UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA
INSTITUTO DE COMPUTACIÓN

Identificador de tendencias en Redes Sociales

Juan Benelli, Diego Alberti
Tutores: Juan José Prada, Mathías Etcheverry

Setiembre 2021, Montevideo, Uruguay

Agradecimientos

En primer lugar queremos agradecer a nuestros tutores, por el excelente clima de trabajo desde el comienzo del proyecto y por todas sus sugerencias y correcciones que tanto nos ayudaron. A nuestros amigos que se tomaron el tiempo de leer y hacer comentarios sobre versiones menos elaboradas de este informe, gracias por ayudarnos con su gran grano de arena. A todos nuestros compañeros y amigos que conocimos a lo largo de esta carrera, gracias por tantos momentos dedicados a cumplir esta meta. Y finalmente, muchas gracias a nuestras familias que tanto nos apoyaron a lo largo de toda la carrera, nada de esto hubiera sido posible sin ustedes.

Resumen

En la actualidad, millones de personas utilizan redes sociales para compartir mensajes, noticias y opiniones sobre diversos temas. Algunos de ellos se vuelven populares producto de una gran cantidad de mensajes publicados. En el caso específico de Twitter, los mensajes públicos llamados *tweets* generan tendencias, en inglés llamadas *Trending Topics*. En este trabajo, para cada tendencia se intenta encontrar el *tweet* que generó la mayor repercusión, candidato a ser el que disparó la popularidad de la tendencia. De forma automatizada, se conecta con Twitter para obtener las tendencias de Uruguay y sus *tweets* asociados, utilizando scripts implementados con el lenguaje Python.

Dado que la versión utilizada de la API de Twitter devuelve los *tweets* que nombran las tendencias de forma textual, tuvimos que dividir el problema en dos.

En primer lugar, dada la masa de *tweets* cuyo texto contiene la expresión que es tendencia, fue necesario construir un clasificador utilizando un modelo de Regresión Logística para descartar aquellos que no son relevantes de procesar ya que no refieren a la tendencia. Utilizando la técnica de *Bag of Words* y una medida de distancia relativa entre un *tweet* y una tendencia, calculada mediante la técnica *word embeddings*, se determina si un *tweet* es relevante o no.

En segundo lugar, utilizando los *tweets* que refieren a una tendencia, se intenta encontrar cuál de ellos es su "disparador", intentando contemplar tanto su repercusión en la red, como el hecho de haberse publicado cuando la tendencia aún no surgió. Para esto, se entrena una Red Neuronal que analiza características como la cantidad de interacciones del *tweet*, su tiempo de vida, la cantidad de *tweets* que se generan en la siguiente hora de publicado, entre otras.

Finalmente, a modo de poder visualizar la tarea realizada, se implementa utilizando *NodeJS* y *React* una aplicación web que permite seleccionar una tendencia previamente procesada, para ver cuál fue su disparador.

Los resultados experimentales muestran que el modelo de clasificación determina si el tema de un *tweet* es relevante para su tendencia asociada con un 79 % de *accuracy*, para un corpus de *tweets* construido durante el transcurso del proyecto. Por otra parte, el modelo de predicción del *tweet* disparador presenta un 81 % de *F1-score* bajo las mismas condiciones.

Comparando los resultados de cada modelo respecto a sus líneas bases utilizadas, en ambos casos se puede afirmar que la solución implementada resuelve de mejor manera los problemas enfrentados en este proyecto.

Palabras clave: Inteligencia Artificial, Aprendizaje Automático, Aprendizaje Supervisado, Procesamiento de Lenguaje Natural, Clasificación, Redes Sociales, Tendencia, Twitter.

Índice general

Capítulo 1

Introducción

En este capítulo se brinda una introducción sobre la temática tratada en este trabajo. En la sección 1.1 se presenta la motivación para este proyecto. En la sección 1.2 se exponen los principales objetivos que se plantean. El cronograma y las actividades realizadas se detallan en la sección 1.3. Por último, en la sección 1.4 se describe la organización de este informe.

1.1. Motivación

Las redes sociales son plataformas digitales utilizadas por millones de individuos en el mundo, quienes las usan para estar en contacto con sus pares, intercambiar información, expresar sus sentimientos, emitir opiniones, manifestar sus intereses o simplemente subir la foto de un paisaje.

Hoy en día se estima que alrededor del 60 % de los usuarios de internet usan redes sociales, lo que equivale a más de 3.800 millones de usuarios, quienes pasan en promedio 2 horas y 34 minutos por día navegando en redes, es decir, más de un tercio del tiempo que el usuario utiliza internet [29]. Dado este uso masivo, se puede afirmar que las redes sociales juegan un papel importante a la hora de reflejar los temas de interés de las personas en todo el mundo, por lo cual su contenido se convierte en información muy valiosa de analizar en diferentes ámbitos, como por ejemplo en campañas de marketing o políticas.

Estas redes se pueden agrupar en dos tipos, por un lado las redes sociales verticales como Flickr[1] o Linkedin[2], las cuales relacionan personas con intereses específicos en común, y por otro lado las redes sociales horizontales, las cuales no poseen una temática determinada sino que apuntan a todo tipo de usuarios. Estas últimas funcionan como medios de comunicación, información o entretenimiento [30] y son más relevantes a la hora de determinar los intereses de sus usuarios. Ejemplos de estas son Facebook[3], Instagram[4] o Twitter[5].

Esta última se ha transformado en los últimos años en una plataforma donde usuarios de todas partes del planeta comparten noticias, comentarios y opiniones de temas de diversa índole en tiempo real. Esto permite dar a conocer información relevante al instante, que puede cambiar la conducta de las personas como por ejemplo: la suspensión de un evento social o deportivo, vías cortadas por accidentes de tránsito, organización de manifestaciones, etc.

[1] https://www.flickr.com/
[2] https://www.linkedin.com/
[3] https://www.facebook.com/
[4] https://www.instagram.com/
[5] https://twitter.com/

El contenido generado por los usuarios de Twitter se publica en los llamados *tweets*. Los *tweets* son publicaciones que poseen un conjunto acotado de caracteres donde los usuarios expresan y comparten sus ideas sin un destinatario específico, es decir, que pueden ser vistos por el resto de los usuarios de la red. Su limitante de 280 caracteres por *tweet*, los obliga a contener información precisa y reducida, haciendo que sean fáciles de leer para el resto de los usuarios. Se estima que se generan 656 millones de *tweets* por día (7600 por segundo) [31], y que existen más de 330 millones de usuarios activos por mes [32].

Por otra parte, a diferencia de otras redes horizontales como Facebook o Instagram, Twitter introduce el concepto de *Trending Topic* (TT). Los TT son las palabras o expresiones más mencionadas por los usuarios de Twitter en una zona geográfica específica en tiempo real. Se podría decir que los TT hacen referencia a los temas de los que se está hablando en Twitter en este momento. Twitter se encarga de ordenar los TT según su popularidad, para que el usuario conozca el grado de relevancia de cada uno de ellos. Cada usuario puede personalizar las tendencias que le son mostradas, de acuerdo a sus temas de interés, su ubicación geográfica o dependiendo de los usuarios con los que más se relaciona. Estas tendencias por lo general están asociadas con eventos que suceden en la vida real. Particularmente, resulta interesante conocer si existe un *tweet* disparador de su gran repercusión y encontrarlo. Actualmente, existen cuentas en varias partes del mundo que se encargan de indicar de forma manual a qué refieren las tendencias más relevantes utilizando el contexto conocido por el usuario dueño de la cuenta. En Uruguay, un ejemplo de esto es la cuenta @porqtendenciauy[6]. Por otro lado, existe la página Trendsmap [7] que permite filtrar por país y visualizar allí los *tweets* más relevantes y algunas tendencias situadas en Montevideo, para los últimos 7 días.

La motivación de este trabajo es tratar de encontrar una forma automática de reconocer los *tweets* que provocan que ciertos temas se vuelvan tendencia, intentando determinar el *tweet* disparador de cada tendencia que surge en Uruguay.

1.2. Objetivo

Existen dos grandes objetivos en este proyecto. En la sección 1.2.1 se expone el objetivo principal. En la sección 1.2.2 se expone el segundo objetivo del proyecto, que nace producto de una limitante técnica surgida al momento de intentar resolver el anterior.

1.2.1. Encontrar el tweet disparador de una tendencia

El objetivo principal de este proyecto es extraer información de *tweets* publicados y de sus autores, que permitan determinar para cada tendencia de Twitter en Uruguay cuál es el o los *tweets* que convirtieron al tema en *TT*, producto de su gran repercusión. La aplicación de Twitter permite acceder a la lista de *tweets* más destacados de cada tendencia, es decir, los que tuvieron mayor repercusión. Esta lista va cambiando a lo largo del tiempo y nada garantiza que estos *tweets* hayan provocado que el tema al que refieren se convierta en *TT*, ya que pudieron haber obtenido gran difusión y popularidad luego de que la tendencia surgiera. Si bien los *tweets* destacados son relevantes para quien desea conocer el contexto de la tendencia, esto no es suficiente para el objetivo de este proyecto, ya que de estos *tweets*, se intenta encontrar los que presentan un buen balance entre haber sido de los primeros en publicarse y haber tenido gran repercusión, es decir, los disparadores.

[6] https://twitter.com/porqtendenciauy

[7] https://www.trendsmap.com/local/uy/montevideo [online; último acceso 6-Agosto-2020]

En la Figura 1.1 se pueden apreciar distintos *tweets* destacados para la tendencia "Canelones", los cuales tienen varias horas de diferencia entre sí y no hay certeza de que alguno de ellos hubiese sido su disparador.

Figura 1.1: *Tweets* destacados asociados a la tendencia "Canelones"

Para poder identificar los *tweets* disparadores, se intenta estimar su repercusión generada en la red utilizando un enfoque de Inteligencia Artificial de forma de aprovechar la gran cantidad de *tweets* que se generan por día y sus datos relacionados. Se intenta aprender características generales de los *tweets* que impulsan a los diferentes temas a ser tendencia. Además de diseñar el modelo que mejor se ajuste a resolver este problema, un punto importante radica en elegir qué información de estos *tweets* es la más adecuada y combinarla de una forma que asegure buenos resultados al momento de evaluar nuestro modelo.

1.2.2. Detectar los tweets que refieren a una tendencia

Este objetivo surge a partir de que el mecanismo de obtención de *tweets* utilizado retorna todos aquellos que nombran textualmente la expresión que es TT. Esto hace que sea necesario filtrar y descartar aquellos *tweets* que si bien cumplen la condición anterior, no refieren a la tendencia. Por

este motivo, se debe identificar dentro de una masa de *tweets* que contienen cierta expresión, cuáles de ellos refieren al tema relacionado con el *TT*. Para esto se intenta estimar el tema principal del TT en base al contenido de los *tweets* que la nombran. Actualmente, Twitter muestra los *tweets* asociados a una tendencia, permitiendo filtrar en tiempo real por los más destacados o los más recientes. Se entiende que un *tweet* está asociado a una tendencia cuando contiene la expresión que es *TT*. En la Figura 1.2, se pueden ver dos *tweets* asociados a la tendencia "robo"[8]. El primero de ellos no refiere a la tendencia, mientras que el segundo sí hace referencia a su tema principal.

Figura 1.2: *Tweets* con temáticas distintas asociados a una misma tendencia

1.3. Cronograma

El proyecto comenzó en abril de 2020. La comunicación fue realizada a distancia durante todo el proyecto. En un principio se utilizó una metodología de reuniones virtuales semanales del equipo de trabajo. En la fase final del proyecto, las reuniones fueron diarias o día por medio para discutir soluciones y seguir de cerca su avance. Su objetivo también era dividir las tareas entre los 2 integrantes del equipo, de acuerdo al grado de avance y a la disponibilidad de cada uno. Las reuniones de seguimiento con los tutores del proyecto fueron realizadas cada dos semanas, con el objetivo de exponer el avance alcanzado y recibir sugerencias, intercambiando ideas para tomar decisiones y avanzar durante los siguientes 15 días. Esta forma de trabajo permitió afrontar el proyecto desde un enfoque ágil, lo que sirvió para ir modelando el alcance de acuerdo a las expectativas del proyecto y de los objetivos y resultados que fueran interesantes que se contemplaran.

Las tareas realizadas para llevar a cabo el proyecto fueron las siguientes.

1) Profundizar y nivelar los conocimientos en el área del Aprendizaje Automático con un curso realizado en la plataforma Coursera de la Universidad de Stanford [24].

[8]Detectada el 20 de Julio de 2021 a causa de un fallo arbitral en el partido de fútbol Atlético Mineiro vs Boca

2) Investigar el estado del arte en trabajos relacionados a este proyecto. Esto consiste en buscar de artículos relacionados, que posteriormente fueron calificados según su relevancia hasta obtener un conjunto reducido de artículos.

3) Diseñar y desarrollar un módulo de extracción de *tweets* utilizando la *API* de Twitter [10].

4) Diseñar y desarrollar un modelo que dada una tendencia y sus *tweets*, permitiera descartar aquellos que no refieren a los temas principales de la tendencia.

5) Diseñar y desarrollar un modelo que dada una tendencia y sus *tweets*, utilizando técnicas de Aprendizaje Automático, intente predecir cuál es el *tweet* que disparó la tendencia.

6) Confeccionar y anotar los ejemplos de entrenamiento para evaluar los modelos.

7) Diseñar y desarrollar un proceso repetitivo (de ahora en más demonio[9]) que para las tendencias y *tweets* que se van obteniendo y guardando en la base de datos, utilice ambos modelos mencionados en los puntos 4 y 5 para calcular el *tweet* disparador de cada tendencia.

8) Realizar las evaluaciones experimentales de los modelos construidos.

9) Desarrollar una aplicación web para mostrar los resultados obtenidos.

10) Elaborar el informe.

La Figura 1.3 muestra la dedicación a estas tareas a lo largo del proyecto.

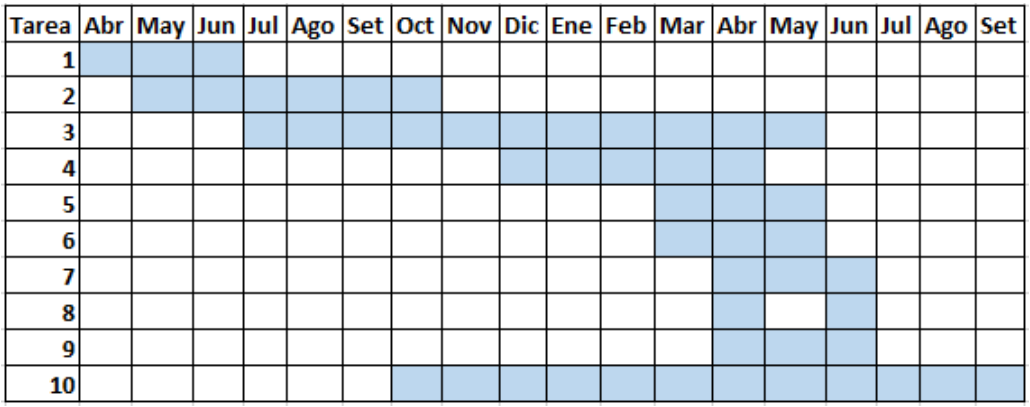

Figura 1.3: Distribución de tareas durante el proyecto

1.4. Organización del informe

El informe se estructura como se explica a continuación. El capítulo 2 presenta el marco teórico del proyecto, donde se definen los principales conceptos utilizados a lo largo del informe. Los

[9]Nombrado de esta manera por su funcionamiento similar a los *daemon* implementados en Unix

trabajos relacionados con este proyecto se exponen en el capítulo 3. El capítulo 4 describe el análisis del problema a resolver, las herramientas utilizadas para la construcción de la solución, y el diseño de la arquitectura implementada. El capítulo 5 contiene el detalle de cada módulo de software construido. Los resultados obtenidos en las pruebas realizadas para cada componente del sistema se presentan en el capítulo 6. El capítulo 7 incluye las conclusiones del proyecto y el posible trabajo a futuro. Sobre el final de este informe se incluye un glosario de términos y dos anexos, donde se ubica el detalle completo de los resultados obtenidos al evaluar los componentes construidos.

Capítulo 2

Marco teórico

En este capítulo se explican algunos conceptos, técnicas y tareas vinculadas a la Inteligencia Artificial, al Procesamiento de Lenguaje Natural y otros conceptos necesarios para enmarcar el proyecto. El propósito de este capítulo es contextualizar al lector en la teoría referenciada en las secciones posteriores de este informe.

En la sección 2.1 se introduce el concepto de Inteligencia Artificial. Luego, en la sección 2.2 se presenta la disciplina de Procesamiento de Lenguaje Natural donde se explican las principales etapas y técnicas utilizadas en este proyecto. En la sección 2.3 se describen los conceptos principales asociados con el Aprendizaje Automático, donde se presentan los modelos en los que se basa la solución de este proyecto. Por último, en la sección 2.4 se describen otros conceptos útiles para este proyecto como las Bases de Datos No Relacionales y las Interfaces Web. Aquel lector que tenga conocimientos sobre estos temas puede saltarse la lectura de este capítulo.

2.1. Introducción a la Inteligencia Artificial

El objetivo de esta sección es introducir al lector en la disciplina, brindando conceptos generales que dan contexto al área en la que se enmarca este proyecto.

Según expone Wolfgang Ertel en su libro *Introduction to Artificial Intelligence* [12], existen diferentes definiciones del concepto Inteligencia Artificial (IA). La primera de ellas, expresada por John McCarthy, plantea que el objetivo principal de la IA es desarrollar máquinas que se comportan como si fueran inteligentes. La Encyclopedia Britannica plantea otra definición: IA es la habilidad de computadoras digitales o robots controlados por ellas, de resolver problemas que normalmente están asociados con las capacidades de alto procesamiento intelectual de los seres humanos. Esta definición presenta una debilidad emparentada con que cualquier sistema sería un sistema de IA, ya que problemas de alto procesamiento intelectual podrían ser la capacidad de recordar un texto extremadamente largo, o la capacidad de multiplicar dos números de 20 dígitos. Sin embargo, estas habilidades no están emparentadas con la IA. El autor Elaine Rich resuelve este dilema: la IA estudia cómo hacer que las computadoras realicen tareas, que al momento, las personas son mejores. Tareas de cómputo de grandes cantidades de operaciones en muy poco tiempo, es uno de los puntos fuertes que presentan las computadoras digitales por sobre las capacidades de los humanos. En cambio, otras tareas propias de la naturaleza humana, son muy difíciles de realizar para un robot. Otro fuerte del ser humano es la capacidad de adaptarnos a diversos cambios en el ambiente que nos rodea y ajustar nuestro comportamiento acorde a ello a través del aprendizaje.

Gracias a que nuestra capacidad de aprender es superior a la de una máquina, el Aprendizaje Automático es una rama central de la IA.

En las siguientes secciones se describen las ramas principales de la IA que dan contexto a este proyecto.

2.2. Procesamiento de Lenguaje Natural

El Procesamiento de Lenguaje Natural (PLN) es la disciplina de la IA que estudia cómo programar computadoras para que sean capaces de leer y entender el lenguaje humano. Esta disciplina se enfoca en la interacción entre la ciencia de datos y el lenguaje natural [34].

2.2.1. Técnicas utilizadas

A continuación se definen las principales etapas y técnicas del PLN utilizadas en este proyecto.

Tokenización
Es el proceso de separar un texto en las palabras que lo componen. En la lengua española, los espacios en blanco suelen ser separadores de palabras, pero esto no siempre es suficiente [36]. Generalmente, se utiliza como etapa anterior al etiquetado léxico. Para la siguiente oración, se muestra su separación en tokens.

"Suarez no toca al defensa y el arbitro nos roba el empate"

Lista de tokens: Suarez, no, toca, al, defensa, y, el, arbitro, nos, roba, el, empate.

Como se muestra en el ejemplo, las distintas apariciones de una misma palabra son tokens diferentes.

Análisis léxico
El Análisis Léxico, o también llamado Etiquetado Gramatical (en inglés, *Part of Speech Tagging*, o *POS Tagging*), se encarga de procesar un texto escrito en lenguaje natural y asignarle una etiqueta a cada palabra dependiendo de la función gramatical que cumpla [35].

En la Tabla 2.1 se presentan las etiquetas gramaticales utilizadas en el idioma español, según el sitio de dependencias universales [23].

Para la oración de ejemplo, el etiquetado gramatical de cada token sería el que se muestra en la Tabla 2.2.

Bag of Words
Es una técnica utilizada para representación de textos, donde se define una estructura que almacena una bolsa de palabras, es decir, un conjunto de palabras sin orden donde se ignora su posición en el texto original, y su único dato relevante es su frecuencia de aparición. Está basada en el principio de que las palabras que más aparecen en el texto son las que suelen representar mejor su temática general [36].

Reconocimiento de Entidades con Nombre
El Reconocimiento de Entidades con Nombre (NER, por sus siglas en inglés), se encarga de reco-

Etiqueta	Descripción
ADJ	Adjetivo
ADV	Adverbio
INTJ	Exclamación
NOUN	Sustantivo
PROPN	Nombre propio
VERB	Verbo
ADP	Adposición
AUX	Auxiliar
CCONJ	Conjunción coordinativa
DET	Determinante
NUM	Número
PART	Partícula
PRON	Pronombre
SCONJ	Conjunción subordinada
PUNCT	Signos de puntuación
SYM	Símbolo
X	Otro

Tabla 2.1: Etiquetas gramaticales para el español [23]

Suarez	no	toca	al	defensa	y	el	arbitro	nos	roba	el	empate
PROPN	ADV	VERB	ADP	NOUN	CCONJ	DET	NOUN	PROPN	VERB	DET	NOUN

Tabla 2.2: Ejemplo de etiquetado gramatical según el modelo *es_core_news_sm* de la librería Spacy

nocer aquellas palabras o frases que refieren a objetos que tienen un nombre propio, por ejemplo una persona, una localidad geográfica o una organización. Esta etapa consiste en etiquetar las entidades con nombre que aparecen en un texto y etiquetarlas según el tipo de entidad. Las cuatro etiquetas más comunes son: PER (persona), LOC (localidad), ORG (organización) y GPE (entidad geopolítica) [36]. Si bien esta técnica no se utiliza directamente en este proyecto, nos parece interesante incluirla ya que las tendencias de Twitter por lo general refieren a este tipo de expresiones lingüísticas.

2.3. Aprendizaje Automático

La definición de este concepto y de sus ramas está basada en el curso de *Machine Learning* realizado en la plataforma Coursera de la Universidad de Stanford [24]. El Aprendizaje Automático, en inglés *Machine Learning*, es el campo de estudio que le brinda a las computadoras la habilidad de aprender, concluyendo su comportamiento a partir de los datos. Una definición más formal es la siguiente: Un programa computacional se dice que aprende de una experiencia E, con respecto a determinada tarea T, bajo una medida de *performance* P, si su *performance* realizando la tarea T, medida según P, mejora con la experiencia E.

Está compuesto por dos principales ramas: Aprendizaje Supervisado y No Supervisado.

Aprendizaje Supervisado
En esta técnica se cuenta con un conjunto de datos donde se conoce cuál debería ser su valor de predicción, bajo la hipótesis de que existe una relación entre las variables de entrada y su salida esperada. Esto significa que el conjunto de datos de entrada debe estar previamente anotado, tarea que usualmente se realiza de forma manual debido a que su valor de salida es conocido por una persona, quien supervisa el entrenamiento. Los problemas de Aprendizaje Supervisado se categorizan en problemas de regresión y clasificación. En los problemas de regresión, se intenta predecir el resultado de una variable de salida continua, es decir, se mapean las variables de entrada a una función continua. En los problemas de clasificación, se intenta predecir resultados de una variable de salida discreta. En otras palabras, se mapean las variables de entrada en un conjunto de categorías.

Aprendizaje No Supervisado
El Aprendizaje No Supervisado permite encarar problemas partiendo de los datos de entrada, sin conocer cuáles deberían ser sus valores de predicción correspondientes. A pesar de esto, es posible estimar la estructura de la salida detectando características similares entre los elementos del conjunto de entrada. *Clustering* y Reducción de Dimensionalidad son ejemplos de problemas que se resuelven utilizando este tipo de técnicas. En el Aprendizaje No Supervisado no existe retroalimentación una vez obtenidos los resultados de predicción.

2.3.1. Clasificación

El objetivo de la clasificación es extraer las características útiles de un individuo de observación, para poder asignarle una de varias clases. Un método utilizado para clasificación de texto consiste en definir conjuntos de reglas escritas a mano. Una desventaja de este método es que las reglas son costosas de escribir y a medida que crece la cantidad de reglas el modelo se vuelve más complejo. Dado que aprender de los datos es una forma de evitar escribir reglas, tanto en este proyecto como en la mayoría de los problemas de clasificación, se utilizan técnicas de Aprendizaje Supervisado, en las cuales se tiene un conjunto de observaciones de entrada, cada una con su valor de salida esperado [36]. En este proyecto se afrontan problemas de clasificación binaria, donde cada individuo de entrada es asignado a la clase positiva o negativa.

2.3.2. Regresión Logística

La Regresión Logística es el algoritmo de Aprendizaje Supervisado de referencia que se utiliza para resolver problemas de clasificación. Su objetivo es clasificar a cada individuo observado en una de dos clases (o más[1]) [36]. Su función de hipótesis resulta de aplicarle una función g al producto de la matriz X de ejemplos de entrenamiento y el vector *theta* de parámetros del modelo. Los valores de esta función corresponden a la probabilidad que se le puede asignar a los datos de entrada.

[1]Regresión Logística Multinomial

Como función g se utiliza la función *sigmoid*, tal como se muestra en (2.1).

$$\begin{aligned} h_\theta(x) &= g(\theta^T x) \\ z &= \theta^T x \\ g(z) &= \frac{1}{1+e^{-z}} \end{aligned} \tag{2.1}$$

En la Figura 2.1 se aprecia gráficamente que la función *sigmoid* tiene forma de escalón suavizado comprendido entre 0 y 1. Esta función es diferenciable, lo cual se utiliza para poder ajustar los valores aprendidos por el modelo.

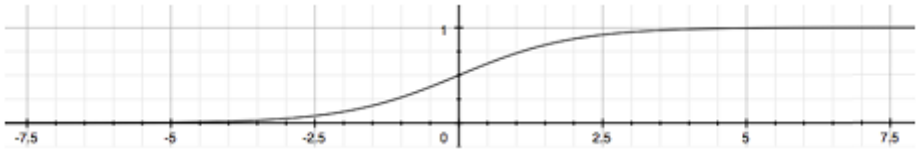

Figura 2.1: Representación gráfica de la función sigmoid

Una vez aplicada esta función se define un umbral de decisión, el cual indica que por encima de este valor se asignará el elemento a la clase positiva y por debajo del umbral a la clase negativa. Como se muestra en (2.2), si el umbral de decisión es 0.5, la categoría asignada a un ejemplo x dependerá de su salida en la función de hipótesis.

$$\begin{aligned} h_\theta(x) \geq 0.5 \rightarrow y = 1 \\ h_\theta(x) < 0.5 \rightarrow y = 0 \end{aligned} \tag{2.2}$$

2.3.3. Red Neuronal

Una Red Neuronal es un grupo interconectado de neuronas artificiales, utilizado para la resolución de problemas de alta complejidad. Las neuronas artificiales son modelos matemáticos que utilizan un vector de pesos como parámetro interno y reciben como entrada un vector de valores. Aplicando una combinación lineal de ambos vectores y utilizando el resultado en una llamada "función de activación", se determina el valor de salida de esta neurona para los valores de entrada, como se muestra en la Figura 2.2. La función de activación es una función continua y diferenciable.

En otras palabras, la salida o de una neurona artificial queda determinada por la siguiente función, donde σ representa a la función de activación, el vector x al vector de entrada, y w al vector de pesos.

$$o = \sigma(\vec{x}.\vec{w})$$

Las funciones de activación pueden ser la denominada función *sigmoid*, la tangente hiperbólica o *relu*.

Estas redes son sistemas adaptativos que cambian el valor de su vector de pesos en base a información externa o interna que fluye a través de esta.

Existen tres tipos de capas de neuronas para una Red Neuronal [22]. La capa inicial se le denomina capa de entrada, la capa final se denomina capa de salida y las capas intermedias son denominadas capas ocultas, como se muestra en la Figura 2.3.

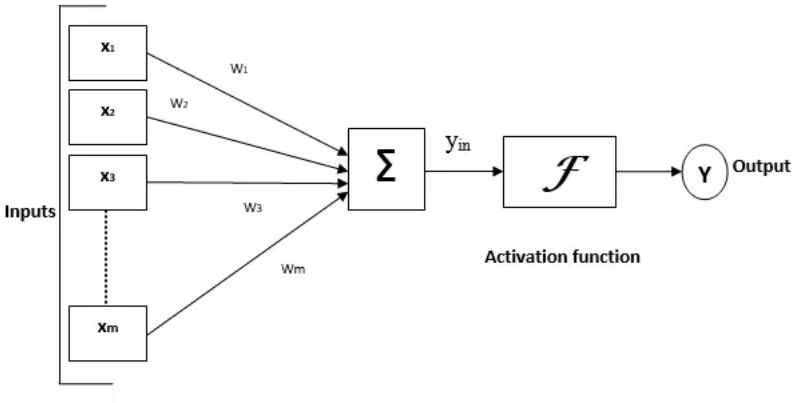

Figura 2.2: Representación de una neurona artificial [41]

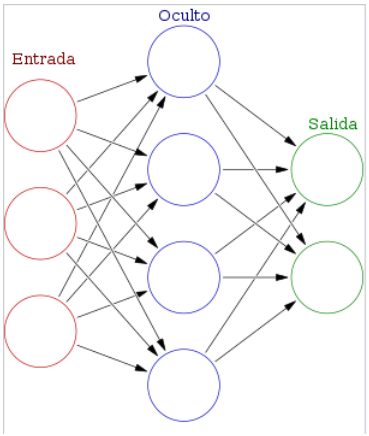

Figura 2.3: Capas de una Red Neuronal [25]

De los distintos tipos de Redes Neuronales que existen, en este trabajo se utilizan las denominadas redes *feed forward*. En este tipo de redes, la entrada de una capa de la red es la salida de la capa anterior, es decir, la información viaja en un solo sentido y no se generan bucles.

2.3.4. Medidas de evaluación

Para evaluar el comportamiento de los modelos de aprendizaje en el marco de un problema de clasificación binaria, es necesario conocer los resultados esperados para el conjunto de datos a evaluar. En este contexto, se define el concepto de matriz de confusión.

Matriz de Confusión: Es una matriz de 2x2 que permite evaluar la predicción de una colección de individuos en dos clases considerando los valores de clase verdaderos y sus valores de predicción. Sea C la matriz de confusión, C[0,0] contiene los verdaderos positivos (VP), C[1,0] los falsos negativos (FN), C[1,1] los verdaderos negativos (VN), y C[0,1] los falsos positivos (FP).

En la imagen 2.4 se muestra como quedan los datos en la matriz de confusión.

Partiendo de esta definición, se pueden calcular las siguientes medidas de evaluación [13].

Figura 2.4: Definición de matriz de confusión

Precision: Es la cantidad de instancias relevantes sobre la cantidad de instancias extraídas como relevantes.

$$Precision = \frac{VP}{(VP + FP)}$$

Recall: Es la fracción de instancias relevantes que fueron extraídas sobre el total de instancias relevantes.

$$Recall = \frac{VP}{(VP + FN)}$$

F1-score: Es una medida que combina *precision* y *recall* para generar un valor de balance entre ambas.

$$F1\_score = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

Accuracy: Es la proporción de predicciones correctas (positivas y negativas) sobre la cantidad total de casos examinados.

$$Accuracy = \frac{(VP + VN)}{(VP + VN + FP + FN)}$$

En la Figura 2.5 se muestra una descripción gráfica de las medidas de precisión y *recall* definidas.

2.3.5. Word embeddings

Es un método de representación vectorial del significado de las palabras a partir del contexto en el cual aparecen [36]. Estos vectores son utilizados en muchas aplicaciones de PLN, ya que ayudan a generalizar y agrupar palabras con semántica similar. El modelo surge como producto de dos grandes ideas: la idea de representar palabras como puntos del espacio vectorial de tres dimensiones, y la propuesta de lingüistas como Joos, Harris y Firth de definir el significado de una palabra por su distribución en el uso del lenguaje, es decir, sus palabras vecinas o sus formas gramaticales. Posteriormente, la idea de representación vectorial se extendió a n dimensiones. Las implementaciones más utilizadas de esta técnica son *word2vec*, *GloVe* [40], y en particular

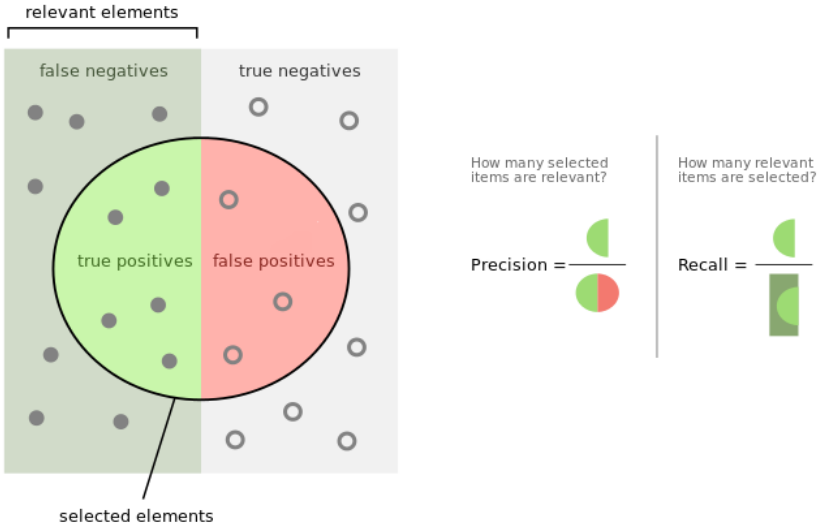

Figura 2.5: Representación gráfica de precisión y *recall* extraída de [39]

la utilizada en este proyecto es *fastText* [16]. En la Figura 2.6 se presenta la gráfica del espacio vectorial reducida a 2 dimensiones donde se muestra la cercanía de la representación de palabras con significado similar.

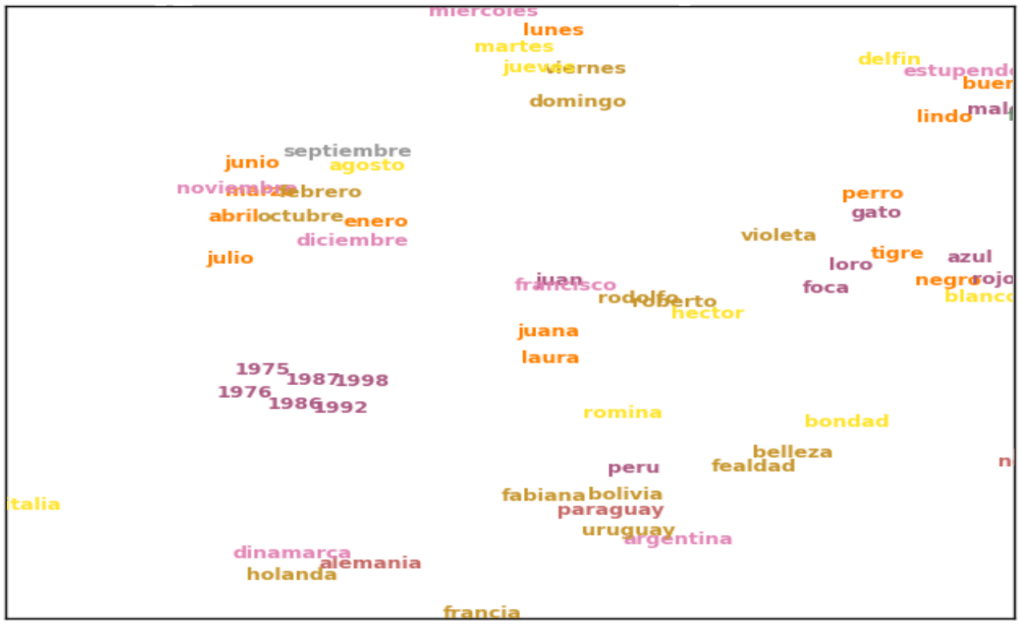

Figura 2.6: Vectores de palabras en el espacio bidimensional [33]

2.4. Otros conceptos adicionales

Esta sección refiere a otros conceptos y estructuras relevantes utilizadas para la construcción de la solución.

2.4.1. Base de Datos No SQL

Las bases de datos No *SQL*, o también llamadas bases de datos No Relacionales, surgen como solución a la necesidad de almacenar grandes cantidades de datos, como por ejemplo una aplicación de correo electrónico o una red social como Twitter, donde los datos son almacenados en forma de documentos. Su alta *performance*, disponibilidad, buena replicación de datos y escalabilidad son ventajas en el uso de este tipo de bases de datos. A diferencia de las bases de datos relacionales tradicionales, los documentos almacenados no tienen una estructura fija. Por más que los documentos de una colección suelen ser similares, estos pueden contener distintos atributos. Los formatos más típicos que se utilizan para representar los documentos almacenados son los formatos *XML*[2] y *JSON*[3] [20].

2.4.2. Web API

Una Web *API* (*Application Programming Interface*) es un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar los componentes de software de una aplicación [27]. Estas permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto otorga flexibilidad, proporciona oportunidades de innovación y simplifica el desarrollo, el diseño, la administración y el uso de las aplicaciones.

En otras palabras, las *APIs* se consideran como contratos, con documentación que representa un acuerdo entre las partes: si una de las partes envía una solicitud remota con cierta estructura en particular, esta estructura determinará cómo debe responder el software de la contraparte.

[2]Extensible Markup Language
[3]JavaScript Object Notation

Capítulo 3

Trabajos relacionados

En este capítulo se describen los trabajos relacionados al manejo de información de Twitter, que si bien no están directamente emparentados con el objetivo de este trabajo, aportan información valiosa a la hora de analizar las características de los *tweets* y sus autores. Al momento de estudiar el estado del arte en proyectos similares no se encontraron artículos que intenten resolver este problema. Si bien esto planteaba un mayor desafío en la propuesta, se presumía su dificultad al momento de comparar los resultados con trabajos similares. Por este motivo, se buscaron trabajos relacionados a la investigación con datos de Twitter y así tomar conocimiento de los temas que fueron desarrollados, las técnicas aplicadas para obtener y manipular los datos, y las conclusiones a las cuales llegaron.

A la hora de filtrar los estudios más relevantes, se tuvo en cuenta parámetros como por ejemplo la cantidad de citas y la cercanía en el tiempo. Una vez analizados aproximadamente 50 artículos, nos quedamos con un conjunto final de 8, los cuales se analizaron en detalle.

Luego de revisar estos artículos, se concluye que a lo largo de la última década, se han utilizado técnicas y atributos en común en varias oportunidades con buenos resultados. En la siguiente sección se expone el análisis realizado sobre los trabajos relacionados que fueron elegidos como base para este proyecto.

3.1. Selección de trabajos relacionados

Los artículos seleccionados refieren a diversos temas como por ejemplo: reconocimiento de eventos, reconocimiento de cuentas similares, clasificación de TT en áreas de interés, detección de *spam* y predicción de la propagación de *tweets*. En las siguientes secciones se resumen los conceptos más relevantes que se extraen de estos artículos de forma de agregar valor al proyecto, por lo cual se agrupan de acuerdo a la forma de utilizar la información obtenida de Twitter, mencionando particularidades y variantes expuestas en cada uno de ellos a medida que son mencionados.

3.1.1. Atributos basados en el usuario y en el *tweet*

Más allá del objetivo particular de cada artículo, al momento de diseñar un modelo de Inteligencia Artificial se afirma que existen dos grupos de atributos. Por un lado existen los atributos relacionados a la información del *tweet* y por otro los relacionados a la información de su autor. Tanto en el artículo de *Alex Hai Wang* [1], como en el de *Sasa Petrovic, Miles Osborne y Victor Lavrenko* [2], se utiliza la combinación de ambos grupos de atributos obteniendo buenos resultados.

Los atributos basados en los usuarios son utilizados para determinar la relevancia que puede llegar a tener una cuenta, el impacto de sus publicaciones, o si directamente sus comentarios no repercuten en la red. En cambio, los atributos basados en el *tweet* evidencian realmente el impacto que este tuvo. Además, utilizando información asociada al *tweet* es posible predecir si tiene potencial para generar repercusión, por ejemplo, si contiene videos, imágenes o *urls* a otros sitios web.

3.1.2. Atributo *"Followers to Followees"*

Cada cuenta de Twitter puede suscribirse como seguidora de otra cuenta, de forma de recibir actualizaciones de estado de los usuarios que son de su interés. Por este motivo, cada usuario se relaciona con un conjunto de usuarios seguidos (*followees*) y un conjunto de usuarios seguidores (*followers*). En el trabajo realizado por *Gerasimos Razis y Ioannis Anagnostopoulo* [5], se utiliza la relación *Followers to Followees* (*FtF*) como forma de medir la importancia o influencia de una cuenta. Esta medida está basada en la hipótesis de que un usuario con muchos más *followers* que *followees*, genera actividad en Twitter sin necesidad de ser influenciado por nadie. A estos se los denomina usuarios activos, quienes pasan más tiempo generando contenido que leyendo publicaciones en la red. Por otra parte, a los usuarios con más *followees* que *followers* se les llama pasivos, ya que su objetivo principal está más emparentado con el consumo de *tweets*.

Los autores calculan esta medida de la siguiente manera.

$$FtF = log_{10}((\frac{followers}{followees}) + 1)$$

Una medida utilizada para distinguir cuentas con un valor similar de *FtF*, es la *Tweets Creation Rate* (*TCR*), calculada a partir de los últimos 100 *tweets* de la cuenta obtenidos de la *API* de Twitter. Ante cuentas con similar *FtF*, se asigna un valor de influencia mayor a los usuarios que tienen una tasa de publicación de *tweets* mayor, es decir, a los más activos.
En el artículo ya mencionado de *Alex Hai Wang* [1] se intenta estimar la reputación R de una cuenta para la detección de *spam*, ya que este tipo de cuentas se caracterizan por tener un valor bajo en esta medida. Para implementar esto, se utiliza un grafo cuyos nodos son los usuarios y las aristas las relaciones *follower-followee*.

$$r = \frac{followers}{(followers + following)}$$

Si bien las medidas *Ftf* y r no son exactamente iguales, están basadas en el mismo principio.

3.1.3. Atributo "Influencia" de una cuenta

Tal como se menciona en la sección 3.1.2, es interesante poder cuantificar la influencia o importancia de una cuenta. En la publicación de *Gerasimos Razis y Ioannis Anagnostopoulo* [5] se analiza que un buen valor de *FtF* no garantiza el poder de influencia de una cuenta. La medida *TCR* puede combinarse con *FtF* para acercarnos a un cálculo más realista en lo que refiere a la influencia. En este trabajo se combinan medidas como la cantidad de *retweets* y favoritos en los últimos 100 *tweets*, el orden de magnitud de la cuenta en base a sus seguidores, el tiempo transcurrido desde el ultimo *tweet* publicado y *FtF* para calcular la influencia de la cuenta.

Entre otros atributos utilizados en este artículo se destacan los siguientes: *followers*, *followees*, *profileLocked* (si el perfil de la cuenta es público o no), *activeAccount*, *replyRatio* (del total de los

últimos *tweets* de la cuenta, cuales fueron en respuesta a otro), *rtPercent* (porcentaje de los últimos *tweets* de la cuenta que fueron *retweets*), cantidad total de *tweets* y cantidad de *tweets* por día.

3.1.4. Detección de eventos

Según *Mateusz Fedoryszak, Vijay Rajaram, Brent Frederick* y *Changtao Zhong* [8], el aumento en la cantidad de *tweets* que hacen referencia a un tema en particular, está relacionado con un evento que se produce en el mundo real que provoca que muchos usuarios compartan su opinión en la red. Por el contrario, existen afirmaciones que manifiestan que algo relevante está sucediendo en el mundo real cuando una entidad relacionada a este evento se vuelve tendencia en las redes sociales. En este artículo, se establece que los eventos se definen por las entidades que lo componen. Por ejemplo, un evento deportivo se vería representado por entidades con nombre como los nombres de los equipos, nombres de los jugadores, etc. Según los autores, para detectar eventos es de vital importancia que el modelo diseñado sea dinámico en lo que refiere a contemplar la transformación de las entidades que son tendencia a lo largo del tiempo.

Los autores extrajeron información que relaciona entidades con nombre y lugares geográficos, de forma de poder agrupar distintas entidades utilizando distancia coseno. De esta forma se puede determinar si dos entidades están asociadas a un mismo evento.

El trabajo de *G. Ifrim, B. Shi* y *I. Brigadir* [4] también presenta un modelo que intenta agrupar *tweets* por tema, calculando su similaridad definiendo una cota para determinar cuándo dos *tweets* refieren al mismo tema.

Capítulo 4

Solución propuesta

En este capítulo se describe la solución propuesta al problema a resolver. En la sección 4.1 se presenta una descripción del problema junto con los desafíos más relevantes que fueron detectados. En la sección 4.2 se listan y detallan las principales herramientas utilizadas a lo largo del proyecto, las cuales facilitaron su realización. La arquitectura utilizada se presenta en la sección 4.3.

4.1. Análisis del problema

En esta sección se brinda una descripción y definición formal del problema a resolver, junto con los desafíos más importantes que presenta.

4.1.1. Descripción

Dada una tendencia de Twitter en Uruguay, el problema consiste en determinar sus *tweets* disparadores. Es decir que dado el conjunto de *tweets* que mencionan un tema, se debe encontrar el *tweet* o conjunto de *tweets* que causan la mayor repercusión y lo convierten en tendencia.

4.1.2. Formalización

Dada una tendencia, existe un conjunto T de *tweets* que la mencionaron recientemente, los cuales generaron una repercusión masiva del tema. El primer objetivo consiste en encontrar el subconjunto U de T, tal que los *tweets* que pertenecen a U refieren al tema o temas principales de la tendencia. El segundo objetivo, es encontrar el subconjunto W comprendido en U, tal que los *tweets* pertenecientes a W son los disparadores de la tendencia. Aplicando teoría de conjuntos, el problema se reduce a encontrar los conjuntos de *tweets* U y W tal que:

$$W \subset U \subset T$$

4.1.3. Desafíos

A continuación se describen los desafíos más importantes que presenta el problema a resolver y, en algunos casos, la solución adoptada.

1) La gran cantidad de *tweets* que se generan en la red

Para que el problema sea escalable, dada la cantidad de *tweets* y tendencias que se generan en la red a cada segundo, fue conveniente diseñar una solución que solamente involucre tendencias de Uruguay. Este enfoque mejora los tiempos de procesamiento ya que reduce la cantidad de *tweets* a almacenar. Esto se debe a que gran parte de las tendencias a procesar repercuten solamente en Uruguay, donde la cantidad total de *tweets* es menor.

2) Se desconoce el algoritmo de Twitter para determinar tendencias

Si bien existen *blogs* y sitios web que exponen pautas necesarias para convertir un tema en tendencia [38], no se encontró de forma oficial el algoritmo que utiliza Twitter para determinarlas. Por este motivo, se obtienen los *tweets* que mencionaron la tendencia en las horas anteriores a que haya surgido. Si bien esto permite excluir de la búsqueda los *tweets* generados posteriormente a que el tema ya se convirtió en tendencia, se define también una cota inferior al horario de creación de los *tweets* extraídos, que será explicada más adelante. Esto permite no guardar innecesariamente publicaciones anteriores que no aportan a que el tema se haya convertido en tendencia en el momento que se la detecta.

3) Obtención de *Trending Topics* de Uruguay

Debido a que la interfaz utilizada para la comunicación con Twitter no ofrece las tendencias de la región Uruguay, ni por código de país ni por sistema de coordenadas, se desarrolla un mecanismo alternativo para obtenerlas. Este mecanismo consta de un algoritmo basado en una técnica de *web scraping*, que se encarga de interactuar con el sitio web de Twitter y de extraer la lista de tendencias sugeridas para un usuario configurado en Uruguay.

4) *Tweets* que nombran la tendencia pero no refieren a ella

Al obtener los *tweets* que nombran una tendencia es necesario poder filtrarlos para poder descartar los *tweets* que, si bien incluyen el texto que es tendencia, no refieren al tema o temas principales de esta. Esto permite procesar una cantidad menor de *tweets* al momento de calcular cuál de ellos fue el que tuvo más difusión para la tendencia y asegurar que este sea representativo del tema que causó tanta repercusión.

5) Subjetividad al anotar el conjunto de datos

Al momento de anotar el conjunto de datos utilizados para entrenar y evaluar los modelos implementados, pueden surgir problemas debido a la subjetividad que conlleva la tarea de interpretar a qué clase debería pertenecer un *tweet*.

Por ejemplo, en el modelo de clasificación temática, a cada *tweet* de entrenamiento de una tendencia se lo puede clasificar en la clase positiva (si realmente refiere a los temas principales de la tendencia), o negativa en caso contrario. Dependiendo de qué tan fácil sea reconocer los temas reales que disparan la tendencia pueden existir *tweets* que según el punto de vista y conocimiento del contexto de la persona que realiza la clasificación, se les pueda asignar la clase positiva o negativa con similar probabilidad.

Un ejemplo de esto es el caso de la tendencia “Diente”, anotada durante el transcurso de este proyecto. La Figura 4.1 muestra un *tweet* de ejemplo que podría referir a la tendencia, ya que según el contexto analizado, la tendencia refiere a un futbolista apodado “Diente”. Sin embargo, en la Figura 4.2, si bien el *tweet* refiere al mismo futbolista, no es tan evidente que se esté hablando del mismo tema. Esta situación le dificulta la tarea a la persona que realiza la anotación del corpus de *tweets* de una tendencia, ya que deberá adoptar un criterio que defina si todos los *tweets* que nombran a este futbolista refieren a la tendencia, o sólo los que refieren a un tema en particular que lo involucra.

Figura 4.1: Ejemplo de tweet de la tendencia “Diente”

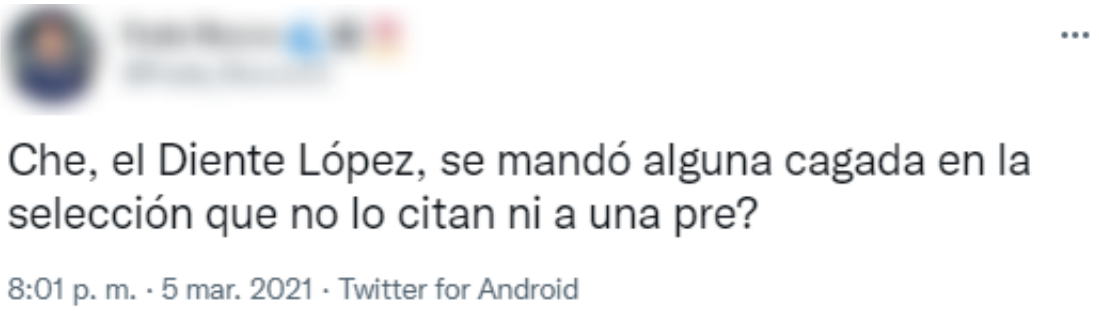

Figura 4.2: Ejemplo de tweet de la tendencia “Diente”

Por otra parte, en la clasificación del modelo de predicción del *tweet* disparador, es posible que dos personas con perspectivas distintas, identifiquen distintos disparadores para un mismo conjunto de datos.

6) Presentar el disparador de una tendencia al usuario en tiempo real

Dada la gran cantidad de *tweets* que pueden mencionar una tendencia, el tiempo de procesamiento para filtrar los que no refieren a sus temas principales y el procesamiento para detectar su disparador, se presenta el desafío de poder realizar estos cálculos con un tiempo de respuesta razonable para una tendencia seleccionada por un usuario final. Para poder cumplir con este objetivo se calcula el *tweet* disparador de cada tendencia a medida que estas surgen. Al usuario final se le

ofrece una lista de tendencias previamente procesadas, de forma que al seleccionar una se despliega rápidamente el resultado.

4.2. Herramientas utilizadas

A continuación se describen las herramientas de software más relevantes utilizadas para la construcción de la solución.

1. **Python:** Es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se lo elige para este proyecto ya que es muy utilizado para desarrollo en el área del Aprendizaje Automático, debido a que cuenta con librerías muy útiles para ello. En este proyecto, se utiliza la versión de Python 3.8.3.

2. **Scikit-Learn:** Es una librería de Python que provee herramientas simples y eficientes para análisis de datos utilizados en modelos de predicción [13]. De esta se utilizan el modelo de Regresión Logística, el modelo de Red Neuronal y métricas que permiten evaluarlos.

3. **Pickle:** Este módulo implementa protocolos binarios para serialización y deserialización de la estructura de un objeto de Python. Este proceso consiste en convertir un objeto de Python en una secuencia de bytes [14]. Su objetivo es persistir el modelo entrenado, generando un archivo serializado.

4. **Spacy:** Es una librería de código abierto utilizado para tareas de PLN avanzado en Python. Está construida para crear aplicaciones de uso productivo que procesan y "entienden" grandes volúmenes de texto. Se utiliza para extracción de información y pre-procesamiento de texto para Aprendizaje Automático [15]. Esta librería se utiliza para obtener el etiquetado POS y poder filtrar palabras con poco contenido semántico.

5. **FastText:** Es un modelo pre-entrenado que se utiliza para obtener representaciones de palabras y clasificación de oraciones. Está entrenado con ejemplos de *Common Crawl* [21] y *Wikipedia*. Está construida en lenguaje C++ [16]. Brinda la representación vectorial de los distintos tokens que aparecen en los *tweets*, lo cual nos ayuda a estimar a qué tema refiere un *tweet*.

6. **MongoDB:** Es una base de datos no relacional, que almacena documentos y ofrece una gran escalabilidad y flexibilidad, brindando un modelo de consultas e indexación avanzado [17]. Su interfaz gráfica *Mongo Compass Community*, nos permitió visualizar y explorar las colecciones de datos de forma más eficiente e intuitiva [17].

7. **API de Twitter v1.1:** Permite obtener de forma sencilla los *tweets* que se generan en la red. Brinda el acceso a diferentes recursos, como por ejemplo: *tweets*, usuarios, mensajes directos, listas, tendencias, archivos multimedia, lugares, etc. Actualmente existen dos versiones, la v1.1 y la v2, con distintas formas de acceso a ellas. Este trabajo se basó la versión 1.1, ya que al momento de comenzar el proyecto, la versión 2 aún no estaba publicada.

8. **Selenium:** Es un software que contiene un gran abanico de herramientas y librerías que permiten automatizar el acceso a navegadores web. Proporciona extensiones que permiten simular las interacciones que realizan los usuarios con los navegadores. A esta técnica se la conoce con el nombre de *web scraping*. El corazón de Selenium es el WebDriver, una interfaz que permite escribir conjuntos de instrucciones que se pueden ejecutar de manera indistinta

en muchos navegadores [18]. En este proyecto es utilizado para acceder a la web de Twitter a través de Google Chrome.

9. **Python-Twitter:** Se utiliza código abierto de Github [9] para la obtención de *tweets* a través de la *API* de Twitter. Provee ejemplos de consultas que nos fueron de gran utilidad, y sirvió como punto de partida. Se analizó la alternativa de usar TwitterAPI [19] pero se descartó debido a que no se contaba con un método que obtenga tendencias.

10. **NodeJS:** Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones network escalables [26]. Fue utilizada para desarrollar la aplicación web y presentar de forma amigable los resultados a través de su librería React.

4.3. Arquitectura de la solución

En esta sección se describe la arquitectura utilizada en la aplicación construida. Para facilitar su explicación se presenta dividida en dos partes, su parte interactiva construida para presentarle los resultados obtenidos a los usuarios, y su parte *batch* que da soporte a los procesos encargados de obtener y procesar la información necesaria para resolver el problema.

4.3.1. Interfaz interactiva

En base a una arquitectura cliente-servidor, se crea una aplicación web utilizando un servidor NodeJS y su librería React para el desarrollo de su interfaz gráfica. El objetivo es ofrecerle al usuario final una lista de tendencias, con opción de filtro por fecha, para que seleccione una y pueda ver cuál fue su *tweet* disparador. Estas tendencias que ya tienen su *tweet* disparador previamente calculado, son obtenidas de la base de datos mediante una *API* implementada en Python la cual utiliza la librería FastAPI [28].

La Figura 4.3 muestra cómo se comunican los distintos actores de la arquitectura interactiva. A continuación se enumeran sus etapas.

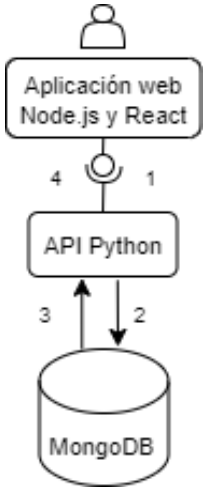

Figura 4.3: Arquitectura interactiva

1. El usuario selecciona filtros de fecha y tendencia a buscar y el servidor web realiza el pedido de tendencias que cumplan con estos filtros a la *API* de Python.

2. La *API* de Python consulta en la base de datos las tendencias que cumplen con los filtros que recibe del servidor web, que ya tengan calculado su *tweet* disparador.

3. La base de datos devuelve a la *API* de Python las tendencias que cumplen con los filtros solicitados.

4. La *API* de Python le devuelve las tendencias encontradas a la aplicación web.

4.3.2. Arquitectura batch

A continuación se presenta la arquitectura de cada componente que participa del procesamiento *batch*, el cual se encarga de la detección de tendencias, extracción de los *tweets* que la mencionan, el almacenado en base de datos y cálculo del *tweet* disparador para cada tendencia.

Extracción de *tweets*

Se implementa un demonio para obtener los *tweets* a utilizar en el cálculo del disparador de cada tendencia de Uruguay. Para cada una de ellas se obtienen los *tweets* públicos que la nombran, utilizando la *API* de Twitter. Los *tweets* obtenidos son almacenados en una base de datos MongoDB, en la cual se define una colección de tendencias y una colección de *tweets*, como se muestra en la Figura 4.4.

Collection Name ▲	Documents	Avg. Document Size
trends	17	107.9 B
tweets	115,869	1.1 KB

Figura 4.4: Colecciones definidas en MongoDB

Para cada *tweet* se registra a qué *trend* está asociado, incluyendo su identificador generado por MongoDB como se muestra en la Figura 4.5.

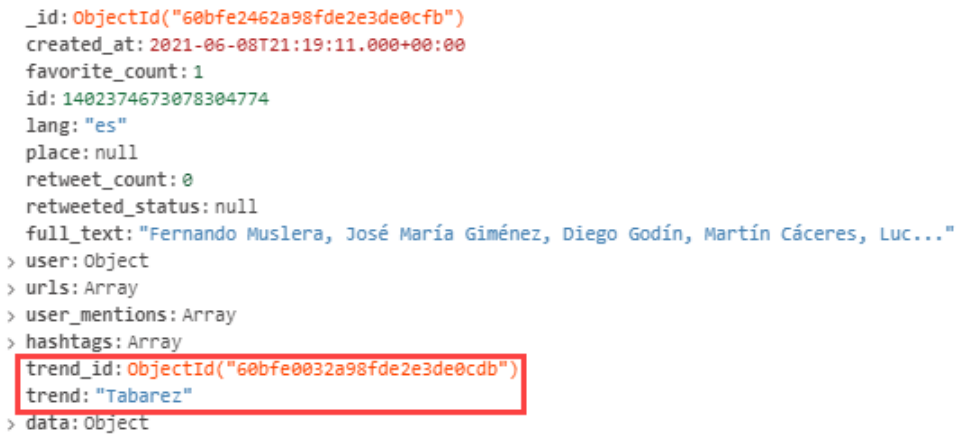

Figura 4.5: Tendencia asociada a cada *tweet*

El campo *created_at* del *tweet* refiere a la fecha y hora de creación en horario UTC[1]. Para las tendencias se registra su fecha/hora en la cual se detectó, también en horario UTC.

En el caso que una tendencia ya almacenada en la base de datos vuelva a surgir, se la vuelve a guardar junto con sus *tweets* asociados, distinguiéndose de la anterior por su fecha/hora de detección. El identificador del objeto generado por MongoDB, permite diferenciar los *tweets* obtenidos para cada tendencia por más que estas sean iguales. En la Figura 4.6 se muestran dos tendencias iguales obtenidas en momentos diferentes, por lo cual son almacenadas de forma separada.

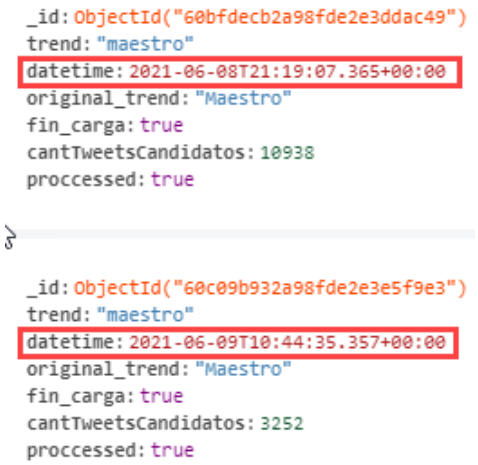

Figura 4.6: Separación de tendencias según su momento de detección

Tanto para la colección de tendencias, como para la colección de *tweets*, se definen índices en la base de datos que mejoran los tiempos de respuesta de los accesos que se realizan tanto al momento de extraer datos de entrenamiento, como las consultas que se realizan desde la aplicación web.

En la Figura 4.7 las filas refieren a los índices definidos para la colección de tendencias. El primer índice es el identificador del objeto, autogenerado por MongoDB. El segundo está formado

[1]Tiempo Universal Coordinado

por la fecha/hora de obtención, si ya fue procesada o no, y su cantidad de *tweets* candidatos. El tercero contiene la fecha/hora de obtención y el texto tendencia.

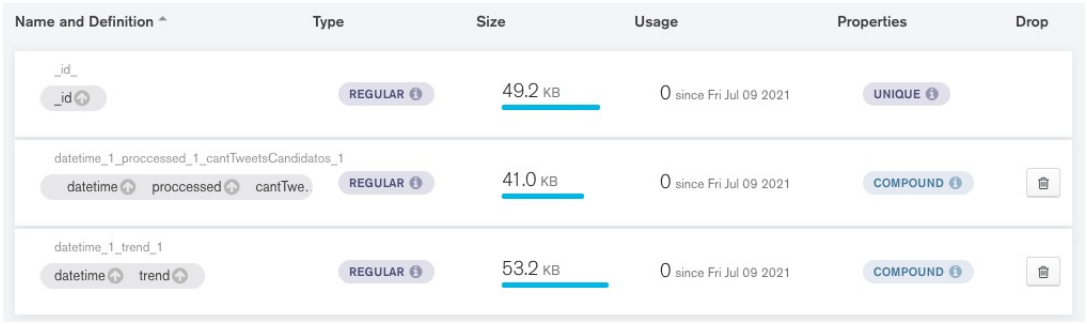

Figura 4.7: Índices en colección de tendencias

En la Figura 4.8 las filas refieren a los índices definidos para la colección de *tweets*. El primer índice contiene el campo identificador del objeto, autogenerado por MongoDB. El segundo contiene su identificador definido por Twitter. El tercero está compuesto por el identificador del objeto tendencia asociado a este, y la fecha/hora de creación del *tweet*.

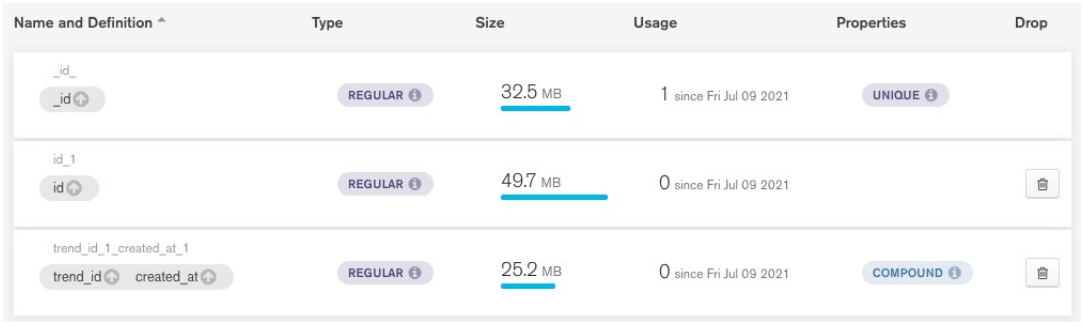

Figura 4.8: Índices en colección de tweets

Modelos de Aprendizaje Automático

Se construyen dos modelos de Aprendizaje Supervisado, uno para cada objetivo de este proyecto. El primero corresponde a un clasificador de *tweets*, cuyo objetivo es determinar cuáles de todos los *tweets* asociados a una tendencia efectivamente refieren a sus temas más relevantes. El segundo modelo es el que determina, para el conjunto de *tweets* seleccionados por el clasificador anterior, cuáles son los que tuvieron la mayor repercusión. Una vez entrenados ambos modelos, se persisten utilizando la librería *Pickle* [14] de Python en archivos serializados con extensión *.pkl*, los cuales son utilizados al momento de predecir el *tweet* disparador de una tendencia.

Pre-procesamiento del *tweet* disparador

El tiempo de respuesta deseable para mostrar el *tweet* disparador de una tendencia seleccionada en la aplicación web, no debería entorpecer la experiencia del usuario[2]. Sin embargo, realizar este cálculo para la tendencia seleccionada en el momento en que el usuario realiza la consulta podría insumir varios minutos, lo cual impide cumplir con este requerimiento no funcional. Como solución a este problema, se implementa en Python otro demonio que para cada nueva tendencia que se almacena en MongoDB calcula su *tweet* disparador utilizando los modelos de Aprendizaje Automático previamente entrenados. Finalmente, la aplicación web le muestra al usuario las tendencias que ya fueron procesadas de forma que al seleccionar una tendencia se muestre el resultado en tiempo real.

Diagrama de procesamiento batch

La Figura 4.9 ilustra la interacción entre los componentes que participan de este proceso.

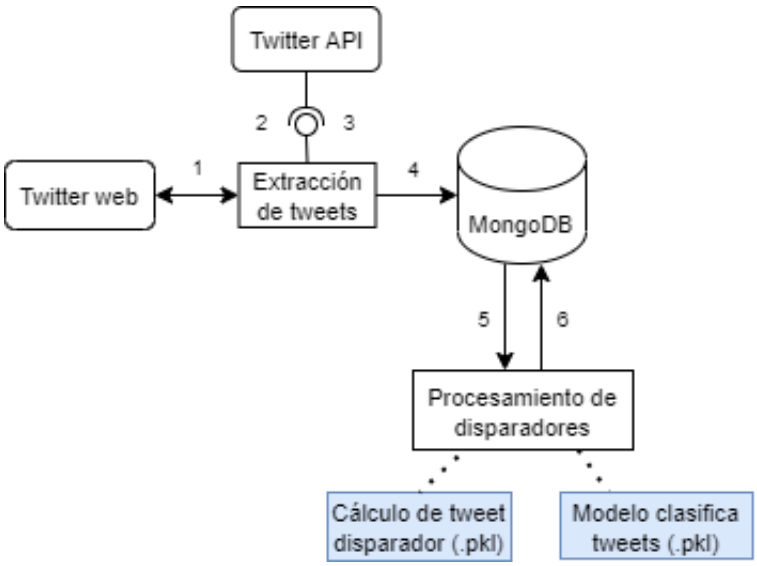

Figura 4.9: Arquitectura *batch*

1. El demonio de extracción obtiene las tendencias de Uruguay realizando *web scraping* en la aplicación web de Twitter.
2. El demonio de extracción solicita *tweets* a la *API* de Twitter, asociados a las tendencias de Uruguay previamente obtenidas.
3. La *API* de Twitter devuelve los *tweets* que mencionan a las tendencias solicitadas.
4. El demonio de extracción de *tweets* guarda en MongoDB las tendencias obtenidas y sus *tweets* asociados.
5. El demonio de procesamiento de disparadores toma las tendencias y sus *tweets* asociados de MongoDB.

[2]Se tomó el criterio de 5 segundos como tiempo máximo aceptable

6. El demonio de procesamiento de disparadores utiliza los modelos de Regresión Logística y Red Neuronal previamente entrenados para calcular el *tweet* disparador de cada tendencia y guarda el resultado nuevamente en MongoDB.

Capítulo 5

Desarrollo de la solución

Este capítulo presenta los módulos implementados como solución al problema planteado en el capítulo 4. En la sección 5.1 se detalla el módulo construido para la extracción de tendencias y *tweets*. Los modelos implementados para la clasificación temática y detección del *tweet* disparador, se presentan en las secciones 5.2 y 5.3 respectivamente. Se describe el proceso desarrollado para el pre-procesamiento de datos en la sección 5.4. Por último, se aborda la aplicación web desarrollada para mostrar los resultados en la sección 5.5. El código implementado se encuentra disponible en GitHub [42] .

5.1. Demonio de extracción de *tweets*

El objetivo de este módulo es poder contar con una gran cantidad de *tweets* almacenados en una base de datos para poder utilizarlos tanto en los modelos de Aprendizaje Automático, como para la aplicación web, en la cual un usuario final pueda acceder a consultar el disparador de una tendencia. Para poder cumplir con este hito surge la necesidad de obtener los *tweets* asociados a una tendencia en el momento en que Twitter la reconoce como tal. Esto significa que es de gran importancia poder analizar los *tweets* con su estado del momento en que se disparó la tendencia. El estado refiere a sus metadatos, como por ejemplo su cantidad de favoritos, *retweets*, citas, etc., ya que en esta información se basan los modelos construidos.

Con esta finalidad, se implementa un demonio que obtiene las tendencias de Twitter en Uruguay junto con sus *tweets* asociados en el momento en que la tendencia surge, y almacena esta información en una base de datos.

5.1.1. Obtención de tendencias

Para dar soporte a este proceso se define una colección de *tweets* y una lista de tendencias en una base de datos MongoDB. Dada la limitante técnica al obtener las tendencias de Uruguay que se describe en el tercer desafío de la sección 4.1.3, se diseña un algoritmo en base a una técnica de *web scraping* que simula la interacción con la aplicación de Twitter, autenticando con un usuario creado específicamente para este proyecto, al cual según sus preferencias se le muestran las tendencias de Uruguay.

La lógica utilizada para obtener las tendencias del momento consiste en una consulta inicial en la cual se obtienen las 15 primeras tendencias en Uruguay. Un minuto después se vuelve a realizar la consulta y las tendencias nuevas que aparecen respecto a la consulta anterior se almacenan en memoria. Paso seguido, para cada tendencia nueva detectada se realiza la búsqueda de *tweets* que

la mencionan, utilizando una heurística que se explica más adelante. Una vez terminadas estas búsquedas, el demonio deja de procesar durante un minuto y vuelve a consultar la lista actual de tendencias. Se repite el proceso de forma indefinida, buscando las tendencias nuevas respecto al conjunto de tendencias que se obtuvo en la consulta inmediatamente anterior.

Este método permite detectar las tendencias a medida que van surgiendo, cumpliendo con el objetivo de reducir la cantidad de información a procesar por cada una de ellas, ya que cuanto antes se reconozca, menos *tweets* se encontrarán en su búsqueda. Esto disminuye la cantidad de *tweets* innecesarios que se guardan en la base de datos. Como el tema ya se convirtió en tendencia, la hora de creación del *tweet* que la disparó debe ser menor que la hora de detección de esta, por lo cual al obtener los *tweets* en el momento en que la tendencia surge, se asegura la presencia del disparador en el conjunto de *tweets* obtenido.

A continuación se muestra un extracto del resultado de la ejecución de este algoritmo en una prueba comenzada el jueves 3 de diciembre a las 20:18hs y culminada el viernes 4 de diciembre a las 21:06hs (1 día de ejecución). En cada iteración se muestra la fecha y hora de consulta seguido de la lista de tendencias obtenidas, ordenadas según la relevancia que Twitter les otorga de acuerdo a su nivel de repercusión en tiempo real. Particularmente, en esta prueba se obtienen las primeras 20 tendencias en cada consulta con el fin de validar en qué posiciones de la lista aparecen las tendencias nuevas, las cuales se muestran en el conjunto *difference*.

1) Se consulta la lista de tendencias por primera vez y posterior a esta el conjunto *difference* permanece vacío como se muestra en la Figura 5.1.

```
Jue 12/03, 20:30:11 - [Damiani, Día Internacional, Ruglio, Ramón Méndez,
 T3 ALBUM OUT NOW, #NuestroNombreIsOut, Tini, Salgado, GACH, Cutcsa, Benfica,
 #WonderAtMidnight, Piquerez, Bloomberg, #LaBajada, Darwin, #dóndeestámili,
 Peñarol y Nacional, Miranda, Día Nacional del Candombe]

difference = []
```

Figura 5.1: Tendencias obtenidas de Twitter: Paso 1

2) En la Figura 5.2 se muestra la siguiente consulta realizada 41 segundos después de la anterior, donde aparece "Valverde" como tendencia nueva en la posición 17. Una vez terminada la búsqueda de *tweets* para esta tendencia, se vuelven a consultar.

```
Jue 12/03, 20:30:52 - [Damiani, Día Internacional, Ramón Méndez,
 T3 ALBUM OUT NOW, #NuestroNombreIsOut, Tini, Salgado, GACH, Cutcsa, Benfica,
 #WonderAtMidnight, Piquerez, Bloomberg, #LaBajada, Darwin, #dóndeestámili,
 Valverde, Peñarol y Nacional, Miranda, Día Nacional del Candombe]

difference = [Valverde] - aparece en lugar 17
```

Figura 5.2: Tendencias obtenidas de Twitter: Paso 2

3) En la siguiente iteración, que se muestra en la Figura 5.3, aparece "Radi" como tendencia nueva en la posición 20 de la lista y se procede a buscar sus *tweets* asociados.

4) En la Figura 5.4, no se detectan tendencias nuevas ya que la lista obtenida contiene las mismas tendencias que la consulta anterior.

5) En la Figura 5.5, aparece "Felicitaciones" como tendencia nueva en la posición 20 de la lista.

Analizando el resultado completo de esta prueba, se concluye que en general la parte baja de la lista cambia con mayor asiduidad que la parte alta. Este resultado es esperable, ya que se

```
Jue 12/03, 20:31:31 - [Damiani, Día Internacional, Ramón Méndez, T3 ALBUM OUT NOW,
 #NuestroNombreIsOut, Tini, Salgado, GACH, Cutcsa, Benfica, #WonderAtMidnight,
 Piquerez, Bloomberg, #LaBajada, Darwin, #dóndeestámili, Peñarol y Nacional,
 Miranda, Día Nacional del Candombe, Radi]

difference = [Radi] - aparece en lugar 20
```

Figura 5.3: Tendencias obtenidas de Twitter: Paso 3

```
Jue 12/03, 20:32:12 - [Damiani, Día Internacional, Ramón Méndez, T3 ALBUM OUT NOW,
 #NuestroNombreIsOut, Tini, Salgado, GACH, Cutcsa, Benfica, #WonderAtMidnight,
 Piquerez, Bloomberg, #LaBajada, Darwin, #dóndeestámili, Peñarol y Nacional,
 Miranda, Día Nacional del Candombe, Radi]

difference = []
```

Figura 5.4: Tendencias obtenidas de Twitter: Paso 4

```
Jue 12/03, 20:32:55 - [Damiani, Día Internacional, Ramón Méndez, T3 ALBUM OUT NOW,
 #NuestroNombreIsOut, Tini, Salgado, Cutcsa, Benfica, #WonderAtMidnight, Piquerez,
 Bloomberg, #LaBajada, Darwin, #dóndeestámili, Peñarol y Nacional, Miranda,
 Día Nacional del Candombe, Radi, Felicitaciones]

difference = [Felicitaciones] - aparece en lugar 20
```

Figura 5.5: Tendencias obtenidas de Twitter: Paso 5

requiere una repercusión importante en muy poco tiempo para que un tema tome tal relevancia que le permita aparecer por primera vez en una parte alta de la lista. Los cambios en el orden de las tendencias en esta lista no son relevantes para el algoritmo, ya que su objetivo es capturar la primera vez que cada tendencia aparece en la lista.

No obstante, a veces puede ocurrir que aparezca una tendencia nueva en un lugar intermedio de la lista como se muestra en el ejemplo de la Figura 5.6.

```
Jue 12/03, 20:32:55 - [Damiani, Día Internacional, Ramón Méndez, T3 ALBUM OUT NOW,
 #NuestroNombreIsOut, Tini, Salgado, Cutcsa, Benfica, #WonderAtMidnight, Piquerez,
 Bloomberg, #LaBajada, Darwin, #dóndeestámili, Peñarol y Nacional, Miranda,
 Día Nacional del Candombe, Radi, Felicitaciones]

difference = []

Jue 12/03, 20:33:35 - [Damiani, Día Internacional, Ramón Méndez, T3 ALBUM OUT NOW,
 #NuestroNombreIsOut, Tini, Salgado, GACH, Cutcsa, Benfica, #WonderAtMidnight,
 Piquerez, Bloomberg, #LaBajada, Darwin, #dóndeestámili, Peñarol y Nacional,
 Miranda, Día Nacional del Candombe, Radi]

difference = [GACH] - aparece en lugar 8
```

Figura 5.6: Ejemplo de tendencia que surge con gran repercusión

Al obtener tendencias se observa que algunas de estas suelen aparecer y desaparecer de la lista obtenida. Al detectar una tendencia nueva, no siempre significa que acaba de surgir, ya que su nivel de repercusión puede haber oscilado en las últimas horas. Como contingencia a este fenómeno, al detectar una tendencia nueva se verifica en la base de datos si ya se buscaron sus *tweets* en las últimas 12 horas. En caso afirmativo se asume, y este es un criterio adoptado por el equipo del proyecto, que la tendencia es la misma que ya se buscó antes y no se vuelven a buscar sus *tweets*, ya que los candidatos a disparadores ya fueron obtenidos. Por el contrario, si la tendencia no fue detectada en las últimas 12 horas se entiende que la misma volvió a surgir, lo que indica que pue-

de existir un nuevo *tweet* disparador. En este caso, se vuelven a buscar los *tweets* que la mencionan.

5.1.2. Obtención de *tweets*

Para obtener los *tweets* asociados a las tendencias se utiliza la versión v1.1 de la *API* de Twitter, la cual mediante su método *GET search/tweets* [10] permite obtener *tweets* públicos de hasta una semana hacia atrás que contengan determinado texto. Se implementa un algoritmo que dada una tendencia obtiene sus *tweets* asociados y los guarda en la base de datos.

Antes de comenzar la búsqueda, se verifica que la tendencia no esté incluida en una lista predefinida de *stopwords*. Esta lista se obtuvo de un sitio de herramientas para procesar texto llamado CountWordsFree [11]. A la lista de *stopwords* se agregan algunos adverbios y adjetivos que usualmente aparecen en Twitter, tales como: "lamentablemente", "obviamente", "impresionante", "imponente", "insólito", "definitivamente". En varias ocasiones se observa que este tipo de expresiones se vuelven tendencia con una gran cantidad de *tweets* asociados que refieren a temas muy diversos. Es importante destacar que por más que se defina una lista de *stopwords*, es difícil contemplar en ella todas las expresiones posibles. La tendencia "Felicitaciones", mencionada en la sección 5.1.1, es un ejemplo de una tendencia que carece de contenido semántico propio, pero que igualmente se puede procesar.

Si la tendencia no es un elemento de la lista de *stopwords*, se procede con las consultas a la *API* de Twitter para obtener sus *tweets*. Dado que la cantidad máxima de *tweets* que devuelve por consulta es de 100, se realizan las consultas en un bucle mientras haya *tweets* para devolver.

La invocación se realiza con los parámetros de entrada que se muestran en la Tabla 5.1.

Parámetro	**Valor**	**Descripción**
q	tendencia	Se asigna el texto que deben contener los *tweets* a buscar.
tweet mode	*extended*	Se indica que se debe devolver el texto completo de los *tweets*.
lang	*es*	Se buscan *tweets* del idioma español.
resultType	*recent*	Este parámetro indica si se deben buscar *tweets* recientes, *tweets* populares, o una mezcla de ambas. Como el objetivo del proyecto es determinar la relevancia y popularidad de un *tweet* candidato a ser disparador de una tendencia, se buscan los más recientes.
count	100	Cantidad de *tweets* a buscar (máximo 100).
max id	último tweet id recibido	Se indica el tweet id del último recibido en la consulta anterior para que no se devuelvan *tweets* repetidos al volver a buscar.

Tabla 5.1: Parámetros utilizados del método search de la Twitter *API*

La ventana de conexión que establece la versión 1.1 estándar de la *API* de Twitter es de 450 consultas cada 15 minutos. Una vez consumida esta cantidad de consultas la ventana de conexión se cierra para la aplicación autenticada. Es por esto que se registran dos aplicaciones distintas en el sitio de Twitter para desarrolladores [10], cada una con sus claves de acceso y autenticación. Esto permite poder cambiar de claves de forma dinámica una vez que expira la ventana de conexión

de alguna de ellas y seguir realizando consultas con las claves de acceso y autenticación de la aplicación que estaba inactiva.

De forma de evitar el procesamiento secuencial de obtención de *tweets*, para cada nueva tendencia a procesar se crea un nuevo hilo de ejecución, utilizando la librería *Threading* de *Python*. Esto es posible, ya que el proceso de búsqueda y persistencia de *tweets* para cada tendencia es independiente. Una vez terminados todos los hilos se procede a la espera de un minuto para volver a ejecutar la consulta de tendencias mediante el *web scraping* mencionado en la sección 5.1.1.

Una alternativa evaluada fue la de no esperar a que terminen los hilos para que el demonio se detenga durante un minuto y vuelva a buscar tendencias. Dado que la búsqueda de *tweets* puede requerir un tiempo prolongado, se descarta esta opción para evitar un eventual decaimiento en la eficiencia del demonio producto de la ejecución concurrente de una gran cantidad de hilos.

5.1.3. Condiciones de parada al buscar *tweets* de una tendencia

Dada la cantidad de *tweets* que se pueden obtener al realizar una consulta con filtro de búsqueda por texto, se definen condiciones de parada para evitar obtener información tanto innecesaria como excesiva.

1) Antigüedad de tweets por tendencia

Al buscar *tweets* de una tendencia, sólo se obtienen aquellos cuyo horario de creación esté dentro del horario actual de búsqueda y 12 horas hacia atrás. Este parámetro se utiliza para reconocer que una tendencia que ya existe en nuestra base de datos se debe considerar como nueva. Esto permite acotar la búsqueda ya que la *API* de Twitter ofrece *tweets* de hasta una semana hacia atrás, lo cual podría generar una gran cantidad de documentos innecesarios en nuestra base de datos. Dentro de los valores experimentados para este parámetro, este es el que mejor se ajusta.

2) Tiempo máximo de búsqueda por tendencia

Se define un parámetro de configuración con el tiempo máximo de búsqueda por tendencia (15 minutos), para evitar obtener una cantidad excesiva de *tweets* y que el tiempo de búsqueda se prolongue de tal manera que al volver a obtener el conjunto de tendencias actuales se detecten muchas tendencias nuevas, o que eventualmente surja alguna nueva tendencia y no pueda detectarse producto de que no se volvió a realizar la consulta. Este valor fue seleccionado en base a la experiencia adquirida de las pruebas realizadas.

3) Cantidad máxima de recolectados

Se define un parámetro de configuración con la cantidad de *tweets* suficientes por tendencia, para terminar con la búsqueda llegado el caso. Se utiliza una cantidad máxima de 30.000 recolectados, valor que surge producto de haber analizado la cantidad de *tweets* obtenidos al procesar tendencias de dimensión mundial. Sin importar la dimensión de la tendencia, se entiende que buscar el disparador, como máximo, en esta cantidad de *tweets* es una tarea realizable, que devuelve un disparador representativo de la tendencia.

5.2. Modelo clasificador temático de *tweets*

Este modelo se encarga de determinar para cada *tweet* asociado a una tendencia si refiere a su tema principal o no. Su propósito es depurar el conjunto de datos que posteriormente será utilizado para detectar el disparador de una tendencia, según se explica más adelante en la sección 5.3. Esto permite descartar los *tweets* que por no referirse al tema principal de la tendencia, no son candidatos a disparadores. Una consecuencia directa de este filtrado es la reducción en la cantidad de *tweets* que se deben procesar al momento de predecir el disparador de una tendencia, lo cual hace más rápida y eficiente su predicción.

Este clasificador surge producto de que el mecanismo de búsqueda de *tweets*, explicado en la sección 5.1.2, obtiene todos los *tweets* públicos que contienen determinada expresión. Por ejemplo, esto hace que para la tendencia "Uruguay" se obtengan *tweets* que refieren a diversos temas como la selección de fútbol, las elecciones presidenciales, la llegada de las vacunas, o alguna otra noticia relevante que refiera a este país. Usualmente, existen *tweets* que refieren a todos ellos en un mismo día. Este clasificador intenta estimar cuáles son los temas más importantes que provocaron la tendencia, para poder descartar los *tweets* que no refieren a ellos, ya que no serán tenidos en cuenta al momento de predecir su disparador.

Otra de las razones que motiva la construcción de este modelo es la existencia de tendencias globales, generadas por una gran cantidad de *tweets* publicados por usuarios de distintas partes del mundo. Esto genera que junto con los *tweets* de las tendencias de Uruguay aparezcan muchos que mencionan la tendencia pero que refieren a temas muy diferentes a los que están mencionando la mayoría de los usuarios en Uruguay. Utilizando este clasificador se intenta filtrar estos *tweets*, de modo de evitar procesar grandes cantidades de información innecesaria.

5.2.1. Bag of Words para una tendencia

Para estimar los temas principales que convierten a una expresión en tendencia se calcula su *Bag of Words (BOW)*. Del conjunto total de *tweets* de cada tendencia, solamente se toman en cuenta aquellos cuya ubicación o ubicación de su autor sea Uruguay. De esta forma, el *BOW* calculado para cada tendencia contiene las palabras que más aparecen en sus *tweets* publicados en Uruguay, lo cual ayuda a determinar qué temas están teniendo repercusión en este país en tiempo real. Es importante aclarar que en este punto no es relevante conocer cuales son exactamente los temas populares que están detrás de una tendencia. Basta con que el BOW construido incluya las palabras más utilizadas en el entorno de la tendencia, para considerar que cuantas más palabras del BOW contenga un *tweet*, mayor probabilidad tiene de referir a los temas principales de la tendencia.

Al día de hoy, Twitter sólo admite añadir la ubicación actual al crear un *tweet* que contiene un archivo multimedia, como por ejemplo una foto. Por este motivo, la cantidad de *tweets* con ubicación asociada es muy baja, lo que impide utilizar una cantidad mayor de datos para calcular el *BOW*. También existe una gran cantidad de usuarios, que en su información personal no tienen cargada su ubicación, a diferencia del ejemplo que se muestra en la Figura 5.7.

Construcción del BOW

Para cada *tweet* de cada tendencia detectada, se aplica el proceso de tokenización utilizando el modelo *es_core_news_sm* de la librería Spacy [15]. Antes de comenzar con el filtrado de tokens, se realiza un procesamiento de normalización a cada uno de ellos:

1. Se convierte el token a minúscula.

Figura 5.7: Ubicación geográfica en perfil de usuario

2. Se normaliza el token utilizando la librería *Unicodedata* [14] de Python para quitar tildes y cambiar letras "ñ" por "n", de forma de unificar palabras que pueden aparecer escritas de forma diferente pero su contenido semántico es el mismo.

3. Si el token termina en signo de puntuación se le quita el signo.

Luego, se agregan a la bolsa inicial de palabras los tokens junto con su cantidad de ocurrencias, que cumplen con las siguientes condiciones:

1. No pertenecer a la lista de *stopwords* definida en la sección 5.1.2. Esta lista contiene también la tendencia, y las palabras que conforman la tendencia. Por ejemplo, si la tendencia es "feliz aniversario", no se agrega al *BOW* las palabras "feliz" ni "aniversario".

2. Sólo se agregan Nombres Propios y Sustantivos.

3. No contener "@" ni "http".

4. El largo del token debe ser mayor que 1 para descartar letras solas, números del 0 al 9, signos de puntuación y emoticones.

Con esta bolsa de palabras se determina el *BOW* final que contendrá las palabras con más ocurrencias. El *BOW* final contiene al menos 20 palabras, y cada una de ellas tiene al menos 2 ocurrencias. Una vez calculado este *BOW*, se divide la cantidad de ocurrencias de cada palabra por la cantidad total de ocurrencias de todas las palabras del conjunto. De esta forma, cada palabra tendrá asociado un "coeficiente peso" que representa su proporción de importancia para la tendencia.

A modo de ejemplo, se presenta en la Tabla 5.2 el cálculo del *BOW* inicial para la tendencia "Shopping" a partir de los *tweets* obtenidos, teniendo en cuenta solamente los tokens de los *tweets* que se tiene certeza que son de Uruguay. En ella muestran solamente las 40 palabras con más ocurrencias para simplificar el ejemplo.

Una vez calculado el *BOW* inicial, se itera sobre él para formar el *BOW* final, quedándonos en cada iteración con los tokens que tienen más ocurrencias hasta llegar a tener por lo menos 20 palabras. Para finalizar, se calcula el peso de cada token en el *BOW*, de forma que las palabras con mayor cantidad de ocurrencias tengan un peso mayor, ya que representan mejor al tema de la tendencia. El *BOW* final para este ejemplo se muestra en la Tabla 5.3.

tokens	ocurrencias	tokens	ocurrencias
compras	296	**escuelas**	42
tweet	290	**casinos**	41
atención	289	**escuela**	36
sorteo	287	**aglomeraciones**	34
2000	287	**gimnasio**	21
vale	287	**gobierno**	20
obsequio	287	**fuentes**	19
seguinos	287	**marchas**	19
26/3	287	**pacha**	18
querias	230	**pascuas**	18
vayas	230	**canastas**	18
bares	131	**productos**	18
personas	131	**garoto**	18
medidas	130	**dale**	18
libertad	127	**publicacion**	18
seguro	121	**salud**	18
paro	118	**ninos**	17
sepan	115	**clases**	17
casino	46	**iglesias**	16
gente	43	**empleados**	16

Tabla 5.2: Tokens del *BOW* inicial para la tendencia "Shopping"

Dado el método utilizado para su construcción, la cantidad de palabras del *BOW* final para cada tendencia dependerá de características propias de esta. Por ejemplo, si se tienen muy pocos *tweets* de la tendencia, su *BOW* posiblemente no llegue a tener 20 palabras.

5.2.2. Atributos considerados

En esta sección se exponen los atributos o características de cada *tweet* analizadas para la construcción del modelo de clasificación.

1. **Atributos basados en el *tweet***

   - **Si el *tweet* es de Uruguay o no:** Vale 1 si el *tweet* es de Uruguay. Se determina que un *tweet* es de Uruguay si su ubicación asociada o la de su autor es Uruguay.

2. **Atributos basados en la tendencia**

   A continuación se exponen los atributos asignados a cada *tweet*, que son obtenidos utilizando la información de la tendencia, de forma de agregarle al *tweet* características de la tendencia a la cual está relacionada.

   - **Proporción de *tweets* de Uruguay que contiene la tendencia:** Analizando la ubicación de los *tweets* de la tendencia, o ubicación de sus autores, se puede estimar qué porcentaje de ellos son de Uruguay. Este porcentaje es una característica de la tendencia

tokens	ocurrencias	peso
compras	296	0.073
tweet	290	0.072
atención	289	0.071
sorteo	287	0.071
2000	287	0.071
vale	287	0.071
obsequio	287	0.071
seguinos	287	0.071
26/3	287	0.071
querias	230	0.057
vayas	230	0.057
bares	131	0.032
personas	131	0.032
medidas	130	0.032
libertad	127	0.031
seguro	121	0.030
paro	118	0.029
sepan	115	0.028
casino	46	0.011
gente	43	0.010

Tabla 5.3: Tokens del *BOW* final para la tendencia de ejemplo

que se incluye en la representación de cada *tweet*. Este atributo permite estimar qué tan uruguaya es la tendencia.

3. **Atributos basados en el *tweet* y la tendencia**

   A continuación se exponen los atributos que combinan información del *tweet* y de su tendencia asociada.

   - **Cantidad de tokens del *tweet* en el *BOW*:** Mientras más palabras tenga el *tweet* en el *BOW* de la tendencia, mayor probabilidad tiene de referirse a su tema principal.

   - **Peso de los tokens del *tweet* en el *BOW*:** Este atributo hace valer el peso de las palabras del *BOW*. La idea es asignar un valor mayor a un *tweet* en este atributo cuanto más ocurrencias tengan en el *BOW* sus palabras. El peso de un *tweet* se calcula sumando el peso de cada uno de sus tokens en el *BOW*.

   - **Word embeddings:** Para agregar información de contexto al modelo se calcula el vector de *word embeddings* tanto para cada *tweet* como para la tendencia. Esto se basa en la hipótesis de que si el tema del *tweet* y la tendencia son similares, entonces sus representaciones vectoriales deberían estar cerca en el espacio de vectores. Para esto, se agregan 600 atributos que corresponden a las 300 entradas del vector de *word embeddings* del *tweet* y las 300 entradas análogas que representan a la tendencia. Para obtener estos

vectores se utiliza el modelo de *FastText*[16]. Ambos vectores se calculan de la siguiente manera:

a) El vector que representa el tema del *tweet*: Se obtiene la representación vectorial de cada token que aparece en el *tweet*. Luego, se suman cada uno de estos vectores resultando un único vector suma.

b) El vector que representa al tema de la tendencia: Este vector se obtiene utilizando el modelo mencionado, pasando como parámetro la tendencia. Esto aplica para tendencias formadas por una o varias palabras.

- **Distancia del *tweet* a la tendencia (*embeddings*):** Este atributo refleja qué tan cerca está semánticamente el *tweet* del tema principal de la tendencia de acuerdo a las palabras que contiene. Para esto se utiliza la técnica de *word embeddings* como se explica a continuación.

  Para estimar la distancia del *tweet* al tema principal es necesario calcular dos valores:

  a) El valor que representa al tema del *tweet*: Se obtiene la representación vectorial de cada token que aparece en el *tweet*. Luego, se suman cada uno de estos vectores, resultando un único vector suma. Finalmente, el valor numérico que representa la temática del *tweet* es la suma de las entradas del vector.

  b) El valor que representa al tema de la tendencia: Para esto se calcula el vector representante de cada *tweet* de Uruguay. Luego, se suman todos los vectores de estos *tweets* y se divide el vector final por la cantidad total de *tweets* de Uruguay que tenga la tendencia, calculando así el promedio de cada entrada del vector. Una vez calculado este vector, se suman todas las entradas del vector obteniendo un número escalar que representa el tema de la tendencia.

  Finalmente, la distancia del tema del *tweet* al tema principal de la tendencia se calcula restando el valor que representa el tema de la tendencia menos el valor que representa el tema del *tweet*. Cuanto menor sea el valor de este atributo significa que el tema del *tweet* es cercano al tema de la tendencia.

5.2.3. Tipos de tendencias

A continuación se exponen los tres tipos de tendencias que se distinguen en este proyecto, los cuales se deducen en base a los casos enfrentados en este trabajo. A grandes rasgos, cada tendencia de Uruguay en Twitter podría ser clasificada en uno de los siguientes grupos:

1) Tendencias globales: En todo el mundo se están generando masivamente *tweets* sobre el tema, incluyendo en Uruguay. Puede referir a un tema específico, como por ejemplo el nombre de un actor o músico famoso. En otros casos, no se refiere a un tema específico, como puede ser la tendencia "China", que si bien se refiere al país, los *tweets* podrían contener diversos temas que lo involucran. La Figura 5.8 muestra un ejemplo de la tendencia global "Hamilton".

Figura 5.8: Ejemplo de tendencia global

2) Tendencias locales: Son tendencias que refieren a un tema con repercusión únicamente en Uruguay y que fuera del país no son tendencia. La mayoría de los *tweets* son de usuarios con ubicación en Uruguay y refieren al tema específico. De acuerdo a su repercusión, puede variar la dimensión de la tendencia en cuanto a su cantidad total de *tweets*. En la Figura 5.9 se muestra una tendencia específica de Uruguay.

3) Tendencias locales diferentes a tendencia global: Son tendencias que refieren a un tema específico de Uruguay, pero que la expresión que la representa es usada a nivel global con otra acepción. En este caso, del conjunto total de *tweets* de la tendencia, un pequeño subconjunto refiere al tema que está generando repercusión en Uruguay. El resto de los *tweets*, refieren en su mayoría a un tema que está teniendo repercusión a nivel mundial y que es diferente al que se habla en Uruguay. En la Figura 5.10 se muestra un ejemplo, donde el primer *tweet* refiere al equipo de fútbol de Uruguay y el segundo al equipo de fútbol inglés que lleva el mismo nombre.

5.2.4. Conjunto de datos

Los *tweets* de ejemplo se dividen en los siguientes conjuntos: entrenamiento, validación y testeo. Para definir estos conjuntos se utilizan *tweets* de distintas tendencias de Uruguay, de forma que estas no se repitan en cada uno de ellos. Este criterio se adopta para establecer un escenario más exigente, ya que si se entrenara y validara el modelo con *tweets* de tendencias iguales, los resultados serían mejores producto de las características similares que mantienen los *tweets* de una misma tendencia. Además, se obtendría una evaluación del modelo no tan real, ya que el objetivo es conocer su desempeño al utilizarlo para evaluar tendencias no conocidas.

Figura 5.9: Ejemplo de tendencia uruguaya

Figura 5.10: Ejemplo de tendencia global y uruguaya a la vez

Por ejemplo, supongamos que se asignan las tendencias "Antel" y "Suarez" al conjunto de entrenamiento, las tendencias "Isla de Flores" y "Argentina" al conjunto de validación, y las tendencias "Parque Viera" y "Shopping" al conjunto de testeo. Esto implica que los conjuntos de *tweets* son disjuntos, ya que no se incluyen *tweets* de las tendencias de uno de ellos en los dos restantes. Se aclara que no se toman todos los *tweets* de las tendencias seleccionadas, sino que se obtiene una muestra al azar de cada una de ellas para contar con datos balanceados, de forma que cada tendencia aporta a su conjunto asignado una cantidad similar de *tweets*. Además, se tiene en cuenta que exista también un balance en cuanto a la cantidad de ejemplos de la clase positiva y negativa que aporta cada tendencia.

No se incluyen tendencias que empiecen con *hashtag*, ya que este tipo de tendencias son creadas

con un fin específico y no genera problema reconocer que todos sus *tweets* refieren al tema principal.

La cantidad de tendencias y *tweets* que se asignan a cada conjunto se muestra en la tabla 5.4.

Conjunto	**Total tendencias**	**Total tweets**	**% corpus**
Entrenamiento	14	1400	70
Validación	6	200	10
Testeo	8	400	20

Tabla 5.4: División del corpus de datos en Regresión Logística

5.2.5. Entrenamiento

Para entrenar este modelo utilizamos un algoritmo de Regresión Logística. Para cada tendencia del conjunto de datos definido, se genera un archivo en formato *CSV* que contiene sus *tweets* asociados, uno por línea. Cada línea está formada por el identificador del objeto *tweet* en MongoDB, su texto, y los atributos mencionados en la sección 5.2.2. Para anotar el conjunto de datos se etiqueta manualmente cada línea de cada archivo agregando un campo con un 1 si el *tweet* refiere al tema principal de la tendencia, o con un 0 si no. Una vez realizada esta tarea para cada archivo, se ejecuta un script que carga en memoria el conjunto de entrenamiento con la información de todos los archivos etiquetados. Lo mismo ocurre para el conjunto de validación y testeo, cada uno con sus tendencias correspondientes. En la Tabla 5.5 se muestran tres *tweets* de la tendencia "Shopping", utilizando solamente dos atributos a modo de ejemplificar el formato de estos archivos. El tercer *tweet* se anota con el valor 1 producto del conocimiento del contexto, ya que la tendencia surgió producto de la apertura de los *shopping* en Uruguay durante la pandemia de COVID-19.

Texto del tweet	**Cantidad de tokens en BOW**	**Es de Uruguay?**	**Valor anotado**
Esta como para irse de shopping y desconocerse con la tarjeta no?	2	NO	0
Salir de shopping con mami no es facil nos gusta todooo jajaja	0	NO	0
Dejen de atacar a los shopping y a los comercios, la gente tiene que comer	4	SÍ	1

Tabla 5.5: Ejemplo de datos de entrenamiento para la tendencia "Shopping"

Para entrenar el modelo, se utiliza la clase *LogisticRegression* de la librería *Scikit-Learn* y se persiste utilizando la librería *Pickle*, la cual genera un archivo serializado *.pkl* con los parámetros aprendidos por el modelo entrenado.

5.3. Modelo de predicción de *tweet* disparador

Este modelo pretende resolver la problemática inicial planteada para el proyecto, que consiste en encontrar el o los *tweets* disparadores para determinada tendencia, para lo cual se utiliza una Red Neuronal *feed forward* entrenada de forma supervisada. En esta sección también se detallan los atributos analizados para entrenar este modelo y la normalización aplicada a cada uno de ellos.

5.3.1. Atributos considerados

Atributos basados en el *tweet*

A continuación se presentan los atributos utilizados para entrenar la Red Neuronal, los cuales se calculan a partir de los metadatos del *tweet*.

- **Diferencia en minutos del *tweet* a la tendencia:** Se calcula la diferencia en minutos entre la fecha y hora de creación del *tweet* y la fecha y hora de detección de la tendencia a la cual corresponde.
- **Interacciones:** Este atributo suma la cantidad de *retweets* y favoritos que tiene el *tweet* en cuestión, el cual fue obtenido en el momento que la tendencia fue detectada.

  En primera instancia este atributo iba a estar separado en dos, *retweets* por un lado y favoritos por otro. Se decide juntarlos ya que a efectos del funcionamiento de Twitter ambos realizan la misma acción de difusión. Un usuario que interactúe de alguna de estas maneras con un *tweet* va a replicarlo a sus seguidores.
- **Interacciones por minuto:** Es el cociente entre la cantidad de interacciones del *tweet* sobre los minutos de diferencia entre el momento de creación del *tweet* y el momento en que se detectó la tendencia.
- **Cantidad de *tweets* por minuto:** Es el cociente entre la cantidad de *tweets* asociados a la tendencia que fueron publicados posteriormente al *tweet* que se está analizando, dividido la diferencia en minutos entre el *tweet* y la tendencia. Cabe aclarar que únicamente se tienen en cuenta los *tweets* que se publicaron antes del momento de detectar la tendencia.
- **Cantidad de *tweets* en la siguiente hora:** Este valor representa la cantidad de *tweets* asociados a la tendencia posteriores al *tweet* que se está analizando, los cuales hayan sido publicados hasta una hora después de su publicación.

Atributos basados en el usuario

A continuación se presentan los atributos utilizados para entrenar la Red Neuronal, los cuales se calculan a partir de los metadatos del usuario que publica el *tweet*.

- **Medida r:** Este atributo intenta representar la influencia de un usuario de la siguiente manera [1]:

$$r = \frac{followers}{(followers + following)}$$

- **Followers:** Es la cantidad de seguidores que tiene el usuario que publicó el *tweet*.

- **Cantidad de listas a las que pertenece el usuario:** Es la cantidad de listas en las que ha sido agregado el usuario que publicó el *tweet*. Cuanto mayor sea la cantidad de listas que un usuario integra, mayor es la difusión que obtienen sus publicaciones, ya que a todos los seguidores de cada lista le son mostrados sus *tweets*, incluso a aquellos que no sigan a este usuario.

5.3.2. Normalización de atributos

Al momento de entrenar el modelo, luego de haber seleccionado los datos para cada conjunto, surge el problema de las diferentes magnitudes que presentan los *tweets* de cada tendencia. Un ejemplo de esto se puede ver en la Tabla 5.6

	Tendencia 1		**Tendencia 2**	
	Interacciones	**Followers**	**Interacciones**	**Followers**
Tweet 1	875	1251	3	3
Tweet 2	19	2200	98	20
Tweet 3	22	15	32	8
Tweet 4	122	168	13	0
Tweet 5	17	8	1	0

Tabla 5.6: *Tweets* de ejemplo y atributos sin escalar

En este caso, las tendencias no tienen magnitudes comparables, ya que el *tweet* que tiene más interacciones de la primer tendencia debería tener un valor comparable con el *tweet* que tiene más interacciones de la segunda tendencia, para que así el aprendizaje del modelo pueda ser utilizado en tendencias no vistas y que su predicción no dependa de sus dimensiones.

Por este motivo se decide escalar los valores de cada atributo a un valor relativo con respecto a los demás *tweets* de la tendencia. Es decir, se agrupan los *tweets* por tendencia y se escalan en conjunto.

Para cada atributo del modelo se determina el valor máximo que se alcanza en el conjunto de datos a procesar. Luego de esto, a cada atributo de cada *tweet* se lo divide por el valor máximo que corresponda. Esta manera de escalar realiza una transformación a cada atributo de forma individual, de manera que el valor absoluto máximo de cada uno de ellos en el conjunto valga 1. Además, no desplaza ni centra los datos por lo cual no destruye la escasez [13].

Para ejemplificar se retoma el ejemplo de la Tabla 5.6. En este caso, se toman los datos de la tendencia 1 y se analizan los máximos de cada atributo. Se puede ver que el máximo de interacciones es 875 y el máximo de *followers* es de 2200. El siguiente paso entonces es dividir cada entrada por el máximo correspondiente a su atributo.

Se realiza este mismo análisis con la tendencia 2 y los resultados se muestran en la Tabla 5.7.

De esta manera, se logra poder comparar *tweets* y tendencias con distintas magnitudes.

5.3.3. Conjunto de datos

Para obtener los datos a utilizar en la validación de este clasificador, se recolectan *tweets* y tendencias procesadas por el clasificador temático y se obtiene como resultado un conjunto de tendencias con los *tweets* asociados a ellas, que refieren a su tema principal.

Luego se dividen estos datos en los conjuntos de entrenamiento, validación y testeo, donde los *tweets* asociados a una tendencia están únicamente en uno de los conjuntos. Para ejemplificar, esto

	Tendencia 1		Tendencia 2	
	Interacciones	**Followers**	**Interacciones**	**Followers**
Tweet 1	1	0.569	0.03	0.15
Tweet 2	0.021	1	1	1
Tweet 3	0.025	0.007	0.327	0.4
Tweet 4	0.139	0.076	0.133	0
Tweet 5	0.019	0.004	0.01	0

Tabla 5.7: *Tweets* de ejemplos y atributos normalizados

quiere decir que si la tendencia "Cancillería" corresponde al conjunto de validación, todos los *tweets* que el clasificador temático predijo que referían a su tema principal van a pertenecer al conjunto de validación. Esto se resuelve de esta manera ya que el disparador que se intenta encontrar surge de analizar en conjunto todos los *tweets* asociados a una misma tendencia.

En la Tabla 5.8 se muestra como fueron divididos los 20722 *tweets* anotados.

Conjunto	**Cantidad de *tweets***	**Cantidad de tendencias**	**Cantidad de disparadores**
Entrenamiento	15512	18	26
Validación	910	7	8
Testeo	4345	9	11

Tabla 5.8: División del conjunto de datos en Red Neuronal

5.4. Demonio de predicción de *tweet* disparador

La implementación de este módulo surge como contingencia a la necesidad de mitigar el tiempo que puede tomar determinar cuál es el *tweet* disparador de una tendencia. Según las evaluaciones experimentales, esta tarea puede insumir desde unos pocos segundos hasta varios minutos, lo cual se vuelve inviable de calcular en tiempo real, una vez que el usuario desea ver el disparador de una tendencia en la web.

5.4.1. Descripción

Este proceso repetitivo se encarga de consultar cada cierto tiempo si existen tendencias nuevas descargadas a la base de datos que aún no hayan sido procesadas.

Para cada nueva tendencia sin procesar, utilizando el clasificador temático, se predice cuales son los *tweets* candidatos a disparadores, descartando los que no refieren al tema principal de la tendencia. Luego, este conjunto es procesado por la Red Neuronal, cuya salida es el conjunto de *tweets* identificados como disparadores.

Por último, a cada uno de estos disparadores se le define un atributo en la base de datos que lo distingue y se marca la tendencia como procesada. Además, a todos los *tweets* analizados en este proceso se les agrega los valores de sus atributos calculados y utilizados en la Red Neuronal, los cuales posteriormente son referenciados en la aplicación web para desplegar gráficas y presentar información.

5.4.2. Desafíos

1) Cuello de botella en accesos a base de datos

Este primer desafío está relacionado con el cuello de botella generado en la conexión a la base de datos. Esto sucede debido a la gran cantidad y complejidad de las búsquedas internas que se realizan en cada paso de este proceso, ya sea para calcular el *BOW*, calcular los atributos de la Regresión Logística o la Red Neuronal.

La resolución de este problema fue muy importante, ya que si el tiempo de procesamiento de tendencias pendientes es mayor que el tiempo en que el demonio de extracción recolecta nuevas, la cola de tendencias a procesar comenzaría a aumentar. Esto tiene como consecuencia que las nuevas tendencias no puedan ser visualizadas en el momento, sino tiempo después. Además, con el paso del tiempo, esta brecha de tiempo entre el momento actual y la fecha de la tendencia procesada más reciente sería cada vez mayor.

Como primera solución a este problema se evaluó la ejecución de hilos paralelos por tendencia. Esto no mejoró los tiempos de procesamiento e incluso en algunos casos, dependiendo de la magnitud de las tendencias que se procesaban en paralelo, la *performance* decaía drásticamente. Para solucionar este problema se cambiaron todas las búsquedas de tendencias y *tweets* que se realizaban sobre la base de datos por una sola búsqueda de mucha información y luego filtrados en memoria con estructuras de datos brindadas por el lenguaje de programación. Asimismo, existen consultas que no se pueden sustituir, o que su implementación en memoria es más compleja y menos eficiente, para las cuales se buscó una mejora de *performance* agregando índices a las colecciones en base de datos.

2) Espacio de almacenamiento en base de datos

El segundo desafío está relacionado la cantidad de espacio de almacenamiento utilizada en base de datos, producto de la cantidad de *tweets* obtenidos. Para esto, se procede con la eliminación de información que deja de ser necesaria luego de procesar cada tendencia, como los *tweets* que no fueron clasificados dentro de la temática de la tendencia, o atributos del *tweet* que no serán utilizados o mostrados en la web.

Al realizar el borrado automático de los *tweets* descartables, surge un problema propio de la base de datos utilizada, la cual reserva memoria cuando inserta los documentos, pero no la libera cuando estos son eliminados o modificados, de tal manera que su tamaño ocupado no disminuye. Para solucionar este problema, luego de terminar de procesar cada tendencia y eliminar los *tweets* que se descartan, el proceso ejecuta un comando en base de datos para compactarla y así liberar el espacio que queda en desuso.

5.5. Aplicación web

Se construye una aplicación web con el propósito de presentarle al usuario final los resultados del sistema de predicción implementado en este proyecto. La interfaz gráfica de esta aplicación se desarrolla en base al lenguaje NodeJS, más precisamente utilizando su librería React. Esta tecnología fue seleccionada para aprovechar la facilidad de su uso, la rapidez con la que se puede obtener buenos resultados y fundamentalmente la experiencia previa que tienen los integrantes del equipo. Por razones similares, su *backend* consta de una web API implementada en Python, expuesta mediante la librería FastAPI.

5.5.1. Descripción

Esta aplicación muestra al usuario final los disparadores de cada tendencia previamente procesada. Como muestra la Figura 5.11, se ofrecen filtros para facilitar la búsqueda de tendencias al usuario. Estos filtros permiten determinar una fecha desde, una fecha hasta, y un texto a coincidir con la tendencia o parte de ella.

Figura 5.11: Pantalla de inicio

Una vez ingresados los filtros, al acceder al botón *BUSCAR* se realiza una consulta a la base de datos para buscar las tendencias que los satisfacen. En este punto, es importante reafirmar que este aplicativo no se conecta con Twitter en tiempo real, sino que realiza la búsqueda en la base de datos que contiene las tendencias previamente recolectadas y procesadas para mejorar la experiencia del usuario. Estas son presentadas en pantalla junto con la fecha en la que fueron encontradas y la cantidad de *tweets* relacionados que se obtuvieron, tal como se muestra en la Figura 5.12.

En esta instancia se permite paginar entre los resultados obtenidos. Una vez elegida una tendencia se accede a sus datos seleccionando la fila correspondiente.

Esto redirige a una pantalla tal como se muestra en la Figura 5.13, donde se visualizan los *tweets* disparadores en la parte izquierda de esta. En el centro de la página se pueden encontrar gráficos de algunos atributos de los *tweets* de la tendencia. Cada punto en cada gráfico representa un *tweet*, el cual puede verse en la parte superior derecha al pasar el *mouse* sobre él. Haciendo clic en los *tweets* que se muestran en pantalla se puede acceder a Twitter para visualizarlos propiamente en esta aplicación.

Figura 5.12: Resultado de búsqueda

Figura 5.13: Disparador de la tendencia

Capítulo 6

Resultados obtenidos

En este capítulo se muestran los resultados que verifican el funcionamiento de los componentes de software que dan soporte a la aplicación construida. La sección 6.1 trata sobre el modelo de clasificación temática y la sección 6.2 sobre el modelo de predicción del disparador de una tendencia. En la sección 6.3 se presentan las pruebas realizadas y resultados obtenidos para el demonio de extracción de *tweets*. Esto mismo se realiza en la sección 6.4 con el demonio de predicción del *tweet* disparador.

6.1. Modelo clasificador temático de *tweets*

En esta sección se presentan los resultados obtenidos al entrenar el clasificador de temas de *tweets*, el cual utiliza una Regresión Logística. En la primera subsección se definen los hiperparámetros utilizados. La subsección 2 presenta la prueba de selección de atributos e hiperparámetros. La subsección 3 expone resultados más completos del modelo final, concluido en la subsección 2.

A continuación se detallan las tendencias utilizadas para los conjuntos de entrenamiento, validación y testeo.

- **Conjunto de entrenamiento:** *Liverpool, Suarez, Telemundo, China, Aldo, San Luis, Leandro, Rojo, Turismo, Diente, Stalin, Día Internacional, Antel, Umpierrez.*
- **Conjunto de validación:** *Mariano, Lito, Atlético, Isla de Flores, Loli, Argentina.*
- **Conjunto de testeo:** *Tony, Lucho, Senadora, Parque Viera, Shopping, Turquía, Salto, Bustillo.*

El conjunto de datos completo se incluye en el repositorio del proyecto [42].

6.1.1. Hiperparámetros del modelo

En esta sección se definen los hiperparámetros de la Regresión Logística, según la documentación oficial de *Sklearn* [13].

- *solver*: Algoritmo utilizado para resolver el problema de optimización. Puede tomar los siguientes valores:
  - 'lbfgs' (por defecto): Funciona bien para problemas de clasificación multiclase.
  - 'newton-cg': Funciona bien para problemas de clasificación multiclase.
  - 'liblinear': Buena opción para conjuntos de datos pequeños. Se limita solamente a resolver problemas de clasificación en esquemas *one-versus-rest*.
  - 'sag': Eficiente para grandes conjuntos de datos.
  - 'saga': Eficiente para grandes conjuntos de datos.
- *max_iter*: Número máximo de iteraciones que realizan los *solvers* en busca de la convergencia.
- *random_state*: Utilizado para barajar los datos de entrada cuando el parámetro *solver* es *'sag'*, *'saga'* o *'liblinear'* .
- C: Es el inverso del factor de regularización, el cual toma un valor positivo de punto flotante. Valores pequeños especifican una mayor regularización, penalizando más la función aprendida por el modelo.

Existen más parámetros, cuyos valores por defecto funcionaron correctamente para el modelo entrenado.

6.1.2. Selección de atributos e hiperparámetros

El objetivo de esta prueba es combinar los potenciales atributos e hiperparámetros del modelo y mostrar los resultados obtenidos para los conjuntos de entrenamiento y validación, de forma de concluir el modelo final. La medida que se toma como referencia de comparación es el *accuracy*.

Atributos utilizados

Se utilizan en total 6 atributos. Estos son:

1. Cantidad de *tokens* del *tweet* en el *BOW*.
2. Peso de los *tokens* del *tweet* en el *BOW*.
3. Proporción de *tweets* de Uruguay que contiene la tendencia.
4. Vector de *word embeddings* de 300 dimensiones que representa al tema del *tweet*. Este vector se calcula como la suma de los vectores de cada token del *tweet*. Además, se incluye un vector de *word embeddings*, también de 300 dimensiones, que representa al tema de la tendencia. Este atributo incluye en total 600 valores, los cuales son tratados de forma atómica.
5. Si el *tweet* es de Uruguay o no.
6. Distancia del *tweet* a la tendencia.

Hiperparámetros utilizados

A continuación se detallan los valores experimentados en cada hiperparámetro, de forma de poder determinar cuales son los que mejor se ajustan a cada uno de los modelos de prueba. Además de estos, fueron probados más valores, los cuales no se incluyen en esta prueba ya que no mejoraban los resultados obtenidos.

- *solver* = ['liblinear', 'lbfgs']
- C = [1.0, 0.75, 0.5, 0.25]
- *max_iter* = [20, 50, 100, 200, 300, 400]
- *random_state* = [None, 0, 1, 42]

Resultados

La Tabla 6.1 muestra un extracto de los resultados obtenidos al realizar esta prueba. Cada subconjunto de atributos define un modelo, para el cual se prueban todas las combinaciones posibles de hiperparámetros, de modo de encontrar los que generan el mejor resultado en el conjunto de validación. En esta tabla se muestra solamente la mejor versión de cada modelo entrenado. En la columna "Atributos" se indica qué atributos incluye el modelo, identificando a cada uno por su numeración del 1 al 6 presentado en esta sección. La tabla completa de resultados (A.1) se muestra en el Anexo I.

Modelo	Atributos	Training	Validation
25	**Sólo 1, 2 y 6**	**0.830**	**0.850**
42	Sólo 1 y 2	0.827	0.835
46	Sólo 1 y 6	0.830	0.835
50	Sólo 2 y 6	0.833	0.835
0	Todos	0.910	0.830
2	Sin 2	0.914	0.830
3	Sin 3	0.910	0.830
14	Sólo 1, 3, 4 y 6	0.907	0.830
16	Sólo 1, 4, 5 y 6	0.915	0.830
30	Sólo 1, 4 y 6	0.907	0.830

Tabla 6.1: Evaluación de atributos del modelo de Regresión Logística

Conclusiones

1. Se descarta el uso de los dos vectores de 300 dimensiones de *word embeddings*, contenidos en el atributo 4. Analizando esta tabla completa, se observa que estos vectores elevan a un 97 % los resultados en el conjunto de entrenamiento. Esto se debe a que el entrenamiento se realiza con cierto conjunto de tendencias que aportan información específica de contexto. Al evaluar este conjunto, como el modelo conoce *tweets* de estas tendencias predice su resultado de forma muy eficaz, a diferencia de los demás conjuntos donde este porcentaje decae, ya que utilizan *tweets* de tendencias no vistas por el conjunto de entrenamiento.

Una posible solución a este problema sería entrenar con una cantidad mucho mayor de tendencias, de forma de abarcar una mayor variedad de temas y contextos en el conjunto de entrenamiento, el cual fue entrenado apenas con 1400 *tweets* de 14 tendencias diferentes. Sin embargo, sería utópico intentar abarcar la totalidad de contextos posibles, debido a la diversidad de temas que se pueden convertir en tendencia.

2. Los atributos finales seleccionados son los del modelo número 25, compuesto por los atributos *1) Cantidad de tokens en el BOW*, *2) Peso de los tokens del tweet en el BOW* y *6) Distancia del tweet a la tendencia*. Este modelo es elegido como modelo final ya que obtiene el mejor resultado con un 85 % de *accuracy* en el conjunto de validación. Si bien los *word embeddings* no generan buenos resultados al agregarlos como atributos del modelo, sí funcionan al ser utilizados para calcular una medida de distancia relativa entre la tendencia y el *tweet*, tal como se realiza en el atributo 6, el cual sí se incluye en el modelo final.

6.1.3. Resultados

A continuación se muestra un resultado más completo del modelo final construido. Si bien el entrenamiento se ejecuta 5 veces para promediar los valores obtenidos, se observa que en cada ejecución los resultados generados son los mismos.

Para evaluar estos resultados se busca una solución más simple que resuelva este problema. Para esto, se define un algoritmo de línea base a efectos de comparar el rendimiento del modelo construido. En este caso, consiste en asignar a la clase positiva todos los *tweets* que contengan la expresión que es tendencia.

Atributos utilizados

1) Cantidad de *tokens* del *tweet* en el *BOW*.
2) Peso de los *tokens* del *tweet* en el *BOW*.
6) Distancia del *tweet* a la tendencia.

Hiperparámetros

- *solver* = 'liblinear'
- $C = 0.5$
- *max_iter* = 100
- *random_state* = *None*

Resultados

La Tabla 6.2 muestra la matriz de confusión para las predicciones del modelo en el conjunto de testeo.

	Positivo	**Negativo**
Positivo	156	10
Negativo	75	159

Tabla 6.2: Matriz de confusión obtenida en el conjunto de testeo

Por otra parte, la Tabla 6.3 muestra la matriz de confusión para la clasificación realizada por el algoritmo de línea base en este mismo conjunto.

	Positivo	**Negativo**
Positivo	231	169
Negativo	0	0

Tabla 6.3: Matriz de confusión obtenida por la línea base

Estos valores son utilizados para calcular las métricas con las cuales se evalúa el modelo, tal como se presentan en la sección 2.3.4. En el conjunto de testeo el *accuracy* es 0.79, el cual indica que el 79 % de las predicciones sobre este conjunto fueron acertadas. Esta medida se calcula realizando la división entre la suma de *tweets* verdaderos positivos y verdaderos negativos, sobre la cantidad de *tweets* totales del conjunto, en este caso $\frac{156+159}{400}$.

Para la clasificación realizada por la línea base, la fórmula $\frac{231+0}{400}$ determina un *accuracy* de 0.57, inferior al obtenido por el modelo construido.

La Tabla 6.4 muestra el resultado de las demás medidas calculadas, comparando el modelo de Regresión Logística con el algoritmo de línea base. La columna *support* indica la cantidad de elementos del conjunto.

	Precision	**Recall**	**F1-score**	**Accuracy**	**Support**
Regresión Logística	0.81	0.81	0.81	0.79	400
Línea base	0.57	1.00	0.72	0.57	400

Tabla 6.4: Resultados del modelo obtenidos en el conjunto de testeo

Conclusiones

Si bien el modelo presenta un porcentaje de acierto del 79 %, analizando la matriz de confusión se observa una cantidad considerable de falsos negativos. Los falsos negativos son *tweets* que refieren a los temas principales de la tendencia y sin embargo son clasificados en la clase negativa. La principal consecuencia de un valor alto de falsos negativos es una mayor probabilidad de pérdida de *tweets* que podrían ser candidatos a disparadores.

Por otra parte, la cantidad de falsos positivos son *tweets* que no refieren a los temas principales de la tendencia pero que el modelo los predijo en la clase positiva. Un valor bajo de falsos positivos disminuye la probabilidad de que el *tweet* elegido como disparador refiera a un tema que no es relevante para la tendencia.

Si bien la evaluación final muestra que se pierden 75 casos positivos sobre un total de 231 positivos totales (32 %) en el conjunto de testeo, un **79 % de *accuracy*** es un resultado aceptable

dada la complejidad del problema, y como ya se vio antes, la falta de trabajos similares con los que comparar.

6.2. Modelo de predicción de *tweet* disparador

En esta sección se presentan los resultados obtenidos al evaluar el modelo de predicción del *tweet* disparador, así como también algunas de las pruebas intermedias que llevan a esos resultados.

Vale aclarar que para determinar los valores de las medidas de evaluación en cada uno de los siguientes ejemplos se toma el promedio de los resultados obtenidos luego de entrenar 5 veces cada modelo.

6.2.1. Hiperparámetros del modelo

En esta sección se definen los hiperparámetros que se ajustaron en la Red Neuronal, según la documentación oficial de *Sklearn* [13].

- *hidden_layer_sizes*: Este parámetro se representa por una tupla donde el i-ésimo elemento representa el número de neuronas en la i-ésima capa oculta. Por defecto tiene una sola capa oculta con 100 neuronas.
- *activation*: Define la función de activación de las capas ocultas, pudiendo ser:
  - *Identity*: $f(x) = x$
  - *Logistic*: $f(x) = \frac{1}{(1+e^{-x})}$
  - *Relu*: $f(x) = max(0, x)$
- *solver*: Determina el algoritmo para realizar la optimización de los pesos entre capas. Dentro de los posibles solvers se tienen:
  - *lbfgs*: Es un optimizador de la familia de métodos quasi-Newton.
  - *sgd*: Se refiere al descenso de gradiente estocástico.
  - *adam*: Es un optimizador estocástico basado en gradientes propuesto por Kingma, Diederik y Jimmy Ba.
- *alpha*: Parámetro de penalización L2 (término de regularización).
- *learning_rate_init*: La tasa de aprendizaje inicial. Controla el tamaño del paso al actualizar los pesos. Este parámetro solo tiene efecto si el solver es *adam* o *sgd*
- *max_iter*: Número máximo de iteraciones. El solver iterará hasta un criterio de parada (determinado por *tol*) o este número de iteraciones.
- *tol*: Tolerancia para la optimización. Cuando la pérdida no mejora en al menos *tol* para *n_iter_no_change* iteraciones consecutivas se detiene el entrenamiento.
- *n_iter_no_change*: Número máximo de iteraciones para no cumplir con la mejora *tol*. Solo es efectivo cuando el solver es *sgd* o *adam*.

Existen más parámetros cuyos valores por defecto funcionan correctamente para la mejor solución del modelo.

6.2.2. Selección de atributos

El objetivo de esta sección es mostrar cómo interactúan entre sí los atributos definidos para este modelo y el valor que aportan a la solución. Como medida de evaluación se tiene en cuenta el *f1-score*.

Previo a realizar esta prueba, se hace un análisis de los hiperparámetros detallados en la sección 6.2.1 y se llega a un conjunto de los mejores modelos posibles, donde todos ellos son entrenados con *adam*.

Tendencias utilizadas

Conjunto de entrenamiento: *Robert, ASSE, Liverpool, FUBB, #UruguaySeVacuna, Bustillo, Alfredo Zitarrosa, #LaHoraCovid, FING, MTOP, David Terans, Carrasco_1, Carrasco_2, Maru Botana_1, Maru Botana_2, Riquelme, #aplaudimosaJorge, Zitarrosa.*

Conjunto de validación: *Cancillería, Quiroga, Herrera, Ana Inés, Umbro, #LaLetraChica, Morabito.*

Conjunto de testeo: *Tony, Senadora, #HayOrdenDeNoAflojar, Parque Viera, Shopping, Turquía, Cabildo Abierto, GACH, Salto.*

El conjunto de datos completo se incluye en el repositorio del proyecto [42].

Atributos utilizados

Los atributos a evaluar son los siguientes:

1. Tiempo entre el *tweet* y el descubrimiento de la tendencia.
2. Cantidad de interacciones.
3. Cantidad de interacciones por minuto.
4. Medida *r*.
5. *Followers* del usuario.
6. Cantidad de listas a las que pertenece el usuario.
7. Cantidad de *tweets* por minuto.
8. Cantidad de *tweets* en la siguiente hora.

Modelos candidatos

En esta sección se presenta el conjunto de modelos candidatos de Redes Neuronales entrenados con *adam*.

La Tabla 6.5 presenta los modelos evaluados y sus parámetros asignados.

Modelo	hidden_layer_sizes	activation	learning_rate_init	alpha	tol	max_iter
1	(8, 4)	*relu*	0.001	1^{-5}	1^{-5}	400
2	(10, 4, 4)	*identity*	0.01	1^{-5}	1^{-5}	400
3	(10, 8, 2)	*identity*	0.01	1^{-5}	1^{-5}	400
4	(10, 8, 6)	*tanh*	0.01	1^{-5}	1^{-5}	400
5	(8, 8, 8)	*tanh*	0.01	1^{-5}	1^{-5}	400

Tabla 6.5: Hiperparámetros de modelos a evaluar

Resultados

En este apartado se presentan los resultados obtenidos del análisis realizado para determinar cuáles son los atributos que más aportan a la solución y cuál es el modelo que mejor se ajusta al problema a resolver. En la columna "Atributos" se indica qué atributos incluye el modelo, identificando a cada uno por su numeración del 1 al 8 presentado en esta sección.

Como primer paso, se analiza cómo impacta quitar los atributos de a uno, comparándolos contra los modelos que utilizan todos ellos.

Modelo	Atributos	Training	Validation
1	Todos	0.839	0.894
2	Todos	0.766	0.867
3	Todos	0.776	0.848
4	Todos	0.880	0.821
5	Todos	0.881	0.886
1	Sin 1	0.818	0.868
2	Sin 1	0.725	0.869
3	Sin 1	0.712	0.888
4	Sin 1	0.862	0.921
5	Sin 1	0.846	0.904

Tabla 6.6: Extracto de pruebas realizadas para primer selección de atributos

En la Tabla 6.6, se presentan los mejores resultados obtenidos. Se puede ver que si se quita el atributo *"tiempo entre el tweet y el descubrimiento de la tendencia"*, la mayoría de los modelos mejoran sus resultados en el conjunto de validación. Para ver la tabla con los resultados completos de esta prueba, dirigirse al apéndice B, más específicamente a la Tabla B.1.

Se procede entonces a quitar este atributo y volver a analizar de los restantes, cuál es el que menos aporta.

En la Tabla 6.7 se presentan los valores obtenidos al evaluar los modelos sin los atributos *"tiempo entre el tweet y el descubrimiento de la tendencia"* y *"cantidad de listas a las que pertenece el usuario"*, donde se aprecia que estos resultados están dentro de un margen de $\pm 2\,\%$ respecto de los modelos evaluados sin el atributo *"tiempo entre el tweet y el descubrimiento de la tendencia"*, por lo que se puede concluir que este atributo no aporta gran valor.

Sumado a esto, se puede ver que al quitar cualquiera de los demás atributos se obtienen peores resultados, lo cual se puede consultar en la Tabla B.2 del apéndice B. Entonces, se procede a repetir el análisis sin los dos atributos mencionados para intentar inferir si se puede prescindir de otro atributo.

Modelo	Atributos	Training	Validation
1	Sin 1 y 6	0.818	0.865
2	Sin 1 y 6	0.728	0.88
3	Sin 1 y 6	0.715	0.871
4	Sin 1 y 6	0.859	0.931
5	Sin 1 y 6	0.837	0.898

Tabla 6.7: Extracto de pruebas realizadas para segunda selección de atributos

Modelo	Atributos	Training	Validation
1	Sin 1, 2 y 6	0.34	0.36
2	Sin 1, 2 y 6	0.566	0.595
3	Sin 1, 2 y 6	0.476	0.658
4	Sin 1, 2 y 6	0.643	0.69
5	Sin 1, 2 y 6	0.55	0.665
1	Sin 1, 3 y 6	0.807	0.824
2	Sin 1, 3 y 6	0.728	0.856
3	Sin 1, 3 y 6	0.699	0.857
4	Sin 1, 3 y 6	0.777	0.815
5	Sin 1, 3 y 6	0.838	0.811
1	Sin 1, 6 y 7	0.799	0.805
2	Sin 1, 6 y 7	0.743	0.827
3	Sin 1, 6 y 7	0.783	0.817
4	Sin 1, 6 y 7	0.843	0.917
5	Sin 1, 6 y 7	0.835	0.883

Tabla 6.8: Extracto de pruebas realizadas para tercer selección de atributos

La Tabla 6.8 muestra que al quitar tanto el atributo *"cantidad de interacciones"*, *"cantidad de interacciones por minuto"* o *"cantidad de tweets por minuto"*, todos los modelos presentan peores resultados que los evaluados en la Tabla 6.7. Se concluye entonces que estos tres atributos son importantes para los modelos por lo cual se dejarán fijos y se permutará la aparición o no de los que aún no han sido descartados. Los resultados completos de esta prueba aparecen detallados en la Tabla B.3 del apéndice B.

De esta última prueba se deduce que el mejor resultado en el conjunto de validación se obtiene para el modelo 4, con un valor de 94.1 % en el *f1-score*, el cual utiliza los siguientes atributos:

2. Cantidad de interacciones
3. Cantidad de interacciones por minuto
4. Medida r
5. *Followers* del usuario
7. Cantidad de *tweets* por minuto

8. Cantidad de *tweets* en la siguiente hora

Se puede consultar los resultados completos de esta prueba en la Tabla B.4 del apéndice B.

6.2.3. Resultados

En esta sección se ilustran más en detalle los resultados obtenidos por el modelo concluido como el más apropiado según el análisis presentado en la sección 6.2.2.

Para evaluar estos resultados se busca una solución más simple que resuelva este problema. Para esto, se define un algoritmo de línea base a efectos de comparar el rendimiento del modelo construido. En este caso, consiste en predecir como disparador al *tweet* que tenga más interacciones dentro de los que refieren a una misma tendencia.

La Tabla 6.9 presenta la matriz de confusión de la predicción realizada por el modelo en el conjunto de testeo.

	Positivo	**Negativo**
Positivo	11	5
Negativo	0	4329

Tabla 6.9: Resultados Matriz de confusión obtenida en el conjunto de testeo

Por otra parte, la Tabla 6.3 muestra la matriz de confusión para la clasificación realizada por el algoritmo de línea base en este mismo conjunto.

	Positivo	**Negativo**
Positivo	7	2
Negativo	4	4332

Tabla 6.10: Matriz de confusión obtenida por la línea base

En base a estos datos se calculan las métricas de *precision*, *recall* y *f1-score* para los *tweets* anotados como disparadores en este conjunto de datos, comparando el modelo construido contra su línea base, tal como se muestra en la Tabla 6.11.

	Precision	**Recall**	**F1-score**	**Support**
Red Neuronal	0.69	1.00	0.81	11
Línea base	0.77	0.63	0.70	11

Tabla 6.11: Resultados del modelo obtenidos en el conjunto de testeo

A partir de estos resultados, se verifica que el modelo tiene un *recall* de 1, lo cual se traduce en que se encuentran todos los disparadores y que no existen falsos negativos. Este resultado plantea la incertidumbre sobre la cantidad y variedad de ejemplos anotados para este conjunto. Por otra parte, los resultados arrojan que la precisión es aceptable, a menos de algunos falsos positivos. Recordar que el etiquetado de estos *tweets* tiene una carga de subjetividad, por lo que estos falsos

positivos merecen ser analizados en detalle. Analizando los resultados obtenidos por la línea base, se observa que encuentra apenas un 63 % de los disparadores, lo cual presenta una diferencia considerable con respecto al modelo construido. Esto se debe a que, por construcción, el algoritmo de línea base detecta solamente un disparador por tendencia, por lo cual obtiene resultados más bajos producto de que algunas de ellas están anotadas con más de un disparador. Esto no sucede en el modelo construido, ya que es capaz de encontrar más de un disparador por tendencia.

En resumen, para la Red Neuronal construida se obtiene un **81 % de *f1-score***, siendo a nuestro entender un resultado tanto satisfactorio como mejorable.

6.3. Demonio de extracción de *tweets*

En esta sección se presentan los resultados obtenidos al validar la *performance* del demonio de extracción de *tweets*. Esta prueba intenta analizar la magnitud de las tendencias en cuanto a la cantidad de *tweets* obtenidos y el espacio en disco para almacenarlos. Por otra parte, se analiza qué tan factible es contar con una consulta histórica de tendencias.

Para ello se ejecuta el demonio de extracción de *tweets* durante una semana con el fin de obtener un resultado parcial y luego poder extrapolarlos al período deseado. Este lapso de tiempo fue elegido en base a que existen eventos recurrentes que generan tendencias en Uruguay como ser eventos deportivos o programas televisivos, los cuales tienen una periodicidad definida.

Se analizaron ventajas y desventajas de tres posibilidades para almacenar la información.

Opción 1:
Como primera alternativa, se evaluó almacenar la información tal cual se obtiene de Twitter, sin aplicarle ningún filtro. Esto tiene como desventaja una gran masa de *tweets* y tendencias que generan un gran uso del espacio de almacenamiento. Como ventaja principal, permite poder volver a evaluar los datos en caso que surja una modificación en alguno de los modelos implementados al contar con la información completa de los *tweets*.

Opción 2:
Como segunda alternativa, se propone eliminar todos aquellos *tweets* para los cuales el clasificador temático determinó que no referían al tema principal de la tendencia. Su ventaja más importante es un menor gasto de recursos para almacenar los *tweets*, ya que un número considerable de ellos es eliminado, pero le quita la posibilidad a este clasificador de poder volver a entrenar y calcular las predicciones sobre las tendencias ya descargadas. Esto no sucede con el modelo de detección del *tweet* disparador ya que se mantienen todos los datos necesarios para calcular los atributos que intervienen en este modelo.

Opción 3:
Por último, se propone eliminar toda aquella información que no sean necesaria para presentar en la web, es decir, se eliminan todos los *tweets* que no refieren al tema principal de la tendencia, así como toda la *metadata* almacenada de cada *tweet* que no se despliegue en pantalla. Esto tiene una enorme ventaja en cuanto al almacenamiento utilizado, ya que se reducen considerablemente los recursos necesarios, lo que también permite mantener más información de tendencias históricas para mostrar en la aplicación web. Como desventaja principal, ninguna corrección en el entrenamiento de los modelos podrá ser ejecutada sobre los *tweets* de las tendencias ya descargadas.

La Tabla 6.12 muestra los datos experimentales asociados a cada opción planteada.

	Cantidad de tweets	Cantidad de tendencias	Almacenamiento necesario
Opción 1	6,116,378	1,055	1.7 GB
Opción 2	2,625,329	1,055	962.6 MB
Opción 3	2,625,329	1,055	486.2 MB

Tabla 6.12: Almacenamiento requerido para *tweets* recolectados durante una semana

Suponiendo que es posible desestimar el impacto que pueden llegar a tener eventos excepcionales, y asumir que todas las semanas hay un comportamiento regular de los usuarios de Uruguay en Twitter, en la Tabla 6.13 se presenta la extrapolación del almacenamiento necesario estimado para distintos plazos.

	Opción 1	Opción 2	Opción 3
Almacenamiento una semana	1.7 GB	962.6 MB	486.2 MB
Cantidad de *tweets* una semana	6.1 M	2.6 M	2.6 M
Almacenamiento un mes	7.2 GB	4 GB	2 GB
Cantidad de *tweets* un mes	26.1 M	11.1 M	11.1 M
Almacenamiento un año	88.6 GB	49 GB	24.7 GB
Cantidad de *tweets* un año	318 M	135.5 M	135.5 M

Tabla 6.13: Proyección de almacenamientos

Conclusiones

Si bien la primera opción no genera un problema real para la infraestructura disponible en la actualidad, en cuanto a lo que almacenamiento se refiere, el exceso de documentos en la base de datos podría llegar a impactar en la *performance* de la aplicación final.

En cuanto a las dos alternativas restantes, si bien van a tener la misma cantidad de documentos guardados y el tamaño del almacenamiento no difiere en cantidades significativas, no se encuentra necesidad de guardar los datos para volver a entrenar un modelo sí y el otro no, en caso que fuera necesario. Por lo tanto, se entiende que la tercera opción es la más adecuada, ya que minimiza el espacio ocupado por cada *tweet* y no genera inconvenientes al guardar información histórica de varios meses, sin la necesidad de contar con una infraestructura más potente que la de un ordenador de uso doméstico.

6.4. Demonio de predicción de *tweet* disparador

En esta sección se presentan los resultados obtenidos en las pruebas realizadas al evaluar el funcionamiento del demonio que predice los *tweets* disparadores de tendencias.

Medición de tiempo de procesamiento:

Se realiza una prueba que intenta determinar el tiempo necesario para calcular los *tweets* disparadores de los datos recopilados durante una semana y verificar que no se produzca un aumento significativo en el tamaño de la cola de espera. Por lo tanto, con el mismo conjunto de datos utilizados en la sección 6.3 y los modelos pre-entrenados, sin tener calculada ninguna predicción, se ejecuta el demonio y se mide el tiempo necesario para procesar las 1055 tendencias. Este proceso

insumió 27,5 horas en procesar toda la información recolectada durante una semana, ejecutando en un equipo que cuenta con un procesador de 2.6 GHz, Intel Core i7 de seis núcleos y 16 GB de *ram* DDR4 de 2667 MHz.

Por lo tanto, se concluye que el demonio de predicción de *tweets* disparadores procesa las tendencias más rápido de lo que se recolectan, es decir, la información que se recolecta en un instante tendrá un *delay* hasta aparecer disponible en la aplicación final, pero no será significativo.

Efectividad en el procesamiento de tendencias:

En esta segunda prueba se intentó determinar a qué porcentaje de las tendencias que fueron encontradas se les pudo calcular su disparador, con el fin de poder comprobar si existen casos en los cuales producto de las características de los *tweets* asociados y su cantidad, no se determine un disparador. Luego de procesar las 1055 tendencias, se observó que sólo 73 de ellas no tuvieron ningún resultado positivo, es decir, que a un 93 % de las tendencias encontradas se les pudo encontrar al menos un *tweet* disparador.

Dado que estas 73 tendencias no tenían *tweets* de Uruguay, no fue posible calcular un *BOW* con al menos 2 palabras para cada una de ellas, por lo cual no se determinó un disparador para estas tendencias.

Capítulo 7

Conclusiones y trabajo a futuro

En este capítulo se presentan las conclusiones del trabajo realizado junto con propuestas de mejoras a futuro. En la sección 7.1 se exponen las conclusiones empíricas del proyecto junto con una reflexión a nivel personal acerca de este. En la sección 7.2 se detallan las principales mejoras sugeridas para intentar mejorar el sistema construido.

7.1. Conclusiones

El objetivo de este trabajo es poder detectar cuáles son los *tweets* que tuvieron gran repercusión en la red social Twitter, como para convertir a un tema en tendencia. Con esta finalidad se construyen dos modelos de Aprendizaje Automático. Uno de ellos consiste en un clasificador basado en una Regresión Logística, cuyo objetivo es filtrar *tweets* dependiendo de la temática a la cual refieren. El segundo modelo está basado en una Red Neuronal que intenta determinar los *tweets* disparadores de cada tendencia. Ambos modelos son entrenados utilizando Aprendizaje Supervisado en base a distintos conjuntos de datos. Las tendencias de estos conjuntos fueron extraídas utilizando un proceso de *web scraping* y sus *tweets* mediante la *API* de Twitter v1.1. Estos datos son recolectados en distintos lapsos de tiempo, basándose siempre en tendencias de Uruguay.

El primer modelo plantea una solución para poder distinguir de todos los *tweets* relacionados con una expresión que es tendencia, cuáles de ellos efectivamente hablan del tema que se está discutiendo en Uruguay. Este modelo fue entrenado con 1400 *tweets* y presenta un porcentaje de acierto del 79 %. Si bien no es un resultado excelente, lo consideramos bueno debido a la dificultad del problema.

El segundo modelo se encarga de predecir qué *tweets* fueron disparadores de tendencias y fue entrenado con un volumen de 15512 *tweets*. Este encontró el 100 % de los disparadores etiquetados en el conjunto de testeo, siendo un resultado que da para desconfiar sobre la variedad o cantidad de ejemplos en este conjunto. Por otro lado, del total de disparadores predichos, el 30 % de ellos son falsos positivos. Si bien este número es un poco elevado, vale la pena analizarlos caso a caso dada la subjetividad que existe en su etiquetado, donde se podría concluir de forma más certera qué tan errónea es la predicción del modelo en cada uno de ellos. Más allá de esto, en términos de *f1-score*, este modelo tuvo un resultado de 81 %, entendido como satisfactorio, por más que posea un margen para ser mejorado. En base a estos resultados se concluye que es posible identificar cuáles son estos *tweets* que alcanzan grandes niveles de repercusión para generar tendencias en Twitter.

Otro punto a destacar de este desarrollo es su versatilidad, ya que si bien el trabajo estuvo

orientado a tendencias de Uruguay debido al reducido volumen de sus datos (si se lo compara con países de mayor población), existe la posibilidad de enfocarlo a cualquier país de habla hispana realizando cambios sutiles en la implementación del sistema.

Se aclara que al investigar trabajos relacionados en el área no se encontraron desarrollos similares que atacaran este problema, lo cual no quiere decir que no existan. Esto tiene como consecuencia la falta de un estudio que permita comparar resultados obtenidos en este trabajo.

A modo de conclusión personal nos gustaría expresar lo que nos dejó este proyecto. Para ambos integrantes significó una primera incursión en el campo de la IA. En el transcurso de este trabajo hemos conocido la cantidad de aplicaciones que tiene el Aprendizaje Automático, lo cual nos motivó a estudiar y aprender para poder llevar adelante el proyecto, sabiendo que quizás pueda ser un área de investigación al cual dedicarnos de aquí en más. Particularmente, el hecho de poder comunicarnos y obtener información de una de las redes sociales más populares en el mundo como Twitter (u otras), y luego procesar estos datos con el fin que uno quiera otorgarle, nos parece realmente interesante. Nos enfrentamos a problemas que nos demostraron las complejidades que presenta procesar texto de forma automatizada y la cantidad de variantes que presenta el lenguaje natural, las cuales si bien son difíciles de abarcar, les pudimos encontrar una buena solución. El hecho de haber definido el alcance del proyecto mientras se desarrollaba, debido a la complejidad del desafío, provocó que este se prolongara en el tiempo más de lo que nos hubiera gustado, pero al fin y al cabo por nuestra parte estamos realmente conformes con el trabajo realizado.

7.2. Trabajo a futuro

En esta sección se detallan las propuestas de mejoras a futuro para cada uno de los componentes que integran la solución construida.

7.2.1. Mejoras al demonio de extracción de *tweets*

Para la comunicación con Twitter se utiliza la versión 1.1 de su *API*. Una posible mejora a futuro sería adaptar la lógica de obtención de *tweets* para interactuar con la versión 2.0, de forma de aprovechar sus nuevas utilidades.

En cuanto a la obtención de tendencias mediante el proceso de *web scraping*, durante la etapa de desarrollo se utilizó un usuario creado específicamente para esto, sin actividad diaria. Al momento de ejecutar este proceso de manera prolongada comenzó a fallar, ya que los inicios de sesión reiterados de un usuario sin actividad eran detectados como sospechosos. Para subsanar esto, se utilizó un usuario personal de los integrantes del equipo. Se plantea como trabajo a futuro investigar una manera alternativa de obtener las tendencias de Uruguay, o mediante la solución ya desarrollada pero buscando una solución al problema de los usuarios a utilizar.

7.2.2. Mejoras al modelo de clasificación de *tweets*

El modelo de clasificación de *tweets* podría ser mejorado para que filtre mejor los que hablan del tema principal de la tendencia. Para esto, un cálculo que podría ser precisado es el del atributo *es de Uruguay*, el cual no está incluido en el conjunto de atributos del modelo final. Para esto, si la ubicación del *tweet* y del usuario son ambas vacías, se podrían buscar sus *retweets* almacenados en la base de datos. Para su conjunto de *retweets*, utilizando la ubicación de sus autores, se calcula cuantos de ellos son de Uruguay. Finalmente, si la cantidad de *retweets* de usuarios uruguayos supera cierto umbral, entonces también se podría afirmar que el *tweet* original es de Uruguay.

De forma de brindar mayor variedad de tendencias al conjunto de entrenamiento, se propone agregar más ejemplos de tendencias no vistas. Dado el esfuerzo que requiere leer cada *tweet* para determinar si habla del tema principal de la tendencia, se cuenta con 1400 ejemplos de entrenamiento, una cantidad reducida para la dimensión del problema que intenta resolver este modelo. Dada la inmensidad de temas posibles, no es posible llegar a cubrir una cantidad de casos de entrenamiento que por su variedad temática nutran de buena manera a este modelo. Sin embargo, agregar casos de entrenamiento podría contribuir a mejorar sus valores de predicción.

Otra alternativa que podría mejorar la solución implementada, sería utilizar un modelo SVM [37] o una Red Neuronal en lugar de una Regresión Logística para resolver.

7.2.3. Mejoras al modelo de cálculo de *tweet* disparador

Para mejorar el modelo que determina el *tweet* disparador se podría invertir tiempo y análisis para agregar nuevos atributos. Con más *metadata*, ya sea brindada por la versión 2.0 de la *API* de Twitter, una versión *premium* de esta *API* o cualquier otro método que permita obtener nueva información relevante, se podría intentar reconstruir en base a una estructura de grafos, la repercusión de cada *tweet*, involucrando citas, respuestas y *retweets*. Cabe aclarar que esto último se intentó pero sin mayor éxito debido a los datos con los que se contaba.

Algo que podría ayudar a mejorar los resultados de este modelo, sin tener que modificar los atributos que ya tiene, es volver a entrenarlo utilizando más cantidad y variedad de información en sus conjuntos de entrenamiento, validación y testeo, además de definir heurísticas para eliminar subjetividades a la hora de supervisar los ejemplos.

Otra idea que podría ayudar a que el modelo una vez instalado continúe aprendiendo de los nuevos casos vistos, es el llamado *Online Learning* [24]. Este método permite que el sistema se nutra de la información de contexto que tienen los usuarios finales, los cuales podrían indicar si el disparador calculado para una tendencia les parece correcto o no. En este trabajo no se estudia ni se aplica *Online Learning*, pero creemos que hoy en día un sistema de *IA* es bueno que cuente con ello.

Glosario

API: Application Programming Interface.

API key: Clave de acceso a una *API*.

BOW: Bag of Words.

Corpus: Es un conjunto muy grande y estructurado de textos que generalmente representan ejemplos de uso de una lengua en un contexto [36].

CSV: Archivo de valores separados por comas (Comma Separated Values).

Demonio: Proceso de ejecución repetitiva.

IA: Inteligencia Artificial.

Metadatos: Datos que describen a los datos. En el contexto de este proyecto, los metadatos del *tweet* son su cantidad de favoritos, *retweets*, citas, etc.

NodeJS: Es un entorno de ejecución de JavaScript orientado a eventos asíncronos, el cuál está diseñado para crear aplicaciones network escalables [26] .

PLN: Procesamiento de Lenguaje Natural.

POS: Etiquetado gramatical de palabras (*Part of Speech*).

React: Biblioteca de JavaScript para construir interfaces de usuario[43].

Retweet: Acción de volver a publicar un *tweet* de otro usuario, donde se conserva el autor original del *tweet*.

Stopwords: Palabras y signos de puntuación que carecen de contenido semántico.

SVM: Support Vector Machines.

Token: Cadena de caracteres delimitada por un espacio al principio y al final.

TT: Tendencia en Twitter (*Trending Topic*).

Tweet: Texto publicado en la red Twitter.

Bibliografía

[1] Alex Hai Wang. Don't Follow Me: Spam Detection in Twitter. 2010.

[2] Sasa Petrovic, Miles Osborne and Victor Lavrenko. RT to Win! Predicting Message Propagation in Twitter. 2011.

[3] Jinan Fiaidhi, Sabah Mohammed and Aminul Islam. Developing a Hierarchical Multi-Label Classifier for Twitter Trending Topics. 2013.

[4] G. Ifrim, B. Shi and I. Brigadir. Event Detection in Twitter using Aggresive Filtering and Hierarchical *tweet* Clustering, Proceedings of the SNOW 2014 Data Challenge at WWW14, Korea, April 2014.

[5] Gerasimos Razis and Ioannis Anagnostopoulos. Discovering similar Twitter accounts using semantics. 2015.

[6] Xunhu Sun and Philip K. Chan. Estimating effectiveness of twitter messages with a personalized machine learning approach. 2017.

[7] Antonio Fonseca and Jorge Louca. Explaining the emergence of online popularity through a model of information diffusion. 2017.

[8] Mateusz Fedoryszak, Vijay Rajaram, Brent Frederick and Changtao Zhong. Real-time Event Detection on Social Data Streams. 2019.

[9] Examples of using Python for Twitter social data mining, using the python-twitter-tools framework: https://github.com/ideoforms/python-twitter-examples [online; último acceso 29-Mayo-2021]

[10] https://developer.twitter.com/en/docs/twitter-api [online; último acceso 7-Junio-2021]

[11] https://countwordsfree.com/stopwords/spanish [online; último acceso 16-Noviembre-2020]

[12] Introduction to Artificial Intelligence, Second Edition, Wolfgang Ertel, Springer, 2017

[13] https://scikit-learn.org/stable/index.html [online; último acceso 15-Junio-2021]

[14] https://docs.python.org/3/library/ [online; último acceso 10-Junio-2021]

[15] https://spacy.io/usage/spacy-101 [online; último acceso 4-Junio-2021]

[16] https://fasttext.cc/docs/en/support.html [online; último acceso 26-Mayo-2021]

[17] https://www.mongodb.com[online; último acceso 29-Marzo-2021]

[18] https://www.selenium.dev/documentation/es/ [online; último acceso 30-Marzo-2021]

[19] https://github.com/geduldig/TwitterAPI [online; último acceso 29-Mayo-2021]

[20] Elmasri and Navate. Fundamentals on Database Systems, Seventh Edition.

[21] Common Crawl. https://commoncrawl.org [online; último acceso 30-Marzo-2021]

[22] Tom M. Mitchell - Machine Learning. 1997.

[23] Universal POS tags. https://universaldependencies.org/u/pos/all.html [online; último acceso 29-Abril-2021]

[24] Stanford University. Machine Learning. https://www.coursera.org/learn/machine-learning/home [online; último acceso 5-Julio-2021]

[25] Wikipedia: Red Neuronal Artificial. https://es.wikipedia.org/wiki/Red_neuronal_artificial [online; último acceso 30-Mayo-2021]

[26] Acerca de Node.js. https://nodejs.org/es/about/ [online; último acceso 5-Julio-2021]

[27] Web Api. https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces [online; último acceso 29-Junio-2021]

[28] FastAPI. https://fastapi.tiangolo.com/ [online; último acceso 28-Mayo-2021]

[29] DXMEDIA: Las redes sociales con más usuarios: 2020 https://dxmedia.net/redes-sociales-usuarios-2020/ [online; último acceso 22-Junio-2021]

[30] Concepto.de: Concepto de REDES SOCIALES https://concepto.de/redes-sociales/#ixzz6yZeMgWiJ [online; último acceso 22-Junio-2021]

[31] Forbes: How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/?sh=3bab081b60ba [online; último acceso 22-Junio-2021]

[32] Zephoria: Top 10 Twitter Statistics – Updated March 2021 https://zephoria.com/twitter-statistics-top-ten/ [online; último acceso 22-Junio-2021]

[33] Mathias Etcheverry, Dina Wonsever. Spanish word vectors from Wikipedia. https://www.aclweb.org/anthology/L16-1584.pdf [online; último acceso 24-Junio-2021]

[34] Your Guide to Natural Language Processing (NLP). https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1 [online; último acceso 6-Julio-2021]

[35] Stanford Log-linear Part-Of-Speech Tagger. https://nlp.stanford.edu/software/tagger.shtml [online; último acceso 6-Julio-2021]

[36] Dan Jurafsky and James H. Martin. Speech and Language Processing (3rd ed. draft). https://web.stanford.edu/~jurafsky/slp3 [online; último acceso 6-Julio-2021]

[37] Introduction to Support Vector Machines. http://www.svms.org/introduction.html [online; último acceso 7-Julio-2021]

[38] ¿Quieres ser un Trending Topic en Twitter? Así es cómo se hace. https://blog.hootsuite.com/es/trending-topic-en-twitter/ [online; último acceso 7-Julio-2021]

[39] Precisión y Recall. https://en.wikipedia.org/wiki/Precision_and_recall [online; último acceso 26-Julio-2021]

[40] Representación de palabras como vectores. https://eva.fing.edu.uy/pluginfile.php/103446/course/section/13185/REPRESENTACIO%CC%81N%20DE%20PALABRAS%20COMO%20VECTORES%20v2.pdf [online; último acceso 28-Julio-2021]

[41] Artificial neural network model. https://www.hebergementwebs.com/tutorial-on-the-artificial-neural-network/artificial-neural-network-basic-concepts [online; último acceso 1-Agosto-2021]

[42] Código implementado. https://github.com/diegoalberti8/pg_trending_topics [online; último acceso 12-Agosto-2021]

[43] React https://es.reactjs.org/[online; último acceso 02-Setiembre-2021]

Apéndice A

Anexo I: Tablas de resultados del modelo de clasificación temática

La Tabla A.1 presenta el resultado obtenido en la prueba de selección de atributos del modelo de Regresión Logística que se expone en la sección 6.1.

Modelo	Atributos	Training	Validation
0	Todos	0.910	0.830
1	Sin 1	0.962	0.795
2	Sin 2	0.914	0.830
3	Sin 3	0.910	0.830
4	Sin 4	0.832	0.815
5	Sin 5	0.915	0.825
6	Sin 6	0.923	0.825
7	Sólo 1, 2, 3 y 4	0.922	0.820
8	Sólo 1, 2, 3 y 5	0.842	0.815
9	Sólo 1, 2, 3 y 6	0.847	0.820
10	Sólo 1, 2, 4 y 5	0.920	0.825
11	Sólo 1, 2, 4 y 6	0.912	0.825
12	Sólo 1, 2, 5 y 6	0.824	0.820
13	Sólo 1, 3, 4 y 5	0.925	0.820
14	Sólo 1, 3, 4 y 6	0.907	0.830
15	Sólo 1, 3, 5 y 6	0.829	0.815
16	Sólo 1, 4, 5 y 6	0.915	0.830
17	Sólo 2, 3, 4 y 5	0.961	0.790
18	Sólo 2, 3, 4 y 6	0.955	0.795
19	Sólo 2, 3, 5 y 6	0.845	0.820
20	Sólo 2, 4, 5 y 6	0.962	0.790
21	Sólo 3, 4, 5 y 6	0.956	0.730
22	Sólo 1, 2 y 3	0.843	0.825
23	Sólo 1, 2 y 4	0.920	0.825
24	Sólo 1, 2 y 5	0.811	0.815
25	**Sólo 1, 2 y 6**	**0.830**	**0.850**
26	Sólo 1, 3 y 4	0.920	0.825
27	Sólo 1, 3 y 5	0.845	0.815
28	Sólo 1, 3 y 6	0.838	0.815
29	Sólo 1, 4 y 5	0.924	0.825
30	Sólo 1, 4 y 6	0.907	0.830
31	Sólo 1, 5 y 6	0.805	0.805
32	Sólo 2, 3 y 4	0.953	0.800
33	Sólo 2, 3 y 5	0.842	0.820
34	Sólo 2, 3 y 6	0.835	0.825
35	Sólo 2, 4 y 5	0.960	0.790
36	Sólo 2, 4 y 6	0.956	0.790
37	Sólo 2, 5 y 6	0.807	0.820
38	Sólo 3, 4 y 5	0.955	0.700
39	Sólo 3, 4 y 6	0.943	0.680
40	Sólo 3, 5 y 6	0.718	0.705
41	Sólo 4, 5 y 6	0.953	0.730
42	Sólo 1 y 2	0.827	0.835
43	Sólo 1 y 3	0.838	0.810
44	Sólo 1 y 4	0.920	0.825
45	Sólo 1 y 5	0.802	0.805
46	Sólo 1 y 6	0.830	0.835
47	Sólo 2 y 3	0.836	0.825
48	Sólo 2 y 4	0.955	0.795
49	Sólo 2 y 5	0.806	0.820
50	Sólo 2 y 6	0.833	0.835
51	Sólo 3 y 4	0.942	0.655
52	Sólo 3 y 5	0.718	0.705
53	Sólo 3 y 6	0.659	0.655
54	Sólo 4 y 5	0.956	0.715
55	Sólo 4 y 6	0.944	0.630
56	Sólo 5 y 6	0.661	0.625

Tabla A.1: Evaluación de atributos del modelo de Regresión Logística

Apéndice B

Anexo II: Tablas de resultados del modelo de predicción de *tweet* disparador

A continuación se presentan los resultados obtenidos al evaluar los modelos candidatos presentados en la sección 6.2.

En la Tabla B.3 se presentan los resultados de evaluar los modelos candidatos.

Modelo	Atributos	Training	Validation
1	Todos	0.839	0.894
2	Todos	0.766	0.867
3	Todos	0.776	0.848
4	Todos	0.88	0.821
5	Todos	0.881	0.886
1	Sin 1	0.818	0.868
2	Sin 1	0.725	0.869
3	Sin 1	0.712	0.888
4	Sin 1	0.862	0.921
5	Sin 1	0.846	0.904
1	Sin 2	0.549	0.566
2	Sin 2	0.586	0.716
3	Sin 2	0.639	0.687
4	Sin 2	0.659	0.717
5	Sin 2	0.649	0.611
1	Sin 3	0.82	0.848
2	Sin 3	0.748	0.871
3	Sin 3	0.707	0.871
4	Sin 3	0.828	0.828
5	Sin 3	0.807	0.857
1	Sin 4	0.87	0.875
2	Sin 4	0.686	0.803
3	Sin 4	0.741	0.896
4	Sin 4	0.909	0.854
5	Sin 4	0.894	0.856
1	Sin 5	0.666	0.666
2	Sin 5	0.671	0.837
3	Sin 5	0.746	0.875
4	Sin 5	0.887	0.894
5	Sin 5	0.844	0.888
1	Sin 6	0.846	0.882
2	Sin 6	0.735	0.853
3	Sin 6	0.762	0.849
4	Sin 6	0.88	0.839
5	Sin 6	0.855	0.876
1	Sin 7	0.861	0.868
2	Sin 7	0.693	0.853
3	Sin 7	0.743	0.818
4	Sin 7	0.895	0.85
5	Sin 7	0.862	0.867
1	Sin 8	0.835	0.887
2	Sin 8	0.759	0.861
3	Sin 8	0.666	0.901
4	Sin 8	0.873	0.811
5	Sin 8	0.873	0.895

Tabla B.1: Resultados de entrenamientos para modelo de Red Neuronal en la primer selección de atributos

Modelo	Atributos	Training	Validation
1	Sin 1 y 2	0.599	0.659
2	Sin 1 y 2	0.45	0.682
3	Sin 1 y 2	0.39	0.704
4	Sin 1 y 2	0.675	0.668
5	Sin 1 y 2	0.669	0.651
1	Sin 1 y 3	0.638	0.641
2	Sin 1 y 3	0.74	0.836
3	Sin 1 y 3	0.738	0.887
4	Sin 1 y 3	0.836	0.867
5	Sin 1 y 3	0.823	0.805
1	Sin 1 y 4	0.783	0.878
2	Sin 1 y 4	0.766	0.836
3	Sin 1 y 4	0.739	0.856
4	Sin 1 y 4	0.841	0.895
5	Sin 1 y 4	0.854	0.881
1	Sin 1 y 5	0.809	0.867
2	Sin 1 y 5	0.588	0.793
3	Sin 1 y 5	0.731	0.889
4	Sin 1 y 5	0.821	0.904
5	Sin 1 y 5	0.81	0.852
1	Sin 1 y 6	0.818	0.865
2	Sin 1 y 6	0.728	0.88
3	Sin 1 y 6	0.715	0.871
4	Sin 1 y 6	0.859	0.931
5	Sin 1 y 6	0.837	0.898
1	Sin 1 y 7	0.812	0.851
2	Sin 1 y 7	0.777	0.797
3	Sin 1 y 7	0.712	0.849
4	Sin 1 y 7	0.848	0.917
5	Sin 1 y 7	0.846	0.941
1	Sin 1 y 8	0.834	0.861
2	Sin 1 y 8	0.718	0.91
3	Sin 1 y 8	0.741	0.91
4	Sin 1 y 8	0.831	0.826
5	Sin 1 y 8	0.843	0.838

Tabla B.2: Resultados de entrenamientos para modelo de Red Neuronal en la segunda selección de atributos

Modelo	Atributos	Training	Validation
1	Sin 1, 2 y 6	0.34	0.36
2	Sin 1, 2 y 6	0.566	0.595
3	Sin 1, 2 y 6	0.476	0.658
4	Sin 1, 2 y 6	0.643	0.69
5	Sin 1, 2 y 6	0.55	0.665
1	Sin 1, 3 y 6	0.807	0.824
2	Sin 1, 3 y 6	0.728	0.856
3	Sin 1, 3 y 6	0.699	0.857
4	Sin 1, 3 y 6	0.777	0.815
5	Sin 1, 3 y 6	0.838	0.811
1	Sin 1, 4 y 6	0.641	0.69
2	Sin 1, 4 y 6	0.682	0.878
3	Sin 1, 4 y 6	0.681	0.911
4	Sin 1, 4 y 6	0.836	0.898
5	Sin 1, 4 y 6	0.85	0.92
1	Sin 1, 5 y 6	0.783	0.916
2	Sin 1, 5 y 6	0.74	0.856
3	Sin 1, 5 y 6	0.77	0.834
4	Sin 1, 5 y 6	0.829	0.887
5	Sin 1, 5 y 6	0.846	0.92
1	Sin 1, 6 y 7	0.799	0.805
2	Sin 1, 6 y 7	0.743	0.827
3	Sin 1, 6 y 7	0.783	0.817
4	Sin 1, 6 y 7	0.843	0.917
5	Sin 1, 6 y 7	0.835	0.883
1	Sin 1, 6 y 8	0.843	0.831
2	Sin 1, 6 y 8	0.772	0.888
3	Sin 1, 6 y 8	0.7	0.9
4	Sin 1, 6 y 8	0.808	0.83
5	Sin 1, 6 y 8	0.825	0.845

Tabla B.3: Resultados de entrenamientos para modelo de Red Neuronal en la tercer selección de atributos

Modelo	Atributos	Training	Validation
1	Solo 2, 3 y 7	0.797	0.824
2	Solo 2, 3 y 7	0.663	0.759
3	Solo 2, 3 y 7	0.751	0.871
4	Solo 2, 3 y 7	0.776	0.843
5	Solo 2, 3 y 7	0.787	0.844
1	2, 3, 4 y 7	0.795	0.859
2	2, 3, 4 y 7	0.75	0.894
3	2, 3, 4 y 7	0.724	0.922
4	2, 3, 4 y 7	0.778	0.876
5	2, 3, 4 y 7	0.793	0.87
1	2, 3, 5 y 7	0.828	0.834
2	2, 3, 5 y 7	0.716	0.922
3	2, 3, 5 y 7	0.767	0.91
4	2, 3, 5 y 7	0.847	0.844
5	2, 3, 5 y 7	0.836	0.844
1	2, 3, 7 y 8	0.771	0.868
2	2, 3, 7 y 8	0.701	0.844
3	2, 3, 7 y 8	0.715	0.867
4	2, 3, 7 y 8	0.817	0.913
5	2, 3, 7 y 8	0.852	0.907
1	2, 3, 4, 5 y 7	0.822	0.857
2	2, 3, 4, 5 y 7	0.756	0.893
3	2, 3, 4, 5 y 7	0.752	0.898
4	2, 3, 4, 5 y 7	0.819	0.816
5	2, 3, 4, 5 y 7	0.843	0.825
1	2, 3, 4, 7 y 8	0.796	0.881
2	2, 3, 4, 7 y 8	0.767	0.824
3	2, 3, 4, 7 y 8	0.709	0.889
4	2, 3, 4, 7 y 8	0.847	0.92
5	2, 3, 4, 7 y 8	0.83	0.907
1	2, 3, 5, 7 y 8	0.828	0.888
2	2, 3, 5, 7 y 8	0.777	0.818
3	2, 3, 5, 7 y 8	0.745	0.869
4	2, 3, 5, 7 y 8	0.813	0.891
5	2, 3, 5, 7 y 8	0.839	0.908
1	2, 3, 4, 5, 7 y 8	0.825	0.861
2	2, 3, 4, 5, 7 y 8	0.732	0.859
3	2, 3, 4, 5, 7 y 8	0.693	0.891
4	**2, 3, 4, 5, 7 y 8**	**0.839**	**0.941**
5	2, 3, 4, 5, 7 y 8	0.858	0.895

Tabla B.4: Resultados de entrenamientos de Red Neuronal para selección final de atributos