	world	lol	rad
hi	whats	up	not	much	chillin	u	same	same
bored	outta	my	head	lol	lol	…	…	…

Tabla 3.16. Vector de palabras, sin signos. Ejemplo 2
Fuente (Elaboración Propia)

3.3.5. Stopwords

En esta etapa se eliminan las palabras que no aportan en nada, estas son palabras que tienen un valor muy bajo en discriminación ya que ocurren en todos los documentos en inglés y por

lo tanto no ayuda a distinguir entre conversaciones con contenido pedófilo y conversaciones sin el contenido pedófilo. Para la eliminación de estas palabras es necesario construir una lista de palabras vacías (stopwords) que generalmente son preposiciones, artículos, pronombres, la creación de esta lista se realizó con la librería de python NLTK que cuenta con una lista de 179 palabras vacías algunas de las cuales son: 'all', 'just', "don't", 'being', 'over', 'both', 'through', 'yourselves', 'its', 'before','should', 'to', 'only'.

Quitando del diccionario esas palabras, el resultado se encuentra en las tablas 3.17 y 3.18:

hi	hi	ton	lazy	m	f	f	ditto	done	absolutely
nothing	day	besides	whatching	stuff	hulu	m	got	home	weekend
trip	tired	tired	oh	cool	family	thing	yeah	gotta	love
…	…	…	…	..	.	..	..	..	..

Tabla 3.17. Vector de conversaciones resultante, excluyendo las palabras vacías
Fuente (Elaboración Propia)

wi	milw	u	like	half	hour	north	milwaukee	kewl
a/s	25/m	town	u	im	tha	north	side	2
suburb	hartford	u	im	menomonee	falls	small	world	lol
rad	hi	whats	much	chillin	u	bored	outta	head
lol	lol	u	fer	livin	carpenter	…	…	…

Tabla 3.18. Vector de conversaciones resultante, excluyendo las palabras vacías, Ejemplo 2
Fuente (Elaboración Propia)

Al realizar la eliminación de los stopwords, el diccionario de la conversación va quedando reducido.

3.3.6. Lematización o stemming

El stemming hace referencia a la estandarización de los tokens, es decir en esta etapa se reduce los tipos de palabras del diccionario determinando por cada palabra un lema[32]. Las palabras se reducen de género, número, adjetivos y tiempos verbales a su raíz. Este paso es aplicado con la ayuda de la librería "lancaster" de **NLTK** en python a cada una de las palabras o tokens del diccionario anterior quedando de la siguiente forma en las tablas 3.19 y 3.20.

32 Lema: o ítem lexical es una unidad autónoma constituyente del léxico de un idioma. Es una serie de caracteres que forman una unidad semántica y que puede constituir una entrada de diccionario.

Lancaster

Nltk.stem.lancaster es un stemmer de palabras basado en el algoritmo de derivación de Lancaster Paice/husk eliminación de afijos.

Este algoritmo consiste en una serie de reglas específicas para cada letra en el alfabeto, al momento de procesar una palabra se toma la última letra, así se encuentra el conjunto de reglas para esta letra, La regla pasa a ser estudiada y es acepta si:

- Concuerda con las ultimas letras de la palabra.
- Cualquier condición especial para esa regla se satisface.
- La longitud del lema resultante no sea menor a un límite establecido o no tenga vocales.

Al ser aceptada se aplica, en caso contrario se prosigue con la siguiente regla. Sin embargo, si se llega a una regla cuya primera letra no coincide con la última letra de la palabra, se implica que no se puede remover la terminación y el proceso finaliza.

hi	hi	ton	lazy	m	f	f	ditto	don	absolv
noth	day	besid	whatch	stuff	hulu	m	got	hom	weekend
trip	tired	tir	oh	cool	fami	thing	yeah	got	lov
…	…	…	…	..	.	..	..	..	..

Tabla 3.19. Reducción de palabras en lemas
Fuente (Elaboración Propia)

wi	milw	u	like	half	hour	nor	milwauk	kewl	a/s
25/m	town	u	im	tha	not	sid	2	suburb	hartford
u	im	monomo	fal	smal	world	lol	rad	hi	what
much	…	…	…	..	.	..	..	..	..

Tabla 3.20. Reducción de palabras en lemas ejemplo 2
Fuente (Elaboración Propia)

3.4. Indexación

Seguidamente del pre-procesado del texto sigue la transformación a un modelo de recuperación de información, donde se une todos los diccionarios de cada conversación formando uno global, donde cada palabra es una característica y no deben repetirse, de esta manera se forma una matriz de medidas numéricas para la predicción, teniendo en cuenta que cada fila será una conversación y cada columna será una característica. Para la representación se usará el modelo booleano y el modelo de frecuencias.

3.4.1. Modelo booleano

Al ser este modelo el más simple, es sencillo de implementar basándose solo en la presencia o ausencia de palabras que se encuentran en cada conversación referente al diccionario global esta representación se la hace usando números de "1" y "0" y se plasman en la matriz numérica. A continuación, la tabla 3.21. matriz numérica formada a partir de las tablas 3.19. y 3.20 con el modelo booleano.

Conv	wi	milw	hi	ton	lazv	nor	hour	day	got	half	weekend	much	oh	…
C1	0	0	1	1	1	0	0	1	1	0	1	0	1	…
C2	1	1	1	0	0	1	1	0	0	1	0	1	0	…

Tabla 3.21. Matriz Numérica
Fuente (Elaboración Propia)

3.4.2. Modelo vectorial

El modelo algebraico vectorial es una mejora del modelo booleano. En este se mide la frecuencia de palabras que aparece en una conversación mostrando de esta manera la relevancia de cada característica en los documentos. La frecuencia-término es una medida de cuántas veces los términos están presentes en las conversaciones, se define la frecuencia-término como una función de conteo:

$$tf(t,d) = \sum_{x \in d} fr(x,t)$$

donde $fr(x,t)$ es una simple función definida como:

$$fr(x,t) = \begin{cases} 1, & si\ x = t \\ 0, & en\ otros\ casos \end{cases}$$

El cual devuelve $tf(t,d)$, que representa cuántas veces aparece el término t en el documento d o conversación. Un ejemplo es tf ("h_i", c_1) $= 2$ ya que se tiene solo dos apariciones del término "h_i" en la conversación c_1. Luego se continua con la creación del vector del documento, que se representa en la tabla 3.22:

$$\overrightarrow{v_{d_n}} = (tf(t_1,d_n), tf(t_2,d_n), tf(t_3,d_n), \dots, tf(t_n,d_n))$$

Conv	wi	milw	hi	ton	lazv	nor	hour	day	got	half	weekend	much	oh	…
C1	0	0	2	1	1	0	0	1	2	0	1	0	1	…
C2	1	1	1	0	0	1	1	0	0	1	0	1	0	…

Tabla 3.22. TF vector de documento
Fuente (Elaboración Propia)

Al finalizar la aplicación del modelo vectorial se presenta un problema con el enfoque de frecuencia de términos, este da importancia a los términos frecuentes y reduce los términos raros que son empíricamente los que brindan información, que los términos de alta frecuencia. La intuición básica es que un término que aparece con frecuencia en muchos documentos no es un buen discriminador por tal razón se recurre a otra medida de relevancia como lo es la *Frecuencia Inversa de Documentos* o en sus siglas ***IDF***.

El peso **TF-IDF** resuelve el problema de frecuencia de términos. Lo que **TF-IDF** ofrece es qué tan importante es una palabra para un documento en una colección, y es por eso que TF-IDF incorpora parámetros locales y globales, ya que toma en consideración no solo el término aislado sino también el término dentro de la colección de documentos. Lo que **TF-IDF** hace, es reducir los términos frecuentes mientras escala los términos raros; un término que ocurre 10 veces más que otro no es 10 veces más importante que eso, es por eso que TF-IDF usa la escala logarítmica.

Antes de sacar los pesos $\boldsymbol{tf-idf}$ se normalizo el vector. Por ejemplo la primera conversación $c1$, el vector $c1$ tiene muchos elementos y para el ejemplo se va a reducir tomando solo los primeros 13 términos y se representa de la siguiente manera:

$$\overrightarrow{V_{c1}} = (0,0,2,1,1,0,0,1,2,0,1,0,1)$$

Al normalizar el vector, es lo mismo que calcular el vector unitario del vector, y se denotan usando la notación "sombrero": $\hat{v}$. La definición del vector unitario $\vec{v}$ de un vector $\vec{v}$ es:

$$\hat{v} = \frac{\vec{v}}{\|\vec{v}\|_p}$$

Donde $\hat{v}$ es el vector unitario, o el vector normalizado, él $\vec{v}$ es el vector que se va a normalizar y él $\|\vec{v}\|_p$ es la norma (magnitud, longitud) del vector $\vec{v}$ en el espacio L^P. La longitud del vector está definida como:

$$\|\vec{u}\|_p = (\sum_{i=1}^{n} |\vec{u}_i|^p)^{\frac{1}{p}}$$

Donde el valor de p será igual a 2 que es la que representa a la norma *Euclidiana se recurrió a esta norma* por ser la más común y la que viene por defecto en la librería de scikits learn. Se normalizo de la siguiente manera:

$$\hat{V} = \frac{\vec{V}}{\left\|\vec{V}\right\|_p}$$

$$\widehat{V_{c1}} = \frac{\overrightarrow{V_{c1}}}{\left\|\overrightarrow{V_{c1}}\right\|_2}$$

$$\widehat{V_{c1}} = \frac{(0,0,2,1,1,0,0,1,2,0,1,0,1)}{\sqrt{0^2+0^2+2^2+1^2+1^2+0^2+0^2+1^2+2^2+0^2+1^2+0^2+1^2}}$$

$$\widehat{V_{c1}} = \frac{(0,0,2,1,1,0,0,1,2,0,1,0,1)}{3.6056}$$

$$\widehat{V_{c1}} = (0,0,0.5547,0.2773,0.2773,0,0,0.2773,0.5547,0.2773,0,0.2773)$$

a) TF-IDF

Normalizados los términos de frecuencia de la conversación se calculó el **TF-IDF**.

El espacio de los documentos es denotado: $D = \{d_1, d_2, \dots, d_n\}$ donde n es el número de documentos en el corpus.

En la tesis se trabajó con siete mil conversaciones, por lo tanto, para el caso de estudio se tomó en cuenta 2 conversaciones como ejemplos de partida, mismos que ya fueron tratados anteriormente.

Entonces en el espacio $D = |c1, c2|$ que representa la primera y segunda conversación teniendo así una longitud $n_d = 2$.

Ahora él IDF se definió como:

$$idf(t) = log\frac{n_d}{df(d,t)} + 1$$

Donde $df(d,t)$ es el número de documentos en el cual el termino t aparece, además en la formula se adiciono un 1 para que los términos que aparecen en los documentos no sean enteramente ignorados. Ahora se usa la tabla n para sacar los valores IDF de la siguiente manera:

$$idf(t_1) = log\frac{n_d}{df(d,t_1)} + 1 = \log\frac{2}{1} + 1 = log3 = 0.477121255$$

$$idf(t_2) = log\frac{n_d}{df(d,t_2)} + 1 = \log\frac{2}{1} + 1 = log3 = 0.477121255$$

$$idf(t_3) = log\frac{n_d}{df(d,t_3)} + 1 = \log\frac{2}{1} + 1 = log2 = 0.301029996$$

$$idf(t_4) = log\frac{n_d}{df(d,t_4)} + 1 = \log\frac{2}{1} + 1 = log3 = 0.477121255$$

El primer término t_1 representa la primera característica o palabra ***wi***, la cual solo aparece en la segunda conversación denotando $df(d,t_1) = 1$ dando un **IDF** para el primer término de 0,477121255, este resultado representa a casi todos los términos de la tabla debido a que la mayoría de los términos solo se presenta en un documento y es la razón por la cual no se muestra los **IDF** de los demás términos. Pero como se puede notar el tercer término t_3 que representa la característica ***hi*** aparece en ambas conversaciones y dando como resultado un idf de 0,301029996.

La fórmula para **TF-IDF** es:

$$tf - idf(t,d) = tf(t,d) \times idf(t)$$

Para la obtención de los pesos TF-IDF se multiplico cada valor obtenido en la tabla 3.22 de **TF** con los valores de **IDF** obtenidos anteriormente, los resultados se reflejan en la tabla 3.23.

Conv	wi	milw	hi	ton	lazv	nor	hour	day	got	...
C1	0	0	0,6020	0,4771	0,4771	0	0	0,4771	0,9542	...
C2	0,4771	0,4771	0,3010	0	0	0,4771	0,4771	0	0	...

Tabla 3.23. IDF obtenido de la multiplicación
Fuente (Elaboración Propia)

3.5. Algoritmo de Clasificación – Redes neuronales

Para comprender el funcionamiento del Sistema Inteligente Para la Detección de conversaciones con posible contenido Pedófilo, Basado en Redes Neuronales, se explicó la etapa de clasificación de textos, a continuación, se detalla el funcionamiento interno de la Red Neuronal Artificial.

3.5.1. Fase de Creación y Desarrollo

1. **Diseño de la Arquitectura**

Se determinó cual es el proceso adecuado para la clasificación de textos de la investigación. Llegando a la conclusión de que la RNA que más se adaptaba es de Tipo Perceptrón multicapa o MLP (Multi Layer Perceptron).

1.1. Número de Neuronas en la Capa de Entrada

El número de entradas en la primera capa se definió en base a la dimensión de la bolsa de palabras o diccionario que se formó en el entrenamiento de acuerdo a la cantidad de palabras que la red estuvo aprendiendo con cada conversación.

El diccionario contiene **77.135** términos los cuales representan las neuronas de la capa de entrada de la red neuronal, formados a partir de **7500** conversaciones de las cuales 1500 son positivas y 6000 son negativas.

Es decir, mientras más conversaciones se use para entrenar a la red, la red puede ir aprendiendo palabras nuevas y esa cantidad de palabras en total que conozca la red conforma el número de entradas de la misma.

1.2. Número de Neuronas en la Capa de Salida

Cada neurona de salida de una red neuronal sirve para reconocer un patrón diferente, el objetivo de la investigación es reconocer el tipo de conversación de entre dos tipos, positivo o negativo (pedófilo o no pedófilo) por tal razón se tiene dos neuronas en la capa de salida.

A continuación, en la figura 3.2. Arquitectura de La Red Neuronal.

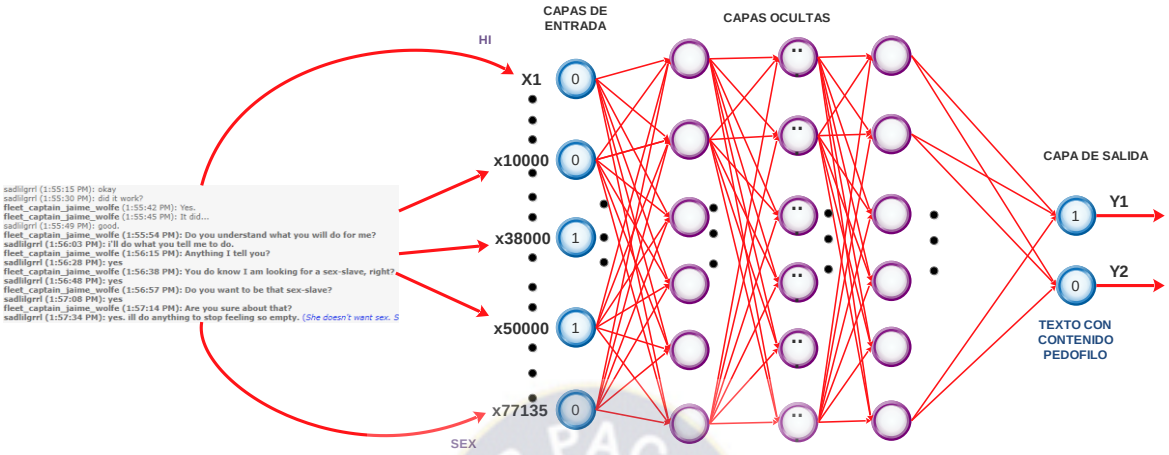


Fig. 3.2. Arquitectura de la RNA para la clasificación de conversaciones con posible contenido pedófilo.
Fuente (Elaboración Propia)

1.3. Numero de neuronas en la capa oculta

Anteriormente se definió el número de neuronas de entrada y de salida, a lo cual se determinó el número de neuronas adecuadas de la capa intermedia. Sin embargo, aunque el funcionamiento de la red depende en forma importante del número de nodos en las capas ocultas, no existe aún un método confiable que permita determinar con precisión el número óptimo de estos (Sánchez,2004), por lo tanto, se elige como número de neuronas en esta capa la cantidad de 15 usadas acertadamente en el reconocimiento de patrones y asimismo se hicieron pruebas a menor escala dando los resultados esperados.

1.4. Funcione de activación

El tipo de problema que se requiere resolver es un problema no lineal y se quiere salidas binarias, por tal razón se utilizó para la RNA una función de activación sigmoide que se empleó para la normalización de los valores, permitiendo describir la evolución del aprendizaje. En la figura 3.3 Función Sigmoidal:

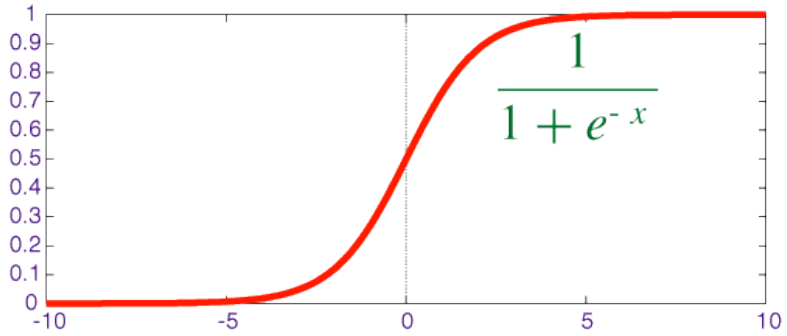


Fig.3.3 Función sigmoidal
Fuente (Elaboración Propia)

El algoritmo de backpropagation que se usa minimiza el error por medio de gradiente descendiente, por lo que la parte esencial del algoritmo es el cálculo de las derivadas parciales de dicho error con respecto a los parámetros de la RNA y para la función sigmoidal se tiene la derivada parcial como se muestra a continuación:

$$y' = y(1 - y)$$

3.5.2. Fase de Entrenamiento de la Red

2.1. Algoritmo de entrenamiento de la RNA

El algoritmo de backpropagation es el algoritmo elegido para entrenar la RNA debido a su capacidad de generalización, facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento y además el uso de un algoritmo supervisado que es el que se requiere. Para la generación del algoritmo se aplica los pasos del algoritmo de entrenamiento de backpropagation en redes neuronales: El Algoritmo Backpropagation para Redes Neuronales, conceptos y aplicaciones del Instituto Politécnico Nacional Centro de Investigaciones en Computación. (Valencia y Yanez, 2006)

Para el algoritmo de entrenamiento se siguen los siguientes pasos:

Paso 1. Inicializar los pesos de la red con valores pequeños aleatorios

La red neuronal tiene dos capas de pesos se inicializa los valores aleatoriamente con media cero, es decir valores pequeños primeramente para la primera capa que combinara las entradas siendo esta la **sinapsis 0** y luego de la misma manera asignamos valores aleatorios para la segunda capa **sinapsis1** de pesos que ayudara a mapear la salida correcta usando las

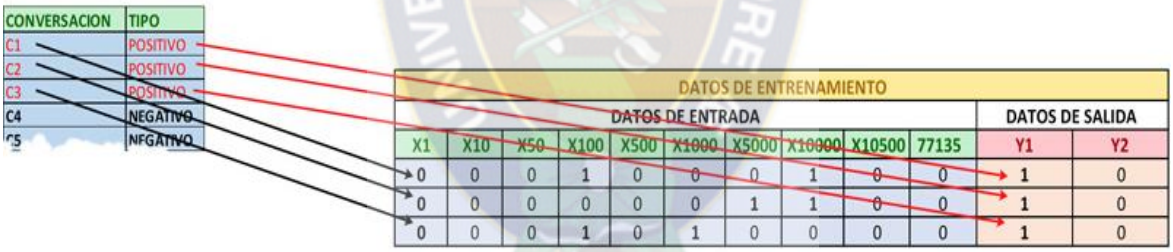
salidas de la primera capa. En el pseudocodigo se ve la inicialización de pesos para la red neuronal.

```
-establecer vector entradas_X (con dimension de 77135)
-establecer vector salidas_Y (con dimension de 2)
-establecer vector sinapsis_0 (con dimension de 77135 x longitud_de(entradas_X)) =
numeros_aleatorios(entre[-0.5 y 0.5])
-establecer vector sinapsis_1 (con dimension de 15 x longitud_de(salidas_Y)) =
numeros_aleatorios(entre[-0.5 y 0.5])
```

Paso 2. Presentar un patrón de entrada y especificar la salida deseada que debe generar la Red

Las conversaciones previamente pre-procesadas que fueron convertidos en datos numéricos son adjuntadas con sus datos de salida, es decir el tipo de conversación.

Como se ha visto las conversaciones son representadas mediante el vector lineal X de 77135 valores binarios, mientras que el tipo de conversación es representado mediante el vector Y con 2 valores binarios como se ve en la figura 3.4 (por razones de espacio, de igual manera que en los casos anteriores solo se mostrara algunas características del vector X con los 3 primeros registros de conversaciones)



CONVERSACION	TIPO
C1	POSITIVO
C2	POSITIVO
C3	POSITIVO
C4	NEGATIVO
C5	NEGATIVO

DATOS DE ENTRENAMIENTO											
DATOS DE ENTRADA										DATOS DE SALIDA	
X1	X10	X50	X100	X500	X1000	X5000	X10000	X10500	77135	Y1	Y2
0	0	0	1	0	0	0	1	0	0	1	0
0	0	0	0	0	0	1	1	0	0	1	0
0	0	0	1	0	1	0	0	0	0	1	0

Fig. 3.4 Base de entrenamiento
Fuente (Elaboración propia)

El total de conversaciones que se usa para el entrenamiento son de 7500, todas estas, una vez transformadas en datos computables con sus respectivas salidas son almacenados en un archivo llamado **datosentrenamiento.json** para que pueda ser usado por el algoritmo de entrenamiento.

Para el algoritmo, el archivo de entrenamiento será leído y recuperado en dos variables $x0$ que representará los vectores de entrada y $y0$ los vectores de salida como se muestra a continuación:

```
-abrir archivo 'datosentrenamiento.json' en variable_archivo
   -establecer vector entradas_X0 = variable_archivo['X']
   -establecer vector salidas_Y0 = variable_archivo['Y']
```

Paso 3. Calcular la salida actual de la red

Para ello se presentan las entradas a la red y se calcula la salida de cada capa hasta llegar a la capa de salida, ésta será la salida de la red.

Cuando se presenta un patrón p de entrada $X^p: X_1^p, \dots, X_i^p, \dots, X_N^p$, éste se transmite a través de los pesos w_{ji} desde la capa de entrada hacia la capa oculta. Las neuronas de esta capa intermedia transforman las señales recibidas mediante la aplicación de una función de activación proporcionando, de este modo, un valor de salida. Este se transmite a través de los pesos v_{kj} hacia la capa de salida, donde aplicando la misma operación que en el caso anterior, las neuronas de esta última capa proporcionan la salida de la red. Este proceso se resume en lo siguiente:

La entrada total o neta que recibe una neurona oculta j, net_j^p, es:

$$net_j^p = \sum_{i=1}^{N} w_{ji} x_i^p + \theta_j$$

donde θ puede ser opcional, pues actúa como una entrada más y para la red neuronal que se construye no fue utilizada.

El valor de salida de la neurona oculta j, y_j^p, se obtiene aplicando la función de activación $f(\cdot)$ sobre su entrada neta:

$$y_j^p = f(net_j^p)$$

La representación del algoritmo se hiso de la siguiente manera:

```
-establecer entradas_netas_0 = entradas_X0*sinapsis_0
-establecer vector salidas_patron_Y0 = funcion_sigmoidal(entradas_netas_0)
```

Donde primero se obtiene la entrada total de x0 y w y luego es aplicada la función sigmoidal.

De igual forma, la entrada neta que recibe una neurona de salida k, net_k^p, es:

$$net_k^p = \sum_{j=1}^{H} v_{kj} y_j^p + \theta_k$$

Por último, el valor de salida de la neurona de salida k , y_k^p , es:

$$y_k^p = f(net_k^p)$$

Y para el algoritmo de igual manera:

-establecer entradas_netas_1 = salidas_patron_Y0*sinapsis_1 -establecer vector salidas_patron_Y1 = funcion_sigmoidal(entradas_netas_1)

Paso 4. Calcular los términos de error para todas las neuronas.

En la etapa de aprendizaje, el objetivo es hacer mínimo el error entre la salida obtenida por la red y la salida deseada por el usuario ante la presentación de un conjunto de patrones denominado grupo de entrenamiento.

La función de error que se pretende minimizar para cada patrón p viene dada por:

$$E^p = \frac{1}{2} \sum_{k=1}^{M} (d_k^p - y_k^p)^2$$

donde d_k^p es la salida deseada para la neurona de salida k ante la presentación del patrón pEl delta de la capa de salida para el error es determinado de la siguiente manera:

$$\delta_k^p = \left(d_k^p - y_k^p\right) f'\left(net_k^p\right)$$

Y para la capa oculta es:

$$\delta_j^p = f'(net_j^p) \sum_{k=1}^{M} \delta_k^p v_{kj}$$

Se puede observar que el error o valor delta asociado a una neurona oculta j, viene determinado por la suma de los errores que se cometen en las k neuronas de salida que reciben como entrada la salida de esa neurona oculta j. De ahí que el algoritmo también se denomine propagación del error hacia atrás.

El código para estas funciones en python no son necesarias al usar tensorflow pues la librería ayuda hacerlo de manera automática al aplicar el gradiente descendiente minimizando el error. Para una mejor comprensión el algoritmo se lo representa de la siguiente manera:

```
-vector error_1 = salidas_Y0-salidas_patron_Y1
-vector delta_1 = error_1*derivada_funcion_sigmodial(salidas_patron_Y1)
-vector error_0 = delta_1*sinapsis_1
-vector delta_0 = layer_1_error*derivada_funcion_sigmodial(salidas_patron_Y0)
```

Paso 5. Actualización de los Pesos

La actualización se la hace de manera iterativa que consiste en aplicar la regla de la cadena a la expresión del gradiente y añadir una tasa de aprendizaje η que fue determinada con un valor de 0.00005. Así, en una neurona de salida:

$$\Delta v_{kj}(n+1) = -\eta \frac{\partial E^p}{\partial v_{kj}} = \eta \sum_{p=1}^{P} \delta_k^p y_j^p$$

Donde n indica la iteración.

Y para la neurona oculta:

$$\Delta w_{kj}(n+1) = \eta \sum_{p=1}^{P} \delta_j^p x_i^p$$

Para el código en python la librería de tensorflow ayuda al minimizar el error usando el gradiente descendiente de esta misma forma ya mencionada, pero para mejor entendimiento se representa el algoritmo de la siguiente manera:

```
-establecer tasa_aprendizaje = 0.00005
-sinapsis_1 = sinapsis_1 + (tasa_aprendizaje*delta_1*salidas_patron_Y1)
-sinapsis_1 = sinapsis_1 + (tasa_aprendizaje*delta_0*salidas_patron_Y0)
```

Paso 6. Aceptación del término de error

En este paso se repite todo el proceso hasta que el termino de error resulta aceptablemente pequeño para cada uno de los patrones aprendidos. Y para que el error se reduzca se debe iterar hasta lograrlo. Para este caso se hiso pruebas para determinar el número de iteraciones y otros parámetros que se necesita para la clasificación de textos, con las pruebas se concluye que el número adecuado es 300000 iteraciones.

En el algoritmo conjunto con las iteraciones se la hace de la siguiente manera:

```
-Mientras el contador < 300000
   -contador = contador + 1
   -vector entradas_netas_0 = entradas_X0*sinapsis_0
```

-vector salidas_patron_Y0 = funcion_sigmoidal(entradas_netas_0)
-vector entradas_netas_1 = salidas_patron_Y0*sinapsis_1
-vector salidas_patron_Y1 = funcion_sigmoidal(entradas_netas_1)
-vector error_1 = salidas_Y0-salidas_patron_Y1
-vector delta_1 = error_1*derivada_funcion_sigmodial(salidas_patron_Y1)
-vector error_0 = delta_1*sinapsis_1
-vector delta_0 = layer_1_error*derivada_funcion_sigmodial(salidas_patron_Y0)
-establecer tasa_aprendizaje = 0.00005
-sinapsis_1 = sinapsis_1 + (tasa_aprendizaje*delta_1*salidas_patron_Y1)
-sinapsis_1 = sinapsis_1 + (tasa_aprendizaje*delta_0*salidas_patron_Y0)

Para una mejor comprensión del algoritmo construido, en la figura 3.5. se puede ver una visualización grafica del proceso de trabajo, esta grafica fue generada por tensorflow.

6.1. Evolución del error en la fase de entrenamiento

Como se mencionó anteriormente el entrenamiento se realizó con 7500 conversaciones y 300000 iteraciones este proceso fue realizado con la ayuda de la librería de TensorFlow, al finalizar proporciona la siguiente figura 3.6. Se puede apreciar, en la gráfica se ha alcanzado aproximadamente 1.3607e-5 de error es decir se consiguió un objetivo y se puede afirmar que se tiene una RNA bien entrenada para la clasificación de conversaciones con contenido pedófilo. Se debe de tener presente que el entrenamiento concluye guardando en archivos los pesos sinápticos de la RNA, en este caso se ha guardado estos pesos con la ayuda de tensorflow en archivos como se ve en la siguiente figura 3.7. y estos serán utilizados más adelante cuando se vea el proceso de clasificación de textos.

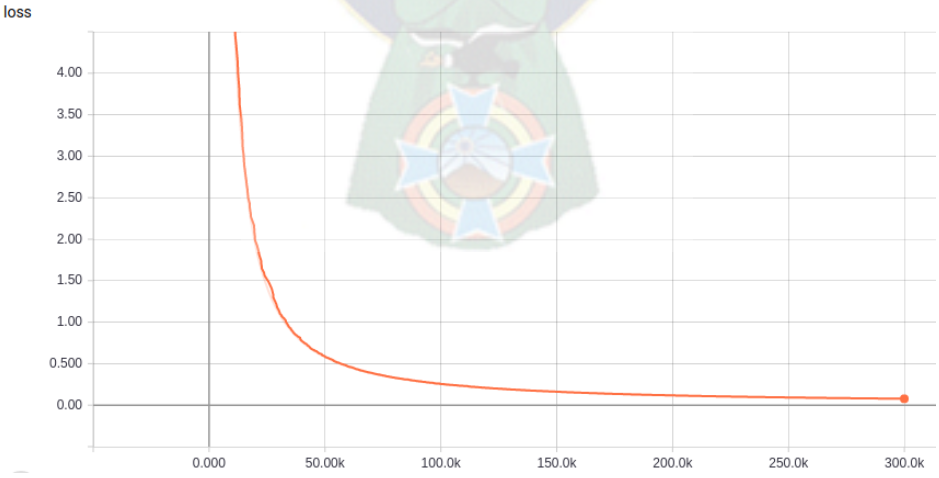


Fig. 3.5. Grafica del Proceso de Trabajo
Fuente (Elaboración Propia)

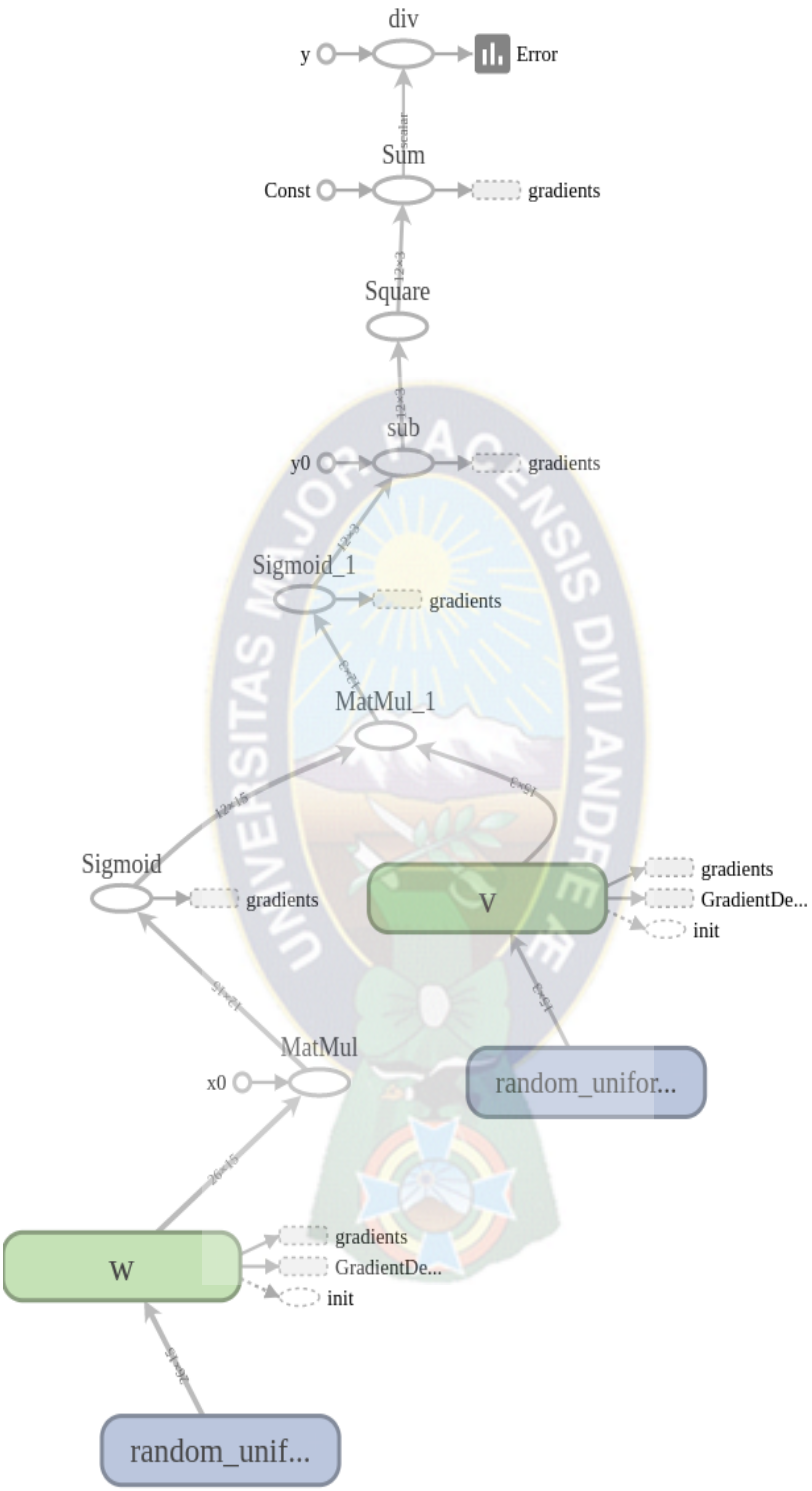


Fig. 3.6. Grafica de la minimización del error en base al número de iteraciones
Fuente (Elaboración Propia)

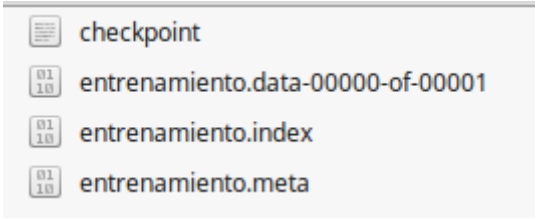


Fig. 3.7. Archivos de Pesos Sinápticos de la RNA.
Fuente (Elaboración Propia)

La fase de validación para la red neuronal junto con el paso de evaluación de rendimiento en la clasificación de textos es presentada en el siguiente capítulo, en el que se le pide al modelo que responda a características diferentes a las mostradas en la fase de entrenamiento, es decir, se proporcionó conversaciones totalmente distintas a las que se usó en el entrenamiento, estas conversaciones fueron seleccionadas de manera que abarque todos los posibles casos en cuanto a conversaciones de pedófilos.

CAPITULO IV

PRUEBAS Y RESULTADOS

4.1. Datos de entrenamiento y de validación

Es importante antes de entrenar la RNA separar los datos de entrenamiento y los datos de validación, con los datos de entrenamiento es que se hará los ajustes necesarios para que la red pueda aprender, una vez terminado el aprendizaje la red puede generalizar y dar respuestas correctas ante patrones de estímulos nuevos y para validar que la RNA es capaz de clasificar correctamente se somete a una validación con los datos de validación.

La división de los datos o conversaciones se realizó de la siguiente manera, figura 4.1.

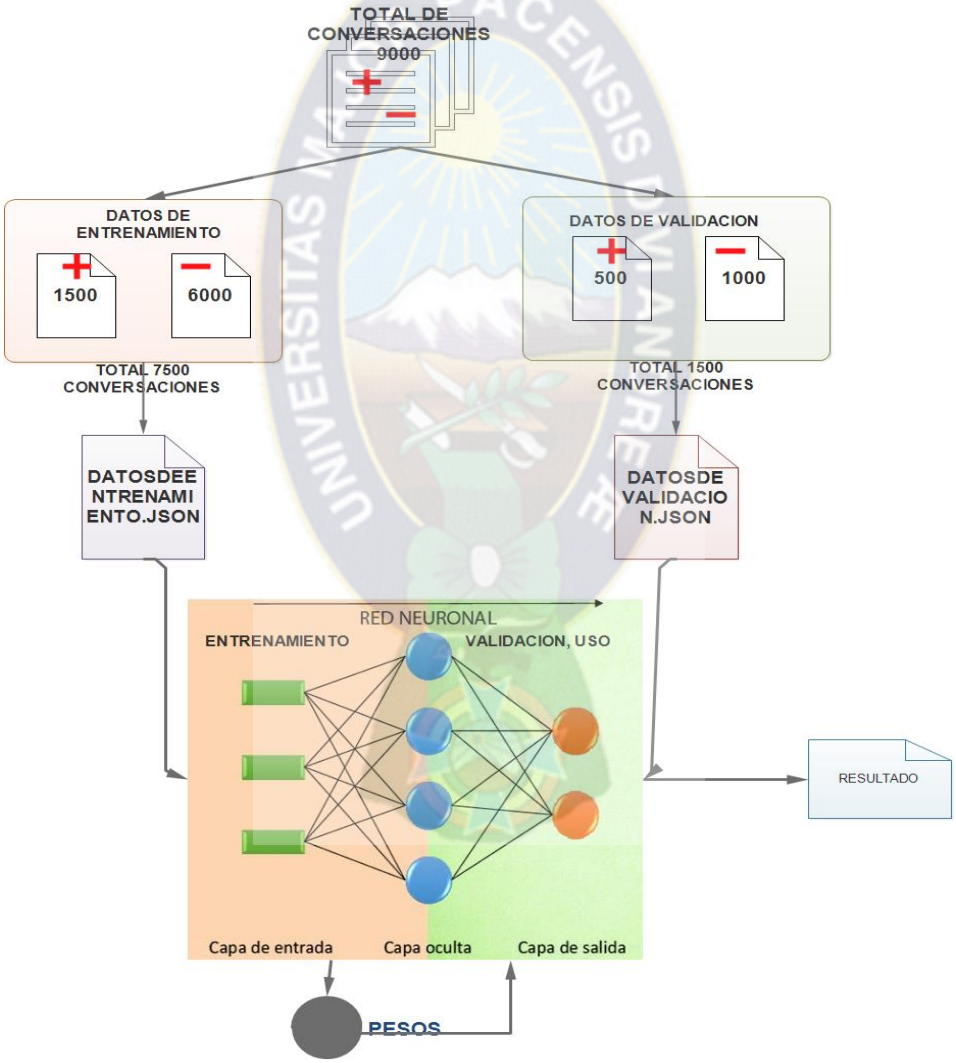


Fig. 4.1. División de Datos
Fuente (Elaboración Propia)

4.2. Pruebas de entrenamiento de la RNA en la clasificación de conversaciones

En la fase de diseño y entrenamiento de la RNA que se vio en el anterior capitulo se mencionan ciertos parámetros como el número de iteraciones, el número de capas ocultas y el valor de la taza de aprendizaje que son determinantes para que la RNA tenga un correcto aprendizaje.

Para la construcción y entrenamiento de la RNA se utilizó Tensorflow conjuntamente con Google Cloud ML Engine que permite usar la potencia, flexibilidad y escalabilidad de sus servidores para poder entrenar los modelos de aprendizaje automático sin dificultad en cuanto a Hardware.

En la figura 4.2. el proceso que se sigue para entrenar un modelo en Google Cloud ML Engine con Tensorflow.

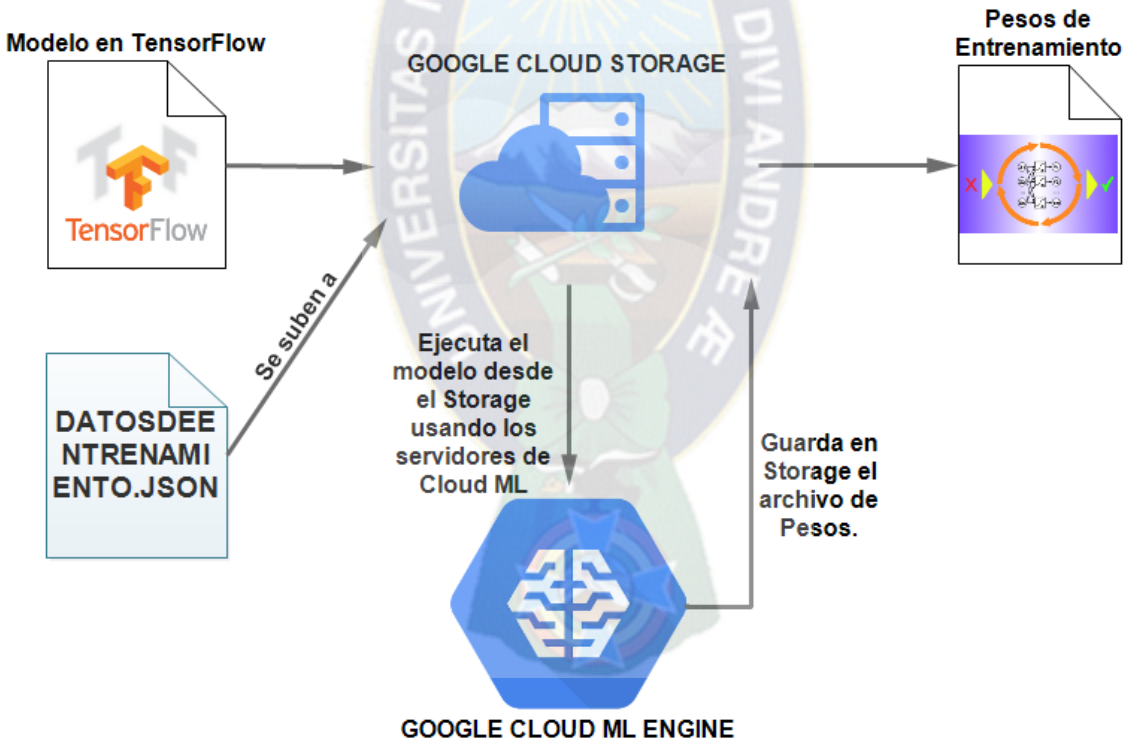


Fig. 4.2. Proceso de entrenamiento en Google Cloud ML Engine con Tensorflow
Fuente (Elaboración Propia)

Al momento de decidir los parámetros de la RNA mencionados anteriormente se tuvo que hacer diferentes pruebas cambiando los valores de esos parámetros para encontrar aquellos

con los cuales la RNA aprende de una manera correcta. Se realizó más de 90 pruebas entre ellas se incluye modelos que generan errores, pero de esas pruebas se tiene 50 pruebas validas que permiten determinar ciertos valores para la red. A continuación, se muestra solo algunas pruebas, las más importantes que influyeron al momento de seleccionar los parámetros.

4.2.1. Pruebas con cambio de parámetros

Se mencionó anteriormente que el número de datos o conversaciones que se empleó para entrenar es de 7500, pero pruebas con esa dimensión llegan a ser muy extensas en tiempo, por tal razón al momento de hacer las pruebas se elige muestras pequeñas y se va subiendo el número de conversaciones hasta dar con los parámetros correctos y definidos con el total de las conversaciones de entrenamiento. Se prueba por primera vez con los siguientes datos:

Numero de conversaciones: 200 **Numero de Iteraciones:** 10000 **Tasa de aprendizaje:** 1.0 **Numero de capas ocultas:** 10 **Modelo:** Booleano

Tabla 4.1 Prueba 1
Fuente (Elaboración Propia)

Los resultados que se obtuvieron con el primer entrenamiento son reflejados en la siguiente figura 4.3:

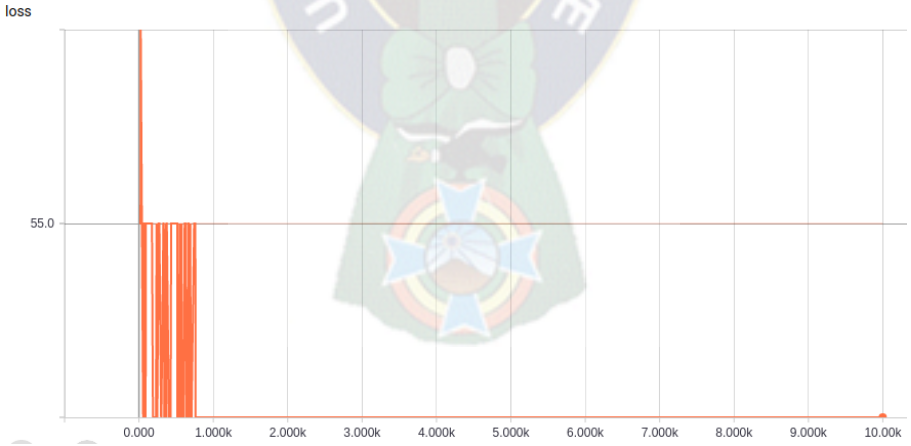


Fig. 4.3. Grafica de la Primera Prueba
Fuente (elaboración Propia)

En la gráfica que se muestra el error no logra minimizarse debido a que no existe precisión al momento de ajustar los pesos el error empieza con 55 y no minimiza nada.

Para corregir el problema de la minimización para la segunda prueba se reduce la taza de aprendizaje con los siguientes parámetros:

Numero de conversaciones: 200 **Numero de Iteraciones:** 10000 **Tasa de aprendizaje:** 0.1 **Numero de capas ocultas:** 10 **Modulo:** Booleano

Tabla 4.2. Prueba 2
Fuente (Elaboración Propia)

Resultados:

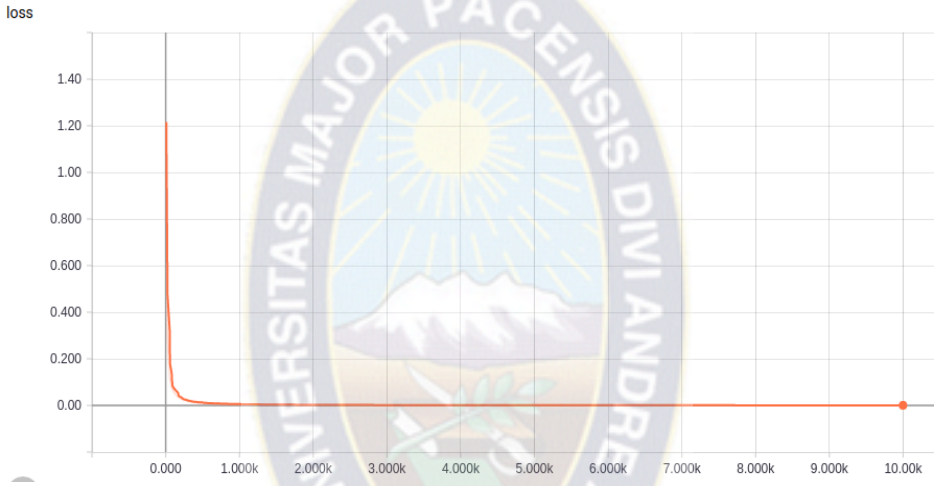


Fig. 4.4. Grafica de aproximación de la segunda prueba
Fuente (Elaboración Propia)

En el nuevo grafico se puede notar claramente que se minimiza el error hasta un mínimo de 4.88e-4. De esta manera se realizan sucesivas pruebas aumentando el número de conversaciones.

Numero de conversaciones: 3000 **Numero de Iteraciones:** 40000 **Tasa de aprendizaje:** 0.0001 **Numero de capas ocultas:** 13 **Modulo:** Vectorial TF-IDF

Tabla 4.3. Prueba 3
Fuente (Elaboración Propia)

Resultados:

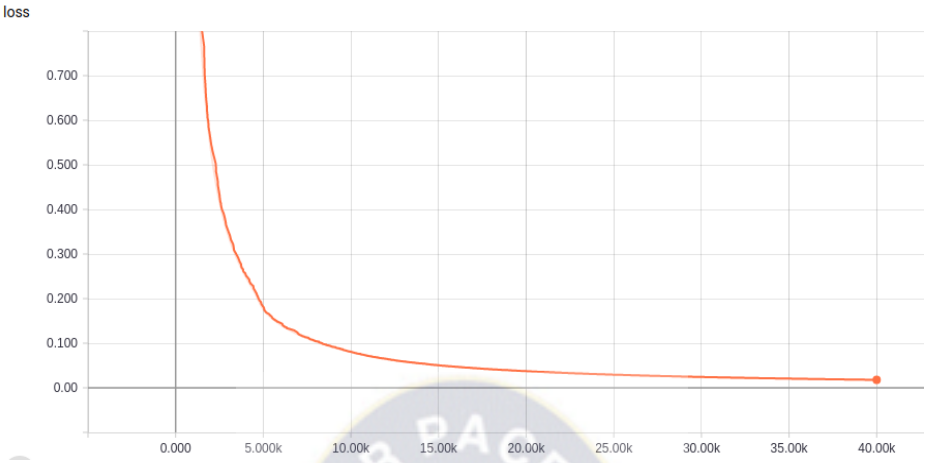


Fig. 4.5. Grafica de la tercera prueba modelo TF-IDF
Fuente (Elaboración Propia)

El error se minimizo hasta 0.02 con las 3000 conversaciones por lo cual se determinó que la taza de aprendizaje aun no es precisa. En la siguiente tabla, una de las últimas pruebas con los 7500 casos de entrenamiento.

Numero de conversaciones: 7500
Numero de Iteraciones: 300000
Tasa de aprendizaje: 0.00005
Numero de capas ocultas: 15
Modulo: Booleano

Tabla 4.4. Prueba 50
Fuente (Elaboración Propia)

Resultados:

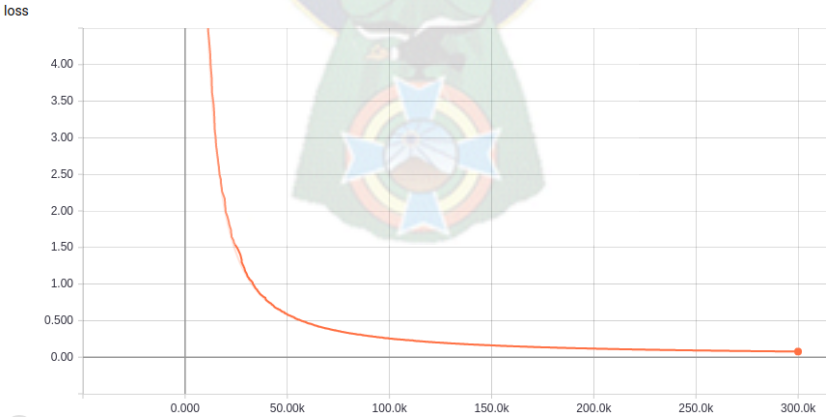


Fig. 4.6. Grafica de la prueba 50 error minimizado

Fuente (Elaboración Propia)

En los últimos resultados se abarco los datos de entrenamiento minimizando a un error de 0.015.

En la figura 4.7 se puede contemplar los detalles de la tarea en Google Cloud Machine Learning Engine, este indica un total de 6 días con 16 horas de trabajo empleados en el aprendizaje de las conversaciones con los parámetros de la tabla 4.4.

Las pruebas fueron ejecutadas en los servidores de Google usando TensorFlow, los tiempos de las pruebas es proporcional al número de iteraciones, numero de capas ocultas y el número de entradas que tiene cada prueba. En la siguiente figura 4.8. las tareas ejecutadas en Cloud ML Engine.

Las tareas marcadas con el color verde son las que se entrenaron correctamente sin problemas, las de color rojo son las tareas que tuvieron un error de código, memoria o algún otro y se detuvieron y los de color gris son aquellos que se han parado manualmente, en la última celda está el tiempo de trabajo en la ejecución.

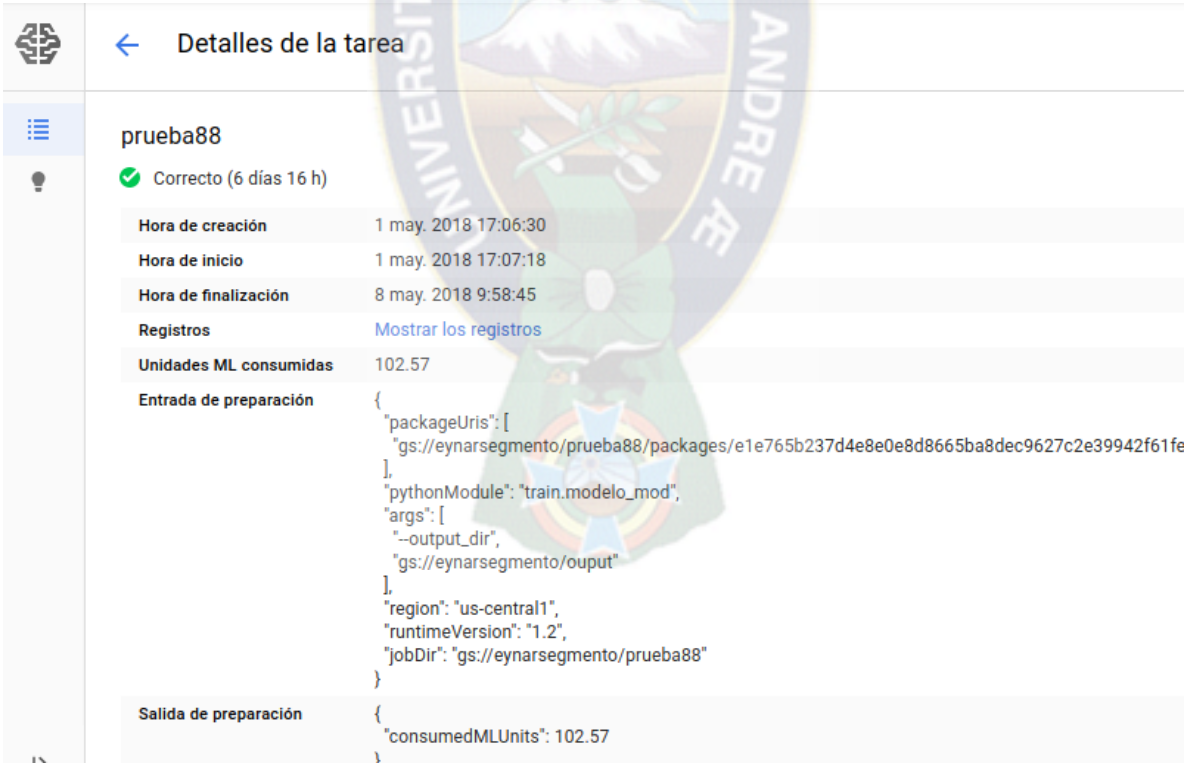


Fig. 4.7. Detalle del entrenamiento con Google Cloud ML Engine
Fuente (Elaboración Propia)

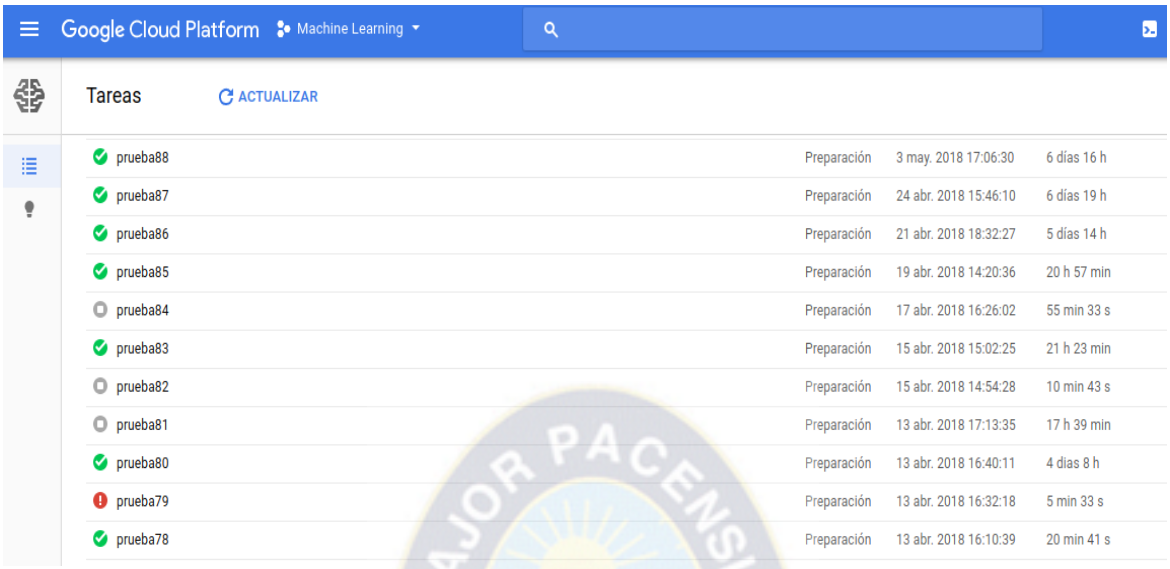


Fig. 4.8. Tareas Ejecutas en Google Cloud ML engine
Fuente (Elaboración Propia)

4.3. Evaluación del rendimiento en la clasificación de textos

Tras la fase de entrenamiento viene la fase de ejecución, durante la que se pidió a la red que responda a estímulos diferentes a los presentados durante la fase de entrenamiento. Gracias a los ejemplos aprendidos del juego de ensayo, la red deberá ser capaz de generalizar y dar respuestas correctas ante patrones de estímulos nuevos.

En otras palabras, una vez terminado el aprendizaje, una red puede generalizar; es decir, ante entradas similares a las de su juego de ensayo, producirá salidas correctas. Hay que tener en cuenta que es muy difícil conseguir la capacidad de generalización de una red sin utilizar grandes cantidades de datos y que estos sean muy variados (Gestal,2010).

En esta fase las evaluaciones para el clasificador de conversaciones pedófilas se la deben realizar de manera experimental, en lugar de analíticamente. La evaluación experimental de los clasificadores, en lugar de concentrarse en cuestiones de Eficiencia, por lo general trata de evaluar la efectividad de un clasificador, es decir, su capacidad de tomar las decisiones correctas de categorización. (Vandana y Namrata, 2012)

Para medir el rendimiento del clasificador de conversaciones pedófilas se usan las mismas medidas que se usa en cualquier clasificador de textos como ser precisión y exhaustividad.

4.3.1. Precisión y Exhaustividad para la Clasificación

Es una métrica empleada en la medida del rendimiento de los sistemas de búsqueda y recuperación de información y reconocimiento de patrones [Gómez Díaz,2003]. En este contexto se dice que la precisión es la proporción de material recuperado realmente relevante, mientras que la exhaustividad es la proporción de material relevante recuperado, del total de los documentos.

En la figura 4.9. se muestra la forma de aplicar la precisión y exhaustividad en el área de Aprendizaje Automático para clasificadores binarios, de esta manera se tiene las medidas del diagnóstico de evaluación del modelo.

		Actual Class y		
		Positive	Negative	
$h_\theta(x)$ **Test outcome**	Test outcome positive	**True positive** (TP)	**False positive** (FP, Type I error)	Precision = $\frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}}$
	Test outcome negative	**False negative** (FN, Type II error)	**True negative** (TN)	Negative predictive value = $\frac{\#\text{TN}}{\#\text{FN} + \#\text{TN}}$
		Sensitivity = $\frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}}$	Specificity = $\frac{\#\text{TN}}{\#\text{FP} + \#\text{TN}}$	Accuracy = $\frac{\#\text{TP} + \#\text{TN}}{\#\text{TOTAL}}$

Fig. 4.9. Evaluación de precisión y exhaustividad
Fuente (Gómez Díaz,2003)

4.3.2. Evaluación de Precisión y Exhaustividad del modelo Booleano

A continuación, se ve los resultados de evaluación obtenidos del modelo de clasificación de conversaciones con contenido pedófilo en la tabla de precisión y exhaustividad para el modelo booleano:

	Positivos	Negativos	
Resultados de prueba positivos	492	8	0.984

	Positivos	Negativos	
Resultados de prueba negativos	11	989	0.989
	0.97813	0.99197	0.98733

Tabla 4.5. Evaluación de Precisión y exhaustividad del modelo Booleano
Fuente (Elaboración Propia)

4.3.3. Evaluación de Precisión y Exhaustividad del modelo Vectorial TF-IDF

A continuación, se ve los resultados de evaluación obtenidos del modelo de clasificación de conversaciones con contenido pedófilo en la tabla de precisión y exhaustividad para el modelo vectorial de TF-IDF:

	Positivos	Negativos	
Resultados de prueba positivos	494	6	0.988
Resultados de prueba negativos	7	993	0.993
	0.98602	0.99399	0.99133

Tabla 4.6. Evaluación de Precisión y Exhaustividad del modelo Vectorial TF-IDF
Fuente (Elaboración Propia)

Precisión:

La precisión intenta responder a la siguiente pregunta:

¿Qué proporción de identificaciones positivas fue correcta?

La precisión se define de la siguiente manera para el modelo Booleano:

$$Precision = \frac{TP}{TP + FP} = \frac{492}{492 + 8} = 0.984$$

Y para el modelo Vectorial TF-IDF:

$$Precision = \frac{TP}{TP + FP} = \frac{494}{494 + 6} = 0.988$$

El primer modelo tiene una precisión de 0.984 y el segundo modelo tiene 0.988 mejorando solo un poco la precisión del primer modelo, entonces se puede afirmar que usando el modelo TF-IDF al momento de detectar conversaciones con contenido pedófilo, se acierta el 99 % de las veces.

Exhaustividad:

La exhaustividad intenta responder a la siguiente pregunta:

¿Qué proporción de positivos reales se identificó correctamente?

La exhaustividad se define de la siguiente manera para el modelo Booleano:

$$Exhaustividad = \frac{VP}{VP + FN} = \frac{494}{492 + 11} = 0.9781$$

Y para el modelo Vectorial TF-IDF:

$$Exhaustividad = \frac{VP}{VP + FN} = \frac{494}{494 + 7} = 0.9860$$

De igual manera que en la precisión, la exhaustividad en el segundo modelo es más efectiva al tener un valor de 0.9860 en comparación con el primer modelo que tiene 0.9781, usando el modelo de TF-IDF, se puede decir que identifica correctamente el 98% de las conversaciones con contenido pedófilo.

F1-SCORE

Una medida que combina precisión y recuperación es la media armónica de precisión y recuperación, la medida tradicional F o la puntuación F equilibrada:

Esta medida es aproximadamente el promedio de los dos cuando están cerca, y es más generalmente la media armónica [David M. W., 2012]. Está definida de la siguiente manera.

$$F = \frac{2 * Precision * Exhaustividad}{Precision + Exhaustividad}$$

Para el modelo Booleano se tiene:

$$F = \frac{2 * 0.984 * 0.9781}{0.984 + 0.9781} = 0.9810$$

Y para el modelo Vectorial TF-IDF se tiene:

$$F = \frac{2 * 0.988 * 0.986}{0.988 + 0.986} = 0.9869$$

De esa manera con TF-IDF, se tiene un promedio mayor al de del Booleano y se puede afirmar que el sistema inteligente puede detectar conversaciones con contenido pedófilo usando Redes Neuronales con una precisión de 98.8% y una exhaustividad de 98.6%.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- Al finalizar el trabajo de investigación, se pudo demostrar que el Sistema Inteligente para la detección de conversaciones con posible contenido pedófilo, basado en Redes Neuronales, realiza la detección de este tipo de conversaciones con un error de 1.31%, el cual es aceptable, y se concluye que el modelo elaborado efectivamente puede detectar conversaciones con contenido pedófilo en el idioma Ingles.
- Una tarea que trajo complicaciones al momento de realizar esta investigación fue la de recolectar los documentos de texto con las conversaciones positivas y negativas, es claro que nadie querría compartir sus conversaciones privadas ante el público, pero gracias a la organización de Perverted-Justice y grupos de Linux se pudo lograr el objetivo de recopilar estos Documentos.
- Se concluye que el pre-procesamiento de las conversaciones textuales, pueden ser tratadas para su posterior análisis, integrando técnicas de Minería de Texto y Procesamiento del Lenguaje Natural, disciplinas que generalmente se trabajan de manera independiente.
- Se ha diseñado una Red Neuronal de tipo perceptrón multicapa con alimentación hacia adelante que ha sido capaz de aprender de 7500 conversaciones, todas en el idioma inglés, y detectar nuevas conversaciones que puedan tener contenido pedófilo.
- Desde el aspecto social se propone un modelo de Red Neuronal que puede ser aplicado en la detección de pedófilos que se encuentran en las redes sociales y así proteger a los niños.

5.2. Recomendaciones

- Se recomienda trabajar con el modelo de Perceptrón Multicapa para el reconocimiento de patrones de texto ya que se puede lograr buenos resultados con márgenes de error muy pequeños.
- Se recomienda poder recolectar mayor cantidad de datos de conversaciones y que estas sean cuidadosamente seleccionadas ya que de ello depende el éxito del entrenamiento de la Red Neuronal.
- Se recomienda la extracción de nuevas características semánticas que no hayan sido consideradas en este trabajo, de tal manera que pueda tomarse en cuenta otros aspectos de la conversación, como la hora o fechas en que se dieron dichas conversaciones.

Bibliografía

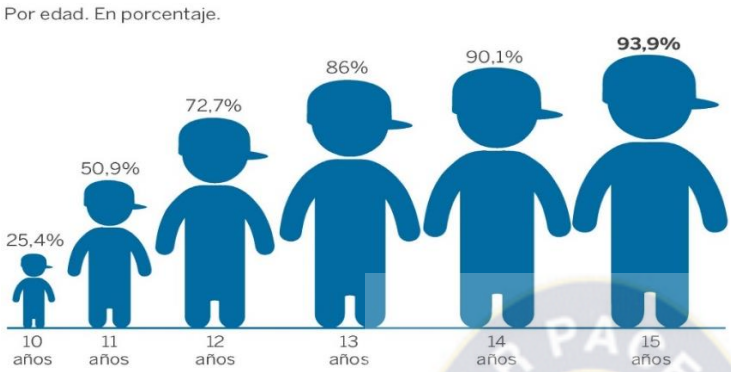
- Aleman, Y., vilariño, D., pinto, D., y tovar, M. (2014). *Detección de Depredadores Sexuales Utilizando un Sistema de Consulta y Clasificación Supervisada* (Tesis de grado), Benemérita Universidad Autónoma de Puebla, Puebla, México.
- Alacmeon, (2012, 26 de abril). *Revista Argentina de Clínica Neuropsiquiatrica* vol. 17, N°3, Argentina, P. 268 a 276
- Albanese, D., Merler, G., Jurman, S., y Visintainer, R. (2008). *MLPy: high-performancePython package for predictive modeling. In NIPS, MLOSS workshop*,
- Beltrán Gómez, A. y Ordóñez Salinas, S. (2014). *Sistema inteligente para la detección de diálogos con posibles contenidos pedofílicos*. Revista Virtual Universidad Católica del Norte, 42, 164-181.
- Bronshtein I., Semendiaev K. (1973*). Manual de matemáticas para ingenieros y estudiantes.* 2da Edición. Moscú, Editorial Mir.
- Dubois, F. (2007). *Python: batteries included, volume 9 of Computing in Science &Engineering. IEEE/AIP*.
- Feldman, R. Sanger, J. (2006). *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructed Data*. CandBridge University, Reino Unido.
- Gestal, P. M., (2010). *Introducción a las Redes de Neuronas Artificiales*. Universidad de Coruña. España.
- Greemberg, D., bradford, J., y Curry, S., (1993a). *Treatment of the paraphilic disorders: Areview of te role of the selective serotonin reuptake inhibitors. Sexual Abuse*. Vol.9 No.4.
- Greemberg, D., bradford, J., y curry, S.*:* (1995b). *A comparison of treatment of pharaphilias with three serotonin reuptake inhibitors: A retrospective study. Bulletin of the American Academy of Psychiatry and the law*. Vol. 11, No. 15
- Gonzalo, p. M.-s., y de la Cruz Garcia, J. M. (2010). *Aprendizaje automático.* Barcelona, España: Ra-Ma.

- Goodfellow, I., y Bengio, Y. (2016) *Deep Learning* Massachusetts Institute of technology, paginas 777, Estados Unidos de America.
- Kao, A. y Stephen, R. P, (2007), Springer Science+ Busines Media. LLC. “Natural Language Processing and Text Mining” 259paginas. Springer- Verlag London Limited
- Korde, V. y Namrata, M. C. (2012). Text Classification anda Classifiers a Survey, International Journal of Artificial Intelligence & Applications (IJAIA). Vol. 3, No. 2, India
- Klimosky, G. (1994). *Las desaventuras del conocimiento científico Una Introducción a la epistemología*, buenos Aires, Argentina: A-Z editora, pag 133.
- Laorden-Gomes, C., Galan-Garcia P., Santos-Grueiro I., Garcia-Bringas, P., Sanz-Urquijo, B., y Gomez-Hidalgo, J. (2013). *NEGOBOT: Agente conversacional Basado en Teoria de Juegos para la Deteccion de Conductas Pedofilas*. Recuperado de http://paginaspersonales.deusto.es/patxigg/publicaciones/2012/Negobot.pdf
- Martinez, C. J. A., (2008), *Los modelos clásicos de Recuperación de la información y su vigencia*. Universidad complutense de Madrid, España.
- Matich-Jorge, D., (2001). *Redes Neuronales: Conceptos Básicos y aplicaciones*. Universidad Tecnológica Nacional. Rosario, Argentina.
- Murillo, W. (2008 18 de abril). *La investigación científica*. Recuperado de http//www.monografias.com/trabajos15/invest-científica/investcientífica.shtm
- McCulloch, W. S. y Pitts, W. (1943). "*A logical calculus of the ideas immanent in neurons activity", Bull. Math. Biophys*., 5, 115-133p.
- Penda, N. (2007). *Toward spotting the pedophile telling victim from predator in text chats. In: Proceedings of the International Conference on Semantic Computing*. ICSC '07, IEEE Computer Society, Washington, DC, USA. pp. 235-241
- Ponce, C. P. (2010). *Inteligencia Artificial con Aplicaciones a la Ingeniería, Matlab anda Simultak.*

- Russell. S., y Norving, P., (2004). *Inteligencia Artificial un enfoque moderno.* 2° edición, Madrid, Pearson Prentice Hall editorial, paginas 1240.
- Sanchez, G., Perez, H., y Nakano, M., (2004). Growing Cell Neuronal Network Using Simultaneous Perturbation. D. F. México, México, Vol. 15. No. 5. P45-52.
- Salkind, N. J., (1999), *Métodos de Investigación*, México, Prentice Hall Hispanoamérica S.A., pp. 380
- Salton, G., Wong. A., y Yang, C. S. (1973) *A Vector Space Model for Automatic Indexing*. Monterrey Editor, Information Retrieval and Lenguage Processing, p.620.
- Salton, G. y McGill, M. J. (1986). *Introduction to modern information retrieval*. McGraw-Hill. España.
- Sparck, K. J., (1972). *A statistical interpretation of term specificity and its application in retrieval*", Journal of Documentation, MCB UP Ltd. Vol. 28 Issue: 1, pp.11-21, University of Cambridge, Cambridge. Reino Unido.
- Sholom, M. y Indurkhya, W. N. (1995). *Rule-based Machine Learning Methods for functional Prediction, Journal of Artificial Intelligence Research*.
- Villatoro, T. E., Juarez, G. A., Escalante, H. J., Montes-y-Gomez, M., Pineda, L.V. (2012), A *two-step approach for efective detection of misbehaving users in chats. In: CLEF (Online Working Notes/Labs/Workshop)*, Mexico.
- Valens, O. P. (2016). *Diseño de Herramienta Para Detectar Usuarios Depredadores en Facebook*, Universidad de San Buenaventura Cali, Colombia.

ANEXOS

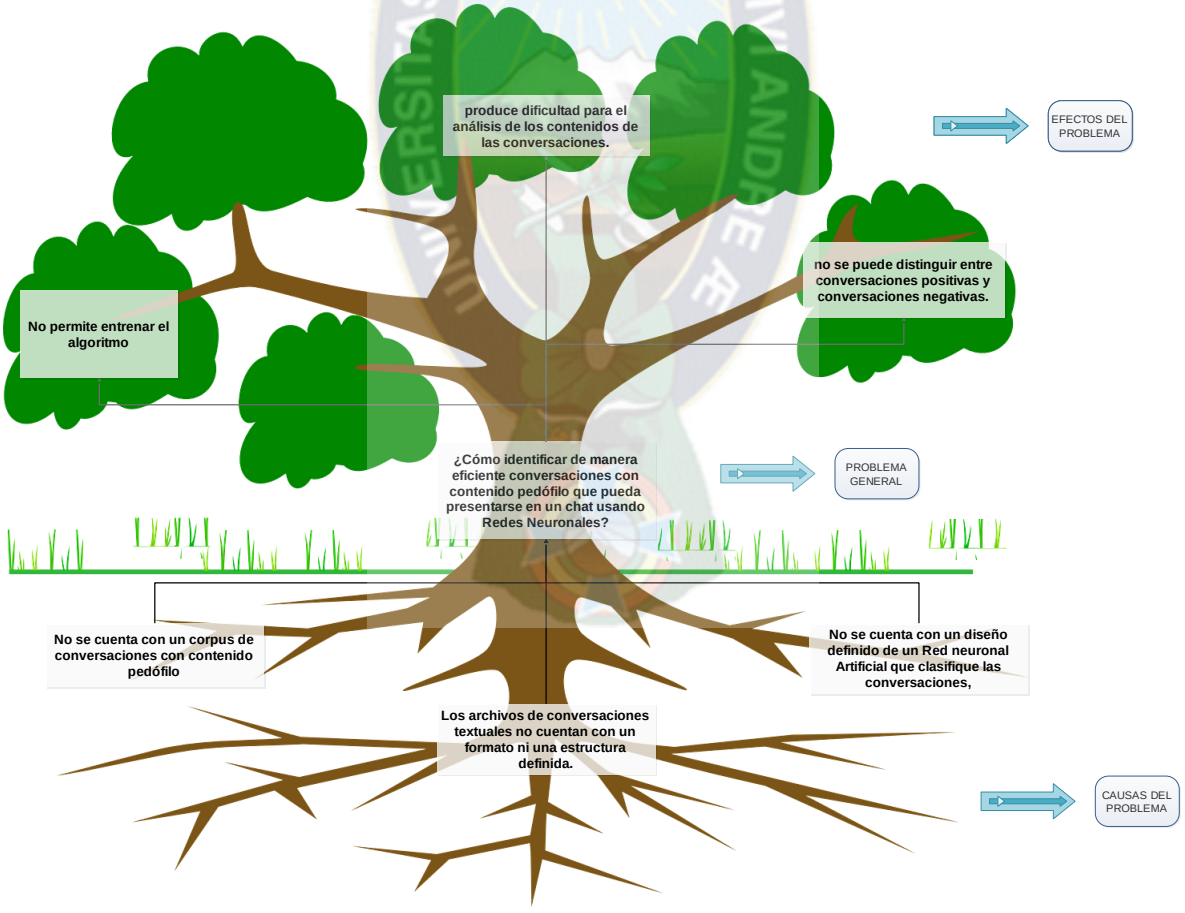
Anexo A



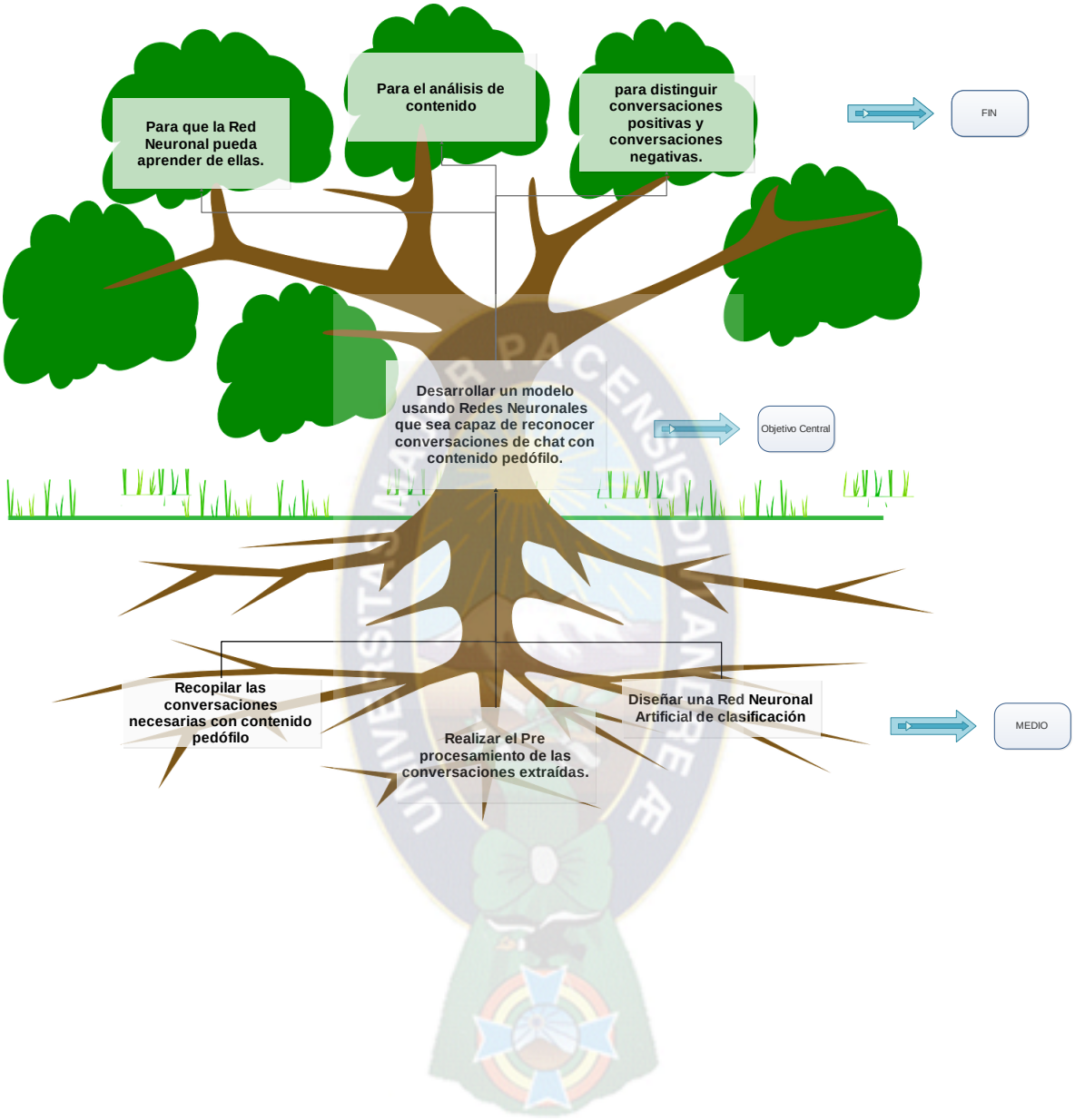
gráfica 1. estadísticas de usuarios menores de edad en internet 2017

fuente [mediaclick.es, 2017]

Anexo B



Anexo C


Para que la Red Neuronal pueda aprender de ellas.
Para el análisis de contenido
para distinguir conversaciones positivas y conversaciones negativas.
FIN
Desarrollar un modelo usando Redes Neuronales que sea capaz de reconocer conversaciones de chat con contenido pedófilo.
Objetivo Central
Recopilar las conversaciones necesarias con contenido pedófilo
Realizar el Pre procesamiento de las conversaciones extraídas.
Diseñar una Red Neuronal Artificial de clasificación
MEDIO