Universidad de la República, Facultad de Ingeniería, Instituto de Computación

Proyecto de Grado

Predicción de género para usuarios de Twitter.

Jimena Rodríguez Pérez
Jimena Díaz Iglesias

Tutores:
Diego Garat
Juan José Prada

Junio 2018
Montevideo, Uruguay

Resumen

Predecir de forma precisa los atributos demográficos de los usuarios de las redes sociales, en particular el género, es de gran utilidad, entre otros, para el marketing y para las investigaciones legales que involucren perfiles falsos con fines maliciosos.

El siguiente proyecto tiene como objetivo detectar el género de una persona utilizando la información disponible en su cuenta de Twitter. Para esto, se utilizan distintos algoritmos de aprendizaje automático: *naïve bayes*, árboles de decisión, bosques aleatorios, *support vector machine (SVM)*, redes neuronales y redes neuronales recurrentes.

Para aplicar los algoritmos, se construye un *corpus* en español, el cual es pieza fundamental para su ejecución. El *corpus* consta de un total 1.370.154 *tweets* correspondientes a 14240 usuarios de los cuales 7079 son del género femenino y 7161 del género masculino.

Los mejores resultados muestran un acierto del 76.6 % y 75.4 % para los modelos obtenidos con los algoritmos de redes neuronales y SVM respectivamente, lo cual se posiciona más de 25 puntos por encima del 50 % que se toma como línea base.

Palabras Clave: detección automática de género, redes sociales, *naïve bayes*, árboles de decisión, bosques aleatorios, SVM, redes neuronales, redes neuronales recurrentes.

Agradecimientos

En primer lugar, agradecemos a la empresa IDATHA por su colaboración, la base de datos de usuarios provista y la disposición para ayudarnos fue fundamental para lograr el objetivo.

Agradecemos también a Mathias Etcheverry que nos proporcionó los vectores de palabras preentrenados y nos guió en su utilización.

Finalmente, agradecemos al grupo de PLN del InCo por permitirnos utilizar el PC con GPU para la realización de pruebas.

Índice

1. Introducción

Las redes sociales son sitios de Internet que permiten a las personas conectarse de manera virtual y compartir contenidos, interactuar, crear comunidades sobre intereses similares: trabajo, lecturas, juegos, amistad, relaciones amorosas, relaciones comerciales, etc. El origen de las redes sociales se remonta a 1995, cuando el estadounidense Randy Conrads crea el sitio Web `classmates.com`. En 2002 comienzan a aparecer los primeros sitios Web que promocionan redes de círculos de amigos en línea o relaciones en las comunidades virtuales. La popularidad de estos sitios crece rápidamente y se va perfeccionando hasta conformar el espacio de las redes sociales en Internet [9].

El concepto de red social ha adquirido una gran importancia a lo largo del tiempo y se ha convertido en una nueva expresión del lenguaje común. Desde la llegada de la Web 2.0, las redes sociales en Internet ocupan un lugar relevante en el campo de las relaciones personales.
Con la Web 2.0 o Web Social, se ha revolucionado el concepto de red, las formas de comunicación han cambiado e Internet ha adoptado nuevas características de colaboración y participación. A diferencia de la Web 1.0 de sólo lectura, la Web 2.0 es de lectura y escritura, donde se comparte información que está en constante actualización. Los profesores de la Universidad de Indiana, Andreas M. Kaplan y Michel Haenlein [6], definen los medios sociales como: "un grupo de aplicaciones basadas en Internet que se desarrollan sobre los fundamentos ideológicos y tecnológicos de la Web 2.0, y que permiten la creación y el intercambio de contenidos generados por el usuario". Otros especialistas plantean estos servicios como herramientas informáticas que permiten la creación de una red social *on-line* y que, para ello, tratan de operar en tres ámbitos de forma cruzada, "las 3Cs": Comunicación, Comunidad y Cooperación [6].

Los avances tecnológicos en el campo de la comunicación siempre han sido objeto de estudio de las ciencias sociales, debido a que las nuevas formas de relación social generan un impacto en la sociedad. El uso de las redes sociales ha permitido la generación de un nuevo vocabulario, nuevas formas de expresión y comunicación entre los individuos, todo a un ritmo muy acelerado. Nuestros datos se comparten en la red y, al mismo tiempo, información de todo tipo nos invade. Otro de los efectos de las redes sociales más estudiado es el poder de manifestación que poseen, la inmediatez de la comunicación, y transmisión de opiniones y contenido las convierte en un potente instrumento social.

Es de gran interés construir automáticamente los perfiles sociodemográficos de los usuarios tomando en cuenta variables como la edad, género, orientación política, región geográfica, dado que en muchas redes sociales no es obligatorio que las declaren, o en caso de serlo, es posible declararlas con datos falsos. Esta información tiene múltiples usos que varían desde el diseño de campañas de mercado o campañas políticas, hasta la detección de perfiles falsos construidos con fines delictivos o fines sexuales, entre otros.

Este proyecto tiene como objetivo estudiar la detección automática de una de estas variables, el género, construyendo un modelo para su predicción en usuarios hispanoparlantes, utilizando técnicas de aprendizaje automático y procesamiento del lenguaje natural.
Conocer esta variable permite, a modo de ejemplo, a una empresa que promociona un producto

destinado a las mujeres, publicitar su producto a los usuarios cuyo género sea femenino. En el caso de las campañas políticas, es posible realizar distintas campañas dependiendo del género del votante. Además, existen problemáticas relacionadas con las falsas identidades, un usuario puede fingir su género para intentar acercarse a la víctima, por ejemplo, un hombre pedófilo puede mentir su género para contactarse con una adolescente fingiendo una amistad.

En varios de los trabajos relacionados [18, 20, 23, 4] se hace referencia al estudio de la escritura de los usuarios en las redes sociales para poder determinar su género. Los mensajes intercambiados por lo general no utilizan un lenguaje estándar y presentan variaciones según el idioma y la región. Se encuentran modelos para analizar estos mensajes, pero son entrenados y testeados para el idioma inglés. La ausencia de un *corpus* similar para el español, nos motiva a construir un *corpus* compuesto por usuarios de habla hispana para realizar un estudio semejante.

Para crear el *corpus*, se investiga qué redes sociales permiten acceder a la información de los usuarios y se decide utilizar la red social Twitter dado que a diferencia de otras redes sociales, da la posibilidad de conseguir muchos datos sobre sus usuarios. En Twitter los datos y las publicaciones de los usuarios son generalmente públicos y cualquier persona puede verlos e interactuar con ellos aunque no tenga una cuenta.

Este proyecto se encuentra disponible en GitHub [1], donde se encuentran los recursos generados durante su desarrollo.

En las siguientes secciones se presentan las características principales de la red social Twitter, la creación de cuentas de usuario, el detalle de sus atributos y la particularidad de los mensajes intercambiados, el cronograma y la organización de este documento.

1.1. Twitter

Twitter es una red social muy popular que tiene unos 313 millones de usuarios activos mensualmente. Su misión, según la página oficial de Twitter [26], es: "Dar a todos el poder de crear y compartir ideas e información al instante, sin barreras". Esta red le permite a los usuarios enviar mensajes cortos denominados *tweets*, con un máximo de 280 caracteres.[2]
En muchas redes sociales es sencillo definir un nombre, edad, género y ubicación falsa. En particular, en Twitter [26] no es obligatorio declarar ninguna de las anteriores, solo pide un correo electrónico y nombre de usuario para la apertura de una cuenta. Esto permite a los usuarios con perfiles criminales, como pedófilos, mentir con facilidad. Por esto, es útil monitorizar los perfiles de los usuarios y analizar el texto que escriben dando como resultado qué perfiles son falsos. [18]

Al crear una cuenta de Twitter hay que completar el campo nombre de usuario que se muestra debajo de la foto de perfil, una cuenta de correo, contraseña y finalmente un alias que lo identifica, *screen name*. En la Figura 1 se puede observar un ejemplo de la creación de una cuenta.

[1] https://github.com/ProyectoGradoGen/pGen

[2] Durante la construcción del *corpus* de este proyecto la máxima cantidad de caracteres es 140; este límite se duplicó a partir de Noviembre de 2017.

Figura 1: Cuenta de Twitter

Una vez creada la cuenta, es posible añadir o cambiar la foto de perfil y foto de portada. También una biografía donde se puede agregar una descripción para definirse a sí mismo en menos de 160 caracteres, una ubicación, un *link* a un sitio web y fecha de cumpleaños. En la Figura 2, se pueden ver los campos que es posible completar.

Figura 2: Campos cuenta de Twitter

Cuando un usuario *twittea* (envía un mensaje a través de Twitter), en cada *tweet* se muestra su nombre de usuario, *screen name*, foto de perfil y su contenido. Dentro del contenido, además de texto, el usuario puede incluir entre otros, emoticones, *links*, menciones a usuarios y *hashtags* (etiquetas de Twitter formadas por el carácter # y una o varias palabras). En la Figura 3 se presenta un ejemplo de un *tweet* y se indican algunos de los términos más importantes a tener en cuenta.

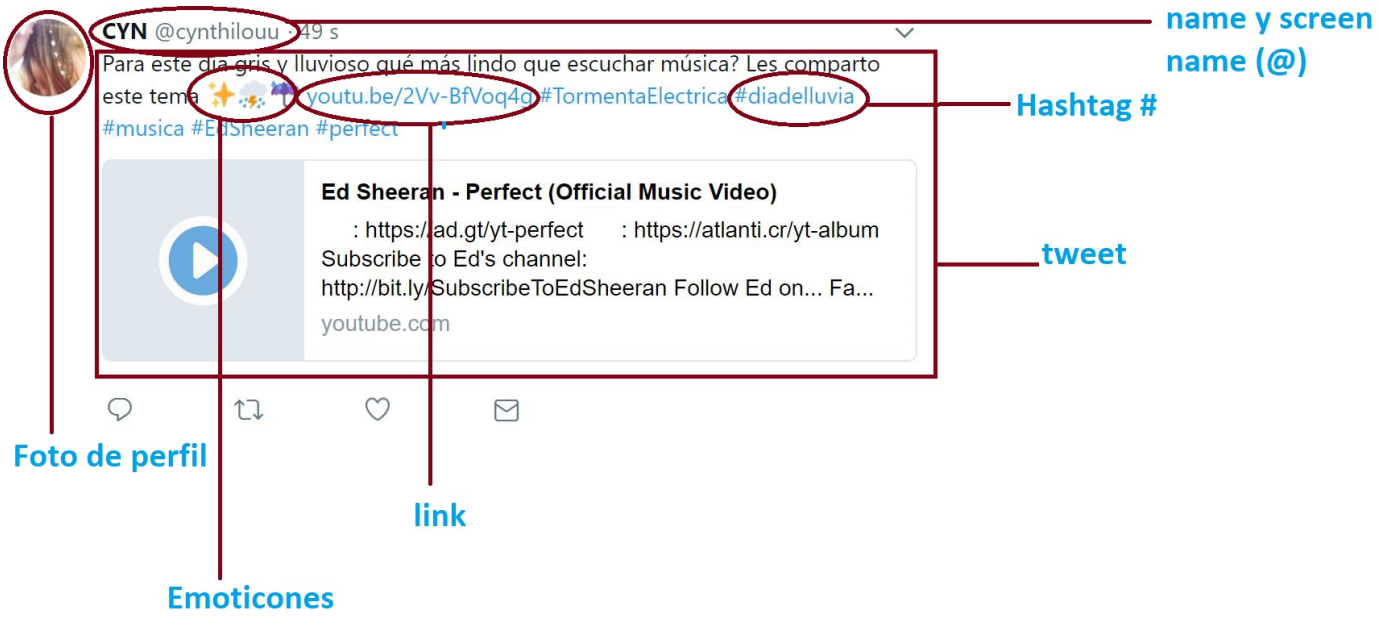

Figura 3: Tweet

1.2. Cronograma

En el Cuadro 1 se resumen las tareas que se realizan a lo largo de todo el proyecto y el tiempo dedicado a cada una. Se presenta una división de las tareas a nivel macro, puesto que, a lo largo del documento se explica detalladamente cada una de ellas.

	Agosto-Diciembre 2016	**Enero-Abril 2017**	**Mayo-Agosto 2017**	**Agosto-Diciembre 2017**	**Enero-Junio 2018**
Informe de estado del arte					
Armado del *corpus*					
Investigación y primeras pruebas con algoritmos de clasificación					
Investigación de redes neuronales recurrentes					
Pruebas en nuestros PC					
Pruebas en los clusteres de Fing					
Optimización de clasificadores					
Evaluación					
Informe final					

Cuadro 1: Cronograma

1.3. Estructura del documento

En este documento se describe el proceso para el trabajo que se realiza en el contexto del proyecto de grado. En primera instancia, en el capítulo 2, se presenta el marco teórico, se hace una breve introducción del concepto de aprendizaje automático y de los algoritmos que se utilizan, así como también una descripción de proyectos relacionados.

En el capítulo 3 se detalla el proceso de construcción del corpus, la selección y procesamiento de los usuarios y atributos. Luego, en el capítulo 4 se presenta la configuración de los algoritmos que se emplean en la implementación, la base de datos y las especificaciones del hardware y software.

En el capítulo 5 se especifican las métricas de performance consideradas, los experimentos que se realizan y los mejores resultados que se obtienen. Finalmente, en el capítulo 6, se encuentran las conclusiones y las posibles mejoras a futuro.

2. Marco Teórico

En este capítulo se presentan algunos conceptos básicos que nos ayudan a dar un marco teórico y a tener una idea general de los distintos algoritmos de clasificación, modelos y técnicas que son de utilidad para lograr el objetivo planteado, muchos de ellos utilizados en proyectos de similares características.

Además, se presentan trabajos relacionados a la detección de género de usuarios de distintas redes sociales.

2.1. Aprendizaje automático

En términos generales, el aprendizaje automático es la rama de la inteligencia artificial que se dedica al estudio de los programas que aprenden de su propia experiencia, esto es, mejoran su rendimiento en una tarea sin la necesidad de explícitamente programar cómo resolverla. Los algoritmos de aprendizaje automático se clasifican en dos grandes familias, de acuerdo al tipo de experiencia: los algoritmos supervisados y los no supervisados. [15]

El aprendizaje supervisado tiene por objetivo inferir una función a partir de un conjunto de entrenamiento formado por instancias de pares con la entrada y su salida esperada. Se espera que el algoritmo construya un modelo a partir de este conjunto de entrenamiento, de forma tal que sea capaz de predecir el valor correcto sobre nuevas instancias no vistas. A modo de ejemplo, si se quiere predecir el valor de una vivienda a partir de sus características (metros cuadrados, cantidad de habitaciones, distancia al centro, etc), el conjunto de entrenamiento debe estar conformado por instancias que describan las características de la vivienda (entrada) y su valor (salida esperada). [15]

A diferencia del caso supervisado, en aprendizaje no supervisado las instancias del conjunto de entrenamiento no están etiquetadas con un valor de salida esperado. El proceso busca inferir una función que explica la estructura de los datos de entrada, a partir de la detección de patrones en los ejemplos. Para fijar ideas, dado un conjunto de clientes a los cuales se quiere dirigir una campaña de marketing, mediante un algoritmo de aprendizaje no supervisado es posible encontrar patrones que permitan ofrecer distintos productos a cada grupo de clientes. [15]

A continuación se detallan los algoritmos estudiados a lo largo del proyecto y posteriormente utilizados para las pruebas.

2.1.1. Clasificador SVM

Support Vector Machines (SVM) es un algoritmo de aprendizaje supervisado basado en el concepto de planos de decisión, separadores lineales o hiperplanos que definen justamente cuáles son los límites de decisión ya sea en el espacio original de los ejemplos de entrada, o en un espacio transformado (espacio de características) si los ejemplos no son separables linealmente en el espacio original.

La idea es seleccionar un hiperplano de separación que equidista de los ejemplos más

cercanos de cada clase para conseguir lo que se denomina un margen máximo a cada lado del hiperplano. Además, a la hora de definir el hiperplano, sólo se consideran los ejemplos de entrenamiento de cada clase que caen justo en la frontera de dichos márgenes. Estos ejemplos reciben el nombre de vectores de soporte.

Como ejemplo básico, se muestra el de un clasificador lineal. En el caso para el cual el conjunto de entrenamiento es linealmente separable en dos clases en un espacio bidimensional, existe un hiperplano que deja a todos los vectores asociados a cada clase del lado correspondiente del hiperplano como se observa en la Figura 4.

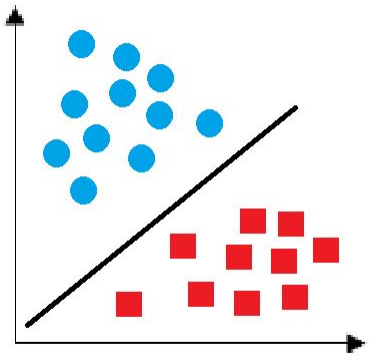

Figura 4: Clasificador Lineal

Sin embargo, no siempre es tan simple; a menudo se necesitan estructuras más complejas para hacer una separación óptima. Para los casos en que el conjunto no es linealmente separable, se aplican transformaciones lineales y se agregan dimensiones al espacio. A estas funciones se les llama *kernels*. Varios son los ejemplos de *kernels*, como ser el lineal, polinómico o gaussiano. Son funciones que toman un espacio de entrada de baja dimensión y lo transforman en un espacio de dimensión superior, transformando al problema no separable en uno separable.

En la Figura 5 se muestra un ejemplo linealmente no separable en un espacio de dos dimensiones x-y. Al agregar una tercera dimensión z, es posible separar el conjunto. Si los puntos en el plano z se representan como:

$$z = x^2 + y^2$$

es posible considerarlos como la distancia de un punto al origen del plano z. De esta forma, si se grafica en el eje z, se puede trazar una línea de separación, cuando se transforma esta línea al plano original x-y, se mapea al límite circular.

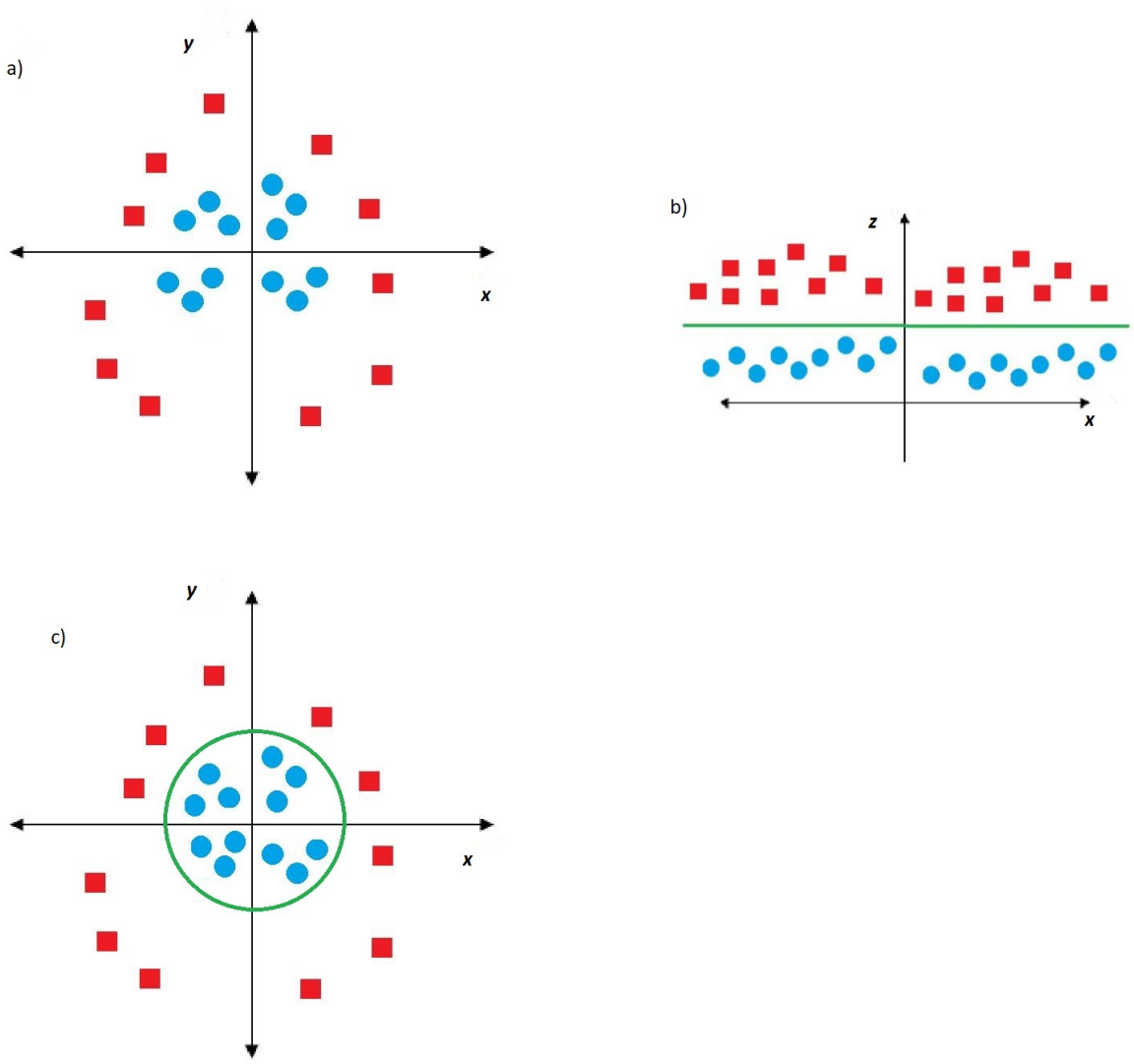

Figura 5: a) problema original; b) transformación con z= $x^2 + y^2$; c) representación en el plano original

Se puede decir entonces, que SVM realiza tareas de clasificación mediante la construcción de hiperplanos en un espacio multidimensional que separa los casos de diferentes clases. SVM soporta tareas de regresión y clasificación y puede manejar múltiples variables continuas y categóricas. [8]

2.1.2. Árboles de decisión

Los árboles de decisión son utilizados para clasificar objetos en un conjunto de clases predeterminadas, basándose en sus atributos. La clasificación parte desde la raíz hasta llegar a algún nodo hoja. Cada nodo en el árbol especifica algún atributo de la instancia, y cada rama descendente de ese nodo corresponde a uno de los valores posibles para este atributo. Para clasificar una instancia, se comienza desde el nodo raíz probando el atributo especificado y desplazándose por la rama del árbol correspondiente al valor del atributo de la instancia dada. Este proceso se repite entonces para el subárbol enraizado en el nuevo nodo. [15]

Los árboles de decisión son adecuados cuando las instancias del concepto son representadas por pares atributo-valor, la función objetivo tiene valores de salida discretos, las descripciones del objeto son disyuntivas y el conjunto de aprendizaje tiene errores o es incompleto.

En la Figura 6 se observa un ejemplo, sencillo, de árbol de decisión utilizado para determinar si una persona está en forma o no, tomando en cuenta la edad, si hace ejercicio y si come habitualmente pizza.

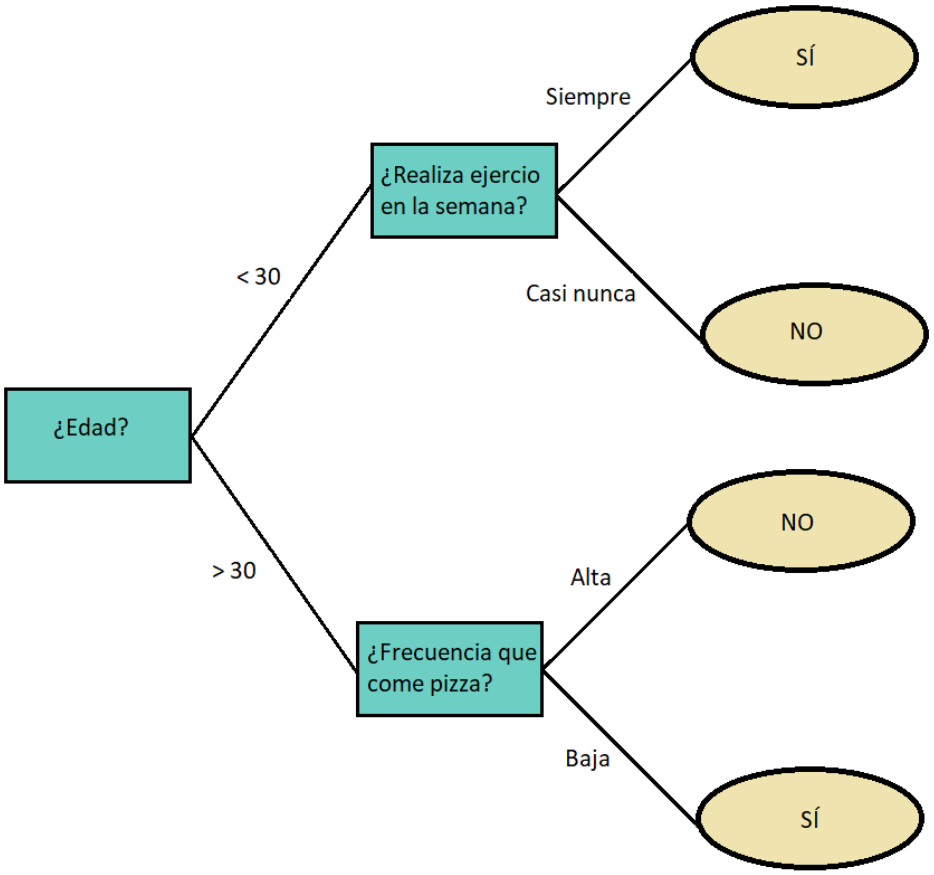

Figura 6: Árbol de decisión para determinar si una persona está en forma

Bosques Aleatorios

Bosques Aleatorios es un algoritmo de clasificación supervisada, el cual se compone por un conjunto de árboles de decisión no correlacionados a los que luego se les aplica el promedio. Al utilizar este algoritmo se intenta mitigar un sobreajuste a los datos. Este algoritmo es utilizado tanto para clasificación como para regresión.

La idea detrás de los bosques aleatorios es combinar varios árboles de decisión en un mismo modelo. Las predicciones individuales de los árboles de decisión pueden no ser exactas, pero si se promedian seguramente se obtiene un mejor acierto. Como se puede observar en la Figura 7, varios árboles con el mismo conjunto de datos pueden predecir distinto, esto se debe a que los distintos árboles toman un conjunto al azar de atributos para definir los nodos del árbol. [3].

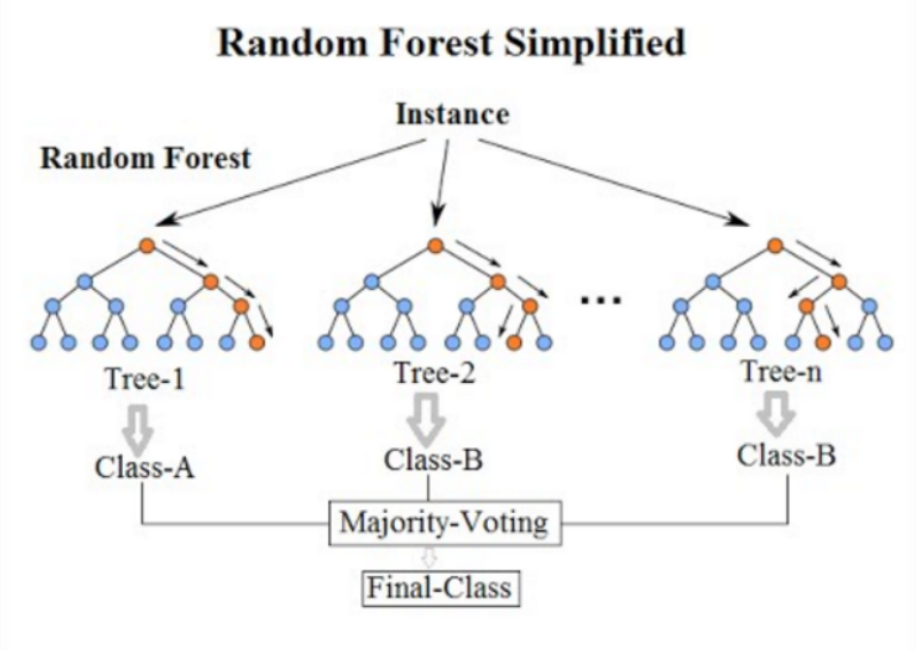

Figura 7: Árbol de decisión Imagen extraída de *Medium* [12]

2.1.3. Naïve Bayes

El algoritmo de clasificación Naïve Bayes es un clasificador probabilístico. La palabra Naïve (ingenuo en inglés) proviene del hecho que el algoritmo utiliza técnicas Bayesianas pero no tiene en cuenta las dependencias entre los atributos de un ejemplo que se desea clasificar. Entre sus ventajas se encuentra que es un algoritmo sencillo de implementar y generalmente obtiene buenos resultados.

Dado un ejemplo x representado por k valores que describen sus atributos, el clasificador se basa en encontrar la hipótesis más probable *f(x)* que describa al ejemplo x, asumiendo que todos

los atributos son independientes. Si la descripción de ese ejemplo viene dada por los valores {a1, a2, .., an}, la hipótesis más probable es aquella que cumple el mayor valor objetivo probable v_{map}:

$$v_{map} = argmax_{v_j \in V} P(v_j | a1, .., an)$$

Es decir, la probabilidad de que el ejemplo pertenezca a la clase v_j conocidos los valores que lo describen (donde v_j es el valor de la función de clasificación f(x) en el conjunto finito V). Aplicando el teorema de Bayes, se traduce a:

$$v_{map} = argmax_{v_j \in V} P(a1, a2, .., an | v_j) P(v_j)$$

Los valores aj que describen un atributo de un ejemplo cualquiera x son independientes entre sí conocido el valor de la categoría a la que pertenecen. Así la probabilidad de observar la conjunción de atributos aj dada una categoría a la que pertenecen es justamente el producto de las probabilidades de cada valor por separado, entonces la hipótesis más probable es [15, 5]:

$$v_{map} = argmax_{v_j \in V} \sqcap_i P(ai | v_j) P(v_j)$$

2.1.4. *Clustering*

Clustering es una técnica de aprendizaje automático no supervisado que consiste en agrupar una serie de instancias según algún criterio en grupos o clústeres (el criterio suele ser la similitud) sin tener información sobre su clasificación. Es útil en minería de datos y en muchas técnicas exploratorias de análisis de patrones.
Dado un conjunto de instancias, se puede aplicar un algoritmo de *Clustering* para clasificarlas en un grupo específico. En teoría, aquellas que están en el mismo grupo tienen propiedades o características similares, mientras que las que se encuentran en diferentes grupos tienen propiedades o características muy diferentes. Algunos de los algoritmos más comunes son *K-Means Clustering*, *Mean-Shift Clustering*, *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)*, etc. [22]

En la Figura 8 se ve cómo se representan en el plano un conjunto de datos correspondientes a 4000 conductores citadinos y rurales al utilizar el algoritmo de clasificación *K-Means*, tomando k=2. Esto implica que se agrupan en dos *clústeres* (grupos) diferentes. Se ve en el eje de las x uno de sus atributos, la distancia promedio recorrida por día y en el de las y otro de ellos, el porcentaje medio de tiempo que un conductor sobrepasa el límite de velocidad. Se realiza una clara separación según la característica de distancia y se puede suponer que el grupo 1 corresponde a los conductores citadinos y el grupo 2 a los rurales debido a que en este último caso las distancias recorridas son mayores.

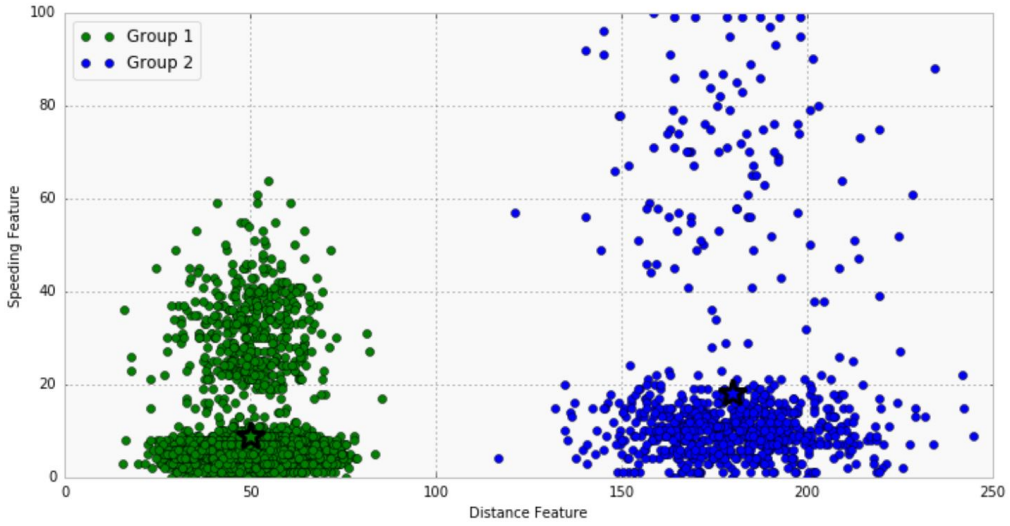

Figura 8: *Clustering* Imagen extraída de *Introduction to K-means Clustering* [24]

2.1.5. Redes Neuronales

Los métodos de aprendizaje basados en redes neuronales proporcionan un enfoque robusto para aproximar funciones de valores reales, discretos y vectoriales. Para algunos problemas, como aprender a interpretar datos complejos de sensores del mundo real, las redes neuronales artificiales, denominadas RNAs, se encuentran entre los métodos de aprendizaje más eficaces actualmente conocidos.

El estudio de las RNAs está inspirado inicialmente por la observación de los sistemas de aprendizaje biológico, que están construidos a partir de redes muy complejas de neuronas interconectadas. En una analogía, las redes neuronales artificiales se construyen a partir de un conjunto densamente interconectado de unidades simples, donde cada unidad toma una serie de entradas de valor real (posiblemente las salidas de otras unidades) y produce una única salida real (que puede llegar a ser la entrada a muchas otras unidades). [15] [7]

Una RNA se determina por las neuronas o unidades y su matriz de pesos. Hay tres tipos de capas de unidades: capa de entrada, capa oculta y capa de salida. Se basan en una idea sencilla: dados parámetros de entrada, hay una forma de combinarlos para poder predecir un resultado. Se trata de encontrar esa combinación de parámetros y aplicarla al mismo tiempo. El proceso de aprendizaje de la unidad puede ser entrenamiento supervisado, es decir, se cuenta con un conjunto de valores objetivo que sirve de patrón de comparación para la salida que retorna la red. Cuando la información fluye a lo largo de las interconexiones de las unidades de forma unidireccional, sin que haya ciclos, la RNA se denomina *feedforward*. Diversos algoritmos se utilizan para el aprendizaje de las RNAs, uno de los más populares es el de propagación reversa (*backpropagation*), y se fundamenta en una estrategia para reducir el error cuadrático

medio.[15, 7]

Como se observa en la Figura 9, la unidad recibe una entrada desde otras unidades o de una fuente externa de datos. Cada entrada tiene un peso asociado w_i, que se va modificando en el proceso de aprendizaje. Cada unidad aplica una función de activación dada σ (típicamente tangente hiperbólica o función sigmoide) a la suma de las entradas ponderadas por los pesos, retornando 1 si es positiva y -1 en otro caso.[7]

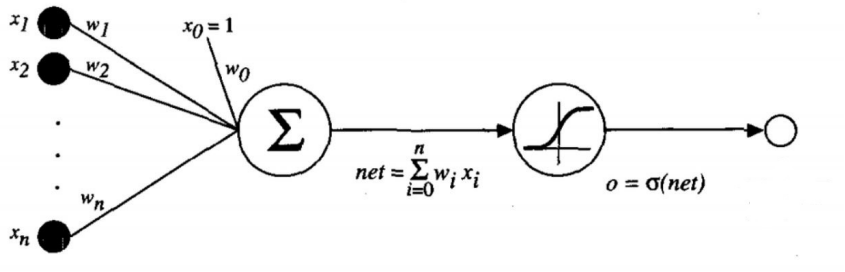

Figura 9: Función Red Neuronal Imagen extraída de *Tom M. Mitchell* [15]

La Figura 10 ejemplifica una red neuronal artificial *feedforward* con n neuronas en la capa de entrada, m en la capa oculta y una única salida.

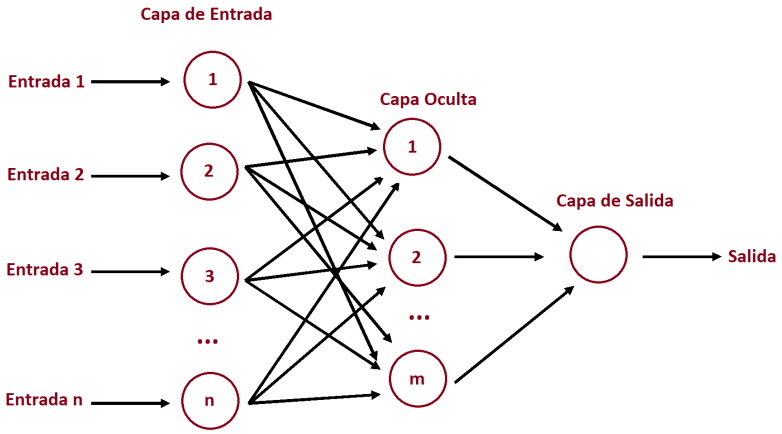

Figura 10: Red Neuronal feedforward

2.1.6. Redes Neuronales Recurrentes

Las redes neuronales recurrentes, más conocidas como RNN, pertenecen a la familia de redes neuronales que son utilizadas para procesar datos secuenciales.

A diferencia de las redes tradicionales, las *RNN* utilizan como información de entrada la salida del nodo anterior, el cual se toma en cuenta para el procesamiento del nodo actual. Por esto, las RNN son capaces de relacionar información previa a la actual; algunas veces, solo se necesita ver información reciente para realizar la tarea actual. Sin embargo, también hay casos en los que se necesita más contexto y es posible que exista una brecha muy grande entre la información relevante y el punto en el que se la necesita. A medida que la brecha crece, las RNN dejan de aprender a vincular esta información. A este inconveniente se le llama problema de "dependencias a largo plazo"("long-term dependencies").

En la Figura 11 se puede ver una representación de una red neuronal recurrente. A la izquierda de la figura se muestra la representación generalizada de la recurrencia, mientras que a la derecha se puede observar la secuencia desplegada para cada tiempo t.

Cada estado A se encuentra definido por el estado anterior, una entrada x_t y una función de activación que transforma la salida, *aplanando* los valores para determinar si se activa o no el nodo.

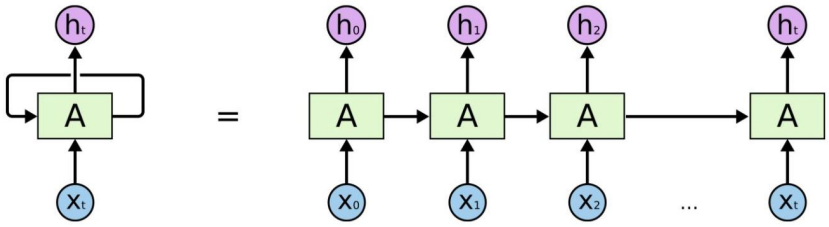

Figura 11: Red Neuronal Recurrente

Matemáticamente, el proceso de guardado de memoria, puede describirse mediante la siguiente fórmula.

$$h_t = f_w(h_{t-1}, x_t)$$

Donde h_t es el estado interno oculto (tiempo t), h_{t-1} el estado interno anterior (tiempo t-1), x_t la entrada tiempo actual (tiempo t) y f_w la función de recurrencia RNN que depende de los pesos W y es independiente de t.

Todas las redes neuronales recurrentes tienen la forma de una cadena de nodos repetitivos de red neuronal. Las LSTM no son la excepción, pero en su caso, el módulo de repetición tiene una estructura diferente. En cada estado, en lugar de tener una única entrada correspondiente al estado anterior, tiene dos.

Estas redes se diferencian de las *RNN* tradicionales en que toman como entrada, no solo el ejemplo actual, sino que también consideran los ejemplos previos.

En la Figura 12 se puede observar una red neuronal LSTM y la estructura de cada nodo de repetición.

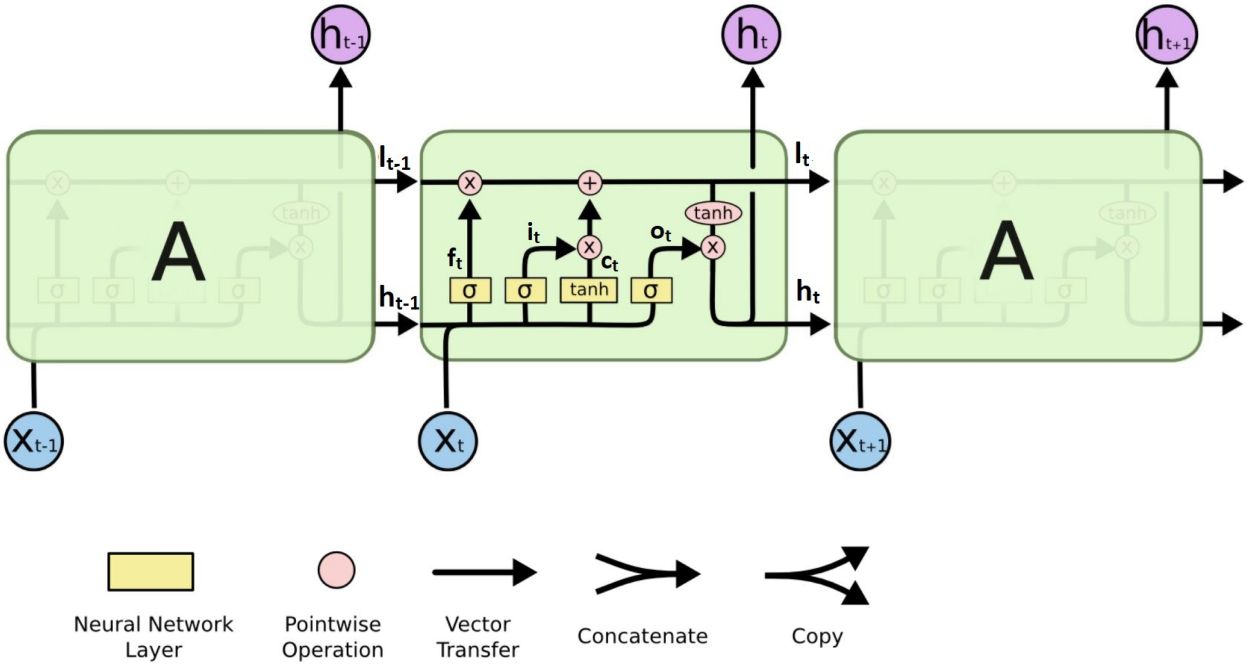

Figura 12: Red Neuronal LSTM Imagen extraída de *Universo Machine Learning* [10]

Las LSTM poseen dos estados de memoria, una memoria a corto plazo y otra a largo plazo. La memoria a corto plazo es esencialmente la salida del estado anterior h_{t-1}. La memoria a largo plazo l_{t-1} se puede ver en la línea horizontal que se extiende en la parte superior del diagrama. La LSTM tiene la capacidad de eliminar o agregar información a esta memoria, regulado por estructuras llamadas puertas. Las puertas son una forma de dejar pasar la información, en la Figura 12 se pueden ver tres capas *sigmoidales* y otra *tanh*. Cada capa *sigmoidal* retorna valores entre {0,1} y describe la cantidad de cada componente que se debe dejar pasar. Un valor de 0 significa "no dejar pasar nada", mientras que un valor de 1 significa "dejar pasar todo". La capa *tanh* retorna valores entre {-1,1}, los negativos indican que se debe quitar la información.

Las distintas puertas representan que información se debe olvidar de la memoria a largo plazo (f_t), que es importante recordar de la nueva información (i_t), si hay nueva información a recordar, cómo se combina con la memoria a largo plazo (l_t) y cuál es la salida del nodo (o_t).

Las redes neuronales recurrentes permiten operar sobre secuencias de vectores en la entrada, en la salida o en el caso más general, en ambas. Se pueden formar distintos tipos de redes según estas secuencias, en este proyecto se utilizan las arquitecturas *Many to One* y una arquitectura *Many to Many* con una *Many to One*.

Las *LSTM Many to One* se tiene como entrada una secuencia de vectores y una única salida, esta arquitectura puede utilizarse en el caso de análisis de sentimiento, donde una oración dada se clasifica como expresión de sentimiento positivo o negativo, o la clasificación del género

de un usuario de Twitter dados ciertos atributos de entrada. En el caso *LSTM Many to Many*, se tiene una secuencia de vectores como entrada y otra secuencia como salida. Un ejemplo de utilización de este tipo de redes puede ser la clasificación de un video donde se quiere etiquetar cada uno de sus fotogramas. [17, 2]

2.2. Proyectos relacionados

En el estudio de proyectos con objetivos similares al presente, se encuentran distintos trabajos relacionados a la detección de género de usuarios de Twitter o de redes sociales que comparten de forma parcial el mismo conjunto de características. Se observa que todos se centran, casi exclusivamente, en el idioma inglés y se enfocan en el contenido de los mensajes que escribe el usuario. Algunos trabajos también utilizan otros atributos, como ser el nombre de usuario, *screen name* y descripción [23, 4] . Otra similitud de todos ellos es que etiquetan a sus usuarios por género masculino o femenino mediante un proceso manual.

En los trabajos *Predicting Age and Gender in Online Social Networks* y *Classifying latent user attributes in Twitter* [18, 20], se considera muy importante el contenido de los mensajes para obtener los atributos y se evalúan las distintas expresiones de los usuarios de cada género: por ejemplo, la repetición de símbolos de puntuación, expresiones o abreviaciones y emoticones.

En particular, uno de ellos cuenta con un *corpus* que consta de 500 usuarios pertenecientes a cada clase (masculino y femenino) y 405.151 *tweets*. Se ignoran a los candidatos que tienen muchos seguidores porque generalmente son compañías o celebridades y se descartan los nombres de usuario argumentando que muchos utilizan nombres que nada tienen que ver con su género o con un nombre real, como ser por ejemplo "graywolf". Aquí los autores describen que para obtener los atributos consideran tres modelos diferentes. Uno de los modelos, al que le llaman sociolingüístico, está basado en las expresiones que utilizan los usuarios de cada género en sus *tweets*; con éste obtienen 3.774 atributos diferentes. Otro se basa en ngramas; para esto normalizan y segmentan el texto, preservando los emoticones y signos de puntuación y luego aplican la frecuencia *tf-idf* considerando unigramas y bigramas. La dimensión de los atributos obtenidos en este caso es 1.256.558. El último de los modelos, al que le llaman *stacked*, resulta de la combinación de los anteriores y con él se logra un acierto de 72,33 % [20].

En *Predicting Age and Gender in Online Social Networks* [18] se utiliza la red social europea llamada *Netlog*[3] para predecir la edad y el género de sus usuarios. En Netlog es obligatorio el ingreso de estos datos al registrarse, y si bien un usuario puede mentirlo, en la mayoría de los casos no ocurre y se ingresa el género real, lo cual permite etiquetar a los usuarios con un margen de error relativamente bajo. Los autores seleccionan los atributos más relevantes de los mensajes utilizando la métrica chi2 y luego de esta selección, en lugar de utilizar su frecuencia en los mensajes, utilizan valores binarios que representan la presencia o ausencia del atributo. Los mejores resultados son obtenidos al experimentar con 50000 atributos, logrando un acierto cercano al 90 %.

[3] https://netlog.com

En el trabajo *Detecting the Gender of a Tweet Sender* [23] se considera el contenido de los *tweets* separando el texto en palabras y eliminando el conjunto de no deseadas (*stop words*), pero además, a diferencia de otros proyectos, se toma en cuenta el nombre de usuario; se logra un acierto del 95.3 %. Si bien este resultado es muy positivo, los autores utilizan el nombre usuario tanto para clasificar manualmente el género como para predecirlo y por tal motivo se obtiene un muy buen resultado.

En el estudio realizado en *Discriminating Gender on Twitter* [4] se añade al contenido de los *tweets*, el *user name*, *screen name* y descripción del usuario para determinar su género. Se incluyen usuarios de distintas nacionalidades, por lo que, en su conjunto de *tweets* hay presencia de varios idiomas (inglés, portugués, español, etc). Para la realización de pruebas se toma en cuenta cada *tweet* de un usuario, un documento conteniendo todos los *tweets* y el resto de los atributos mencionados anteriormente por separado. Luego de estas pruebas se consideran todos los atributos en conjunto obteniendo el mejor resultado y logrando un 91.8 % de acierto.

A modo de síntesis, los atributos más nombrados en los artículos estudiados para la detección del género de un usuario de Twitter, se pueden distinguir entre los que provienen de la cuenta de Twitter del usuario y aquellos que se extraen del propio contenido de sus *tweets*. En el Cuadro 2 y Cuadro 3, se muestran cuales son.

Atributo	Descripción
Nombre de Usuario	Nombre del usuario de Twitter.
Descripción del usuario	Descripción que realiza el usuario de Twitter en su biografía.

Cuadro 2: Atributos propios de la cuenta de Twitter del usuario

Atributo	Descripción
Texto de los *tweets* originales	Corresponde a los textos de los *tweets* de un usuario de Twitter. Generalmente se descartan *hashtags* y *urls*.
emoticones	Se trata de detectar aquellos emoticones que son más utilizados por los usuarios de cada género.
Palabras o expresiones más utilizadas	Refiere a las palabras, símbolos de puntuación o expresiones que son más utilizadas por los usuarios de cada género.

Cuadro 3: Atributos extraídos del texto escrito en los *tweets*

Aquellos trabajos que obtienen mejores resultados tienen la particularidad de considerar

en su conjunto de atributos alguno de los datos que utilizan para etiquetar manualmente los usuarios. En su gran mayoría, se elige como algoritmo de clasificación al SVM [23, 20, 18, 4], pero en otros se utilizan además, Naïve Bayes y una implementación de Winnow2 [4], algoritmo similar a RNA.

3. Corpus

Se decide construir un *corpus* para el estudio de variables demográficas para clasificar el género de un usuario de Twitter de habla hispana, ya que hasta el momento no existe uno. Para esto se seleccionan 7.000 usuarios de género femenino y 7.000 de género masculino, de los cuales se obtienen un total de 649.777 y 705.782 *tweets* respectivamente. Además se relevan 240 usuarios que se reservan para los experimentos, 161 son de género masculino y 79 de género femenino, para ellos se obtiene un total de 9.901 y 4.694 *tweets* respectivamente.

A continuación se detalla el armado del *corpus* de datos, las decisiones tomadas y los problemas encontrados durante el proceso.

3.1. Construcción del corpus

Una de las fases iniciales e importantes del proyecto consta de elegir una muestra de perfiles de Twitter de habla hispana que permita obtener sus atributos y armar el conjunto de datos a utilizar. Dado que los distintos países de habla hispana tienen diferentes formas de expresarse y usan distintas palabras, inicialmente se decide abarcar la región del Río de la Plata para asegurar cierta homogeneidad en las expresiones de todos los usuarios.

Al querer obtener los usuarios de esta región, se encuentra el primer problema: Twitter permite completar el campo ubicación con una ciudad, pero si el usuario prefiere completarlo con una frase también puede hacerlo. La gran mayoría de los usuarios optan por colocar una frase por lo que se dificulta saber a qué país o región pertenecen y resulta muy complicado compararlo con una base de datos de ciudades y países. Como no es posible determinar la región ya que el atributo ubicación no tiene información relevante para la mayoría de los casos, se decide considerar los usuarios de habla hispana, sin restringirse al Río de la Plata, asumiendo el riesgo de que los usuarios escriban de manera muy diferente.

Al considerar los usuarios cuyo idioma nativo es el español, hay que enfrentar otro problema, muchos tienen *tweets* en varios idiomas. Se toman 10 usuarios que *twittean* en varios idiomas, se analizan los *tweets* y se detecta que todos tienen más cantidad de *tweets* en su idioma nativo que en cualquier otro. Por esta razón, para seleccionar los usuarios, se obtienen sus *tweets* y se consideran aquellos que tienen la mayoría escritos en español.

Se arma un conjunto de 240 usuarios etiquetados manualmente por género; para 50 de ellos se sabe con veracidad su género porque pertenecen a personas conocidas por las autoras de este trabajo, y el resto son famosos con una cuenta verificada por Twitter, es decir, que se conoce que realmente la cuenta es del famoso que dice ser. Incluir estos últimos conlleva tomar otro riesgo: los usuarios que son famosos muchas veces no manejan sus propias cuentas sino que alguien lo hace por ellos. Es factible que, por ejemplo, a un hombre famoso le administre la cuenta una mujer y *twitee* por él o viceversa.

Como la cantidad de usuarios es muy baja para poder entrenar los algoritmos de aprendizaje automático y además contar con un conjunto para testeo, se obtiene una base de datos, cedida por la empresa IDATHA, de 50.000 usuarios de Twitter ya clasificados por género. Estos

usuarios son etiquetados utilizando diccionarios de nombres sobre el atributo *name* del usuario, y si eso no resuelve el género, se aplica un modelo que utiliza el atributo *name* y *screen name* del usuario. Las fuentes de datos utilizadas son las mismas que las del proyecto *gender-detector*[14], el cual ofrece diccionarios de nombres para Estados Unidos, Inglaterra, Argentina y Uruguay.

Los usuarios tienen tres clasificaciones posibles: hombre, mujer o indeterminado. Para cada clasificación se cuenta con 30.013, 15.209 y 4.778 usuarios respectivamente. La clasificación como indeterminado sucede cuando el nombre no se encuentra en el diccionario o cuando un nombre es ambiguo. En este proyecto se descartan estos usuarios, considerando únicamente los clasificados como hombre o mujer.

Se analizan varios usuarios por separado, se descartan aquellos que ya no tienen una cuenta de Twitter y luego se reduce la cantidad de hombres y mujeres para obtener conjuntos de usuarios de igual tamaño. Una vez obtenido el conjunto de usuarios compuesto por 7000 usuarios etiquetados como mujer y 7000 usuarios etiquetados como hombre, más los 240 usuarios etiquetados manualmente, se utilizan los primeros 14.000 para el entrenamiento y ajuste de los algoritmos, y los otros 240 se decide reservarlos para la evaluación final de los modelos obtenidos.

3.2. Selección de atributos

Luego de analizar por algunos días varias cuentas de distintos usuarios, resulta interesante considerar el contenido de los *tweets* dado que otorga información sobre la forma de escribir del usuario y lo que comparte, además muchos de los trabajos similares estudiados se basan en este contenido para la clasificación por género[18, 16, 19, 20, 23]. Por otro lado, se considera la descripción y foto de perfil del usuario, descartándose los atributos *name* y *screen name* dado que existe una correlación entre ellos y los que utiliza la empresa IDATHA para etiquetar a los usuarios que nos proveen. De considerarlos, se estaría sesgando el algoritmo para que prediga con ese atributo determinante el género.

Se descartan también los *retweets*[4] puesto que no aportan información sobre la forma de escribir del usuario y tampoco se consideran los símbolos # que indican el comienzo de los *hashtags*, pero sí el conjunto de caracteres que viene luego de este símbolo porque se cree que puede tomarse como una nueva palabra que muchos usuarios mencionen en sus *tweets*.

A modo de ejemplo, si se considera un usuario para el cual se obtienen únicamente dos *tweets* (para simplificar), como es el caso que se muestra en la Figura 13, el conjunto de atributos para este usuario consta de los *tweets* simples, la descripción, un documento que contiene los textos concatenados de los dos *tweets* más la descripción, una lista de emoticones que utiliza en sus *tweets* y el conjunto de etiquetas y categorías correspondientes a la foto de perfil. En el Cuadro 4 se muestra el detalle.

[4] *retweet* es la acción que permite reenviar un tweet de otra persona

Figura 13: Ejemplo

Atributos	
tweets simples	{*Tatuados en el brazo me quedan tus abrazos y en el pecho esta canción De una extraña manera se que tu voz me espera en algún lugar*} {*Inventar otro final después del temporal, la vida se revela y viene por más... Cuando digan que ya está escrito el guión es cuando se hace urgente una revolución. Darse la oportunidad...*}
Descripción	{ *Fanática del carnaval Uruguayo*}
Documento con *tweets* y descripción concatenados	{*Tatuados en el brazo me quedan tus abrazos y en el pecho esta canción De una extraña manera se que tu voz me espera en algún lugar Inventar otro final después del temporal, la vida se revela y viene por más... Cuando digan que ya está escrito el guión es cuando se hace urgente una revolución. Darse la oportunidad... Fanática del carnaval Uruguayo* }
Emoticones	🎶 💕 🎭
Etiquetas y Categorías de la foto de perfil	{ *"people_young", "woman", "wall", "indoor", "person", "posing", "beautiful", "smiling", "black", "pretty"*}

Cuadro 4: Atributos

3.2.1. Atributos de texto

Como atributos de texto se considera el contenido de los *tweets* y la descripción del usuario. Mediante el servicio de Twitter [27] se obtienen los últimos 200 *tweets* de cada uno de los usuarios seleccionados y se guardan como texto, etiquetados según el género en una base de datos MongoDB. Luego se unen en un solo documento los *tweets* de cada usuario, concatenándolos y obteniendo un único texto con todos ellos (de ahora en más les llamamos *tweets concatenados*).

Analizando el contenido de los distintos *tweets* se observa que las imágenes y videos se representan con una *url*. Se nota que las *urls* se encuentran con el formato *t.co*, formato propio de Twitter [25]. Este formato modifica todas las *urls* para que tengan 23 caracteres. Además de modificarle el largo, una misma *url* no se representa siempre de la misma forma al aplicarle este formato. Por ejemplo, si un usuario *twittea* la *url*: *www.gogle.com* en más de una oportunidad, no necesariamente son iguales las *urls* en formato *t.co*. Por esta razón, si se dejan las *urls* en este formato es altamente probable que no se pueda aprender algún patrón de ellas, por lo que se decide modificar todas las *urls* de los *tweets* almacenados.

La descripción del usuario en caso de tenerla se procesa como texto y se decide unir al documento que posee los *tweets* concatenados del usuario correspondiente.

Para utilizar los distintos algoritmos de clasificación, se debe transformar la información de cada usuario para que los clasificadores la tomen como datos de entrada y la interpreten de forma adecuada. Este problema se resuelve de dos maneras distintas dependiendo de que clasificador es utilizado.

Para los clasificadores Árboles de Decisión, Bosques Aleatorios, Naïve Bayes, SVM, Clustering y Redes Neuronales, se utiliza una bolsa de palabras para obtener los atributos del texto de los *tweets*. Una bolsa de palabras es una representación de texto que describe la ocurrencia de palabras dentro de un documento, para medir esta ocurrencia se utiliza *tf-idf* (*Term Frequencies - Inverse Document Frequency*). Esta medida expresa cuán relevante es una palabra para un documento en una colección de ejemplos.

tf(t,d,D) corresponde al número de ocurrencias de un término t en un documento d de la colección D.

idf(t,D) mide la importancia de un término específico por su relevancia dentro del documento, se expresa mediante la fórmula:

$$idf(t,D) = log\frac{|D|}{1 + |\{d \in D/t \in d\}|}$$

Finalmente la medida *tf-idf*:

$$tf\text{-}idf(t,d,D) = tf(t,d) * idf(t,D)$$

Donde t denota los términos, d cada documento y D la colección de documentos.

En la Figura 14 se observa como se visualiza un *tweet* de usuario con una *url* en la aplicación de *Twitter*, mientras que en la Figura 15 se muestra como devuelve el mismo *tweet* el servicio

con la *url* en el formato que la representa Twitter. Para poder tomar en cuenta exactamente lo que *twittea* el usuario, se calcula *tf-idf* con este *tweet* considerando la *url* original. Se obtiene la matriz del Cuadro 5 en el cual se muestra la frecuencia de cada palabra. Se considera como una palabra, aquella que se encuentra entre dos delimitadores, por ejemplo un espacio, los símbolos de puntuación o por la / en el caso de las urls. *Tf-idf* retorna una matriz en donde las columnas representan a cada uno de estas palabras (atributos del texto del *tweet*) y las filas representan a cada documento. En cada celda de la matriz se encuentra la frecuencia correspondiente antes explicada.

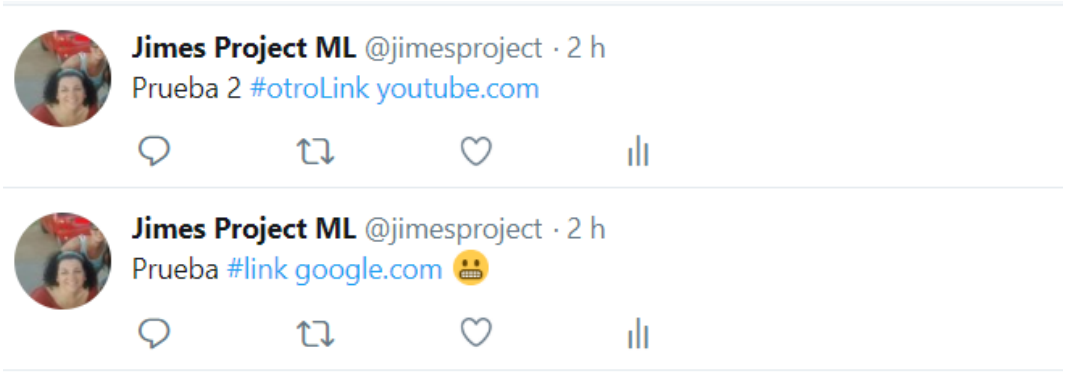

Figura 14: Tweet de usuario

```
Tweet 1: Prueba 2 #otroLink https://t.co/jPa3YjOnhU

Tweet 2: Prueba #link https://t.co/LV4YXXk76O
```

Figura 15: Tweet del usuario devuelto por el servicio

Palabras del *tweet*/ Frecuencia tf-idf	com	google	http	https	link	otroLink	Prueba	www	youtube
Tweet 1	0.3552	0	0	0.4992	0	0.4992	0.3552	0	0,4992
Tweet 2	0.3177	04465	04465	0	04465	0	0.3177	04465	0

Cuadro 5: Matriz de tf-idf para el *tweet* de ejemplo

Por otro lado, para las redes LSTM se utiliza una representación distinta a la antes explicada dado que se deben traducir los atributos a vectores que los representen en el espacio. Para esto, se debe hacer un entrenamiento de las palabras en determinado contexto (en este caso el

contexto son los *tweets*) y de allí se obtienen los vectores que muestran que tan vinculados están los atributos mediante la distancia a la que se encuentran representados en el espacio. Dado que este entrenamiento lleva mucho tiempo, se recurre a vectores de palabras preentrenados de dimensión 300, proporcionados por Mathias Etcheverry del grupo de PLN del Inco [28].

Para obtener los vectores correspondientes a los atributos de los usuarios utilizados para la clasificación, se toma cada palabra del *tweet* de un usuario y se obtiene el vector asociado. Si la palabra no existe en el conjunto de vectores, no se toma en cuenta. Una alternativa a esta solución puede ser tomarlo como un vector nulo en caso de su inexistencia, pero dado que los vectores que se utilizan están entrenados en otro contexto, sucede que muchas palabras que se utilizan en Twitter no se encuentran representados y por eso se descartan, para no incluir muchos vectores nulos que no agregan información.

3.2.2. Atributos generales

Se consideran como atributos generales a un usuario para los clasificadores Árboles de Decisión, Bosques Aleatorios, Naïve Bayes, SVM, Clustering y Redes Neuronales, la foto de perfil y los emoticones. A pesar que en los artículos relacionados no se menciona la foto de perfil del usuario como un atributo para determinar el género, entendemos que pueden ser de utilidad y proporcionar información sobre las características de los usuarios. En el caso de asumir que un usuario coloca una imagen verdadera de su persona, es posible determinar con distintas herramientas muchas de sus características, por ejemplo, su género, una aproximación de su edad, rasgos faciales, estado de ánimo, entre otras. Generalmente, esto es algo que no podemos asumir porque los usuarios suelen falsificar imágenes, colocar imágenes de familiares o de famosos a los que idolatran, así como imágenes de lugares u objetos. De todas formas, resulta interesante contar con este análisis para poder recolectar distintas características y verificar si es posible obtener patrones que luego ayuden a clasificar a los usuarios.

Para cada usuario, se cuenta con la *url* de su foto de perfil, esto lo provee el json que retorna el servicio de Twitter. En un principio, se piensa en obtener cada una de estas imágenes y guardar el binario en la base de datos ya que el usuario a lo largo de las pruebas puede cambiar la foto de perfil.

Con el servicio de Microsoft Computer Vision API [1], se logra analizar cada foto de perfil y obtener etiquetas (palabras que etiquetan a la imagen) y categorías (categorías a las que pertenece la imagen) que la describen. Se decide consultar una única vez el servicio de Microsoft con la imagen y almacenar la respuesta json con las categorías y las etiquetas en la base de datos en vez de guardar la imagen en sí misma. Para incorporarlas se tratan como palabras y se determina si un usuario las tiene o no luego de analizar su foto de perfil, considerando el valor 1 en caso positivo y el valor 0 en caso negativo.

Para añadir de los emoticones al *corpus*, se recorren los documentos que contienen los *tweets* concatenados de cada usuario y se obtienen todos los emoticones que son *twitteados*. Antes de agregarlos se deben agrupar aquellos emoticones que representen lo mismo pero que son de distinto color, a modo ejemplo, la mayoría de las personas utilizan el emoticón del pulgar

para arriba con distintos tonos de piel. Se decide que el color no aporta información por lo que se agrupan todos los emoticones de pulgar para arriba como un solo emoticón. Una vez realizada esta agrupación, se procede de la misma manera que con las imágenes.

En el caso de las redes LSTM se utilizan únicamente los emoticones como atributos generales. Se representan de una forma distinta a la antes explicada dado que se deben traducir los atributos a vectores que los representen en el espacio. Para esto, se debe hacer un entrenamiento de los emoticones en determinado contexto (en este caso el contexto son los *tweets*) y de allí se obtienen los vectores que muestran que tan vinculados están los atributos mediante la distancia a la que se encuentran representados en el espacio, o realizar un reentrenamiento de los vectores de palabras proporcionados por Mathías Etcheverry. Puesto que tanto este entrenamiento como el reentrenamiento llevan mucho tiempo, se recurre a vectores preentrenados de emoticones obtenidos del proyecto *emoji2vec* [21]. Este conjunto de vectores es obtenido en un contexto distinto al de vectores de palabras pero coincide en su dimensión.

3.3. Análisis de atributos

Es de interés analizar los distintos atributos seleccionados para determinar si existe algún tipo de relación entre los usuarios de cada género. Dado que un usuario se puede representar con un vector que contiene los valores de sus atributos para la mayoría de los clasificadores, se toma esta representación para hacer un análisis gráfico. Puesto que los vectores tienen como dimensión la cantidad de atributos que es del orden de los 100000 si se consideran todos, es difícil visualizar su representación en el espacio, por esta razón se utiliza la librería *TSNE* de *sklearn* que conserva la relación de distancia de los vectores de atributos de un usuario en una dimensión menor.

En la Figura 16 se observan los atributos representados en tres dimensiones. Los puntos rojos corresponden a los de las mujeres y los azules a los de hombres. Si bien se obtiene una representación que se puede visualizar, es muy difícil identificar atributos determinantes para cada clase.

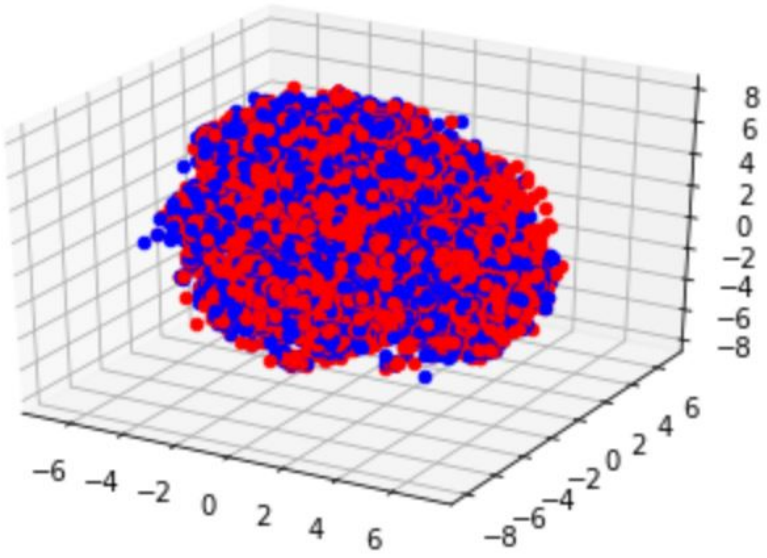

Figura 16: Atributos tf-idf en 3 dimensiones

Considerando que la representación anterior no es válida para las RNN LSTM porque cada usuario se representa por un conjunto de vectores que corresponden a las palabras y emoticones que utiliza, se presenta también un análisis de las palabras y emoticones más frecuentes utilizados por los usuarios.

Para visualizar de forma sencilla las palabras más *twitteadas* por los usuarios de cada género, se muestra una nube de palabras para cada caso, discriminando entre los usuarios que se utilizan para el entrenamiento y los que se reservan para las pruebas. Una nube de palabras o etiquetas es una representación visual en la cual se muestra la importancia relativa de cada palabra en base al tamaño y color, siendo las de mayor tamaño aquellas que aparecen con más frecuencia.

Para los emoticones, se muestra también una nube de palabras que contienen las etiquetas que los representan y se exponen gráficamente los más utilizados.

3.3.1. Palabras Comunes

En las Figuras 17 y 18 se observan las palabras más utilizadas por hombres y mujeres del conjunto de entrenamiento. Se notan claramente con mayor frecuencia de utilización las palabras: {amigo, ante, año, buena, cosa, clase} para el género masculino y las palabras:{amiga, amigo, año, clase, cosa, bueno, día} para el género femenino, entre otras. Si bien tienen muchas en común, también hay otras como ser “huevo”, “grande” o “gana” que son más utilizadas por los hombres y no aparecen en el conjunto de palabras de las mujeres con la misma frecuencia.

En las Figuras 19 y 20 se observan las palabras más utilizadas por hombres y mujeres del

conjunto de prueba. En este caso también coinciden algunas de las palabras más utilizadas por ambos, entre ellas: “amigo” y “año”. Se destacan para el género femenino las palabras “aire”, “amiga”, “beso” y “fiesta”. Mientras que para el género masculino, se destacan las palabras “ante”, “hora” y “grande”.

Si se analizan las palabras más frecuentes por género para los conjuntos de entrenamiento y prueba, se observan varias coincidencias, pero también se encuentran palabras que se destacan, en particular, la palabra “ante” es muy frecuente en ambos conjuntos para los hombres y la palabra “amiga” es muy frecuente para las mujeres.

Figura 17: Palabras más *twitteadas* por los Hombres Entrenamiento

Figura 18: Palabras más *twitteadas* por las Mujeres Entrenamiento

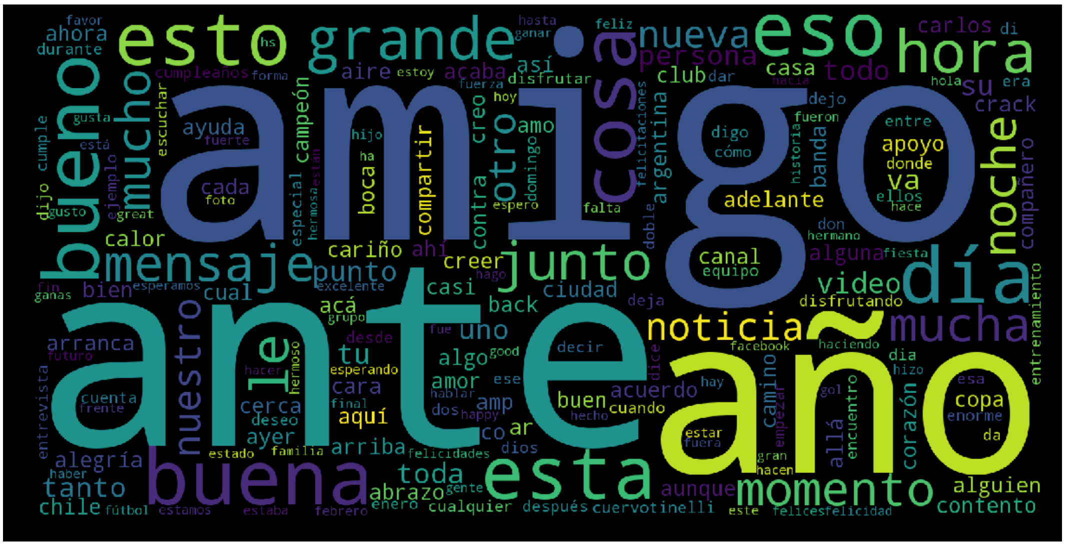

Figura 19: Palabras más *twitteadas* por los Hombres Prueba

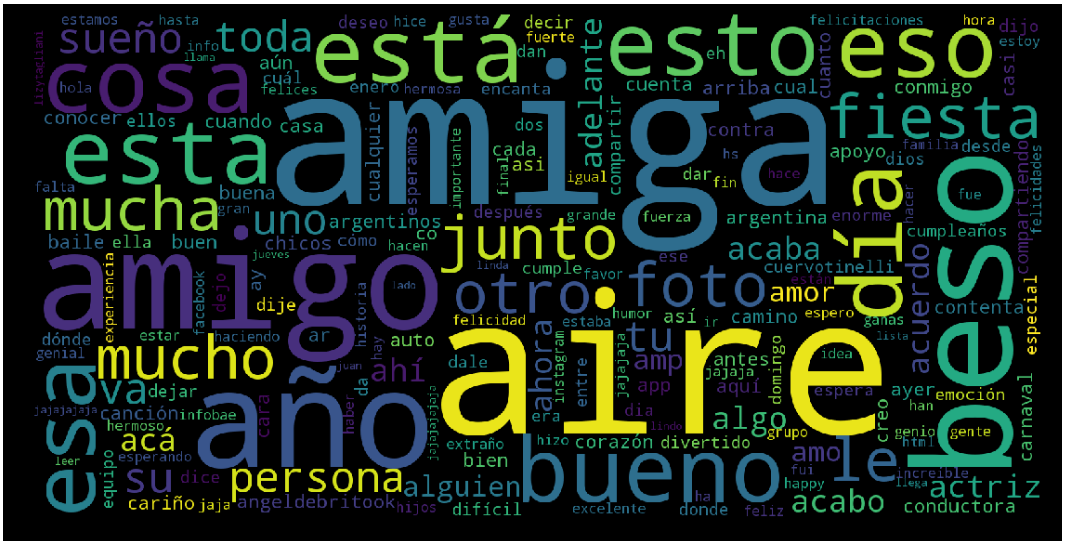

Figura 20: Palabras más *twitteadas* por las Mujeres Prueba

3.3.2. Nube de Etiquetas de Emoticones

En las Figuras 21 y 22 se presentan los emoticones más utilizados por género para los usuarios del conjunto de entrenamiento. Al igual que sucede con las palabras más frecuentes, varios son utilizados tanto por hombres como mujeres y con frecuencias similares, es el caso de los siguientes: “cara con lágrimas de alegría”, “cara sonriente con ojos de corazón”, “ojos”, “corazón rojo” o “manos aplaudiendo”. Sin embargo, hay algunos emoticones que si bien tienen una menor frecuencia de uso, son propios de cada género; por ejemplo, “pelota de fútbol” en los

hombres y “signo femenino” en las mujeres.

En las Figuras 23 y 24 se presentan los emoticones más utilizados por género para los usuarios del conjunto de prueba. En este caso, aunque se encuentran coincidencias en la utilización de algunos emoticones, hay una diferencia más notoria en cuanto a los más utilizados por hombres y mujeres. Se destacan para los hombres: “bíceps flexionado”, “corazón rojo intenso”, “signo de pulgar hacia arriba”, “pelota de fútbol” y “círculos de colores”; mientras que para las mujeres se destacan: “rodando por el suelo de risa”, “palo de corazones”, “signo femenino”, “cara sonriente con ojos de corazón” y “rostro pensativo”.

Si se observan los conjuntos de entrenamiento y prueba por género, se encuentra que estos son más parecidos en el caso de los hombres ya que comparten muchos de los emoticones más frecuentes. No es tan así en el caso de las mujeres, para las cuales aparecen varios emoticones nuevos en el conjunto de prueba.

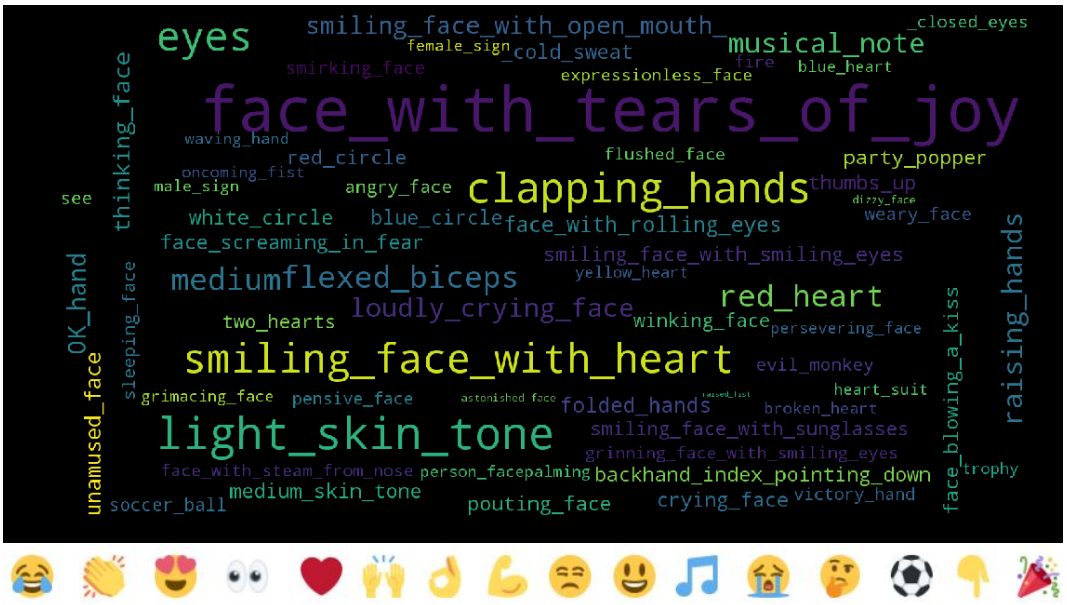

Figura 21: Emoticones más *twitteadas* por los Hombres Entrenamiento

Figura 22: Emoticones más *twitteados* por las Mujeres Entrenamiento

Figura 23: Emoticones más *twitteadas* por los Hombres Prueba

Figura 24: Emoticones más *twitteados* por las Mujeres Prueba

4. Implementación

En este capítulo se explica cuales son las decisiones tomadas para construir los diferentes modelos, algoritmos de clasificación y las librerías más importantes utilizadas. Además, se detallan las herramientas, *hardware* y *software* que se utilizan para la implementación.

Como se explica en el Capítulo 3, se construye un *corpus* de datos que contiene los últimos 200 *tweets* de cada usuario, un documento que contiene todos los *tweets*, etiquetas y categorías de la foto de perfil, la descripción y emoticones que utiliza. Para determinar el género de un usuario de Twitter se proponen dos alternativas:
(a) considerar todos los atributos disponibles de su cuenta;
(b) tomar en cuenta un *tweet* y sus emoticones. En las siguientes secciones se detalla que alternativa se utiliza en la aplicación de cada algoritmo.

4.1. Algoritmos de clasificación

Para la implementación se utilizan distintos algoritmos de clasificación: Árbol de Decisión, Bosques Aleatorios, Naïve Bayes, Red Neuronal, Clustering y SVM. Para todos ellos se considera la alternativa (a) en la cual cada ejemplo de entrada contiene todos los atributos disponibles de una cuenta de usuario. Dado un usuario, un ejemplo de entrada se representa como un vector u como el siguiente:

$$u =< F, EM, ET, C >$$

Donde F representa las frecuencias *tf-idf* de las palabras, EM, ET y C los valores binarios que indican si el usuario utiliza determinado emoticón, si contiene en su foto de perfil una etiqueta o categoría respectivamente. Esto se corresponde con la matriz explicada en el Capítulo 3. En el caso de las RNN LSTM se considera la alternativa (b); se toma como ejemplo de entrada un conjunto de vectores de palabras de tamaño equivalente al promedio de palabras que contiene un *tweet* de un usuario y además se toman en cuenta los emoticones que utiliza.

4.1.1. Naïve Bayes

Para implementar este modelo se utiliza el algoritmo MultinomialNB de la librería *sklearn*. Este algoritmo aplica naïve bayes para datos distribuidos multinomialmente que es una de las variantes utilizadas en la clasificación de texto.

Se toma en cuenta el parámetro α que suaviza la distribución multinomial considerando que existen atributos con valor cero en muchos ejemplos; el valor de α que maximiza la perfomance del algoritmo es: 0.03. También se configura el parámetro *fit_prior* que determina que se aprenda de la probabilidad de los valores de los distintos atributos de los ejemplos previos para calcular la actual, aumentando la performance del algoritmo.

4.1.2. Árbol de Decisión

Para implementar este modelo se utiliza el algoritmo DecisionTreeClassifier de la librería *sklearn*.

La configuración que maximiza la performance de este algoritmo es aquella que para elegir cada nodo del árbol utiliza el criterio de menor entropía, lo que implica que el atributo que se elige otorga mayor información. El número mínimo de ejemplos que debe haber para determinar si un nodo es una hoja es uno y el mínimo para dividir un nodo interno es 1500. La profundidad del árbol y la cantidad de nodos se deja crecer de forma ilimitada. Con estos parámetros se obtiene un árbol con 89 nodos y una profundidad máxima de 30.

4.1.3. Bosques Aleatorios

Como implementación de este modelo se utiliza el algoritmo RandomForestClassifier de la librería *sklearn*. En este algoritmo se toman distintos subconjuntos de igual tamaño correspondientes a los ejemplos de entrenamiento y para cada uno de ellos se construye un árbol de decisión que conforma el bosque.
Se crea un bosque compuesto por diez árboles, cada uno de ellos tiene la misma configuración que la explicada en la sección anterior. Esta configuración es la que maximiza la performance del algoritmo.

4.1.4. Clustering

En el caso de *clustering* se utiliza el algoritmo no supervisado KMeans de la librería *sklearn*, el cual dado un conjunto de ejemplos como el vector u antes mencionado, calcula la distancia mínima entre ellos y los agrupa según su cercanía. Como se quiere predecir el género de un usuario retornando si es masculino o femenino, se definen dos *clusteres* para que los ejemplos se agrupen en alguno de ellos y se consideran diez iteraciones para la elección de los centroides.

4.1.5. Redes Neuronales

En este modelo se utiliza el algoritmo MLPClassifier de la librería *sklearn*. La red se conforma por m cantidad de nodos de entrada, siendo m la cantidad de atributos, 20 capas ocultas que contienen 20 nodos cada una y una capa de salida con un único nodo que retorna el valor cero en caso de que el género sea masculino y el valor uno en caso de que sea femenino. Se utiliza una función rectificadora como función de activación y un gradiente estocástico con el fin de optimizar los pesos. Se realizan 200 iteraciones en las cuales se consideran lotes de 200 ejemplos. Además, se configura un parámetro de finalización temprana y como consecuencia se finaliza el entrenamiento en caso de que el acierto promedio no mejore en dos iteraciones consecutivas.

4.1.6. SVM

Para implementar este modelo se utiliza el algoritmo LinearSVC de la librería *sklearn* el cual se compone de un *kernel* lineal. Se configura una penalización para evitar la clasificación errónea de los ejemplos; un valor grande en este parámetro implica que se elijan hiperplanos con un margen más pequeño y un valor pequeño que se elijan hiperlanos con mayor margen. El valor de este parámetro llamado C que maximiza la performance del algoritmo es 0.85 y la función de penalización que se utiliza es la norma euclídea.

4.1.7. Redes Neuronales Recurrentes LSTM

Los algoritmos para la Red Neuronal Recurrente LSTM se obtienen de la librería *Keras* [11]. Estas tienen como entrada un conjunto de tamaño fijo de vectores, denominado ventana. En nuestra implementación, una ventana contiene la representación de las palabras de un *tweet* y los emoticones de un usuario como vectores. Se toman en cuenta solo aquellas palabras y emoticones existentes en el conjunto de vectores preentrenados que conforman el *corpus*.

La particularidad de estas redes es que pueden vincular información a lo largo del tiempo, por lo que es importante aprovechar en el análisis de los *tweets*, el contexto en el que aparecen las palabras y emoticones. Dado que el promedio de palabras por *tweet* es 12, se decide considerar una ventana de este tamaño. Para aquellos *tweets* que contengan menos de 12 palabras se completa la ventana con el *tweet* siguiente. En los casos en que no existan más *tweets* se completa con valores nulos.

Se decide probar la arquitectura de LSTM *Many to One* y una arquitectura que se compone de una *Many to Many* con una *Many to One*, para esto se modifican los parámetros de la capa LSTM de *Keras*.

Dado que no es posible asegurar que las relaciones de los vectores de palabras y de emoticones sean las correctas, se decide armar una red LSTM de vectores de palabras y una red LSTM de vectores de emoticones. Ambas se entrenan por separado y luego se unen las salidas utilizando la capa *Concatenate* de *Keras*.

La construcción de esta red neuronal se hace en etapas. Primero se determina la LSTM de vectores de palabras y una vez obtenida, se procede a definir e incorporar la LSTM de emoticones. A modo de ejemplo en la Figura 25 se muestra la arquitectura de una red *LSTM Many To One* con una ventana de 4 vectores, cada uno de ellos es entrada de cada nodo. Luego se procesan y el último nodo retorna el resultado de la predicción, en este caso, el género del usuario. Se aplica una función de activación *sigmoidal* para *aplanar* el valor de salida. Esta función se modifica en las pruebas para determinar la variación más adecuada.

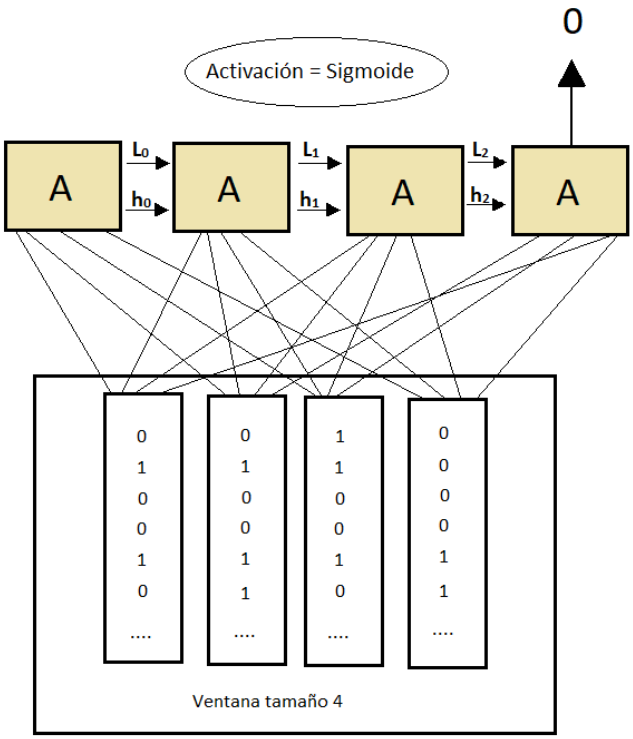

Figura 25: Implementación LSTM Many to One

En la Figura 26 se puede ver la arquitectura de una red *LSTM Many To Many*, en la cual cada nodo retorna un resultado intermedio que luego se toma como entrada de una red *LSTM Many to One* que realiza la predicción del género del usuario. En esta arquitectura también se aplica una función de activación *sigmoidal* para *aplanar* el valor de salida.

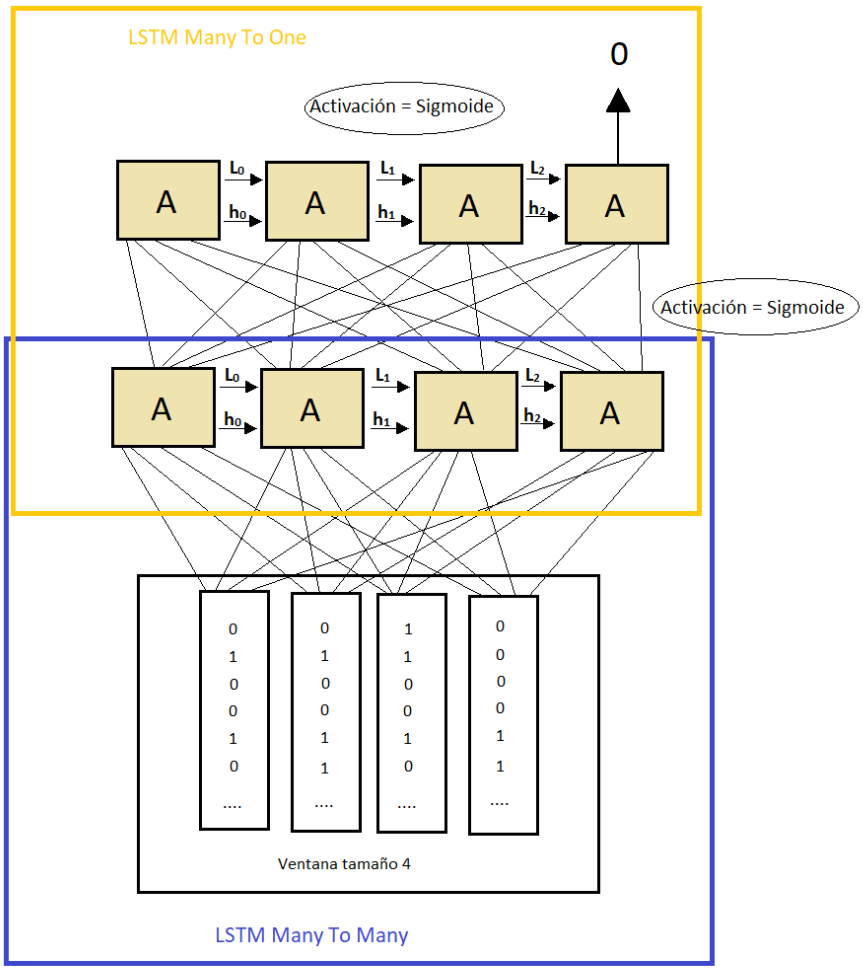

Figura 26: Arquitectura LSTM Many to Many con LSTM Many to One

En la Figura 27 se muestra la concatenación de dos redes *LSTM Many To One*, una correspondiente a los vectores de palabras y otra a los vectores de emoticones, la capa de concatenación recibe como entrada los resultados de ambas redes y realiza la predicción final del género del usuario. La capa de salida es una capa densa con función de activación *sigmoidal*.

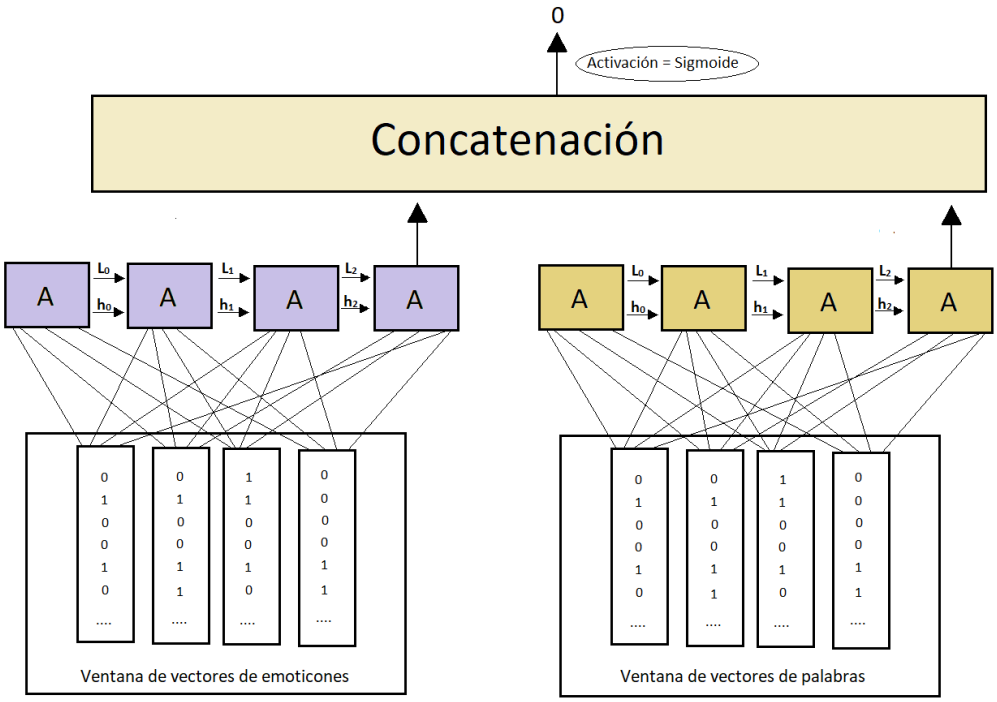

Figura 27: Concatenación de LSTM de palabras y emoticones

4.2. Estructura de la base de datos

Una vez seleccionados y procesados los usuarios, la siguiente decisión a tomar es en donde se almacenan los datos y de qué manera. En primera instancia se prueba con bases relacionales como MySql y PostgreSql pero se encuentran inconvenientes con la representación de los emoticones y por tal motivo se descartan. Luego investigando, se encuentra que la base de datos no relacional MongoDB es la que mas se adecúa a lo que se busca. Los emoticones se representan correctamente, permite guardar objetos en colecciones con todos los atributos de un *tweet* en formato *json* y además, IDATHA lo utiliza para la representación de todos los usuarios que nos proveen. Se guardan los datos de los *tweets*, la descripción y las imágenes correspondientes a las fotos de perfil de los usuarios, se trabaja con una base de datos no relacional a la que se le llama *proyectodb* y se crean colecciones para guardar los datos. Cada elemento de esta colección es un documento *json* que contiene la información correspondiente.

En el Cuadro 6 se muestran las colecciones creadas para almacenar la información correspondiente a los 14000 usuarios proporcionados por IDATHA y su descripción.

Además de estas colecciones, se crean colecciones homólogas para los 240 usuarios etiquetados manualmente.

Colección	Descripción
usuarios_mujeres, usuarios_hombres	En cada documento *json* de estas colecciones se encuentran datos del usuario como ser el *screen_name*, *name* y descripción.
tweets_mujeres_simple_url_cambiada, tweets_hombres_simple_url_cambiada	Contienen los últimos 200 *tweets* con la *url* real y completa de cada usuario.
tweets_mujeres_concat_url_cambiada, tweets_hombres_concat_url_cambiada	En estas colecciones hay un documento por usuario , el cual contiene los últimos 200 *tweets* concatenados con la *url* real y completa.
foto_perfil_mujeres, foto_perfil_hombres	Se encuentran los datos de categorías y etiquetas correspondiente a la foto de perfil de cada usuario.
emojis_mujeres, mojis_hombres	Para cada usuario contiene una lista de emoticones utilizados en sus *tweets*.

Cuadro 6: Colecciones de la base de datos proyectodb

4.3. Herramientas, Software y Hardware

Para este proyecto se generan programas y *notebooks* en el lenguaje de programación Python en su versión 3.6.0 y Anaconda 4.3.0 (64-bit).

Los algoritmos de clasificación se importan en su gran mayoría de la librería *scikit-learn* en su versión 0.18.1. Esta librería se utiliza también para la extracción de atributos. Los algoritmos y modelos de redes neuronales recurrentes se obtienen de la librería *Keras* en su versión 2.0.6

Se utiliza el servicio de Twitter para la obtención de los *tweets* de los usuarios y Computer Vision API de Microsoft para la obtención de los atributos de las fotos de perfil.

Para obtener los emoticones, se utiliza la librería *emoji* de Python en su versión 0.4.5, en particular se utiliza la función *demojize* para traducir el emoticón a una palabra.

Como motor de base de datos se utiliza MongoDB en su versión 3.4 y la librería *pymongo* también en su versión 3.4.

El ambiente de trabajo se arma principalmente en nuestras computadoras personales, una de ellas con sistema operativo Windows 10 de 64 bits, 16 GB de memoria RAM, 4 procesadores Intel(R) Core™ i7-HQ CPU @ 2.20GHz 2.20 GHz y la otra con sistema operativo Windows 10 de 64 bits, 16 GB de memoria RAM, 4 procesadores Intel(R) Core™ i7-4702HQ CPU @ 2.80GHz 2.81 GHz y GPU Nvidia GeForce GTX 1050 con 640 Núcleos CUDA.

También se utilizan los clusteres de CPU y GPU de la Facultad de Ingeniería (FING) [5] para la realización de algunas pruebas particulares que demoran en su ejecución y el PC con GPU que pertenece al grupo de PLN de la FING.

[5] https://www.fing.edu.uy/cluster/index.php

5. Experimentación

En este capítulo se presentan las distintas pruebas realizadas a lo largo del proyecto y los resultados más relevantes que se obtienen. También se explican las métricas de performance tomadas en cuenta para los resultados.

Las pruebas se dividen en dos partes debido a que se tienen dos variantes del *corpus* construido dependiendo del algoritmo que se utiliza para la clasificación: por un lado, se realizan las que corresponden a las RNN LSTM con el *corpus* que contiene los vectores de palabras y emoticones, y por otro, las que corresponden al resto de los algoritmos con el *corpus* que contiene todos los atributos declarados en el capítulo 3.

Las pruebas se ejecutan en distintas etapas y de forma incremental al igual que en la construcción del *corpus*. Se comienza considerando cada uno de los atributos elegidos por separado, luego se van incorporando y combinando con otros atributos hasta llegar a la realización de pruebas con todos los atributos en conjunto. Además, a lo largo de las pruebas, se van aumentando la cantidad de usuarios que se considera para los conjuntos de entrenamiento y prueba, llegando a las pruebas finales con 14000 usuarios para entrenar y 240 para evaluar.

Se establece como línea base un 50 % para todas las etapas de experimentación; este valor equivale a elegir al azar el género de un usuario.

5.1. Métricas de Performance de los Algoritmos

Una vez que se tienen los modelos entrenados, nos interesa medir su performance, que tan capaces son de realizar una buena predicción. Para cumplir con ello, utilizamos medidas estándar como lo son el acierto, precisión, recuperación, medida f y matriz de confusión que a continuación se detallan.

Además, se presenta la técnica de validación cruzada que nos permite evaluar los resultados y tratar de garantizar que son independientes de la partición entre datos de entrenamiento y prueba que se está tomando.

5.1.1. Acierto

La medida de acierto en términos estadísticos, está relacionada con el sesgo de una estimación. Cuanto menor es el sesgo, más exacta es una estimación. Cuando se expresa la medida de acierto de un resultado, se expresa mediante el error absoluto que es la diferencia entre el valor experimental y el valor verdadero. Nos dice que tan exacta es la estimación que se está realizando, la proximidad entre el resultado que nos retorna el clasificador y la clasificación real o exacta, se mide con la siguiente relación:

$$Acierto = \frac{VerdaderosPositivos + VerdaderosNegativos}{MuestraTotal}$$

5.1.2. Precisión

La precisión es la calidad de respuesta del clasificador. Se expresa mediante la siguiente relación:

$$Precisión = \frac{VerdaderosPositivos}{VerdaderosPositivos + FalsosPositivos}$$

5.1.3. Recuperación

Es la habilidad del clasificador de encontrar todas las muestras positivas. En otras palabras, expresa cuántos de los verdaderos positivos fueron encontrados, comparado con el total de los positivos. Se expresa con la siguiente relación:

$$Recuperación = \frac{VerdaderosPositivos}{VerdaderosPositivos + FalsosNegativos}$$

5.1.4. Medida F

Se emplea en la determinación de un valor único ponderado de la precisión y la recuperación. La fórmula general para el cálculo de esta medida es:

$$F_\beta = (1 + \beta^2)\frac{Precisión.Recuperación}{\beta^2.Precisión + Recuperación}$$

Siendo β un número real positivo, si $\beta > 1$ se le da más importancia al *recuperación*, si $\beta < 1$ se le da más importancia a la *precisión*, si es 1 tienen la misma ponderación.

5.1.5. Matriz de Confusión

Es una matriz que representa en cada columna las predicciones de cada clase y en cada fila la instancia de cada clase. En el Cuadro 7 se presenta un ejemplo acorde a nuestro proyecto en el caso de la clasificación de usuarios como hombre y mujer.

	Predicción Mujer	Predicción Hombre
Mujer	Verdaderos Positivos	Falsos Negativos
Hombre	Falsos Positivos	Verdaderos Negativos

Cuadro 7: Matriz de Confusión para 2 clases (Hombre y Mujer)

5.1.6. Validación Cruzada

Resulta necesario validar la estabilidad de los modelo de aprendizaje automático, para asegurase que no hay sobreajuste (*overfitting*) de los datos de entrenamiento, y esto sucede

cuando un modelo se ajusta demasiado a las características del *corpus* en el que fue entrenado y no le permite generalizar para luego predecir el resultado ante nuevos ejemplos.

En la validación cruzada K, los datos se dividen en k subconjuntos. El método se repite k veces de forma que, cada vez, uno de los k subconjuntos se utiliza como el conjunto de prueba y los otros subconjuntos k-1 se juntan para formar un conjunto de entrenamiento. La estimación del error se promedia en todos los k ensayos para obtener la efectividad total del modelo.

Cada conjunto de datos llega a estar en un conjunto de validación una sola vez, y k-1 veces en un conjunto de entrenamiento. Esto reduce significativamente el sesgo ya que se utilizan la mayoría de los datos para el ajuste, y también reduce significativamente la varianza ya que la mayoría de los datos también se utilizan en el conjunto de validación. El intercambio de los conjuntos de entrenamiento y prueba también se suma a la efectividad de este método. Como regla general y evidencia empírica, generalmente se prefiere un valor para K de 5 o 10. [13]

5.2. Experimentos

En esta sección se describen los experimentos para los clasificadores SVM con kernel Lineal, Naïve Bayes, Árbol de Decisión, Red Neuronal, Clustering y Bosques Aleatorios. Durante la explicación de las distintas etapas de experimentos se nombran como *SVM*, *NB*, *DT*, *RNA*, *CL* y *RF* respectivamente. Los resultados obtenidos para los distintos algoritmos en cada etapa se muestran mediante cuadros ilustrativos.

En todos los casos, para la obtención de los conjuntos de entrenamiento y prueba, se divide el conjunto de usuarios conformándose el conjunto de entrenamiento con el 75 % de ellos y el conjunto de prueba con el 25 % restante.

Etapa 1

Para realizar las primeras pruebas se decide tomar en cuenta subconjuntos desde 500 a 2000 usuarios del total de los 14000 que se tienen para el entrenamiento de los distintos algoritmos; eligiendo siempre la mitad de ellos hombres y la otra mitad mujeres. Estos subconjuntos se dividen a su vez en un conjunto de entrenamiento y en otro de prueba.

Se realizan pruebas con los *tweets* simples de los usuarios por un lado y con los documentos que contienen los *tweets* concatenados por otro, sin modificar las *urls* que hay en ellos.
Se observa que el tiempo de procesamiento tanto para el entrenamiento como para la prueba es mucho menor cuando se utilizan los *tweets* concatenados en vez de los *tweets* simples, hablamos del orden de la mitad del tiempo o menos en el caso de los concatenados. Por otra parte, en cuanto a la performance de los clasificadores, es similar en ambas pruebas en varios de los casos y aun en aquellos en que la diferencia de performance no es tan significativa, el tiempo de ejecución es mucho menor.

En esta etapa se ejecutan también pruebas para obtener los mejores parámetros para la configuración de cada uno de los clasificadores antes mencionados. Se utiliza un algoritmo de búsqueda exhaustiva sobre valores de parámetros especificados para un estimador, retornando

los mejores. No se logra terminar con todos los casos debido al largo tiempo de ejecución que lleva este algoritmo y su amplio consumo de procesador: son tiempos mayores a una semana sin que la ejecución del algoritmo finalice o en su defecto termine de forma abrupta e inconclusa.

En los Cuadros 8 y 9 se detallan algunos de los resultados para 1000 y 2000 usuarios. Se puede ver la performance de cada uno de los clasificadores para distintas cantidades de usuarios y atributos, así como el tiempo de ejecución de las pruebas.

Considerando únicamente el contenido de los *tweets*, para los algoritmos de clasificación *SVM*, *NB* y *RNA* se logra un acierto que ronda el 69 %, resultando como mejor valor un 69.39 % para SVM con *kernel* lineal. Al utilizar los *tweets* concatenados se obtiene un acierto de 69.6 % para RNA, el mayor valor conseguido, y es de destacar la disminución del tiempo de entrenamiento necesaria para todos los casos.

Clasificador	#Us.	#Atr.	Acierto	Tiempo
SVM	1000	95226	**0.6939**	0.933 s
	2000	159449	0.6927	2.915 s
NB	1000	95226	0.6900	0.048 s
	2000	159449	0.6911	0.097 s
DT	1000	95226	0.5755	1.179 m
	2000	159449	0.5669	4.644 m
RNA	1000	95226	0.6804	3.708 h
	2000	159449	0.6703	8.316 h
CL	1000	95226	0.5376	2.221 m
	2000	159449	0.5123	4.384 m

Cuadro 8: Pruebas Etapa 1 *tweets* simples

Clasificador	#Us.	#Atr.	Acierto	Tiempo
SVM	1000	103378	0.656	0.063 s
	2000	180957	0.666	0.228 s
NB	1000	103378	0.572	0.013 s
	2000	180957	0.612	0.028 s
DT	1000	103378	0.528	1.156 s
	2000	180957	0.550	3.063 s
RNA	1000	103378	0.648	3.107 m
	2000	180957	**0.696**	7.796 m
CL	1000	103378	0.512	27.695 s
	2000	180957	0.554	8.590 s

Cuadro 9: Pruebas Etapa 1 *tweets* concatenados

Etapa 2

En esta etapa, al igual que en la anterior, se toma un subconjunto de usuarios y se obtienen los conjuntos de entrenamiento y prueba.

A diferencia de la etapa anterior, en esta se hacen modificaciones en los *tweets*, transformando a todas las *urls* del formato de Twitter a su correspondiente *url* real y completa con el objetivo de verificar cuánto influyen las *urls* correspondientes a imágenes, videos y enlaces que un usuario comparte.

Se repiten las pruebas de la Etapa 1 considerando los *tweets* simples por un lado y los concatenados por otro con la modificación en las *urls*. En cuanto a los tiempos de procesamiento y resultados para ambas pruebas, se observa nuevamente que utilizando los *tweets* concatenados el procesamiento se realiza en un tiempo menor.
También se realizan pruebas utilizando únicamente la descripción de los usuarios.

En el Cuadro 10 se presentan las pruebas con los *tweets* simples y se advierte una mejora en las métricas de performance para los algoritmos de clasificación (con excepción de *CL*), se logra un 73 % de acierto para *SVM* con *kernel* lineal y *NB*, ambos con 1000 usuarios.

En el Cuadro 11 se exponen las pruebas con los *tweets* concatenados. No se encuentra una mejora respecto a resultados anteriores, pero se sigue notando una gran diferencia en los tiempos de entrenamiento en relación a las pruebas con los *tweets* simples. Se logra un 66.2 % de *Acurracy* para el algoritmo *RNA*.

En el Cuadro 12 se detallan los resultados de considerar únicamente la descripción de los usuarios, si bien no se obtienen grandes resultados, se supera en su gran mayoría la línea base del 50 % y se logra un acierto del 58.6 % para RNA.

Clasificador	#Us.	#Atr.	Acierto	Tiempo
SVM	1000	98780	**0.7302**	0.898 s
	2000	166478	0.7224	3.007 s
NB	1000	98780	**0.7302**	0.042 s
	2000	166478	0.7196	0.092 s
DT	1000	98780	0.6066	1.144 m
	2000	166478	0.5733	4.629 s
RNA	1000	98780	0.7133	4.898 h
	2000	166478	0.7022	10.661 h
CL	1000	98780	0.5393	1.626 m
	2000	166478	0.4939	3.46 s

Cuadro 10: Pruebas Etapa 2 *tweets* simples con url completa

Clasificador	#Us.	#Atr.	Acierto	Tiempo
SVM	1000	109424	0.592	0.115 s
	2000	188200	0.644	0.298 s
NB	1000	109424	0.464	0.012 s
	2000	188200	0.612	0.021 s
DT	1000	109424	0.508	1.684 s
	2000	188200	0.528	4.262 s
RNA	1000	109424	0.616	3.705 s
	2000	188200	**0.662**	9.924 m
CL	1000	109424	0.452	38.491 s
	2000	188200	0.480	1.059 m

Cuadro 11: Pruebas Etapa 2 *tweets* concatenados con url completa

Clasificador	#Us.	#Atr.	Acierto	Tiempo
SVM	1000	2683	0.520	0.006 s
	2000	4885	0.566	0.009 s
NB	1000	2683	0.568	0.002 s
	2000	4885	0.568	0.003 s
DT	1000	2683	0.520	0.051 s
	2000	4885	0.534	0.149 s
RNA	1000	2683	0.552	9.776 s
	2000	4885	**0.586**	19.371 s
CL	1000	2683	0.472	0.282 s
	2000	4885	0.494	0.283 s

Cuadro 12: Pruebas Etapa 2 descripción de los usuarios

Etapa 3

En esta instancia de pruebas se toma en cuenta primero un subconjunto de los usuarios y luego a todos ellos. Se obtiene en cada uno de los casos los conjuntos de entrenamiento y prueba respectivos.

Las primeras pruebas se realizan únicamente considerando las fotos de perfil de los usuarios para tener una idea del valor de los atributos que nos proveen las imágenes. En el Cuadro 13 se observan valores parecidos a los obtenidos con la descripción del usuario, si bien no son grandes resultados, para el algoritmo *RNA* se logra un 58 % de acierto.

Las siguientes pruebas se realizan uniendo los atributos de la descripción y fotos de perfil a los *tweets* concatenados con las *urls* completas para un subconjunto de usuarios y para todos los usuarios. Respecto a pruebas anteriores que contienen menos atributos, se logran mejoras en los resultados. Se toman como positivas porque se considera que los nuevos atributos generan su

aporte y se decide continuar utilizando la combinación de todos ellos en las sucesivas pruebas. En el Cuadro 14 se detallan los resultados de los experimentos que se realizan con todos los atributos antes mencionados, el algoritmo que tiene una mejor performance es el *SVM* con *kernel* lineal, alcanzando un 70 % de acierto.

Además, se realizan pruebas con reducción de atributos, se establece un umbral mínimo y un umbral máximo en *tf-idf*. Estos umbrales indican la menor y mayor frecuencia que debe tener una palabra para ser considerada, descartándose todas aquellas que no se encuentran dentro del rango. En el Cuadro 15 se pueden observar los mejores resultados obtenidos. Al considerar como máximo una frecuencia de 0.6 se logra reducir la cantidad de atributos, reducir el tiempo de entrenamiento y se mantienen muy similares las métricas de performance de los algoritmos *RNA* y *SVM* logrando en éste último el mejor valor con un 70 % de acierto.

Luego se varían los parámetros de los distintos clasificadores para tratar de encontrar una combinación que mejore los resultados obtenidos hasta el momento. Las pruebas se realizan en todos los casos para los 14000 usuarios. En el Cuadro 16 se muestran los resultados. Se observa que el algoritmo que tiene la mejor performance es *RNA*, en el cual se logra un 71 % de acierto.

En el Cuadro 17 se muestran los resultados obtenidos luego de ejecutar la técnica de validación cruzada de 10 iteraciones para los algoritmos de clasificación. Se utiliza en estas pruebas la mejor configuración para cada clasificador. Se presenta el promedio de los 10 valores obtenidos de acierto. Los algoritmos que presentan mejor performance en este caso son *SVM* con *kernel* lineal y *RNA*.

Clasificador	#Us.	#Atr.	Acierto	Tiempo
SVM	1000	970	0.528	0.014 s
	2000	970	0.490	0.027 s
NB	1000	970	0.520	0.005 s
	2000	970	0.482	0.030 s
DT	1000	970	0.544	0.043 s
	2000	970	0.480	0.113 s
RNA	1000	970	**0.580**	5.453 s
	2000	970	0.484	9.605 s
CL	1000	970	0.512	0.490 s
	2000	970	0.518	0.940 s

Cuadro 13: Pruebas Etapa 3 foto de perfil de los usuarios

Clasificador	#Us.	#Atr.	Acierto	Tiempo
SVM	1000	95712	0.564	0.279 s
	2000	161182	0.688	0.680 s
	14000	721900	**0.703**	10.599 s
NB	1000	95712	0.444	0.018 s
	2000	161182	0.618	0.039 s
	14000	721900	0.607	0.296 s
DT	1000	95712	0.532	1.226 s
	2000	161182	0.552	3.226 s
	14000	721900	0.582	1.623 m
RNA	1000	95712	0.640	2.949 m
	2000	161182	0.670	7.605 m
	14000	721900	0.684	3.023 h
CL	1000	95712	0.512	22.682 s
	2000	161182	0.510	30.608 s
	14000	721900	0.496	4.557 m

Cuadro 14: Pruebas Etapa 3 foto de perfil, *tweets* concatenados con *url* completa y descripción de los usuarios

Clasificador	min	max	#Atr.	Acierto	Tiempo
SVM	-	0.6	721858	**0.7005**	8.697 s
NB	0.01	-	7351	0.6557	0.834 s
DT	0.05	0.5	2353	0.5805	12.989 s
RNA	0.04	0.5	2617	0.6825	2.990 h
CL	0.05	-	2353	0.5042	1.670 m

Cuadro 15: Pruebas Etapa 3 reducción de atributos

Parámetros	Acierto	Tiempo
SVM	0.709	1.347 s
NB	0.670	0.336 s
DT	0.640	16.358 s
RNA	**0.714**	1.419 s
CL	0.504	1.667 s

Cuadro 16: Pruebas Etapa 3 mejor configuración 14000 usuarios

Clasificador	Promedio Acierto
SVM	**0.70**
NB	0.66
DT	0.62
RNA	**0.70**
CL	0.49

Cuadro 17: Pruebas Etapa 3 Validación Cruzada

Etapa 4

En la etapa final de pruebas, se agregan a los *tweets* concatenados, descripción y foto de perfil de los usuarios, los atributos proporcionados por los emoticones. Aquí se realiza la unión de todos los atributos disponibles y se consideran los 14000 usuarios para el conjunto de entrenamiento y los 240 usuarios relevados manualmente y reservados para el final de las pruebas como el conjunto de testeo.

Se incorpora un nuevo algoritmo de clasificación supervisado: *RF* correspondiente a Bosques Aleatorios para estudiar su performance y compararlo con los resultados obtenidos hasta el momento para árboles de decisión.

En el Cuadro 18 se detallan los resultados. Para cada clasificador se ejecutan pruebas con la mejor configuración obtenida y se muestran los valores de las métricas de performance discriminados por género, siendo H los correspondientes a los hombres y M a las mujeres.

Agregar los atributos de los emoticones representa una mejora respecto a evaluaciones anteriores, alcanzando buenos resultados con un valor de acierto del 76.6 % para el algoritmo *RNA*. Se observa que todos los algoritmos predicen mejor a los usuarios de género masculino y en particular, el algoritmo *RNA* logra un 92 % y un 60 % en Precisión para los hombres y mujeres respectivamente, también presenta los mejores valores para Medida F1 en ambos casos con un 80 % y 71 %. En la Figura 28 se presenta la matriz de confusión para este algoritmo.

Clasificador	Acierto	Precisión		Recuperación		Medida F1	
		H	M	H	M	H	M
SVM	0.754	0.880	0.594	0.733	0.797	0.800	0.681
NB	0.729	0.828	0.574	0.751	0.683	0.788	0.624
DT	0.708	0.793	0.552	0.763	0.594	0.778	0.573
RNA	**0.766**	**0.920**	**0.600**	0.714	0.873	**0.804**	**0.711**
CL	0.687	0.762	0.526	**0.776**	0.506	0.769	0.516
RF	0.666	0.909	0.496	0.559	**0.886**	0.692	0.636

Cuadro 18: Pruebas Etapa 4 foto de perfil, *tweets* concatenados con *url* completa, descripción y emoticones de los usuarios

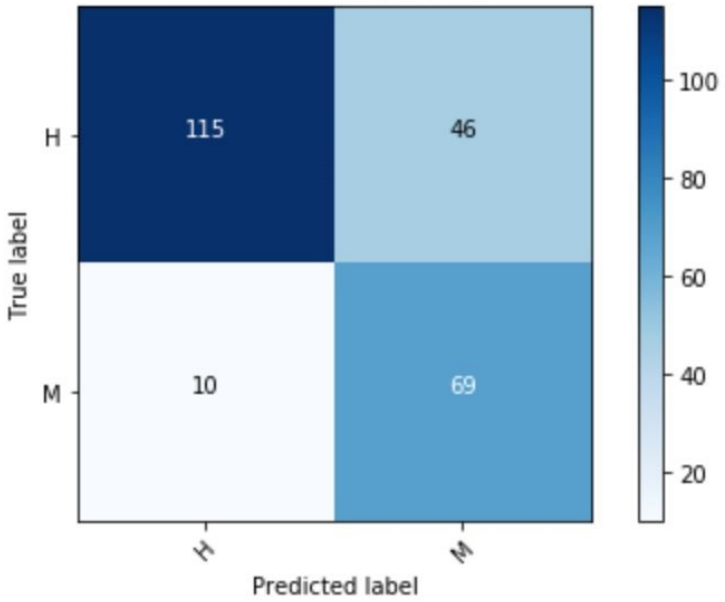

Figura 28: Matriz de Confusión para el clasificador RNA

5.3. Experimentos RNN LSTM

Aquí se describen las pruebas realizadas para las redes neuronales recurrentes LSTM. Los resultados obtenidos en el proceso de los experimentos se muestran mediante cuadros ilustrativos. Para elegir el conjunto de entrenamiento y prueba se implementa una función f que dado un conjunto de usuarios elige aleatoriamente un subconjunto de ellos. Esto se debe a que la cantidad de ejemplos iniciales es muy grande y los entrenamientos de las redes LSTM tienen alta duración en el tiempo. Luego se considera el 75 % para el conjunto de entrenamiento y el 25 % restante para el conjunto de prueba.

Etapa 1

En la primer etapa de pruebas con LSTM, solo se toman en cuenta los *tweets* con la *url* real y completa. Se varía la cantidad de usuarios, la cantidad de ejemplos a considerar y también se aplican distintos modelos de redes LSTM, modificando en cada uno de ellos distintos parámetros que cambian la estructura de la red.

Inicialmente se consideran los 14.000 usuarios con los que contamos en la base de datos. Luego de hacer algunas pruebas en los distintos ambientes de trabajo con ejecuciones de una semana de duración y algunas de hasta incluso más tiempo, como no se logran mejores resultados se descarta esta opción y se comienza a disminuir la cantidad de usuarios y ejemplos. La idea es que se puedan realizar ejecuciones de modelos que finalicen el entrenamiento en un tiempo menor y que mejoren los resultados. Se va variando así la cantidad de usuarios y ejemplos logrando

ejecuciones de no más de 24 horas y manteniendo la performance del algoritmo.

Otra configuración importante que se debe tener en cuenta es la cantidad de *epochs* que se utilizan en cada modelo para estas redes debido a que influye y mucho en los tiempos de ejecución. La cantidad de *epochs* refiere a cuántas veces el conjunto de entrenamiento recorre la arquitectura completa de la red, es decir, la cantidad de veces son usados los datos para ajustar los pesos. Si el conjunto de entrenamiento se divide en *batchs*, por cada uno de los *batchs*, se recorre la cantidad de *epochs* especificada. Se comienza considerando pocos *epochs* (entre 2 y 10) y se va aumentando la cantidad hasta visualizar que la performance de entrenamiento de la red neuronal no aumenta, mirando particularmente las medidas de acierto y los valores de la función de pérdida (*loss*). Una vez que la red neuronal ya *no aprende* en el entrenamiento, no es necesario agregar más *epochs*, eso solo insume tiempo y no realiza ningún aporte. En cada prueba que se realiza, se aumenta la cantidad de *epochs* siempre y cuando siga aumentando el acierto de entrenamiento de la red, una vez que la red ya no aprende más, se detiene y no se varía más este parámetro.

Se divide la prueba en dos arquitecturas, *Many to one* y *Many to Many* con *Many to One*, para determinar con cuál se obtiene un mejor resultado y mejor tiempo de procesamiento.

Primero se realizan distintas pruebas con la arquitectura de la LSTM *Many to One*. La diferencia en cada una de las pruebas es la cantidad de ejemplos para definir cuál es el mejor valor a utilizar sin afectar el acierto y tiempo de entrenamiento.

Observando el Cuadro 19 se encuentra que todos los valores de acierto son similares; lo único que varía notoriamente es el tiempo que demora en ejecutar la red. Por este motivo se decide seguir con 280 usuarios para poder realizar la mayor cantidad de pruebas posibles.
Luego se varía la cantidad de vectores que compone a una ventana. La única forma que la ventana contenga vectores nulos es que su tamaño sea mayor a la cantidad de palabras correspondientes al usuario. Como se puede observar en el Cuadro 20 el mejor tamaño de ventana es 12, coincidiendo con el promedio de palabras por *tweet*.

Teniendo la cantidad de usuarios, la ventana de contexto a considerar en la red LSTM y el tiempo que tarda en correr 5 *epochs*, lo siguiente es variar los parámetros de la red y aumentar la cantidad de *epochs*. En el Cuadro 21 se presentan las pruebas realizadas al variar un conjunto de parámetros, entre ellos, la función de activación y *dropout*. Las distintas combinaciones de estos parámetros se denominan P1...P7, los cuales se encuentran detallados en el Cuadro 28 del anexo. De aquí en más se utiliza esta nomenclatura para la combinación de los parámetros.
Las mejores métricas de performance se obtienen en la prueba con el conjunto de parámetros P3 considerando una función de activación *sigmoidal*, los estados de los *batchs* independientes, sin *dropout*, función de pérdida *entropía cruzada binaria* y optimizador Adam.

En el Cuadro 22 se puede ver que se logra una performance del algoritmo en el entorno del 62 % al considerar 50, 100 o 150 *epochs*, alcanzando el mejor resultado en acierto para 50.

#Us.	#Ejemplos	Acierto	Tiempo
14000	268708	**0.4997**	3.26 h
7000	132719	0.4857	49.36 m
1400	27489	0.4906	11.56 m
700	14732	0.4894	6.22 m
280	5697	0.4784	4.58 m

Cuadro 19: Pruebas Etapa 1 *LSTM Many to One* cantidad de usuarios

#Ventana	#Ejemplos	Acierto	Tiempo
2	6518	0.4718	55 s
4	6265	0.4774	68 s
6	6032	0.4653	82 s
9	5967	0.4812	108 s
12	5384	**0.5191**	119 s
16	5003	0.4726	141 s
20	4660	0.4843	151 s
22	4499	0.4949	159 s

Cuadro 20: Pruebas Etapa 1 *LSTM Many to One* desplazamiento

Parámetros	Acierto
P1	0.5716
P2	0.5638
P3	**0.5935**
P4	0.5729
P5	0.5496
P6	0.5793
P7	0.5896

Cuadro 21: Pruebas Etapa 1 *LSTM Many to One* variación de parámetros

#Epochs	Acierto
50	**0.6206**
100	0.6193
150	0.6116

Cuadro 22: Pruebas Etapa 1 *LSTM Many to One* Aumento de #Epochs

Se realiza un estudio similar con la arquitectura *Many to Many* y *Many to One* combinadas, pero como no se obtienen mejores resultados se descartan y en las siguientes etapas se continua

mejorando la arquitectura de *Many to One*. En el anexo, en el Cuadro 27 se pueden observar algunas de las pruebas que maximizan el acierto para este caso.

Etapa 2

En esta etapa se consideran únicamente los emoticones que utiliza cada usuario. Se construye un modelo con LSTM y se toman los conjuntos de usuarios para entrenamiento y prueba de la misma manera que en la etapa anterior.

Los emoticones agregan valor: tomando en cuenta solo estos atributos se logran buenos resultados, muchos de ellos incluso mejores que los de modelos LSTM de vectores de palabras para los *tweets*.

En el Cuadro 23 se muestran los resultados. Se observa un 60 % como el mejor valor de acierto para el conjunto de parámetros P1 que se corresponde a una LSTM *Many To One* que considera como función de activación *tangente hiperbólica*, sin *dropout*, función de pérdida *entropía cruzada binaria* y optimizador Adam.

Parámetros	Acierto
P1	**0.6050**
P2	0.5882
P3	0.5798
P4	0.5798
P5	0.5966
P6	0.5798
P7	0.5714

Cuadro 23: Pruebas Etapa 2 *LSTM Many to One* de emoticones y variación de parámetros

Etapa 3

Esta última etapa de pruebas para las redes neuronales recurrentes LSTM, une los modelos implementados y probados en las etapas anteriores. Para ello, se construye un nuevo modelo que toma como entradas los vectores correspondientes a las palabras y a los emoticones de los respectivos modelos antes mencionados.

En el Cuadro 24 se presentan los mejores resultados. Se observa que hay una mejora al considerar este modelo respecto de los modelos individuales tanto de vectores de palabras como de vectores de emoticones, logrando un 65.55 % de acierto.

Finalmente, se toman los 240 usuarios reservados para las pruebas finales y se evalúa este modelo con ellos, obteniendo resultados similares al resto de los logrados en el proceso de pruebas. En el Cuadro 25 se muestran los resultados con los usuarios reservados. Se alcanza un acierto igual al 65.45 % para la combinación de parámetros P1 y P7.

Parámetros Emoticones	Parámetros Palabras	Acierto
P1	P7	64.71 %
P7	P7	57.14 %
P7	P3	**65.55 %**

Cuadro 24: Pruebas Etapa 3 *LSTM Many to One* de emoticones y Palabras

Parámetros Emoticones	Parámetros Palabras	Acierto
P1	P7	61.82 %
P7	P7	63.64 %
P1	P3	**65.45 %**

Cuadro 25: Pruebas Etapa 3 *LSTM Many to One* de emoticones y Palabras con usuarios reservados

5.4. Comparaciones con Otros Proyectos

En esta sección se comparan los resultados más relevantes obtenidos con los que se presentan en el artículo *Classifying Latent User Attributes in Twitter* [20] que estudian el mismo problema. Como se expone en el Capítulo 2, en ese trabajo se cuenta con 500 usuarios por género, 405.151 *tweets* y la dimensión de su modelo de atributos de ngramas es de 1.256.558. Este es de los proyectos relacionados, uno de los más semejantes. Descartan los nombres de usuario o *screen name* y se toman como atributos principales los provenientes del contenido del *tweet*. Los resultados que obtienen utilizando sus diferentes modelos de atributos y el algoritmo *Support Vector Machine* , presentan los valores de acierto que se detallan en el Cuadro 26.

Modelo	Acierto
Baseline Model (Prior)	50.00 %
Socolinguistic Feature Model	71.76 %
Ngram-based Feature Model	68.70 %
Stacked Model	**72.33 %**

Cuadro 26: Resultados proyectos similares

Como se puede ver, en su modelo *Stacked* que es una conjunción de los modelos sociolingüísticos y de ngramas que combina todos los atributos extraídos del texto de los *tweets*, se logra un valor de acierto igual al 72.33 %. Existen otros trabajos [18, 23] que obtienen mejores resultados, pero tienen la particularidad de considerar en su conjunto de datos alguno de los atributos que utilizan para etiquetar manualmente los usuarios, o disponen explícitamente de la declaración del género del usuario.

Los resultados del Cuadro 26 son cercanos a los de nuestro proyecto, pero se encuentran 3 y 4 puntos porcentuales por debajo de los mejores valores obtenidos para los clasificadores SVM

con *kernel* lineal y RNAs con un acierto del 76.6 % y 75.4 % respectivamente.

Si bien en *Classifying Latent User Attributes in Twitter* [20] tienen como positivo que los usuarios son etiquetados manualmente y que al considerar una menor cantidad de usuarios se logran buenos resultados, se entiende que son de destacar los resultados que se obtienen en el presente proyecto, en el cual se utiliza una variedad importante de atributos de los usuarios de Twitter y el hecho de contar con un *corpus* que contiene datos de 14.240 usuarios hace que los algoritmos de clasificación se puedan entrenar con una diversidad mayor de ejemplos. Además, se pueden seguir ajustando variables como el etiquetado inicial por género con otras técnicas para mejorarlos aún más a futuro.

6. Conclusiones y Trabajos Futuros

Este proyecto tiene por objetivo la construcción de un modelo para la predicción del género de usuarios de Twitter para el idioma español.

Se inicia el proceso con la creación de un *corpus* que contiene información de usuarios de Twitter etiquetados por género y con textos escritos en español. Este corpus es una pieza fundamental para lograr el objetivo planteado y, al ser inexistente hasta el momento, se considera un gran aporte también para trabajos futuros.

Se obtiene información de 14.240 usuarios de Twitter, de los cuales, 240 se relevan y etiquetan de forma manual y se componen de usuarios conocidos o famosos para los cuales se conoce su género real. Los 14.000 usuarios restantes se obtienen luego del procesamiento de un conjunto de usuarios etiquetados por género que son proporcionados por la empresa IDATHA.

Una vez que se determinan los usuarios que forman parte del *corpus*, se procede a definir cuáles de sus atributos se toman en cuenta. Se estudian trabajos relacionados para el idioma inglés, que se centran básicamente en el contenido de los *tweets* y nombre o *screen name* de los usuarios. En nuestro proyecto se descartan los nombres de los usuarios debido a que hay una correlación entre ellos y la forma en la que se anota inicialmente su género en el *corpus* de trabajo.

En cambio, sí se considera el contenido de los *tweets* incluyendo el texto, las *urls*, palabras de los *hashtag* y emoticones, la descripción del usuario y los atributos correspondientes a la foto de perfil. Resulta interesante porque estos últimos son atributos que no se presentan en otros trabajos similares.

El *corpus* se construye en etapas, de forma iterativa e incremental, en la cual se van incorporando nuevos atributos y variantes de los atributos hasta llegar a una versión final que consta de los últimos 200 *tweets* de cada usuario, un documento de texto conteniendo todos los *tweets*, etiquetas y categorías correspondientes a su foto de perfil, descripción del usuario y emoticones que utiliza.

Para este trabajo se evalúa una variedad de algoritmos de aprendizaje supervisado y no supervisado. Puntualmente, se utilizan árboles de decisión, bosques aleatorios, naïve bayes, *clustering*, *Support Vector Machines*, redes neuronales y redes neuronales recurrentes. Se realizan pruebas con todos ellos en etapas, con distintas variantes del *corpus* y se mide su performance, considerando como medida principal el acierto para poder hacer una comparación entre ellos. En el desarrollo de las pruebas y evaluaciones, los atributos que se consideran más determinantes surgen del contenido de los *tweets* de los usuarios y en particular se observa que considerar la *url* completa y real y los emoticones tienen un buen aporte e influye positivamente en la mayoría de algoritmos.

Al analizar los algoritmos de clasificación, los que presentan una mejor performance a lo largo de todas las pruebas realizadas son los correspondientes a SVM con *kernel* lineal y redes neuronales, logrando en las pruebas finales un acierto del 75.4 % y 76.6 % respectivamente.

Los valores alcanzados se encuentran por debajo de los que se presentan en algunos de los

trabajos estudiados; esto se debe principalmente a que en esos trabajos se etiquetan de forma manual todos los usuarios, y además, se consideran como parte del *corpus* los atributos que se utilizan para el etiquetado por género. De todas formas, los resultados que se obtienen se entiende que son positivos: se encuentran entre 15 y 26 puntos por encima de la línea base establecida del 50 % que equivale a dejar al azar la determinación del género de los usuarios y superan los resultados de algunos trabajos relacionados con características similares como es el caso de los trabajos de *Deep Rao y David Yarowsky* [19] y *Deep Rao, David Yarowsky, Abhishek Shreevats y Manaswi Gupta* [20]. El proyecto está disponible en GitHub [6] donde se encuentran *notebooks* de Python que se pueden utilizar como referencia del desarrollo del trabajo.

Lo que respecta al trabajo a futuro, es posible mejorar el *corpus* construido ampliando los atributos que se consideran:
(a) Aumentar la cantidad de *tweets* por usuario consultando al servicio de Twitter periódicamente y actualizando la base de datos. De esta manera es posible evaluar la forma de escribir de cada uno de ellos considerando una mayor cantidad de ejemplos.
(b) Realizar un tratamiento distinto de las imágenes, videos y *links* tomando en cuenta los contenidos y no únicamente las *urls* que los representan. Analizar los contenidos permite identificar si los usuarios comparten imágenes, videos o *links* de temáticas similares.
(c) Obtener además de las fotos de perfil, las fotos de portada considerando una lista de imágenes en vez de una única imagen. Se logra así tener una historia de las fotos que selecciona el usuario para identificarse.
(d) Procesar la información de las redes sociales asociadas a la cuenta del usuario y los usuarios a los que sigue para tratar de identificar patrones.

Para el caso de los vectores utilizados en las redes recurrentes LSTM, una posibilidad de mejora es reentrenarlos para tomar en cuenta nuevas palabras e incluir en este caso a los emoticones, todo en un mismo contexto. Además, para considerar los atributos de las fotos de perfil de los usuarios, se sugiere generar vectores que tomen en cuenta las etiquetas y las categorías de la imagen. De esta forma, se puede construir un modelo que contenga todo el contenido del *tweet* (palabras y emoticones en un mismo contexto) y unirlo con otro modelo que tenga como entradas los vectores que representan a los atributos que proveen las imágenes.

[6] https://github.com/ProyectoGradoGen/pGen

7. Glosario

CORPUS

Conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación. En el presente proyecto se trata del conjunto de datos que contiene la información de los usuarios de Twitter.

EPOCH

Refiere a la cantidad de veces que el conjunto de entrenamiento recorre la arquitectura completa de la red, es decir, la cantidad de veces son usados los datos para ajustar los pesos.

HASHTAG

En Español se denomina etiqueta. Es una cadena de caracteres formada por una o varias palabras concatenadas y numeral (#).

LSTM

Es un tipo de red neuronal recurrente que tiene la capacidad de aprender dependencias a largo plazo. LSTM es un acrónimo de su nombre en inglés, Long short-term memory (Memoria a largo plazo).

JSON

Es un acrónimo de JavaScript Object Notation. Es un formato de texto ligero para el intercambio de datos.

MONGODB

MongoDB es una base de datos *open-source* no relacional y orientada a documentos.

RETWEET

Es la acción que permite reenviar un tweet de otra persona.

RNA

RNA es un acrónimo de Red Neuronal Artificial. Las redes neuronales artificiales son un modelo de aprendizaje automático inspirado por la observación de los sistemas de aprendizaje biológico que están construidos a partir de redes muy complejas de neuronas interconectadas. Proporcionan un enfoque robusto para aproximar funciones de valores reales, discretos y vectoriales.

RNN

RNN es un acrónimo de Red Neuronal Recurrente. Las redes neuronales recurrentes pertenecen a la familia de redes neuronales que son utilizadas para procesar datos secuenciales.

SOBREAJUSTE

En aprendizaje automático el sobreajuste es el efecto que tiene un modelo cuando se ajusta demasiado a las características del *corpus* en el que fue entrenado.

SVM

Support Vector Machines (SVM) es un modelo de aprendizaje automático supervisado que pertenece a la categoría de los clasificadores lineales.

TWITTER

Es un servicio de microblogueo que permite a sus usuarios enviar y publicar mensajes breves.

TWITTEAR

Acción de enviar un *tweet*. Los *tweets* se muestran en las cronologías de Twitter o se insertan en sitios web y blogs.

TWEET

Mensaje intercambiado entre usuarios de Twitter.

URL

Es un acrónimo de Uniform Resource Locator. Es un identificador de recursos uniforme, cuyos recursos referidos pueden cambiar, por ejemplo, la dirección puede apuntar a recursos variables en el tiempo.

Referencias

[1] Microsoft Azure. Computer Vision API. https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/. [Online]. Accedido Marzo 2018.

[2] Andrej Karpathy blog. The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. [Online]. Accedido Marzo 2018.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[4] John D. Burger, John Henderson, George Kim, and Guido Zarrella. Discriminating gender on twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1301–1309, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[5] Universidad Antonio de Nebrija Constantino Malagón Luque. Clasicadores bayesianos. El algoritmo Naïve Bayes. http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf. [Online]. Accedido Febrero 2017.

[6] Observatorio Tecnológico de España. Redes Sociales. http://recursostic.educacion.es/observatorio/web/es/%20component/content/article/1043-redes-sociales?start=1. [Online]. Accedido Noviembre 2016.

[7] Universidad Carlos III de Madrid. Introducción a las redes neuronales aplicadas. http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf. [Online]. Accedido Febrero 2017.

[8] ETS de Ingeniería Informática Universidad Nacional de Educación a Distancia (UNED) Enrique J. Carmona Suárez, Dpto. de Inteligencia Artificial. Tutorial sobre Máquinas de Vectores Soporte,. http://www.ia.uned.es/~ejcarmona/publicaciones/[2013-Carmona]%20SVM.pdf. [Online]. Accedido Febrero 2017.

[9] Miniesterio de Educación ciencia y tecnología Gobierno Argentino. Redes Sociales. http://escritoriofamilias.educ.ar/datos/redes-sociales.html. [Online]. Accedido Febrero 2017.

[10] Mykola Harvat. Universo Machine Learning. https://conocemachinelearning.wordpress.com/2017/07/24/por-que-funcionan-tan-bien-las-lstm-en-nlp-frente-a-las-re [Online]. Accedido Marzo 2018.

[11] Keras. The Python Deep Learning library. https://keras.io/. [Online]. Accedido Marzo 2018.

[12] William Koehrsen. Random Forest Simple Explanation. https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d. [Online]. Accedido Marzo 2018.

[13] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[14] Jeremy B. Merrill. Gender-detector 0.0.4. `https://pypi.org/project/gender-detector/0.0.4/`. [Online]. Accedido Marzo 2018.

[15] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[16] Rashmi Mohan, Marco Pennacchiotti, and Ana maria Popescu. A machine-learning approach to twitter user classification. In *Proceedings of ICWSM*, 2011.

[17] Christopher Olah. Understanding LSTM Networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. [Online]. Accedido Marzo 2018.

[18] Claudia Peersman, Walter Daelemans, and Leona Van Vaerenbergh. Predicting age and gender in online social networks. In *Proceedings of the 3rd International Workshop on Search and Mining User-generated Contents*, SMUC '11, pages 37–44, New York, NY, USA, 2011. ACM.

[19] Delip Rao and David Yarowsky. Detecting latent user properties in social media. In *In Proc. of the NIPS MLSN Workshop*, 2010.

[20] Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. Classifying latent user attributes in twitter. In *Proceedings of the 2Nd International Workshop on Search and Mining User-generated Contents*, SMUC '10, pages 37–44, New York, NY, USA, 2010. ACM.

[21] Tim Rocktäschel. emoji2vec. `https://github.com/uclmr/emoji2vec`. [Online]. Accedido Marzo 2018.

[22] George Seif. The 5 Clustering Algorithms Data Scientists Need to Know. `https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68`. [Online]. Accedido Marzo 2018.

[23] Regina y Saskatchewan Thomas Oshiobughie Ugheoke. Detecting the Gender of a Tweet Sender. `http://www2.cs.uregina.ca/~hilder/my_students_theses_and_project_reports/ugheokeMScProjectReport.pdf`. [Online]. Accedido Febrero 2017.

[24] Andrea Trevino. Introduction to K-means Clustering. `https://www.datascience.com/blog/k-means-clustering`. [Online]. Accedido Marzo 2018.

[25] Twitter. Acerca del servicio de enlace de Twitter. `https://help.twitter.com/es/using-twitter/url-shortener`. [Online]. Accedido Marzo 2018.

[26] Twitter. Pagina oficial de Twitter. https://about.twitter.com/en_us/company.html. [Online]. Accedido Febrero 2017.

[27] Twitter. Twitter Developer Documentation. https://developer.twitter.com/rest/reference/get/statuses/show//%3Aid. [Online]. Accedido Febrero 2017.

[28] Grupo PLN Udelar. Grupo de Procesamiento de Lenguaje Natural,. https://www.fing.edu.uy/inco/grupos/pln/. [Online]. Accedido Marzo 2018.

Anexo

En este anexo se presenta la configuración de los parámetros para los algoritmos utilizados en la experimentación y los resultados de las pruebas descartadas para las RNN LSTM.

En el Cuadro 27 se detallan los mejores parámetros para las pruebas que maximizan la performance de la LSTM *Many To Many* y LSTM *Many To One* combinadas. Dado que no se logra superar el acierto obtenido para la LSTM *Many To One*, se descarta esta arquitectura.

En el Cuadro 28 se muestra la definición de los conjuntos de parámetros utilizados en las etapas de pruebas de las redes neuronales recurrentes LSTM *Many To One* y sus respectivos valores. Finalmente, en el Cuadro 29 se presentan las configuraciones de los parámetros que maximizan la performance del resto de los algoritmos.

Parámetros Many To Many	Parámetros Many To One	Acierto
dropout:0.0, recurrent_dropout:0.0, activation:'tanh', recurrent_activation: 'hard_sigmoid, use_bias:True, stateful = False, units=1	dropout:0.0, recurrent_dropout:0.0, activation:'tanh', recurrent_activation: 'hard_sigmoid, use_bias:True, stateful = False, units=1	**55.10 %**
dropout:0.0, recurrent_dropout:0.0, activation:'tanh', recurrent_activation: 'hard_sigmoid, use_bias:True, stateful = False, units=300	dropout:0.0, recurrent_dropout:0.0, activation:'tanh', recurrent_activation: 'hard_sigmoid, use_bias:True, stateful = False, units=1	42.90 %
dropout:0.0, recurrent_dropout:0.0, activation:'tanh', recurrent_activation: 'hard_sigmoid, use_bias:True, stateful = False, units=10	dropout:0.0, recurrent_dropout:0.0, activation:'tanh', recurrent_activation: 'hard_sigmoid, use_bias:True, stateful = False, units=1	54.95 %

Cuadro 27: Parámetros LSTM *Many To One* y LSTM *Many To Many*

Prueba	Parámetros
P1	*dropout:0.0, recurrent_dropout:0.0, activation:'tanh', recurrent_activation: 'hard_sigmoid, use_bias:True, stateful = True, units=1*
P2	*dropout:0.0, recurrent_dropout:0.0, activation='sigmoid', recurrent_activation:'hard_sigmoid', use_bias:True,stateful:False, units=1*
P3	*dropout:0.0, recurrent_dropout:0.0, activation='hard_sigmoid', recurrent_activation:'hard_sigmoid',use_bias:True, stateful:False, units=1*
P4	*dropout:0.0, recurrent_dropout:0.0, activation='hard_sigmoid', recurrent_activation: 'hard_sigmoid', use_bias= False, stateful:False*
P5	*dropout:0.0, recurrent_dropout:0.0, activation='sigmoid', recurrent_activation:'hard_sigmoid', use_bias:True,stateful:True, units=1*
P6	*dropout:0.0, recurrent_dropout:0.0, activation='hard_sigmoid', recurrent_activation:'hard_sigmoid',use_bias:True, stateful:True, units=1*
P7	*dropout=0.2, recurrent_dropout:0.0, activation='hard_sigmoid', recurrent_activation:'hard_sigmoid',use_bias:True, stateful:False, units=1*

Cuadro 28: Parámetros LSTM *Many To One*

Algoritmo	Parámetros
SVM	*C=0.85, class_weight=None, dual=True, fit_intercept=True, intercept_scaling=1, loss='hinge', max_iter=40, multi_class=óvr', penalty='l2', random_state=None, tol=0.0001, verbose=0*
NB	*alpha=0.03, class_prior=None, fit_prior=True*
DT	*class_weight=None, criterion=éntropy', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_split=1e07, min_samples_leaf=1, min_samples_split=1500, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best'*
RNA	*activation='relu', alpha=0.0001, batch_size=áuto', beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08, hidden_layer_sizes=(20, 20), learning_rate='constant', learning_rate_init=0.001, max_iter=200, momentum=0.9, nesterovs_momentum=True, power_t=0.5, random_state=None, shuffle=True, solver=ádam', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False*
CL	*algorithm=áuto', copy_x=True, init='k-means++', max_iter=300, n_clústeres=2, n_init=10, n_jobs=1, precompute_distances=áuto', random_state=0, tol=0.0001, verbose=0*
RF	*bootstrap=True, class_weight=None, criterion=éntropy', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_split=1e-07, min_samples_leaf=1, min_samples_split=1500, min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1, oob_score=True, random_state=None, verbose=0, warm_start=True*

Cuadro 29: Mejor configuración de parámetros algoritmos