universidad de león

Escuela de Ingenierías Industrial,
Informática y Aeroespacial

Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

Análisis de técnicas de Procesamiento de Lenguaje Natural y Aprendizaje Automático para detección de contenido erótico en textos con computación en la nube.

Analysis of Natural Language Processing and Machine Learning techniques for erotic content detection in texts with cloud computing.

Autor: Gonzalo Molpeceres Barrientos
Tutor: Rocío Alaiz Rodríguez
Tutor: Víctor González Castro

(Junio, 2019)

UNIVERSIDAD DE LEÓN
Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA
Trabajo de Fin de Grado

ALUMNO: Gonzalo Molpeceres Barrientos

TUTOR: Rocío Alaiz Rodríguez

TUTOR: Víctor González Castro

TÍTULO: Análisis de técnicas de Procesamiento de Lenguaje Natural y Aprendizaje Automático para detección de contenido erótico en textos con computación en la nube.

TITLE: Analysis of Natural Language Processing and Machine Learning techniques for erotic content detection in texts with cloud computing.

CONVOCATORIA: Junio, 2019

RESUMEN:

Con el auge de tecnologías relacionadas con la inteligencia artificial se están consiguiendo automatizar tareas que previamente se realizaban de forma manual, tanto mediante el desarrollo de algoritmos que generen cierta automatización, aunque gobernada y controlada por los seres humanos, como con trabajo directo de seres humanos. Paralelamente, al haber un rápido desarrollo en tecnologías de comunicación, surgen ciertos problemas en cómo la gente se comunica, desde ataques como insultos y comentarios abusivos hasta la creación de contenido censurable como podría ser por ejemplo contenido sexual (videos sexuales, textos eróticos), contenido violento (relacionado con agresiones o incitación al odio) o hasta contenido ilegal (videos de asesinatos, abuso de menores). Este trabajo se centra en la detección de contenido inapropiado escrito en un ámbito de lenguaje natural, más concretamente contenido erótico. Tras una búsqueda de métodos para lograr este objetivo, este trabajo se centra

en el aprendizaje automático, más concretamente en aprendizaje supervisado. Se estudiarán y aplicarán tres extractores de características: Bag Of Words (BOW), Term Frequency, Inverse Document Frequency (TF-IDF) y Word2Vec, además de cuatro clasificadores: Support Vector Machines (SVM), Logistic Regression (LR), K-Nearest Neighbours (KNN) y Random Forest (RF). Se han probado todas las combinaciones de extractor y clasificador además de un proceso de computación en la nube llamado Amazon Comprehend, el cual realiza la función de extracción de características y clasificación automáticamente. Para hacer las pruebas fue necesario crear un conjunto de datos basado en datos públicos de la página web Reddit. Los resultados fueron bastante positivos, llegando a un de tasa de aciertos de 97.12% con TF-IDF + SVM (con kernel lineal) solo siendo superado por Amazon Comprehend, con una tasa de aciertos de 98.51%. Para finalizar, en base a dichos resultados, se han valorado posibles aplicaciones de este trabajo, como su uso para desarrollo de aplicaciones reales de detección de contenido inapropiado como para punto de apoyo para otros trabajos relacionados con esta temática.

ABSTRACT:

With the rise of technologies related to artificial intelligence, tasks that were previously carried out manually were being automated, both through the development of algorithms that generate some automation, although governed and controlled by human beings, as well as by direct work of human beings. In parallel, as there is a rapid development in communication technologies, certain problems arise in how people communicate, problems like attacks such as insults and abusive comments or the creation of censurable content such as, for example, sexual content (sexual videos, erotic texts), violent content (related to aggressions or incitement to hatred) or even illegal content (videos of murders, child abuse). This work focuses on the detection of written inappropriate content in a context of natural language, more specifically erotic content. After a search of methods to achieve this objective, this work focuses on machine learning, more specifically on supervised learning. Three feature extractors will be studied and applied: Bag Of Words (BOW), Term Frequency, Inverse Document Frequency (TF-IDF) and Word2Vec, as well as four classifiers: Support Vector Machines (SVM), Logistic Regression (LR), K- Nearest Neighbors (KNN) and Random Forest (RF). All combinations of extractor and classifier have been tested, in addition to a cloud computing process called Amazon Comprehend, which performs the feature extraction and classification function automatically. To do the tests it was necessary to create a dataset based on public data from the Reddit website. The results were quite positive, reaching a accuracy of 97.12% with TF-IDF + SVM (with linear kernel) only been exceeded by Amazon Comprehend, with an accuracy of 98.51%. Finally, based on these results, possible applications of this work have been evaluated, such as its use for the development of real applications for the detection of inappropriate content as a point of support for other works related to this topic.

Palabras clave: Natural Language Processing (NLP), Text classification, Machine Learning, BOW, TF-IDF, Word2Vec, SVM, Logistic Regression, Random Forest, KNN, AWS, Comprehend.

Firma del alumno:	**VºBº Tutor/es:**

Índice

Índice de figuras

Índice de cuadros y tablas

1. Introducción

El avance de las tecnologías en el campo de la inteligencia artificial está permitiendo que tareas antes desarrolladas con algoritmos escritos directamente por seres humanos se sustituyan por algoritmos generados mediante un proceso en el que se intenta simular e imitar el comportamiento y razonamiento humano. Para ello, los sistemas que desarrollen dichos algoritmos, deben ser capaces de percibir estímulos de diferentes formas, razonar ante ellos y planear cuál puede ser el método más óptimo para alcanzar un objetivo propuesto. Estas tareas se llevan a cabo mediante diferentes tecnologías, yendo desde, por ejemplo, Visión Artificial y Procesamiento de Lenguaje Natural (Natural Language Processing o NLP) para poder percibir diferentes señales y convertirlas en datos, hasta diferentes tipos de Aprendizaje Automático (Machine Learning) como Redes Neuronales o Support Vector Machines, encargados de analizar los datos y de ellos extraer conclusiones

Este trabajo se ha centrado en el Procesamiento de Lenguaje Natural de textos para su posterior detección de contenido inapropiado, en este caso contenido de carácter sexual o erótico.

Este objetivo ha sido seleccionado debido a la necesidad de automatizar tareas que, a día de hoy, realizan seres humanos de una forma no demasiado eficaz, por razones como pueden ser el tiempo en el que realiza la tarea en comparación a un proceso automatizado, el coste que supone para las empresas, el diferente sesgo dependiendo de la persona e incluso el factor psicológico que interviene al moderar cierto contenido dependiendo de la persona (como en el caso de videos, casos de asesinatos o abuso infantil).

Buscando una parte de esta automatización, como se ha dicho previamente, este trabajo se centra en la clasificación automatizada de contenido erótico en texto, aunque las ideas que se plantean en este trabajo podrían servir de punto de apoyo para otro tipo de procesos que complementen el objetivo primordial de detección de contenido inapropiado (como

detección de otro tipo de contenido inapropiado en texto o incluso en otro formato como videos o imágenes, por ejemplo).

Para ello se hará uso del Procesamiento de Lenguaje Natural, valorando diferentes técnicas para extraer características claves (Feature Extraction) de lenguaje natural como Bag Of Words (BOW) [1], Term Frequency, Inverse Document Frequency (TF-IDF) [2] y Word2vec [3] las cuales se pasarán a un sistema de Aprendizaje Automático con Aprendizaje Supervisado, usando como modelos de clasificación Support Vector Machines (SVM) [4], Logistic Regression (LR) [5], K-Nearest Neighbors (KNN) [6] y Random Forest (RF) [7].

Debido a lo costoso que es en el aspecto de hardware y tiempo de computación realizar un entrenamiento con un conjunto de datos de gran tamaño, también se estudiará la utilización de un sistema de computación en la nube, AWS Comprehend [8] que realiza el proceso de extracción de características y clasificación a la vez, completamente transparente para el usuario, y se compararán los resultados obtenidos.

Es necesario un buen conjunto de datos para llevar a cabo este proceso, y en este caso, debido a no encontrar ningún Dataset público adecuado, se creará uno a partir de datos públicos de la página web Reddit[1], filtrando el texto que deseemos y separándolos en contenido erótico y neutral (siguiendo el proceso que se muestra en el Anexo I).

Siguiendo estos objetivos, a continuación, en el Capítulo 2 “Estado del arte” se revisará la literatura existente sobre este tema para, a continuación, en el Capítulo 3 “Metodología” exponer los métodos utilizados en este trabajo. A continuación, los experimentos realizados para evaluar dichos métodos se exponen en el Capítulo 4 “Experimentación”, cuyos resultados se mostrarán en el Capítulo 5 “Resultados”. Finalmente, en el Capítulo 6 “Conclusiones” se expone una revisión de todo el proceso y unas conclusiones sobre los resultados obtenidos.

[1] https://www.reddit.com/

2. Estado del arte

Para un mejor entendimiento de las técnicas utilizadas en clasificación de texto y el procesamiento de lenguaje natural que este conlleva se han tomado en cuenta estudios previos y los resultados de estos.

Muchas investigaciones están basadas en técnicas tradicionales como las que en este trabajo se emplean, pero también se han buscado trabajos que incluyeran el resto de las técnicas utilizadas.

2.1. PROCESAMIENTO DE LENGUAJE NATURAL

La literatura referente al procesamiento de lenguaje natural es muy amplia debido a que esta es una rama de la inteligencia artificial muy compleja.

El NLP (Natural Language Processing, Procesamiento de Lenguaje Natural en español) ha ganado mucha repercusión debido a su función de representar y analizar el lenguaje humano de forma computacional. Es una tecnología que se ha extendido por diferentes campos como el análisis de sentimientos, análisis de contexto, la detección de Spam, contestación de preguntas a nivel usuario, analizar y resumir informes médicos, y codificación del lenguaje [9].

La tarea que se busca realizar en este trabajo es la extracción de información que pueda ser codificada para que las máquinas lo entiendan.

Siendo un problema planteado desde los años 50 e irresoluble por aquella época, la historia del NLP tuvo un avance muy importante a partir de 1980, pues hasta el momento la mayoría de procesos eran llevados a cabo mediante reglas escritas manualmente, pero con la llegada del aprendizaje automático (Machine Learning), muchos modelos estadísticos dieron unos resultados muy avanzados.

A partir de ese momento, la adición de diferentes tecnologías como N-grams[10] o la implementación de redes neuronales fueron ayudando a que cada vez este proceso fuera más y más preciso, acercándose cada vez más al entendimiento total por parte de una máquina del lenguaje que los humanos usan en entornos reales [11].

2.2. CLASIFICACIÓN DE TEXTO

La clasificación de texto se define como la asignación de una frase a su correspondiente categoría, pudiendo esta variar según el objetivo de la clasificación.

Para ello se hace uso de dos tecnologías diferentes, ambas ramas de la inteligencia artificial, el procesamiento de lenguaje natural y el aprendizaje automático.

El aprendizaje automático (Machine Learning) estudia algoritmos que permiten a los sistemas aprender a través de datos que reciben, hallando de esta manera formas de generalizar ante datos que no hayan visto antes y estimarles un resultado.

La principal ventaja es que, gracias al entrenamiento al que son sometidos, no es necesario programarlos explícitamente para una tarea, siendo esto muy útil para problemas cuya resolución no puede ser abordara mediante la creación de algoritmos debido a la complejidad intrínseca del mismo.

Para resolver el problema de la clasificación de texto se siguen unas etapas que llevarán del archivo que se pretende clasificar hasta una decisión de clasificación de este. Los pasos que se han de seguir son la obtención de datos y el preprocesamiento de estos (adaptarlos a una forma más amigable, pero sin cambiar su contenido), la extracción de características clave de los datos que sirvan para a continuación el entrenamiento de un clasificador, encargado de tomar la decisión final.

La obtención de datos es un proceso clave, ya que un conjunto de datos que no sea representativo arruinará toda posibilidad de obtener un resultado satisfactorio. Este conjunto

de datos puede estar previamente clasificado o no, depende del tipo de clasificador que queramos utilizar, supervisado a no supervisado.

La extracción de características es la parte que, hoy en día, está menos avanzada en este proceso, siendo la encargada de la parte más clave y que diferencia este problema de otros relacionados con inteligencia artificial, la codificación del lenguaje humano.

Diferentes estudios como los encontrados en [12], [13] y [14] proponen el uso de técnicas tradicionales de extracción de características como Bag of Words (BOW) y Term Frequency – Inverse Document Frequency (TF-IDF) combinándolas con clasificadores como Support Vector Machine (SVM), Naïve Bayes, Logistic Regression o Árboles de Decisión.

Los resultados suelen ser positivos, la mayoría sobrepasando el 90% de precisión en la clasificación, pero concluyendo que la complejidad radica sobre todo en la extracción de características, como se dijo previamente, y no en la clasificación en sí, ya que si se tuviera un conjunto de características que representen totalmente el contenido del texto la clasificación en si se vuelve mucho más sencilla.

Es por ello que en trabajos como [15], [16] y [17] proponen el uso de modelos de espacio vectorial (más concretamente el modelo Word2Vec), los cuales tienen la capacidad de generar Word Embedding y con ello dotar a las características de muchos más matices, como podría ser un entendimiento del contexto en el que se encuentran, relaciones con otras palabras o parte del significado de la palabra en sí misma.

El método revisado en este trabajo será Word2Vec y se compararán sus resultados con BOW y TF-IDF.

2.3. APLICACIÓN DE COMPUTACIÓN EN LA NUBE

La computación en la nube es el término con el que se define cualquier proceso informático llevado a cabo en servidores a través de internet en vez de en el equipo local

Pese a no tener ninguna relación directa en su concepción con ninguna de las ramas de la inteligencia artificial, la computación en la nube se está convirtiendo en un pilar importante de esta gracias a la facilidad que otorga al usuario y su bajo coste asociado, pudiendo además escalar los servicios rápidamente ante una alta demanda.

En este trabajo se revisará Amazon Web Services (AWS) un importante proveedor de computación en la nube y su servicio dedicado a NLP, Amazon Comprehend, pese a que todos sus productos suelen estar enfocados en un aspecto comercial más que de investigación.

2.4. TEXTO CON CONTENIDO ERÓTICO

Siendo el objetivo último de este trabajo la correcta clasificación de texto con contenido erótico, se ha hecho una búsqueda de literatura existente hasta el momento, siendo que, pese a que no hay nada tan concreto como lo que se busca, sí que hay estudios que avalan la viabilidad de esta clasificación al estudiar los resultados de problemas similares como puede ser la clasificación de contenido violento en texto [18] o el análisis de sentimientos [19] [20].

Pese a numerosas leyes que obligan a mantener un control sobre cierto contenido inapropiado en la web (como acceso a páginas pornográficas o a ciertas redes sociales), es muy difícil conseguir una efectividad total en el cumplimiento de dichas reglas.

Una de las ramas más difíciles de controlar y que pasa más desapercibida es el contenido erótico en textos, sin necesidad de contenido multimedia. Es por eso importante llegar a un resultado satisfactorio en este trabajo, debido a que estos resultados se podrían aplicar, por ejemplo, para complementar proyectos que tienen como objetivo la detección de contenido no apropiado para menores y la protección de estos ante un contenido más enfocado para adultos.

3. Metodología

3.1. CONJUNTO DE DATOS

A la hora de entrenar cualquier sistema de aprendizaje automático es importante disponer de un buen conjunto de datos o Dataset que represente bien todas las posibles opciones que dicho sistema se pueda encontrar para que así aprenda de ellas. Así mismo, es importante que disponga de una cantidad suficiente de ejemplos para establecer patrones y generalizar bien ante casos que no se hayan visto en el entrenamiento.

Ante la falta de un conjunto de datos público que dispusiera de estas características para el objetivo de este trabajo, fue necesario crear uno personalizado. Para ello se utilizaron de base una serie de conjunto de datos públicos de la página web Reddit que contienen todos los datos de todas las publicaciones separadas por meses, filtrándolos según lo que se necesita para crear un conjunto de datos separado en dos categorías: Erótico y Neutral.

El proceso de filtrado y creación del conjunto de datos se detalla en el Anexo I.

3.2. EXTRACTORES DE CARACTERÍSTICAS

Es importante, a la hora de enfrentarse a un problema de clasificar texto que es lenguaje natural, tener en cuenta que no es algo que se pueda hacer de madera directa, el texto ha de ser transformado del lenguaje humano un lenguaje codificado que pueda ser procesado por el clasificador.

Para ello se hace uso de los extractores de características, los cuales, aplicando diferentes técnicas, analizan cuáles son las características más representativas de un texto y les asigna un peso.

Así pues, el texto es transformado en una lista mapeada de términos o características con un peso diferente según la importancia que posee el término dentro del texto de entrada, siendo finalmente este valor numérico el que emplearán los clasificadores.

Es importante mencionar que, dependiendo de cómo sea el texto de entrada, puede ser necesario o recomendable someterlo a un preprocesamiento. Este preprocesado varía dependiendo de cómo sea dicha entrada, pudiendo ir desde eliminar etiquetas HTML si el texto proviene de una página web, hasta eliminar las palabras más comunes de un idioma que no otorgan significado al texto (conocidas como Stop-Words).

En este trabajo además se hará uso de "N-grams" en todos los extractores de características, técnica que consiste en coger las palabras de N en N y analizarlas como una sola, añadiendo así un posible significado más complejo que por separado no tienen (p.e. "Madre tierra" no tiene nada que ver con "Madre" y "Tierra" por separado).

A continuación, se revisarán los modelos de extracción de características usados en este trabajo: Bag of Word, Term Frequency – Inverse Document Frequency y Word2Vec.

3.2.1. **BAG OF WORDS (BOW)**

La primera forma que utilizaremos para representar el lenguaje natural será Bag of Words (BOW), técnica definida por primera vez por Zellig Harris en 1954 en un artículo sobre estructura de distribución [1].

BOW es una técnica tradicional de extracción de características que consiste en la representación del texto mediante el número de ocurrencias de las palabras en el documento.

Para ello, se hace uso de dos elementos: un vocabulario de las palabras conocidas y la medida de cuánto aparecen estas.

Así, en el proceso de entrenamiento, se añaden al vocabulario aquellas palabras que no existen y a la hora de pasar una frase por el modelo ya creado, este devolverá un vector contabilizando la frecuencia de cada palabra respecto al vocabulario.

Ejemplo:

Modelo entrenado con las frases:

- "mi perro es el azul"
- "el cielo es azul"
- "yo no veo como es mi perro"

Tendrá un vocabulario tal que: "mi", "perro", "es", "el", "azul", "el", "cielo", "yo", "no", "veo", "como"

Al recibir una frase como: "mi perro es azul como el cielo azul" se codificará de la siguiente manera:

Tabla 3.1 Ejemplo BOW

mi	perro	es	el	azul	cielo	yo	no	veo	como
1	1	1	1	2	1	0	0	0	1

Devolviendo un vector tal que [1, 1, 1, 1, 2, 1, 0, 0, 0, 1]

3.2.2. **TERM FREQUENCY, INVERSE DOCUMENT FREQUENCY (TF-IDF)**

La segunda técnica escogida es Term Frequency, Inverse Document Frequency (TF-IDF), propuesta por primera vez en 1972 por Karen Sparck Jones [2] en su artículo en el que sugiere que las ocurrencias en documentos deberían tratarse estadísticamente respecto a otros documentos en vez de individualmente.

La novedad en este método fue la aparición del IDF, factor que mide la importancia del término en todo el conjunto de documentos, contrastando con el TF, que solo mide la

frecuencia de aparición de un término en el propio documento. Gracias a la combinación de ambos se obtiene un valor que disminuye el peso de los términos que aparecen en muchos documentos y que no aportan valor sobre su contenido, y aumenta el peso de aquellos que aparecen en pocos documentos pero que les otorgan significado.[2]

Así, por ejemplo, el peso *w* de una palabra *x* en el documento *y* se calcularía como la frecuencia de *x* en *y* multiplicado por el logaritmo de la fracción inversa de la frecuencia de documentos que contienen el termino *x*.

$$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term *x* within document *y*

$tf_{x,y}$ = frequency of *x* in *y*
df_x = number of documents containing *x*
N = total number of documents

Figura 3.1 Funcionamiento TF-IDF [3]

De esta forma, esta técnica devolverá también un vector (con posiciones equivalentes al vocabulario, igual que en BOW) con los pesos de las diferentes palabras.

Ejemplo:

Siguiendo con el ejemplo de BOW, se tenían las frases:

- "mi perro es azul"
- "el cielo es azul"
- "yo no veo como es mi perro"

[2] Fuente: [22] http://www.tfidf.com/
[3] Fuente: https://mropengate.blogspot.com/2016/04/tf-idf-in-r-language.html

Con un vocabulario tal que: “mi”, “perro”, “es”, “el”, “azul”, “cielo”, “yo”, “no”, “veo”, “como”

En este ejemplo, tres frases actúan como los documentos, y tenemos el documento de “mi perro es azul como el cielo azul” de entrada codificándose de la siguiente manera:

Tabla 3.2 Ejemplo TF-IDF

	mi	perro	es	el	azul	cielo	yo	no	veo	como
TF	1/8	1/8	1/8	1/8	2/8	1/8	0/8	0/8	0/8	1/8
IDF	$\log\left(\frac{3}{2}\right)$	$\log\left(\frac{3}{2}\right)$	$\log\left(\frac{3}{3}\right)$	$\log\left(\frac{3}{2}\right)$	$\log\left(\frac{3}{2}\right)$	$\log\left(\frac{3}{1}\right)$	$\log\left(\frac{3}{1}\right)$	$\log\left(\frac{3}{1}\right)$	$\log\left(\frac{3}{1}\right)$	$\log\left(\frac{3}{1}\right)$
w	0.02	0.02	0	0.02	0.04	0.06	0	0	0	0.06

Devolviendo el vector: [0.02, 0.02, 0, 0.04, 0.06, 0.06, 0, 0, 0, 0.06]

3.2.3. **Word2Vec**

Modelo de espacio vectorial enfocado en la creación de Word Embedding y creado en 2013 por un grupo de investigación de Google liderado por Tomas Mikolov [3]

Para definir Word2Vec hay que empezar por entender el concepto de Word Embedding, el cual se podría definir como una forma de representar el vocabulario de un documento mediante representaciones de las palabras en forma de vector, pudiendo capturar así contexto de la palabra, similitudes semánticas y sintácticas, relaciones con otras palabras, e incluso parte de su significado.

Esta perspectiva de representar las palabras le otorga otra dimensión de significado y permite relacionar palabras que con otras técnicas no tendrían ninguna relación conjunta.

Un ejemplo visual de Word Embedding se puede observar en la Figura 3.2, en la cual vemos como se comparan relaciones entre conceptos que, aunque sin tener nada que ver los conceptos que se relacionen, a comparación entre ellos mismos están a la misma distancia el uno del otro (Hombre-Mujer, Presente-Pasado, País-Capital).

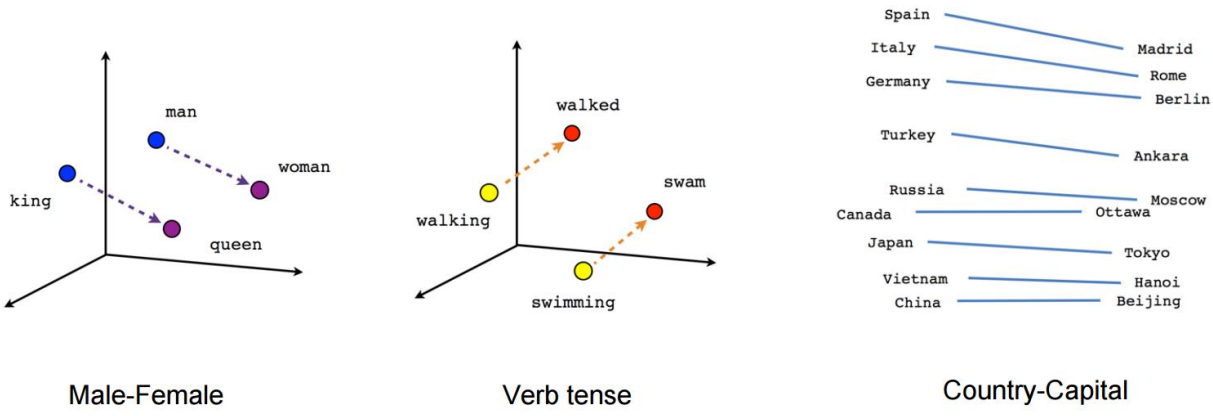

Figura 3.2 Representación Word Embedding.[4]

Una vez se tiene claro la definición de Word Embedding, se puede definir Word2Vec como un método para crear un Word Embedding mediante una Shallow Neural Network (Red Neuronal con pocas capas ocultas), pudiendo crearse con dos métodos diferentes: CBOW (Continuos Bag Of Words) y Skip-gram.

La principal diferencia entre ambos métodos es que mientras CBOW intenta predecir la palabra a través del contexto, Skip-gram está diseñado para predecir el contexto a través de la palabra.

La ventaja de Skip-gram es que trabaja bien con pequeñas cantidades de datos de entrenamiento, representando bien incluso palabras y frases extrañas, mientras que CBOW destaca en su velocidad de entrenamiento y en su precisión con palabras muy frecuentes.

[4] Fuente: [23] http://hunterheidenreich.com/blog/intro-to-word-embeddings/

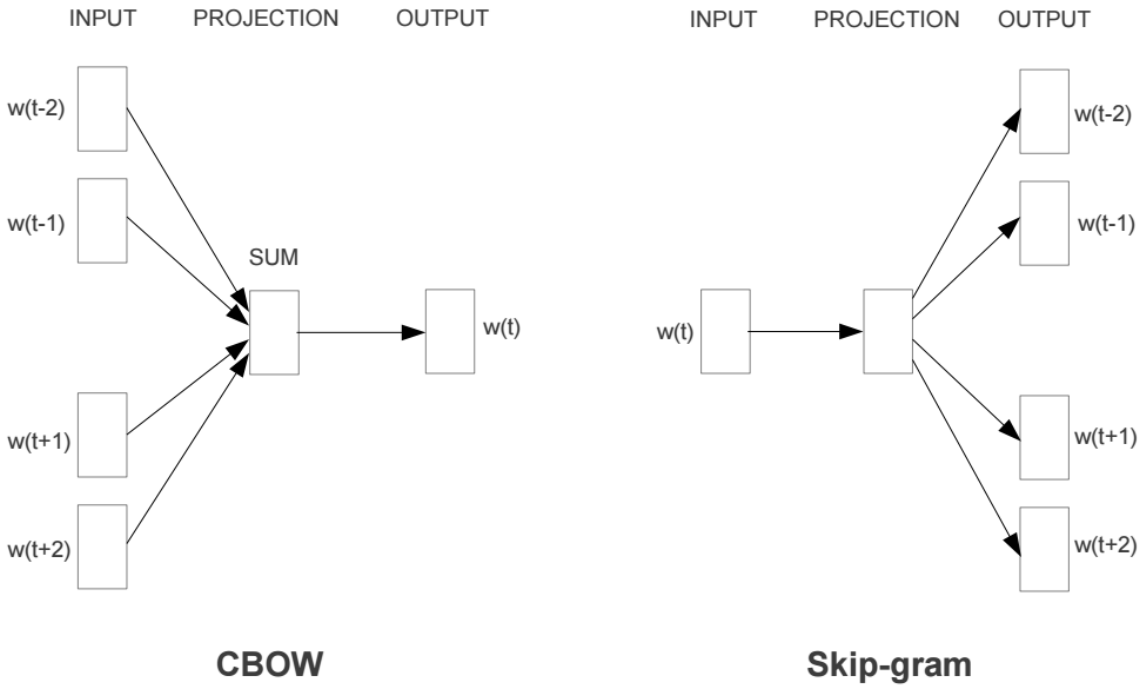

Figura 3.3 Representación de cómo trabajan CBOW y Skip-gram[5]

3.3. CLASIFICADORES

Una vez el lenguaje natural ha sido codificado por alguno de los métodos anteriores es posible pasar al proceso de aprendizaje automático.

En él, nos centraremos en la técnica de aprendizaje supervisado, la cual consiste en el entrenamiento del modelo con datos ya etiquetados previamente con el objetivo de ir calibrando poco a poco un modelo para que sea capaz de clasificar mediante generalizaciones casos diferentes a aquellos con los que ha entrenado.

El aprendizaje supervisado se puede dividir en dos categorías según sea el problema a resolver:

[5] Fuente: [24] https://rohanvarma.me/Word2Vec/

- Clasificación: En el cual la salida ha de ser una variable discreta dentro de un conjunto previamente planteado en el entrenamiento, como en el ejemplo de este trabajo, el cual realiza una clasificación de dos categorías (“Erótico” y “Neutral”)
- Regresión: En este caso la salida debe ser una variable continua, ajustándose el algoritmo lo más posible a esta. Un ejemplo podría ser el cálculo de una medida como peso, temperatura o cantidad de un valor.

A continuación, se revisarán los algoritmos de clasificación usados en este trabajo.

3.3.1. SUPPORT VECTOR MACHINES (SVM)

Propuesto por primera vez en 1995 por Corinna Cortes y Vladimir Vapnik [4], este modelo realiza la clasificación encontrando el hiperplano que maximice la distancia entre las clases.

Debido a que no todas las clasificaciones son linealmente separables, se puede aplicar una transformación al conjunto de datos que consiga que se pueda separar de esa manera. A estas transformaciones se les llama kernels, y en este trabajo se utilizarán el kernel lineal y el kernel Radial Basis Function (RBF). Con los resultados que obtengan estos dos diferentes kernels además se podrá comprobar si el ejemplo de este trabajo es linealmente separable o no.

Es importante recalcar que la función del kernel no es alterar el método de cálculo de los SVM, siendo que cualquiera que sea el kernel, la división entre clases será lineal igualmente, pero el kernel transformará el conjunto de datos para poder hacer que un problema no linealmente separable se pueda separar con ese método en otra dimensión (como en el ejemplo de la Figura 3.4).

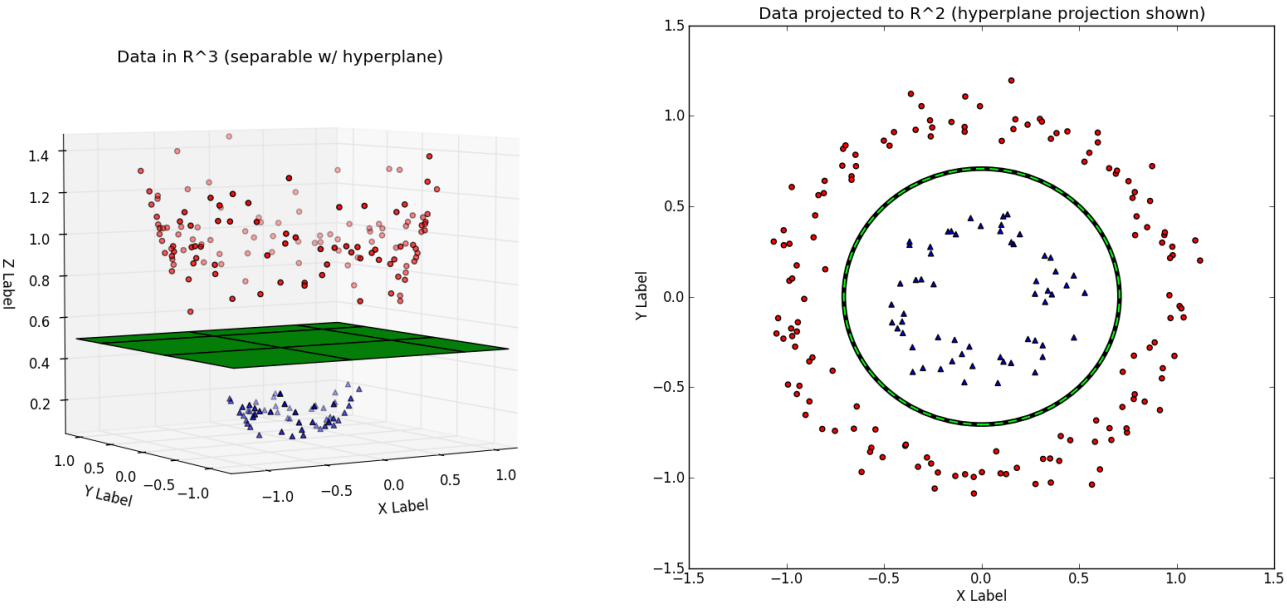

Figura 3.4 Ejemplo de funcionamiento de kernels en SVM[6]

Además de los kernels, un valor importante de los SVM es la regularización (normalmente representado como C), ya que este valor nos indica cuan estrictos queremos ser a la hora de clasificar algunos ejemplos mal, y que cuanto más alto, menos lejos permitirá que los datos estén de su zona separada por el hiperplano.

Otro parámetro importante es gamma, aplicable solo en kernels con separación no lineal, y que indica el límite en el que queremos que un dato influya al calcular el hiperplano, siendo que cuanto más bajo sea gamma, más lejano será el límite en el que tome en cuenta para el cálculo.

En la Figura 3.5 podemos observar las fronteras de decisión formadas por los hiperplanos situados por SVM con los dos tipos de kernels que se utilizarán.

[6] Fuente: [25] https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

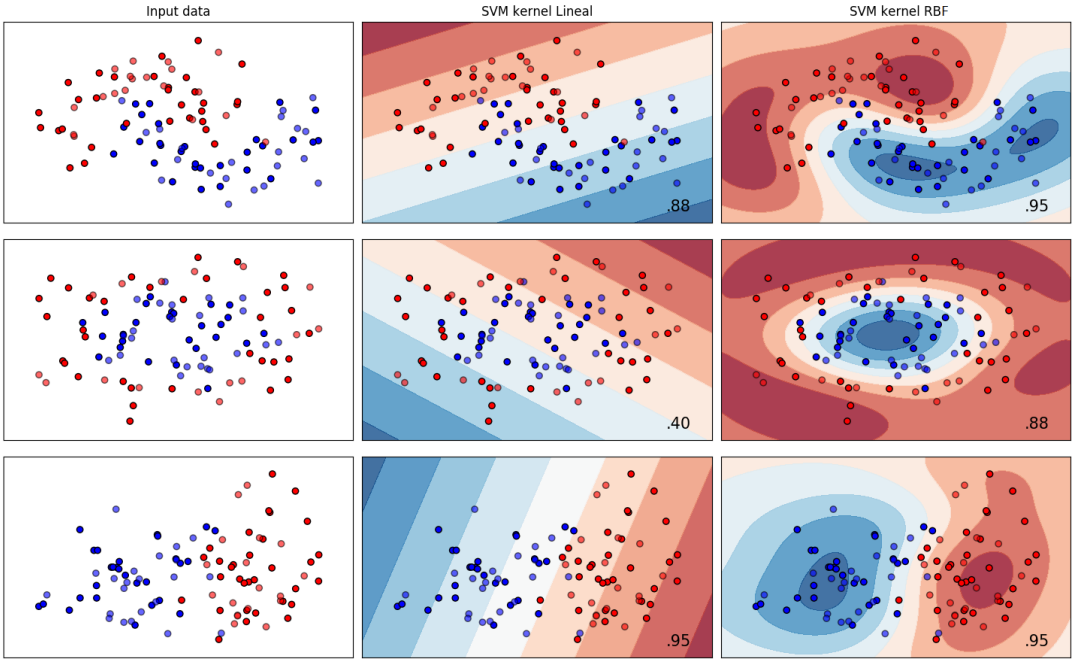

Figura 3.5 Ejemplo de los dos kernels en SVM

3.3.2. **LOGISTIC REGRESSION (LR)**

Algoritmo propuesto por David Cox en 1958 [5] basado en la Regresión lineal, este método halla la probabilidad de que una determinada variable sea de una clase u otra.

Este cálculo se realiza aplicando los datos de salida de una función lineal a una función sigmoidal, reduciéndolos así al rango [0,1]. Una vez el modelo esté entrenado, para clasificar se valorará la entrada respecto a la curva creada y se clasificará en la clase que corresponda, comprándolo con un umbral seleccionado.

En el ejemplo de la Figura 3.6 podemos ver como resultarían las curvas en el caso de valorar el eje X e Y de los datos. Este cálculo se realizaría igualmente en ejemplos con más dimensiones, como se puede ver en la Figura 3.7, la cual representa una curva tridimensional que se crearía con una entrada de dos dimensiones.

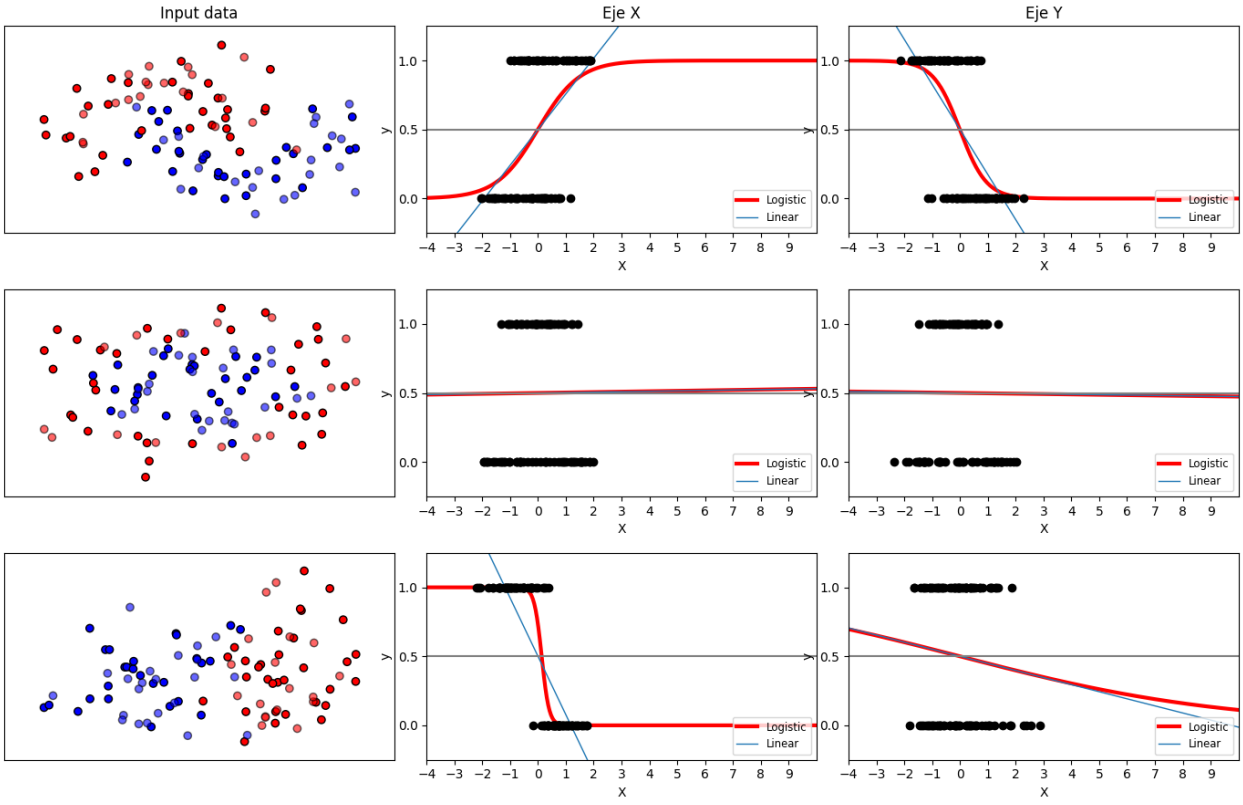

Figura 3.6 Ejemplo curvas Logistic Regression

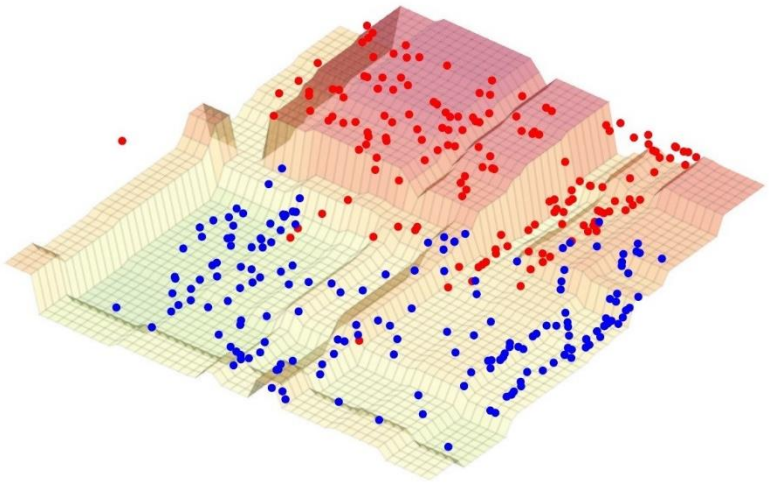

Figura 3.7 Ejemplo curva tridimensional Logistic Regression[7]

[7] Fuente: [26] https://towardsdatascience.com/plotting-decision-boundaries-in-3d-logistic-regression-and-xgboost-e68ce0535b4b

Con los resultados obtenidos con las curvas logísticas al entrenar el modelo, ya se podría crear además una frontera de decisión como la mostrada en la Figura 3.8

En este modelo de clasificación vuelve a ser importante el parámetro de regularización C, indicando que cuanto mayor sea, menor regularizado estará y más estricto será en el entrenamiento con todos los datos.

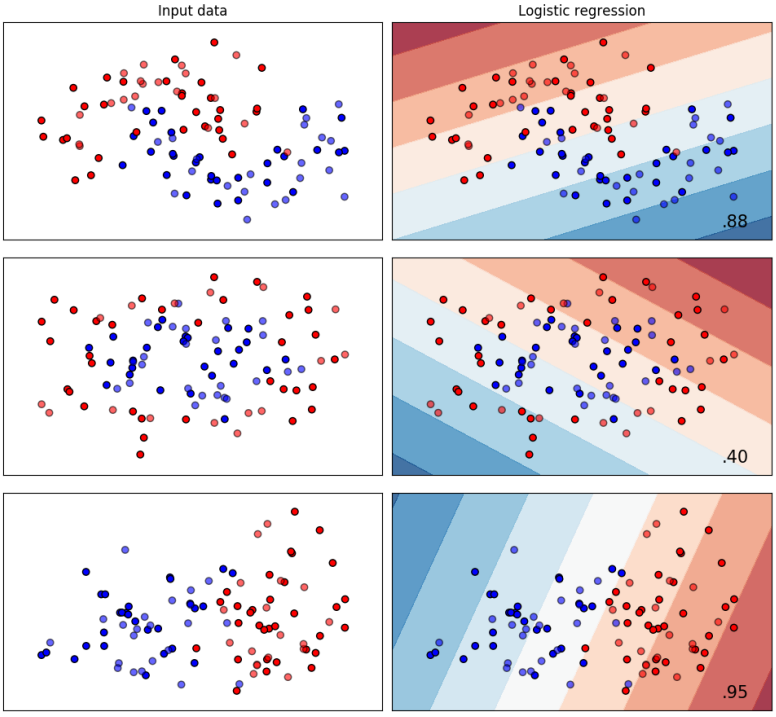

Figura 3.8 Ejemplo fronteras de decisión Logistic Regression

3.3.3. **K-NEAREST NEIGHBORS (KNN)**

Planteamiento desarrollado por James Keller en 1985 [6], KNN propone la asignación de un dato según sea la categoría de los K elementos más cercanos.

Definido como un “algoritmo vago”, KNN carece de generalización, lo cual se podría entender como que KNN no saca conclusiones del significado que puede tener los datos de

entrenamiento, simplemente compara las características recibidas y valora su similitud con el resto.

Siendo un algoritmo bastante simple, el parámetro más importante es el número de vecinos en sí mismo, K, pues cuanto mayor sea más generalista será y más suave será la separación entre las clases.

En la Figura 3.9 se puede apreciar como quedaría la frontera de decisión resultante una vez se hubieran analizado todos los datos de entrenamiento.

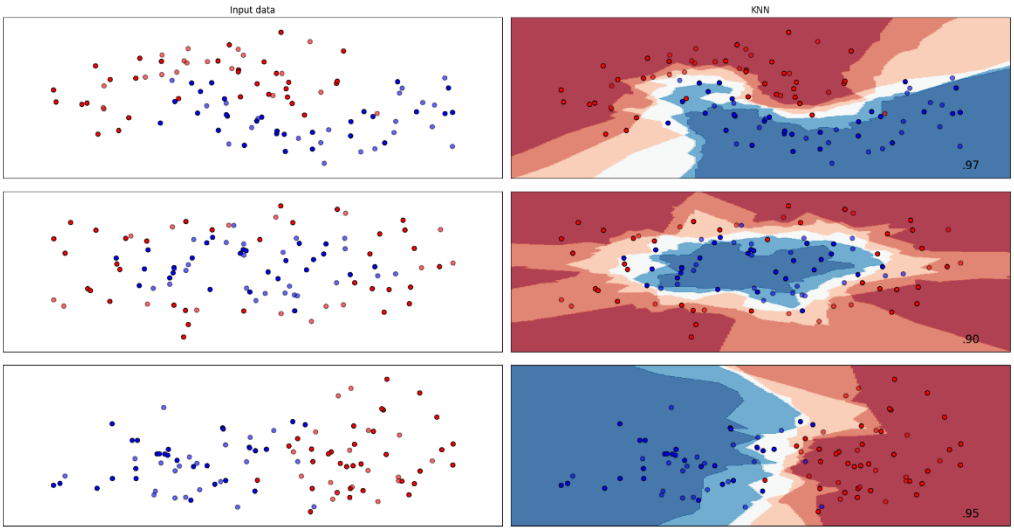

Figura 3.9 Ejemplo KNN

3.3.4. **RANDOM FOREST (RF)**

Basado en los árboles de decisiones, este algoritmo fue desarrollando y extendido por Leo Breiman en 2001 [7].

Para entender el funcionamiento del clasificador Random Forest hay que entender primero el concepto de árbol de decisión, el cual se podría definir como un modelo predictivo basado en una serie de ramificaciones de decisiones mediante características concretas para llegar a una conclusión más generalizada [21].

En la Figura 3.10 se puede apreciar un árbol de decisión simple para una entrada *X* con características determinadas (X_2 y X_5) y una salida *Y*.

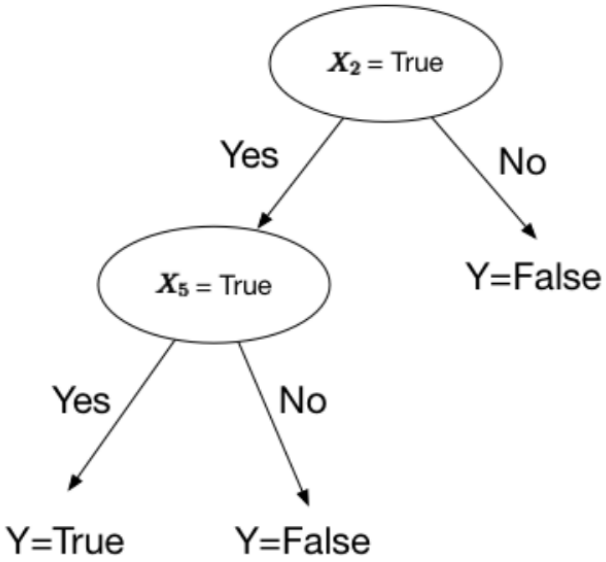

Figura 3.10 Ejemplo árbol de decisión

Sabiendo esto, en un modelo Random Forest se combinan múltiples árboles de decisión, haciendo que cada uno de ellos otorgue una clasificación individual y así valorar cual ha sido la decisión más común.

En este trabajo se han variado dos parámetros a la hora de realizar la clasificación: el número de árboles totales y la profundidad máxima de los árboles.

En la Figura 3.11 se pueden observar las fronteras de decisión resultantes del cálculo con Random Forest.

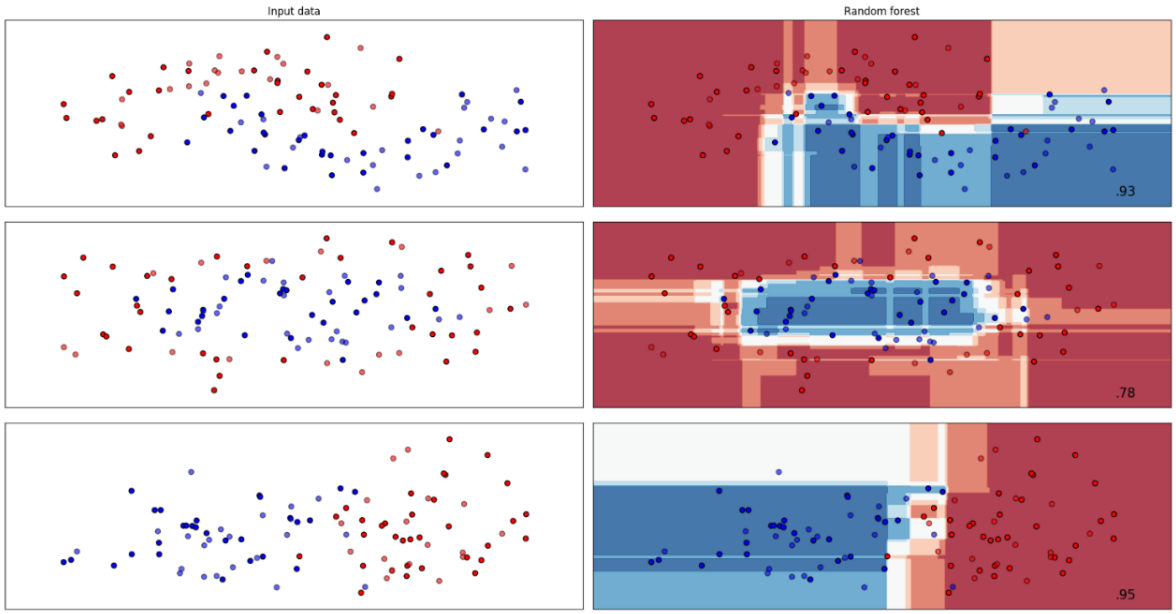

Figura 3.11 Ejemplo Random Forest

3.4. AMAZON COMPREHEND

Amazon Comprehend es una herramienta creada por Amazon Web Services con funciones de procesamiento de lenguaje natural.

Tiene funciones de reconocimiento de entidades, palabras clave, lenguaje, sentimientos y elementos comunes en un documento.

Una función clave, y será la que se usa en este trabajo, es la llamada "clasificación personalizada de Comprehend", la cual, mediante tratamiento de lenguaje natural y aprendizaje supervisado, creará un modelo ya entrenado listo para clasificar directamente documentos en lenguaje natural.

La única condición necesaria para entrenar el modelo es crear un documento con el formato adecuado, el cual consiste en un fichero en formato CSV tal que en cada línea esté un elemento con su categoría y, seguido de una coma, el texto en cuestión ("label1, texto1").

El entrenamiento además separará automáticamente un 10% de los datos para hacer el test y mostrará las métricas resultantes, decidiendo de esta manera si el resultado ha sido satisfactorio o no.

Para finalizar, una vez entrenado, se podrán realizar trabajos de clasificación, tanto directamente como a través de otros servicios de AWS.

Los precios para estos servicios son bastante asequibles, siendo 3$/h para el entrenamiento de modelos, 0.50$ el precio mensual para almacenar el modelo ya entrenado y 0.0005$ por unidad a la hora de realizar un trabajo de clasificación (Una unidad equivale a 100 caracteres)

4. Experimentación

4.1. CONJUNTO DE DATOS

Para este proyecto se utilizaron de base los conjuntos de datos públicos de Reddit de los meses de enero y julio de los años 2015, 2016, 2017 y 2018.

Estos conjuntos de datos contenían en total 98.753.936 archivos y tras la conversión realizada (descrita en el Anexo I) se escribieron 111.834 archivos, lo cuales fueron clasificados 60.913 como contenido neutral y 50.921 como contenido erótico, configurándose además de tal forma que todos los archivos tuvieran mínimo 20 palabras.

Las subcategorías (representadas en el conjunto de datos original como Subreddits) escogidas para hacer el filtrado se pueden observar en la siguiente tabla con su correspondiente número de archivos sacados de las mismas.

Tabla 4.1 Numero de archivos del conjunto de datos

subreddit	number of files	category
sex	25640	erotic
gonewildstories	12943	erotic
sluttyconfessions	1828	erotic
eroticliterature	528	erotic
sexstories	426	erotic
erotica	332	erotic
hotpast	195	erotic
eroticwriting	29	erotic
explainlikeimfive	18760	neutral
pets	11053	neutral
paranormal	10176	neutral
outoftheloop	8133	neutral
getmotivated	8064	neutral
hfy	1865	neutral
ask	1619	neutral
stories	545	neutral
literature	440	neutral
write	199	neutral
kitchen	24	neutral
series	21	neutral
workstories	14	neutral

Paralelamente a la transformación del conjunto de datos se creará un archivo CSV con el formato necesario para poder subirlo a Amazon Comprehend y que este realice el entrenamiento.

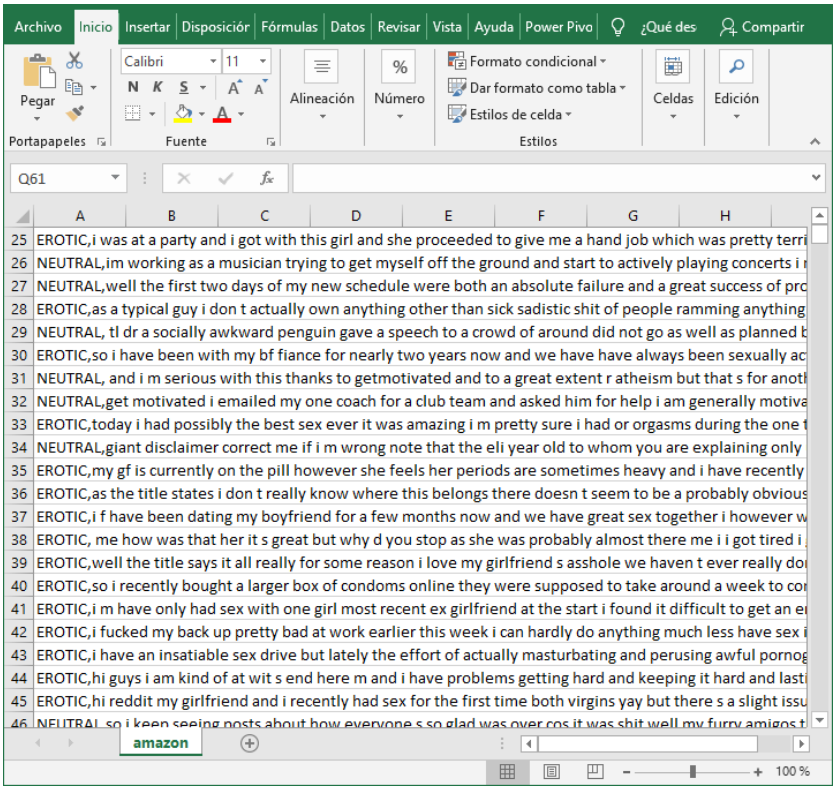

Figura 4.1 Ejemplo de archivo para Amazon Comprehend

A la hora de trabajar con este conjunto de datos con la clasificación manual, es necesario dividirlo en un subconjunto para entrenamiento y otro para test.

Con el objetivo de minimizar la aleatoriedad que conlleva una única división (En la cual se pueden dejar, por ejemplo, unos datos muy fáciles de clasificar en test), se hará uso la validación cruzada K-Fold, técnica con la cual se dividirá el conjunto de datos en K secciones y, realizando K iteraciones se utilizará en cada una de ellas una sección diferente como subconjunto de test. De esta manera se habrán utilizado al final del proceso todos los posibles datos para realizar el test al menos una vez.

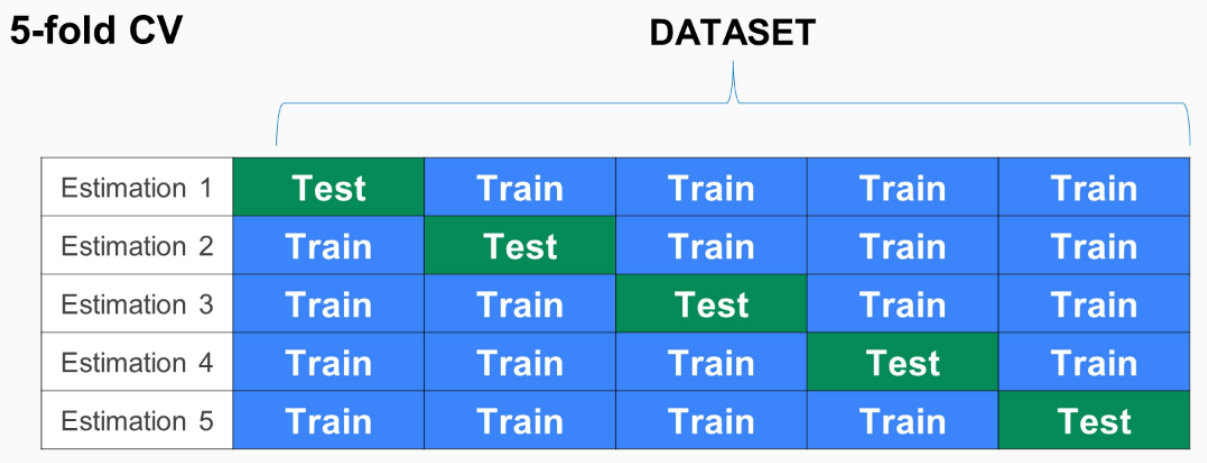

Figura 4.2 Ejemplo Cross-[8]Validation

En este caso, se ha utilizado un valor de K de 5, realizando entonces 5 veces el proceso completo de extracción de características y clasificación determinado, realizando al final una media con los resultados.

4.2. EXTRACCIÓN DE CARACTERÍSTICAS

Una vez ya tenemos el conjunto de datos preparado hay que procesarlo con ayuda de los diferentes extractores de características planteados previamente.

La implementación de BOW y TF-IDF se realizó a través de la librería Scikit-learn[9], teniendo ambos métodos funciones ya establecidas para extraer características claves de texto en forma de lenguaje natural, mientras que Word2Vec se realizará a través de la librería Gensim[10]

En las Figuras 4.3 (BOW), 4.4 (TF-IDF) y 4.5 (Word2Vec) se pueden apreciar los parámetros establecidos.

[8] Fuente: https://alioh.github.io/DSND-Notes-6/
[9] Librería especializada en Machine Learning para Python. Fuente: [27] https://scikit-learn.org
[10] Librería con funciones de topic modeling y procesamiento de lenguaje natural para Pyhton. Fuente: [28] https://radimrehurek.com/gensim/index.html

```
CountVectorizer(
        preprocessor = data_preprocessor,
        tokenizer = data_tokenizer,
        ngram_range = (1,ngram),
        min_df = 3,
        lowercase = False
)
```

Figura 4.3 Declaración modelo BOW

```
TfidfVectorizer(
        preprocessor = data_preprocessor,
        tokenizer = data_tokenizer,
        ngram_range = (1,ngram),
        min_df = 3,
        lowercase = False
)
```

Figura 4.4 Declaración modelo TF-IDF

```
Word2Vec(
        min _count = 3,
        size = 500,
        workers = cores − 1,
        window = 5,
        iter = 30
        )
```

Figura 4.5 Declaración modelo Word2Vec

La variable “ngram” en BOW y TF-IDF establecerá el límite que se busca de n-grams, siendo que en este trabajo se utilizan los ejemplos de unigrama (n=1) y bigrama (n=2).

Por otra parte, las variables “data_preprocessor” y “data_tokenizer” también solo en BOW y TF-IDF son referencias a funciones de preprocesamiento, sobrescribiendo los valores por defecto, los cuales eran referencias a funciones predeterminadas de Scikit-learn, y personalizando así lo que estas harán.

En este caso, el preprocesamiento se encargará de:

1. Transformar todo el texto en minúscula (“Rodrigo” -> “rodrigo”)
2. Eliminar todo aquello que no sean letras y reducir los espacios a uno (“this 5is incredible!!!!” -> “this is incredible”)
3. Eliminar Stop-words, palabras que son tan comunes en un idioma que no aportan significado (“the”, “is”, “a”, etc)
4. Lematización. Este proceso implica la reducción de formas léxicas complejas de una palabra a su raíz, como puede ser el infinitivo de un verbo, el singular de un plural o la base de una familia de palabras. (“am”, “are”, “is” -> “be”, “differents” -> “differ”)
5. Dividir las frases en los tokens (en este caso las palabras) que serán analizadas por el modelo.

Los procesos de eliminación de Stop-words y tokenization se realizaron con ayuda de la librería NLTK (Natural Language Toolkit)[11] mientras que el de lematización fue con la librería Spacy[12]

El modelo de Word2Vec, por otro lado, funciona de forma diferente. La función de la librería Gensim no dispone ni de preprocesamiento automatizado ni de aplicación de n-grams, funciones que se tendrán que realizar de forma manual (Figura 4.6).

[11] Librería especializada en el procesamiento de lenguaje natural en inglés. Fuente: [29] https://www.nltk.org/
[12] Librería especializada en avanzado procesamiento de lenguaje natural. Fuente: [30] https://spacy.io/

```
for i in text_train.index.values:
    text_train[i] = data_tokenizer(data_preprocessor(text_train[i]))

for i in text_test.index.values:
    text_test[i] = data_tokenizer(data_preprocessor(text_test[i]))

text_train = text_train.values.tolist()
text_test = text_test.values.tolist()

for i in range(ngram - 1):
    phrases = Phrases(text_train + text_test)

    phraser = Phraser(phrases)

    text_train = list(phraser[text_train])
    text_test = list(phraser[text_test])
```

Figura 4.6 Extracto de código para la llamada al preprocesamiento manual

Una vez declarados los modelos, el proceso continuará con el entrenamiento de estos en base al conjunto de datos de entrenamiento. Al acabar este proceso el modelo estará preparado para transformar tanto el conjunto de datos de entrenamiento como el de test en las características que posteriormente utilizará el clasificador.

4.3. CLASIFICACIÓN

Utilizando los métodos de clasificación revisados previamente se han clasificado las salidas de todos los extractores de características para así, en igualdad de condiciones, ver cual otorga mejores resultados.

Todos los clasificadores han sido implementados con la ayuda de la librería Scikit-learn tal y como se indica en las figuras 4.7 (SVM con kernel Lineal), 4.8 (SVM con kernel RBF), 4.9 (LR), 4.10 (KNN) y 4.11 (RF)

$$LinearSVC(C = C\_SVM)$$

Figura 4.7 Declaración SVM con kernel lineal

Siendo "C_SVM" la regularización propuesta, en este caso se tomaron los valores de 0.1, 1, 10 y 100 para comparar los resultados.

$$SVC(C = C\_SVM\_rbf,\ gamma = gamma\_SVM\_rbf)$$

Figura 4.8 Declaración SVM con kernel RBF

"C_SVM_rbf" será la regularización propuesta al igual que en el SVM de kernel lineal, en este caso tomando los valores de 0.1, 1 y 10 y combinándolos con "gamma_SVM_rbf" el cual representará el valor que adaptará el parámetro gamma, en este caso 0.1 y 1

$$LogisticRegression(C = C\_LR)$$

Figura 4.9 Declaración LR

De nuevo, la variable "C_LR" tomará diferentes valores para probar la regularización, en este caso 0.1, 1, 10 y 100

$$KNeighborsClassifier(n\_neighbors = neighbors\_KNN)$$

Figura 4.10 Declaración KNN

La variable "neighbors_KNN" representará el número de vecinos (K) que se han tenido en cuenta en el modelo, siendo los valores que adapta 3, 5 y 7.

$$RandomForestClassifier(n\_estimators = est\_RF,\ max\_depth = depth\_RF)$$

Figura 4.11 Declaración RF

La variable "est_RF" nos dará la cantidad de árboles que formarán el modelo de Random Forest, pudiendo ser 3, 10 y 30, a la par que "depth_RF" nos dice la profundidad máxima de dichos árboles pudiendo ser 5 o "None", el cual representará que no lo limita a ninguna máxima.

Una vez declarados los modelos, el proceso debe continuar con el entrenamiento. Para ello, se pasan de parámetros las características de entrenamiento extraídas en la fase anterior y sus correspondientes categorías. Con el modelo entrenado se han obtenido las salidas para los datos de test, obteniendo las categorías predichas para pasar a la fase de análisis de los resultados.

4.4. RESULTADOS

Para evaluar la eficacia de la clasificación, y teniendo como base las categorías predichas y las reales, se han empleado las siguientes métricas:

- Precision: Respecto a una categoría, la fracción de predicciones acertadas entre todas las predicciones realizadas.
- Recall: Fracción de predicciones acertadas de una categoría entre todas las instancias de esa categoría.
- F-score: Media harmónica de Precision y Recall.
- Accuracy: Fracción que representa la cantidad de instancias correctamente categorizadas entre las totales.

4.4.1. **CLASIFICACIONES MANUALES**

A continuación, figuran las tablas con los resultados conseguidos con la combinación de los extractores de características y clasificadores previamente descritos, con todos los diferentes tipos de parámetros y su valor.

En este trabajo se considerarán todas las métricas previamente descritas, utilizándolas todas para mostrar los resultados de la clase Erotic (la cual es realmente la que intentamos predecir) y solo Recall para la clase Neutral (Pudiendo ver de esta forma la sensibilidad condicionada de esta clase), incluyendo además Accuracy, métrica que engloba los resultados de todas las categorías.

Se destacarán los tres mejores resultados de la métrica F-score para la categoría Erotic y los tres mejores resultados de la métrica Accuracy. Para visualizarlo se utilizará el color verde, yendo de mayor a menor intensidad.

Tabla 4.2 Resultados SVM - Kernel Lineal

Suppor Vector Machines - Kernel Lineal			BOW		TF-IDF		Word2Vec	
			1-gram	2-gram	1-gram	2-gram	1-gram	2-gram
C = 0.1	Precision	Erotic	0.9635	0.9697	0.979	0.9701	0.9476	0.9494
	Recall	Erotic	0.9422	0.9335	0.9212	0.9207	0.9468	0.9441
		Neutral	0.9781	0.982	0.988	0.9826	0.9675	0.9685
	F-score	Erotic	0.9527	0.9512	0.9491	0.9447	0.9472	0.9467
	Accuracy		0.9642	0.9632	0.9619	0.9585	0.9599	0.9595
C = 1	Precision	Erotic	0.9398	0.967	0.9718	0.9687	0.9463	0.9477
	Recall	Erotic	0.9428	0.9359	0.945	0.9526	0.9485	0.9451
		Neutral	0.9634	0.9803	0.9835	0.9812	0.9683	0.9677
	F-score	Erotic	0.9413	0.9512	0.9582	0.9606	0.9474	0.9464
	Accuracy		0.9554	0.9631	0.9685	0.9701	0.9601	0.9592
C = 10	Precision	Erotic	0.9277	0.9668	0.9568	0.9668	0.9433	0.9406
	Recall	Erotic	0.9391	0.9359	0.9427	0.957	0.9485	0.9526
		Neutral	0.956	0.9801	0.9745	0.9797	0.9644	0.9616
	F-score	Erotic	0.9333	0.951	0.9496	0.9618	0.9459	0.9465
	Accuracy		0.9493	0.963	0.962	0.971	0.9587	0.959
C = 100	Precision	Erotic	0.9259	0.9668	0.9492	0.9666	0.9117	0.9253
	Recall	Erotic	0.9379	0.9359	0.941	0.9577	0.9541	0.9449
		Neutral	0.9549	0.9801	0.9698	0.9796	0.9451	0.9506
	F-score	Erotic	0.9317	0.951	0.945	0.9621	0.9319	0.9342
	Accuracy		0.9482	0.963	0.9584	0.9712	0.9474	0.9495

Tabla 4.3 Resultados SVM - Kernel RBF

Support Vector Machines - Kernel rbf				BOW		TF-IDF		Word2Vec	
				1-gram	2-gram	1-gram	2-gram	1-gram	2-gram
C = 0.1	gamma = 0.1	Precision	Erotic	1	0	0.9824	0.9802	0.9759	0.9743
		Recall	Erotic	0.0016	0	0.7653	0.543	0.8993	0.9037
			Neutral	1	1	0.9914	0.9934	0.9862	0.9853
		F-score	Erotic	0.0031	0	0.8601	0.6971	0.936	0.9376
		Accuracy		0.6213	0.6207	0.9042	0.8194	0.9526	0.9537
	gamma = 1	Precision	Erotic	0	0	0.9847	0.973	1	1
		Recall	Erotic	0	0	0.8251	0.7118	0.0484	0.0718
			Neutral	1	1	0.992	0.9883	1	1
		F-score	Erotic	0	0	0.8978	0.8215	0.0922	0.1336
		Accuracy		0.6207	0.6207	0.9276	0.8813	0.6385	0.647
C = 1	gamma = 0.1	Precision	Erotic	0.7234	1	0.9816	0.9759	0.9716	0.9608
		Recall	Erotic	0.8097	0.0022	0.9141	0.9069	0.9389	0.9384
			Neutral	0.7932	1	0.9897	0.9864	0.9829	0.9824
		F-score	Erotic	0.7174	0.0044	0.9466	0.9401	0.955	0.9542
		Accuracy		0.7786	0.6215	0.9602	0.9555	0.9659	0.9654
	gamma = 1	Precision	Erotic	0.4	0.4	0.9798	0.9718	0.9948	0.9932
		Recall	Erotic	0.0002	0.0002	0.9306	0.9327	0.2436	0.2764
			Neutral	1	1	0.9884	0.9835	0.9992	0.9988
		F-score	Erotic	0.0005	0.0005	0.9545	0.9517	0.3895	0.4303
		Accuracy		0.6208	0.6208	0.9658	0.9636	0.7102	0.722
C = 10	gamma = 0.1	Precision	Erotic	0.757	1	0.9705	0.9667	0.9698	0.9679
		Recall	Erotic	0.8148	0.0122	0.9464	0.9561	0.9384	0.9403
			Neutral	0.8267	1	0.9826	0.9797	0.9819	0.9808
		F-score	Erotic	0.7402	0.024	0.9583	0.9614	0.9538	0.9538
		Accuracy		0.8019	0.6251	0.9686	0.9707	0.9649	0.965
	gamma = 1	Precision	Erotic	0.4	0.4	0.9785	0.9597	0.9939	0.993
		Recall	Erotic	0.0002	0.0002	0.9366	0.9386	0.2758	0.3118
			Neutral	1	1	0.9876	0.9825	0.999	0.9986
		F-score	Erotic	0.0005	0.0005	0.957	0.9542	0.4297	0.4724
		Accuracy		0.6208	0.6208	0.9677	0.9654	0.722	0.7353

Tabla 4.4 Resultados LR

Logistic Regression			BOW		TF-IDF		Word2Vec	
			1-gram	2-gram	1-gram	2-gram	1-gram	2-gram
C = 0.1	Precision	Erotic	0.9748	0.9749	0.972	0.9531	0.9505	0.9487
	Recall	Erotic	0.9168	0.9132	0.8303	0.7895	0.9468	0.9438
		Neutral	0.9855	0.9854	0.9852	0.9766	0.9694	0.9681
	F-score	Erotic	0.9449	0.943	0.8953	0.8629	0.9486	0.9462
	Accuracy		0.9587	0.9573	0.9253	0.9034	0.961	0.9592
C = 1	Precision	Erotic	0.9672	0.9722	0.9764	0.9619	0.9505	0.9485
	Recall	Erotic	0.9377	0.9282	0.9165	0.9172	0.9483	0.9465
		Neutral	0.9805	0.9835	0.9866	0.9778	0.9693	0.968
	F-score	Erotic	0.9522	0.9497	0.9455	0.939	0.9494	0.9475
	Accuracy		0.9639	0.9621	0.9593	0.9542	0.9617	0.9601
C = 10	Precision	Erotic	0.9558	0.9714	0.974	0.9664	0.9463	0.9472
	Recall	Erotic	0.941	0.9312	0.9388	0.9435	0.9474	0.9457
		Neutral	0.9736	0.9829	0.9847	0.9796	0.9668	0.967
	F-score	Erotic	0.9483	0.9508	0.956	0.9548	0.9468	0.9464
	Accuracy		0.9609	0.9629	0.9668	0.9657	0.9597	0.9593
C = 100	Precision	Erotic	0.9459	0.9708	0.9668	0.9672	0.9452	0.9496
	Recall	Erotic	0.9411	0.9291	0.9421	0.9504	0.9475	0.9467
		Neutral	0.9674	0.9826	0.9804	0.9802	0.9661	0.9683
	F-score	Erotic	0.9434	0.9495	0.9542	0.9587	0.9464	0.9482
	Accuracy		0.9572	0.9619	0.9655	0.9687	0.9592	0.9604

Tabla 4.5 resultados KNN

K-Nearest Neighbors			BOW		TF-IDF		Word2Vec	
			1-gram	2-gram	1-gram	2-gram	1-gram	2-gram
K = 3	Precision	Erotic	0.8893	0.9179	0.8792	0.8829	0.8292	0.8275
	Recall	Erotic	0.6092	0.392	0.8989	0.8836	0.9699	0.9685
		Neutral	0.9551	0.9795	0.9268	0.9302	0.8794	0.8781
	F-score	Erotic	0.7213	0.5456	0.8887	0.883	0.8939	0.8922
	Accuracy		0.8211	0.7531	0.9159	0.912	0.9143	0.9128
K = 5	Precision	Erotic	0.9165	0.9326	0.8898	0.8932	0.8316	0.8324
	Recall	Erotic	0.5766	0.3522	0.9099	0.9037	0.9717	0.9695
		Neutral	0.9686	0.9852	0.9331	0.9359	0.8809	0.8819
	F-score	Erotic	0.7061	0.5084	0.8995	0.8981	0.896	0.8956
	Accuracy		0.8171	0.742	0.9238	0.923	0.9159	0.9157
K = 7	Precision	Erotic	0.9263	0.9407	0.8929	0.8985	0.8308	0.8322
	Recall	Erotic	0.5606	0.3371	0.9186	0.9151	0.9719	0.9701
		Neutral	0.9734	0.9876	0.9347	0.9391	0.8804	0.8816
	F-score	Erotic	0.694	0.4936	0.9053	0.9065	0.8956	0.8957
	Accuracy		0.8137	0.7379	0.928	0.9294	0.9157	0.9157

Tabla 4.6 Resultados RF

Random Forest				BOW		TF-IDF		Word2Vec	
				1-gram	2-gram	1-gram	2-gram	1-gram	2-gram
n_est = 3	depth = 5	Precision	Erotic	0.8315	0.8002	0.8682	0.8466	0.9077	0.9168
		Recall	Erotic	0.378	0.1459	0.3957	0.2519	0.8651	0.862
			Neutral	0.9535	0.9766	0.9658	0.9699	0.9462	0.9379
		F-score	Erotic	0.5127	0.2405	0.527	0.3768	0.8858	0.8786
		Accuracy		0.7313	0.6594	0.7415	0.6924	0.9154	0.9099
	depth = None	Precision	Erotic	0.8171	0.8256	0.8416	0.8148	0.8852	0.8947
		Recall	Erotic	0.8001	0.7162	0.8051	0.7297	0.8775	0.8755
			Neutral	0.8917	0.9083	0.9087	0.8995	0.9313	0.9372
		F-score	Erotic	0.8075	0.7659	0.8223	0.7679	0.8812	0.8848
		Accuracy		0.8558	0.8341	0.8686	0.8331	0.9105	0.9134
n_est = 10	depth = 5	Precision	Erotic	0.8983	0.9947	0.9248	0.9093	0.9318	0.9268
		Recall	Erotic	0.3918	0.1556	0.3934	0.1625	0.8833	0.8767
			Neutral	0.9728	0.9945	0.9812	0.9894	0.9606	0.9582
		F-score	Erotic	0.5336	0.26	0.5469	0.2723	0.9068	0.9009
		Accuracy		0.7454	0.6727	0.7538	0.6739	0.9311	0.927
	depth = None	Precision	Erotic	0.946	0.8681	0.8886	0.8753	0.92	0.9184
		Recall	Erotic	0.8992	0.8371	0.9087	0.8293	0.9137	0.91
			Neutral	0.901	0.9242	0.9308	0.9285	0.9512	0.9498
		F-score	Erotic	0.8709	0.851	0.8982	0.8506	0.9168	0.9141
		Accuracy		0.8996	0.8893	0.9221	0.8889	0.937	0.9349
n_est = 30	depth = 5	Precision	Erotic	0.9193	0.9388	0.9361	0.9409	0.9377	0.9366
		Recall	Erotic	0.3496	0.1211	0.3619	0.1361	0.8885	0.8837
			Neutral	0.9807	0.995	0.9846	0.9938	0.964	0.9633
		F-score	Erotic	0.4988	0.2115	0.5167	0.2321	0.9123	0.9093
		Accuracy		0.7365	0.6612	0.7454	0.6657	0.9351	0.9329
	depth = None	Precision	Erotic	0.8962	0.9007	0.9327	0.9165	0.9498	0.9522
		Recall	Erotic	0.9008	0.8324	0.9016	0.8486	0.9125	0.9134
			Neutral	0.9366	0.9447	0.9609	0.9539	0.9703	0.9718
		F-score	Erotic	0.8978	0.8635	0.9166	0.8804	0.9307	0.9323
		Accuracy		0.9218	0.8992	0.9375	0.9125	0.9482	0.9493

4.4.2. **AMAZON COMPREHEND**

Los resultados obtenidos por la clasificación llevada a cabo por Amazon Comprehend la cual, una vez subido el CSV indicado, separó el conjunto de datos en un 10% para test y 90% para entrenamiento y comenzó automáticamente el trabajo de extracción de características y entrenamiento, se consiguieron tras un periodo de aproximadamente 1 hora y 10 minutos.

Amazon Comprehend DocClassifier		$3.99
Processes Custom Classification model storage request	1 Unit	$0.50
Processes Custom Classification model training request	4,192 seconds	$3.49

Figura 4.12 Tiempo de procesamiento y precio final de una ejecución con Amazon Comprehend

La métrica devuelta por este proceso fue una matriz de confusión para el conjunto de datos de test, la cual contiene los siguientes datos, siendo TE y TN la cantidad de clasificaciones correctas de la clase Erotic y Neutral respectivamente y FE y FN aquellas que han sido predichas como Erotic y Neutral de forma errónea:

Tabla 4.7 Matriz #1 de confusión Amazon Comprehend

	True Erotic	True Neutral
Predicted Erotic	TE = 5800	FE = 46
Predicted Neutral	FN = 120	TN = 5217

Este proceso se repitió una vez más para ratificar los resultados anteriores, devolviendo la siguiente matriz de confusión, muy similar a la primera.

Tabla 4.8 Matriz #2 de confusión Amazon Comprehend

	True Erotic	True Neutral
Predicted Erotic	TE = 5796	FE = 44
Predicted Neutral	FN = 124	TN = 5219

A partir de la media de las matrices de confusión se pueden calcular todas las métricas buscadas a partir de sus respectivas ecuaciones, entregando los siguientes resultados finales:

Tabla 4.9 Resultados Amazon Comprehend

	Precision	Recall	F-Score	Accuracy
Erotic	0.9923	0.9794	0.9858	0.9851
Neutral	0.9772	0.9914	0.9842	

4.4.3. **COMPARATIVAS**

Debido a la gran cantidad de resultados, se han generado dos tablas que resumen los mejores resultados de la métrica F-score para la categoría Erotic y para la métrica Accuracy, escribiendo en ellas solamente el mejor resultado que generaba cada combinación de extractor de características y de clasificador, además de los obtenidos por Amazon Comprehend.

Tabla 4.10 Comparativa Erotic F-score

Erotic F-score				
	BOW	TF-IDF	Word2Vec	Comprehend
SVM-lineal	0.9527	0.9621	0.9474	0.9858
SVM-rbf	0.7402	0.9614	0.955	
LR	0.9522	0.9587	0.9494	
KNN	0.7213	0.9065	0.896	
RF	0.8978	0.9166	0.9323	

Tabla 4.11 Comparativa Accuracy

Accuracy				
	BOW	TF-IDF	Word2Vec	Comprehend
SVM-lineal	0.9642	0.9712	0.9601	0.9851
SVM-rbf	0.8019	0.9707	0.9659	
LR	0.9639	0.9687	0.9617	
KNN	0.8211	0.9294	0.9159	
RF	0.9218	0.9375	0.9493	

4.5. DISCUSIÓN

Debido a algunos de los resultados tan positivos obtenidos, se ve posible diseñar un sistema que detecte contenido erótico muy eficazmente. Este sistema se podría emplear en diferentes campos para la detección de contenido inapropiado, como podría ser la censura a menores o la detección de acceso a sitios inapropiados en situaciones que no se permita.

En cuanto al modelo a implementar para ello, se puede apreciar en los resultados anteriores que, excepto en el clasificador Random Forest, en igualdad de condiciones el extractor que mejores resultados ofrece es TF-IDF. En Random Forest, sin embargo, es ampliamente superado por Word2Vec.

Analizándolos individualmente, el que mejores resultados generales ofrece de los clasificadores valorados es SVM con el kernel lineal, siendo que su valor más bajo de Accuracy en cualquiera de los casos es 0.9474, un resultado muy aceptable, y el mejor 0.9712. Se aprecia una mejora de los resultados cuanto más estricto se es con la regularización (es decir, cuanto mayor es el valor de C).

Un caso con resultados generalmente negativos es SVM con kernel RBF, ya que pese a que se pueden apreciar aceptables resultados en un extractor como TF-IDF, los resultados son muy negativos en el caso de utilizarlo junto a BOW con Word2Vec, como por ejemplo cuando adopta los valores C=0.1 y gamma=1. Al ser el kernel una transformación del conjunto de las características devueltas por el clasificador, se puede concluir que tanto BOW como Word2Vec (Los que obtienen malos resultados en algunos casos) tienen devuelven unos datos menos compatibles con dicha transformación. También habría que valorar el hecho de que los resultados parecen ser mejores con gamma=0.1 que con gamma=1, y que, al igual que en el caso con kernel lineal, mejoran al aumentar el valor del parámetro C.

En el caso de LR, los resultados son bastante aceptables en todos los casos, aunque no tanto como con SVM con kernel lineal, siendo el mejor extractor TF-IDF seguido de cerca por Word2Vec. Al igual que en SVM, se aprecia una mejora al aumentar C.

El clasificador KNN es el que peores resultados ha dado a grandes rasgos, siendo especialmente negativos en el caso de BOW (Especialmente en 2-gram), y mejorando en TF-IDF y Word2Vec, valorando que cuanto más aumentemos el número de vecinos mejores resultados genera.

Para RF podemos apreciar que cuando la profundidad de los árboles es igual a 5, BOW y TF-IDF dan unos resultados pésimos, mientras que Word2Vec rinde bastante aceptablemente, llegando a una Accuracy máxima de 0.9351. Como es lógico y se puede apreciar, los resultados mejoran al no establecer ninguna profundidad máxima y al aumentar el número de árboles, siendo la mejor combinación con Word2Vec y llegando en esos casos a un máximo de 0.9493 en Accuracy.

Respecto al uso de n-grams, excepto en casos concretos no se aprecia mucha diferencia al uso de 1-gram o 2-gram, siendo esto un resultado a tomar en cuenta, ya que al hablar de lenguaje natural en un contexto no formal en el que pueden existir una jerga o diferentes expresiones que se entiendan mejor con 2-grams, en un principio podría parecer que habría más diferencia.

Para finalizar, mirando a la comparativa se puede apreciar que los resultados obtenidos por Amazon Comprehend son mejores que cualquiera de las combinaciones revisadas, obteniendo un 0.9851 en Accuracy mientras que el segundo mejor resultado es SVM con TF-IDF con un 0.9712, una diferencia de más del 1%, a la vez que obteniendo un 0.9848 en Erotic F-score seguido de nuevo por SVM con TF-IDF con 0.9621, con una diferencia de este caso de más del 2%.

5. Conclusiones

Este trabajado ha tenido como objetivo la detección de contenido no apropiado en texto, más concretamente nos hemos centrado en la detección y clasificación de contenido erótico. El texto debía ser lenguaje natural, no categorizado o estructurado.

Para conseguir ese objetivo, primero se ha revisado literatura existente relacionada con el tema, tanto de extractores de características como de clasificadores y su desempeño juntos.

Tras esta revisión se han seleccionado tres extractores de características, siendo dos técnicas tradicionales como son BOW y TF-IDF y una más novedosa, Word2Vec, a la vez que se han seleccionado 4 clasificadores: SVM, LR, KNN y RF.

No fue posible obtener un conjunto de datos ya creado que fuera adecuado para el problema, así que hubo que crear uno basándose en datos públicos de Reddit, obteniendo un amplio Conjunto de Datos con muchos ejemplos diferentes.

Las clasificaciones se llevaron a cabo variando algunos parámetros de cada clasificador y se obtuvieron los resultados de todas las combinaciones posibles.

Gracias a los buenos resultados obtenidos en clasificadores de datos linealmente separables en comparación al resto, se puede concluir que las características generadas del texto para diferenciar ambas categorías son linealmente separables, dando dichos clasificadores resultados muy buenos, llegando al 97.12% de tasa de aciertos para TF-IDF + SVM con Kernel Lineal, siendo esta la mejor combinación de las revisadas.

Paralelamente a ese proceso, se llevó a cabo una clasificación mediante computación en la nube con ayuda de AWS y su servicio Amazon Comprehend. Esta clasificación ha sido la que mejor resultados ha obtenido, con un 98.51% de Accuracy, consiguiendo una diferencia de más del 1% en Accuracy y de más del 2% en F-score (aplicado a la clase Erotic) en comparación al segundo mejor resultado.

Unas diferencias de estas magnitudes en valores tan altos es algo muy difícil de conseguir, haciendo de Amazon Comprehend sin duda la solución más óptima para este problema.

Hay variables a tener en cuenta a la hora de escoger el mejor método para llevar a cabo un trabajo como este, como por ejemplo que el modelo generado por Amazon Comprehend, pese a ser el que mejores resultados consigue, tiene un coste de mantenimiento y de petición de clasificación, además de no poder descargar el modelo para trabajar con él por tu cuenta.

Se puede concluir, pese a quedar cierto margen de mejora, que los resultados obtenidos en este trabajo son suficientemente buenos como para llevar su aplicación a un entorno real y evaluar su verdadero desempeño frente a problemas reales, además de poder aplicar los métodos e ideas propuestos para futuros trabajos como la detección de otro tipo de contenido inapropiado, como podría ser violencia, terrorismo o actividades ilegales.

6. Bibliografía

[1] Z. S. Harris, "Distributional Structure," *WORD*, vol. 10, no. 2–3, pp. 146–162, Aug. 2015.

[2] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1. pp. 11–21, Jan-1972.

[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *papers.nips.cc*, 2013.

[4] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[5] D. R. Cox, "The Regression Analysis of Binary Sequences," *J. R. Stat. Soc. Ser. B*, vol. 21, no. 1, pp. 238–238, Jul. 1958.

[6] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-{NN} neighbor algorithm," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-15, no. 4, pp. 580–585, 1985.

[7] L. Breiman, "Random Forests," vol. 45. pp. 5–32, 2001.

[8] Amazon, "Amazon comprehend," 2019. [Online]. Available: https://docs.aws.amazon.com/comprehend/latest/dg/comprehend-general.html. [Accessed: 07-Jun-2019].

[9] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural Language Processing: State of The Art, Current Trends and Challenges," 2017.

[10] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, and L. Chanona-Hernández, "Syntactic N-grams as machine learning features for natural language processing," *Expert Syst. Appl.*, vol. 41, no. 3, pp. 853–860, Feb. 2014.

[11] D. foot. Keith, "A Brief History of Natural Language Processing," 2019. [Online].

Available: https://www.dataversity.net/a-brief-history-of-natural-language-processing-nlp/. [Accessed: 17-Jun-2019].

[12] M. W. Al Nabki, E. Fidalgo, E. Alegre, and I. de Paz, "Classifying Illegal Activities on Tor Network Based on Web Textual Contents," in *the Association for Computational Linguistics*, 2017, vol. 1, pp. 35–43.

[13] V. S. Chavan and S. S. Shylaja, "Machine learning approach for detection of cyber-aggressive comments by peers on social media network," in *2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015*, 2015, pp. 2354–2358.

[14] S. Sun, C. Luo, and J. Chen, "A review of natural language processing techniques for opinion mining systems," *Inf. Fusion*, vol. 36, pp. 10–25, Jul. 2017.

[15] S. Khurana, "Vector Space Modelling of Natural Language," 2015.

[16] B. Chiu, G. Crichton, A. Korhonen, and S. Pyysalo, "How to Train good Word Embeddings for Biomedical NLP," 2016, pp. 166–174.

[17] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP," in *Procedia Computer Science*, 2017, vol. 117, pp. 256–265.

[18] H. L. Hammer, "Automatic detection of hateful comments in online discussion," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2017, vol. 188, pp. 164–173.

[19] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification," 2015, pp. 1555–1565.

[20] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02*, 2002, vol. 10, pp. 79–86.

[21] P. Gupta, "Decision Trees in Machine Learning – Towards Data Science," *Towards*

Data Science, 2017. [Online]. Available: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052. [Accessed: 21-Jun-2019].

[22] TDIDF, "Tf-idf :: A Single-Page Tutorial - Information Retrieval and Text Mining." .

[23] H. Heidenreich, "Introduction to word embeddings," no. November, pp. 1–10, 2018.

[24] R. Varma, "Language Models, Word2Vec, and Efficient Softmax Approximations," 2017. [Online]. Available: https://rohanvarma.me/Word2Vec/. [Accessed: 13-Jun-2019].

[25] D. Making, L. M. I. Jordan, C. C. (Albany) Rodriguez, and M. Hofmann, "The Kernel Trick," *Learning*, no. x, pp. 1–5, 2004.

[26] S.-T. Li, "Plotting decision boundaries in 3D — Logistic regression and XGBoost," *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/plotting-decision-boundaries-in-3d-logistic-regression-and-xgboost-e68ce0535b4b. [Accessed: 12-Jun-2019].

[27] T. Klikauer, "Scikit-learn: Machine Learning in Python," *TripleC*, vol. 14, no. 1, pp. 260–264, 2016.

[28] R. Rehurek, "gensim: Topic modelling for humans," 2018. [Online]. Available: https://radimrehurek.com/gensim/index.html. [Accessed: 15-Jun-2019].

[29] X. Shu and R. Cohen, "Natural Language Toolkit (NLTK)," 2010.

[30] spacy, "spaCy · Industrial-strength Natural Language Processing in Python," 2019. [Online]. Available: https://spacy.io/. [Accessed: 14-Jun-2019].

Anexo I: Creación del conjunto de datos

Para la creación del conjunto de datos en este trabajo se utilizaros de base unos archivos públicos que contienen todas las publicaciones principales de los posts de Reddit separadas por meses[13]

Estos archivos base estaban en un formato JSON, siendo cada publicación un objeto JSON el cual se separa del resto mediante un salto de línea, los cuales contienen varios campos útiles para nuestro Conjunto de Datos, los cuales son:

- id: ID única para cada objeto
- selftext: En él se encuentra el cuerpo del post y es el contenido principal que posteriormente será analizado.
- subreddit: Categoría en la que el post fue publicado.
- over_18: Variable booleana que indica si el post es adecuado para menores de 18 años. Se utiliza para el filtrado de los posts.

Gracias a la variable "subreddit" se consigue el filtrado principal para que el conjunto de datos creado se le asigne una categoría automáticamente.

Esto se consigue gracias a un estudio previo desde la página web en sí, analizando y recopilando una lista de Subreddits que tengan siempre contenido erótico, como pueden ser historias que algunos usuarios escriben que contienen experiencias sexuales, o preguntas que hacen usuarios sobre temas relacionados con el sexo.

Una vez tenemos la lista de subreddits que tendrán contenido erótico, se analizan también Subreddits que sean neutrales pero variados, seleccionando así por ejemplo uno en el que se habla de cocina, otro de mascotas, otro en el que la gente cuenta historias que les ocurren y uno de preguntas generales. Por si acaso algún post de contenido erótico esté en estos subreddits más generales (como uno en el que alguien haga una pregunta en este subreddit

[13] Fuente: https://files.pushshift.io/reddit/submissions/

general en vez de en el erótico), se tiene en cuenta la variable over_18 y no se incluirá en la lista si esa variable está en positivo.

Una vez que ambas listas están formadas, se pasa a realizar la conversión a nuestro propio conjunto de datos. Para ello se creó un programa en Python que analizaba los archivos en formato JSON y los transformaba en dos archivos de formato CSV diferentes: Uno que contenía la lista de datos (creando los archivos con el contenido del post en un archivo txt aparte) filtrada, y otro con un formato especial para que Amazon Comprehend lo pueda analizar, siendo este formato "label, contenido del texto".

Además del filtrado básico mediante los subreddits se añadieron diferentes variables como una longitud máxima y mínima y casos en los que no quisiéramos que el contenido fuera guardado en la lista (cuando es un post solo con título y sin descripción, por ejemplo, o si el post fue borrado).

Todas las variables de filtro se pueden apreciar en la Figura I.1

```
listSubredditsErotic = {"sex": 0, "sluttyconfessions": 0, "eroticliterature": 0,
                        "sexstories": 0, "erotica": 0, "hotpast": 0,
                        "eroticwriting": 0, "gonewildstories": 0}
listSubredditsNeutral = {"study": 0, "jokes": 0, "jobs": 0, "pets": 0,
                         "stories": 0, "write": 0, "hfy": 0, "literature": 0,
                         "kitchen": 0, "workstories": 0, "ask": 0,
                         "outoftheloop": 0, "series": 0, "getmotivated": 0,
                         "explainlikeimfive": 0}

# Lista de casos que no queremos que sea o contenga el texto
textNotValid = [None, "", "[removed]", "[deleted]"]

# Longitud minima y máxima de los textos
lenEroticMin = 100;
lenNeutralMin = 100;

lenEroticMax = None;
lenNeutralMax = None;
```

Figura I.1 Filtros de creación del conjunto de datos

En la finalización del filtrado, se habrán separado por carpetas según la categoría el contenido erótico del neutral, se habrán guardado los dos archivos de formato CSV previamente descritos y se indicará la cantidad de archivos que había originalmente en el conjunto de datos base y cuántos fueron seleccionados.

```
Writting files dataset_subreddit.csv and amazon.csv

Reading file RS_2012-01
The file RS_2012-01 contains 1981577 JSON objects and the program wrote 4177 (neutral: 2037; erotic: 2140) in 26.29 seconds

Reading file RS_2014-01
The file RS_2014-01 contains 4145541 JSON objects and the program wrote 12689 (neutral: 7974; erotic: 4715) in 72.49 seconds

Reading file RS_2017-01
The file RS_2017-01 contains 9218513 JSON objects and the program wrote 6180 (neutral: 4024; erotic: 2156) in 184.82 seconds


All files contains 13004628 JSON objects and the program wrote 25659 (neutral: 15926; erotic: 9733)
```

Figura I.2 Ejemplo salida de la creación del conjunto de datos