UNIVERSITAT JAUME·I

TRABAJO DE FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN SISTEMAS INTELIGENTES

Detección automática de tweets noticiosos

Autor: Aldo Garcés Matilla

Tutor: Dr. Rafael Berlanga Llavori

Fecha de lectura: octubre de 2019

CURSO 2018-2019

RESUMEN

Las redes sociales, como Twitter, pueden facilitar la distribución de información en tiempo real entre usuarios de todo el mundo. Se ha demostrado que las personas cada vez reciben más noticias de las redes sociales que de las fuentes de noticias tradicionales. El objetivo del presente trabajo es la detección automática de tweets noticiosos. Se ha realizado, primeramente, un etiquetado de tweets noticiosos y un conjunto de no etiquetado, en el cual se encuentran los negativos. Para el proceso de extracción de negativos desde el conjunto de no etiquetados se realiza un proceso de PU-Learning. Además, por desbalanceo del conjunto se desarrolla una aumentación de datos, mediante la creación de tweets sintéticos. Finalmente, se propone un modelo, con aproximaciones de aprendizaje profundo, para la detección de tweets con los conjuntos obtenidos por el PU-Learning. Este modelo alcanza como resultado un 0.86 de F1-score y precisión de 0.98.

Palabras claves:

PU-Learning, aumentación de datos, aprendizaje profundo.

ABSTRACT

Social networks, such as Twitter, can facilitate the distribution of information in real time among users around the world. The people receive more news from social networks than from traditional news sources. The objective of this work is the automatic detection of newsworthy tweets. A labelling of newsworthy tweets and unlabeled was done. For the process of extracting negatives from the set of unlabeled, a PU-Learning process is applied. In addition, due to the imbalance of set, a data augmentation has been developed, through the creation of synthetic tweets. Finally, a model has been proposed, with deep learning approaches, for the detection of tweets with the sets detected by the PU-Learning. This model results in a 0.86 F1-score and precision of 0.98.

Keywords:

PU-Learning, data augmentation, deep learning

Índice

1. INTRODUCCIÓN

El desarrollo explosivo de la World Wide Web desde mediados de la década de 1990 ha avanzado significativamente la forma en que las personas se comunican entre sí. Las redes sociales en línea, como Twitter y Facebook, pueden facilitar la distribución de información en tiempo real entre usuarios de todo el mundo. Con las características de facilidad de uso, bajo costo y velocidad rápida, las redes sociales se han convertido en la principal plataforma para la interacción social en línea y la transmisión de información [1].

Estudios recientes han demostrado que las personas cada vez reciben *más noticias de las redes sociales* que de las fuentes de noticias tradicionales. Por lo tanto, en los últimos tiempos se ha incrementado el análisis de los mensajes transmitidos a través de servicios microblogging [2, 3], enfocándose en muchos casos sobre la red social Twitter [4].

El surgimiento del paradigma de Inteligencia de Negocio Social ofrece nuevas perspectivas de análisis y acción en las redes sociales basadas en métricas e indicadores obtenidos de los mensajes y usuarios en redes sociales. Dentro de este nuevo paradigma, la reducción del ruido e identificación de usuarios o mensajes relevantes son cruciales para realizar un análisis de calidad [5, 6].

Este trabajo se orienta hacia la *detección de tweets noticiosos* en un flujo de datos obtenido. Las maneras tradicionales para crear modelos de clasificación de tweets son por medio de varios profesionales entrenados que emiten su veredicto (*etiquetado manual*) acerca de un objetivo específico (este caso, *tweets noticiosos*). Este tipo de clasificadores requieren un número grande de instancias etiquetadas en ambas clases para su buen funcionamiento. Cuando el conjunto de tweets a revisar es muy grande, esta forma de etiquetado manual, se vuelve muy costoso y muy lenta. Es por esto que se propone resolver este problema utilizando métodos de PU-Learning, el cual tiene entre sus características principales la de aprender a partir de *ejemplos positivos* y requerir solamente una pequeña cantidad de datos *etiquetados (positivos),* junto con un conjunto grande de datos *no-etiquetados* para su entrenamiento.

Los métodos PU-Learning tienen información de una sola clase, la clase positiva (*en nuestro caso la clase de tweets noticiosos*), la cual generalmente es pequeña. Además, se cuenta con otra clase (la clase no-etiquetada), que contiene un gran número de datos, la cual incluyen ejemplos positivos y negativos. En el presente trabajo se utiliza dos aproximaciones de PU-Learning: *bagging* [7] e *iterativo* [8, 9].En general, los métodos PU-Learning usan una estrategia de dos pasos para construir un clasificador de dos clases [10], estos son:

- Identificar automáticamente un conjunto de *relativos negativos* (nuestro caso, *tweets no noticiosos*), el cual se forma extrayéndolos del conjunto no etiquetado.
- Usar un algoritmo de aprendizaje con el conjunto de entrenamiento refinado, para construir un clasificador de dos clases.

Como se ha mencionado los métodos de PU-Learning puede trabajar con muestras positivas (P), pero se tiene como inconveniente que si existe una proporción de muestras positivas con respecto a las no etiquetas muy pequeñas los métodos pueden bajar su eficacia. Por lo tanto, en este caso se usa una estrategia de aumentación de datos (*data aumentación*) de las muestras positivas. Esta estrategia además de resolver el problema mencionado, ayuda al inconveniente de *desbalanceo* bastante común en el aprendizaje automático y contribuiría a clasificador final más eficaz. En el presente trabajo la aumentación de datos de tweets se hace utilizando The Guardian Open Plataform para la creación de tweets sintéticos y las métricas de los usuarios a través de los datos que se tienen de la clase positiva.

Los *word embeddings* son una representación vectorial de palabras con valores real mediante embeddings de significados semánticos y sintácticos obtenidos de grandes corpus sin etiquetar. Es una herramienta poderosa ampliamente utilizada en diferentes tareas modernas de procesamiento del lenguaje natural obteniendo resultados relevantes [11], que incluye análisis semántico, recuperación de información, respuesta a preguntas, y traducción automática. Existen una gran variedad de modelos *de word embeddings* pre-entrenados [12, 13], mediante el uso de textos sin etiquetar, mediante la predicción de contextos y modelos no supervisados. Estos modelos usan una alta representación de palabras y frases, en cambio, muchos están entrenados sobre un dominio específico y carecen de peculiaridades de tareas específicas. Muchas soluciones para este problema es aplicarles *fine tuning* o definir *modelos* o *capas* que adapten los *word embeddings* pre-entrenados a una tarea específica. En el presente trabajo se realiza una propuesta de un modelo utilizando *redes neuronales convolucionales (CNN*) y *Long short-term memory (LSTM)* para la adaptación de los word embeddings al dominio del objetivo propuesto.

El objetivo principal del presente trabajo es el detectar tweets noticiosos a través de un flujo obtenido, donde se considere tanto características de contenido (textuales) como de contexto (usuario). Para este propósito se ha planteado los siguientes objetivos particulares:

- Realizar un análisis del flujo de tweets obtenidos y llevar a cabo el etiquetado de los tweets noticiosos (positivos).
- Seleccionar y extraer características adicionales que ayuden a discernir con mayor precisión los tweets en el objetivo deseado.

- Diseñar y evaluar un método basado en *PU-Learning* para construir datos negativos y positivos que logren discernir entre tweets que sean noticiosos o no.
- Proponer un modelo con enfoques de aprendizaje profundo para la detección de tweets noticiosos.

El presente trabajo se organiza en 3 capítulos. En el Marco Teórico (sección 2) se describen los principales fundamentos teóricos para la solución del problema. En el capítulo de Metodología (sección 3) se detallan aspectos del análisis y desarrollo del trabajo. En capítulo de Resultados (sección 4) se realiza una serie de experimentos con diferentes configuraciones y se realiza un análisis de los resultados obtenidos. Por último, se hacen las conclusiones y se plantean ideas para trabajo futuro.

2. MARCO TEÓRICO

En esta sección se describen los principales fundamentos teóricos para el desarrollo del presente trabajo. En procesamiento de lenguaje natural la representación mediante *word embeddings* de palabras o frase aumenta el rendimiento de los modelos en diferentes tareas, en la sección 2.1 se describe algunos de los modelos de *word embeddings* más utilizados. Además, se exponen los métodos de PU-Learning, algoritmos para clasificación supervisada por el aprendizaje automático y finalmente se da una visión al entorno donde será aplicado el presente trabajo, los microblogging, específicamente Twitter.

2.1. Word embeddings

Los modelos de *word embeddings* transforman cada *palabra* o *frase* de un determinado corpus (conjunto de documentos) a un vector de números reales. Para generar estos modelos son incluidos redes neuronales, modelos probabilísticos y la representación explícita en términos del contexto en el que aparecen las palabras. El aprendizaje se realiza mediante el uso de textos sin etiquetar, mediante la predicción de contextos y modelos no supervisados.

Como estos vectores que representan coordenadas en un determinado espacio vectorial podemos calcular palabras próximas, o análogas, en función de la distancia que exista entre sus vectores. Es decir, cuanto más cerca se encuentren dos vectores (palabra o frase) éstas tendrán una semántica más similar que de encontrarse más alejados. El cálculo de esta similitud o cercanía entre vectores puede hacerse de diversas maneras siendo algunas de las más utilizadas la distancia euclidiana o la similitud del coseno.

En Fig. 1 aparecen representados algunos vectores de palabras y sus relaciones calculadas con Glove [14], el cual es capaz de capturar conceptos como que '*hombre'* es a '*mujer'* como '*rey'* es a '*reina'*, relaciones entre verbos y tiempos verbales o vincular nombres de países con sus capitales.

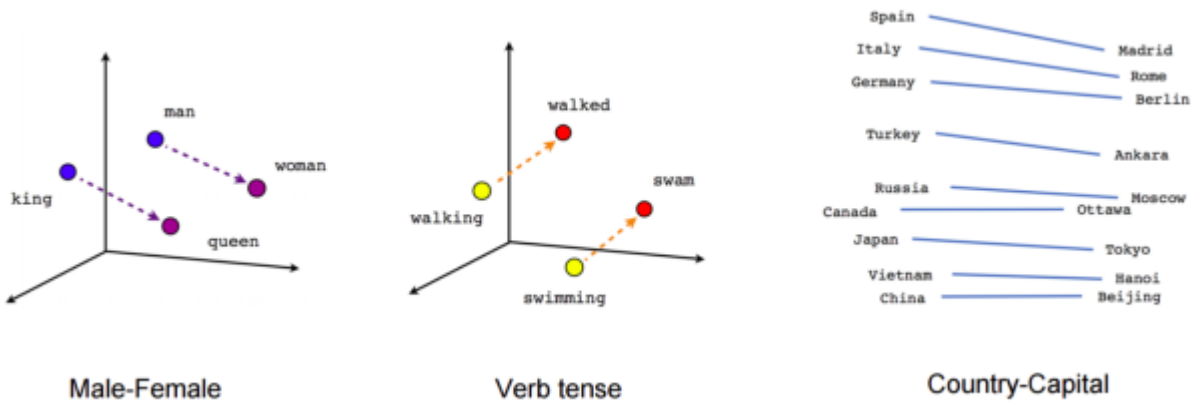


Fig. 1 Representación del concepto de los *word embeddings* capturando información relativa al género, tiempos verbales, entidades, etc [14].

A continuación, se describen los modelos más utilizados de *word embeddings* para la representación de palabras o frases.

Glove

El modelo Glove aprende los word embeddings a partir de una matriz de co-ocurrencia de términos en lugar de una tarea de predicción de palabras [14]. La matriz co-ocurrencia es de dimensiones V × V, donde V es el tamaño del vocabulario. Cada entrada de la matriz corresponde al número de veces que los elementos del vocabulario indicados ocurren juntos dentro de una ventana de contexto pre-especificada, que se mueve a través de todo el corpus. Glove aprende los word embeddings para minimizar el error de reconstrucción entre las estadísticas de co-ocurrencia predichas por el modelo y las estadísticas de co-ocurrencia global observadas en el corpus entrenado. El modelo consta de numerosos hiperparámetros, incluidos la dimensión del embeddings de los vectores y el tamaño de la ventana de contexto. Los vectores de palabras estimados usando Glove son conceptualmente similares a los derivados de word2vec [15], pero usan un modelo basado en conteo subyacente, en lugar del modelo basado en predicción de word2vec. Debido que Glove generalmente calcula estadísticas sobre ventanas de contexto más grandes que word2vec, permite capturar dependencias a más largo plazo, aunque se perderá el orden de esas dependencias. Empíricamente, no ha surgido ninguna ventaja clara para ninguno de los word2vec o Glove, ya que el rendimiento general depende de varios factores, que incluyen: el tipo de datos y la tarea de evaluación que se está considerando.

En la Fig. 2 se representa las probabilidades reales de co-ocurrencia de un corpus de 6 millones de palabras. Como se comprueba en el gráfico, el hielo ocurre más frecuentemente con el sólido que con el gas, mientras que el vapor ocurre más frecuentemente con el gas

que con el sólido. Ambas palabras coinciden con el agua ya que es una de sus propiedades compartidas, y ambas no coinciden con la moda que es una palabra con poca frecuencia para el contexto de las palabras.

Fig. 2 Probabilidades de co-ocurrencia en un corpus de 6 millones de palabras.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

FastText

FastText [13] es una extensión del modelo word2vec. Específicamente, puede manejar términos nuevos y fuera de vocabulario (OOV) al extender el modelo word2vec skip-gram (SG) con información interna de sub-palabras [9], en forma de caracteres n-gramas (por ejemplo, <*artificial*> en < *ar, art, rti, tif, ifi, fic, ici, ial, al*>). El método crea una representación vectorial para una palabra basada en la composición de estas componentes de sub-palabras, lo que permite que el modelo represente la morfología y la similitud léxica de las palabras, además de poder construir vectores para palabras invisibles. Esto ayuda a capturar el significado de palabras más cortas y permite que los *embeddings* comprendan sufijos y prefijos, y funcionen correctamente con palabras que no se ven durante el entrenamiento, lo que es una gran ventaja.

BERT

BERT [16] es otro modelo de representación de palabras contextualizado basado en un transformador-codificador bidireccional multicapa, donde la red neuronal del transformador utiliza capas de atención paralelas en lugar de recurrencia secuencial. BERT está pre-entrenado en dos tareas sin supervisión: (1) un 'modelo de lenguaje enmascarado', donde el 15% de los tokens se enmascaran al azar (reemplazados con el token "[MASK]"), y el modelo está entrenado para predecir los tokens enmascarados, (2) una tarea de 'predicción de la siguiente oración' (NSP), donde el modelo recibe un par de oraciones y está entrenado para identificar cuándo la segunda sigue a la primera. Esta segunda tarea está destinada a capturar más información a largo plazo o pragmática.

BERT está entrenado con el conjunto de datos BooksCorpus (800 millones de palabras) y texto de Wikipedia en inglés. Están disponibles dos tamaños de modelos pre-entrenados para BERT: BERT-Base y BERT-Large, con dimensiones de *embeddings* 768 y 1024 respectivamente. BERT se puede usar directamente desde el modelo pre-entrenado, o aplicar *fine-tunning* en los datos de una tarea específica.

2.2. PU-Learning

En los métodos de PU-Learning se tiene información de una sola clase, la clase positiva (en nuestro caso la clase de *tweets noticiosos*), la cual generalmente es pequeña [9]. Además, se cuenta con otra clase (la clase no-etiquetada), que contiene un gran número de instancias. A diferencia de los clasificadores de una sola clase, la clase no etiquetada es grande, pero sigue siendo una colección finita. Es posible modelarla y representa un conjunto de datos no etiquetados (*unlabeled*) que se compone de instancias de las dos clases (falsas y verdaderas). En general, los métodos PU-Learning usa una estrategia de dos pasos (Fig. 3) para construir un clasificador de dos clases [9], estos son:

- **Paso 1**: Identificar automáticamente un conjunto de *relativos negativos* (nuestro caso, *tweets no noticiosos*), el cual se forma a partir extrayéndolos del conjunto no etiquetado.
- **Paso 2**: Usar un algoritmo de aprendizaje con el conjunto de entrenamiento refinado, para construir un clasificador de dos clases.

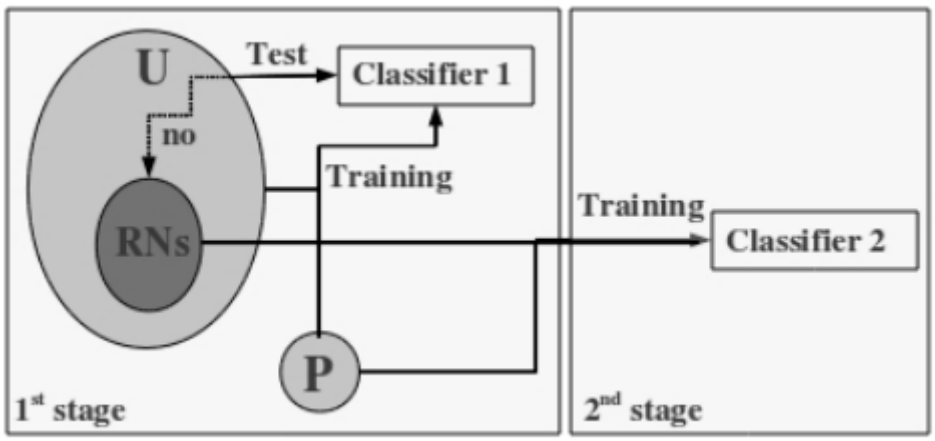


Fig. 3 Esquema de la estrategia de dos paso de los métodos PU-Learning.

PU-Learning Bagging

La estrategia planteada por Mordelet [7], *PU-Learning Bagging*, se ha aplicado con éxito para resolver varios problemas de PU-Learning. El objetivo del método es equilibrar el conjunto

de datos de entrenamiento sub-muestreando (*undersampling*) muestras no etiquetadas (U) aleatoriamente. El sub-muestreo es un método popular para tratar problemas de *desbalanceo*, que utiliza solo un subconjunto de la categoría mayoritaria (U) y, por lo tanto, es muy eficiente. Mediante el *Bagging*, evite la pérdida de información de datos durante el sub-muestreo. Dado que el algoritmo *Bagging* lleva a cabo muestreos para entrenar el modelo cada vez, tiene una fuerte capacidad de generalización y juega un papel importante en la reducción de la varianza del modelo. En Algoritmo 1 es mostrado el *pseudocódigo* del método para el caso *inductivo.* Los parámetros en para el método son: conjunto de entrenamiento que contiene solo *positivos* (P) y *unlabeled* (U), donde K = # de ejemplos de bootstrap y T =# de bootstrap.

Algoritmo 1 PU-Learning Bagging.

Entrada: P, U, K , T = # de bootstrap

Salida: P, N, U

for t= 1 **to** T **do**

Extraer subconjunto U_t de tamaño K desde U
Entrenar clasificador f_t para discriminar P de U_t

end for

$$f = \frac{1}{T}\sum_{t=1}^{T} f_t$$

PU-Learning Iterativo

El algoritmo Liu [8] para el aprendizaje de PU ha mostrado un muy buen desempeño en la clasificación de textos, al igual que derivados como el de Fusilier [9] para detección de spam. Se ha observado que su efectividad está muy relacionada con el nivel de cohesión entre los ejemplos positivos. En consecuencia, en las tareas que muestran una gran similitud entre los ejemplos etiquetados como positivos, el algoritmo *PU-Learning* tiende a hacer una buena selección inicial de las instancias negativas confiables e iteración por iteración, puede ampliar este conjunto con ejemplos negativos más relevantes.

En el Algoritmo 2 se muestra el funcionamiento de este método, la primera parte de este algoritmo (de la línea 1 a la 6) considera la identificación de un conjunto inicial de instancias negativas confiables de U. Se procede de la siguiente manera: primero, todo el conjunto U sin etiquetar se considera como la clase negativa y un clasificador es entrenado usando este conjunto junto con el conjunto P de ejemplos positivos. Luego, este clasificador se usa para

clasificar (etiquetar automáticamente) el conjunto sin etiquetar U. Las instancias del conjunto sin etiquetar clasificadas como negativas se seleccionan para formar el conjunto inicial de instancias negativas confiables (RN). La segunda parte del algoritmo (de la línea 7 a la 13) amplía iterativamente el conjunto de instancias negativas confiables al agregar algunas instancias adicionales de U. Esto se hace entrenando un clasificador binario usando los conjuntos P y RN (de la iteración anterior), y clasificar las instancias restantes en U. Las instancias de U clasificadas como negativas (Q) se agregan al conjunto de instancias negativas confiables de la iteración anterior

Algoritmo 2 PU-Learning

Entrada: P, U

Salida: P, N, U

$i = 1$

$C_i = \text{Generar\_Clasificador}(P,U)$

$Q_i = Extraer\_Negativos(U_i^L)$

$RN_i = Q_i$

$U_i = U - Q_i$

while $Q_i > \emptyset$ **do**

$i = i + 1$
$C_i = \text{Generar\_Clasificador}(P,U)$
$U_i^L = C_i(U_{i-1})$
$Q_i = Extraer\_Negativos(U_i^L)$
$U_i = U_{i-1} - Q_i$
$RN_i = RN_{i-1} + Q_i$

end for

2.3. Aprendizaje profundo

Las investigaciones actuales el área de aprendizaje profundo ha dado lugar al desarrollo de diversos tipos de redes neuronales que presentan características óptimas para el aprendizaje de ciertos tipos de información. Entre estos destacan las redes convolucionales (CNNs) [17], que han demostrado su excepcional capacidad de aprendizaje en el área de visión por computador, donde es necesario extraer correlaciones locales en estructuras espacio-temporales para el reconocimiento y clasificación de objetos. Del mismo modo, este tipo de redes, han sido aplicadas dentro del área de procesamiento de lenguaje natural (PLN) , las cuales son capaces de extraer automáticamente vectores de características sobre *n-gramas*

mediante *filtrado convolucional* y *pooling* consiguiendo aprender relaciones de más alto nivel entre los componentes del texto, tal como demuestra en sus experimentos Kim [18, 19]. Con este procedimiento se puede introducir a una red una matriz de datos (*word embeddings de palabras*) directamente, sin necesidad de "convertir" los valores en un vector unidimensional.

Existen otro tipo de arquitecturas más enfocadas al modelado de series temporales, entre las que destacan las conocidas LSTM (Long Short-term Memory RNN) [20], que incorporan un novedoso mecanismo de memoria para el modelado de dependencias a largo plazo lo que resuelve algunos de los problemas derivados del cálculo del gradiente en redes recurrentes más simples, convirtiéndose en una de las mejores opciones para el desarrollo de modelos de clasificación de texto. Como ampliación a este tipo de redes, Zhou [21] introdujo las LSTM bidireccionales (BiLSTM), las cuales añaden la capacidad de preservar información pasada y también futura.

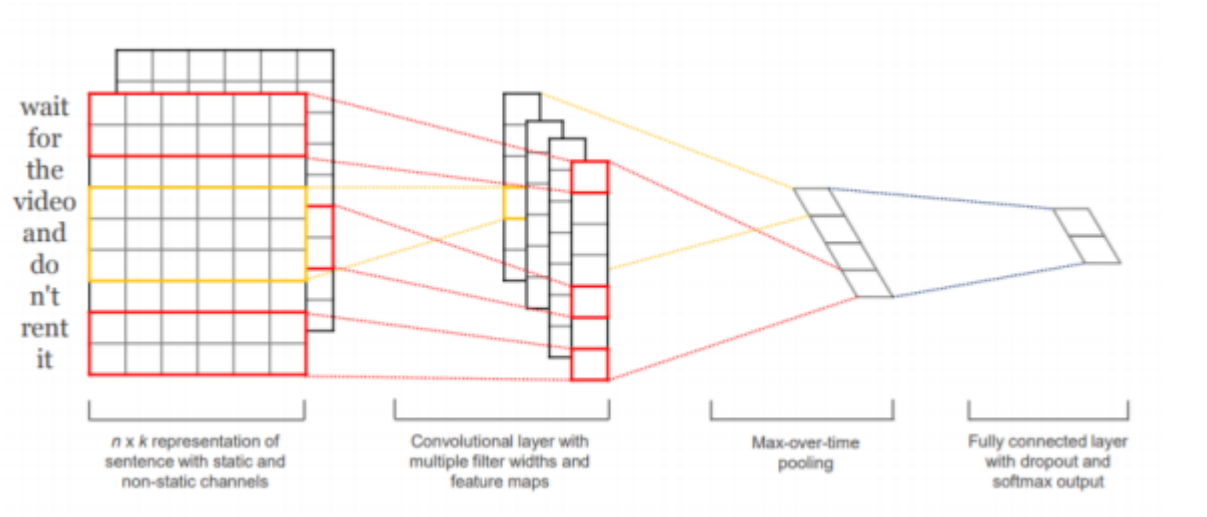


Fig. 4 Redes neuronal convolucional aplicada a texto.

2.4. Escenario de aplicación: Twitter

El escenario de aplicación del presente trabajo es la plataforma Twitter. Esta es una red social basada en *microblogging* que permite mandar mensajes de texto con un tamaño máximo de 140 caracteres, que reciben el nombre especial de *tweet*. Mediante estos textos se puede:

- Publicar textos cortos de infinidad de temas: Todo tipo de información es publicable en esta red social siempre que cumpla con las condiciones y normas de Twitter.
- Tener conversaciones públicas: Tener conversaciones y seguir otras conversaciones.

- Seguir temas de interés: Seguir temas y compartirlos con solo usar el carácter reservado *hashtag*.
- Seguir a personas: formar una red de personas y seguirlas.

Twitter posee características únicas que lo diferencian de las demás redes sociales, estas características son:

- Hashtag: Se usan para referirse a un tema.
- Símbolo @: Se refiere a una entidad o usuario.
- Emoticonos: Reflejan sentimientos directos.
- Palabras reservadas: RT que sirve para publicar temas de otras personas o páginas.
- Unicode que representa símbolos.

Esta red social genera una gran cantidad de datos por hora, es una fuente de datos casi inagotable para la tarea de procesamiento de lenguaje natural, pero estos textos poseen algunos problemas definidos por algunas investigaciones. Esto en resumen indica que los datos válidos o que reflejan algún significado válido, es de menos del 40% de los existentes. El resto de datos posee palabras sin sentido, spam, mensajes repetidos, auto promoción y texto que no valdría la pena clasificar.

2.5. Tecnologías utilizadas

Para llevar a cabo las implementaciones, se utiliza como lenguaje de programación Python3 junto con un conjunto de librerías y herramientas específicas dentro del área del aprendizaje automático, a continuación, se mencionan algunas:

- **Pandas**: es una biblioteca para manipulación y análisis de datos en el lenguaje de programación Python. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales y herramientas para su manipulación.
- **NLTK**: es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural (PLN). Alberga librerías para clasificación, tokenization, stemming, etiquetado, parsing, y razonamiento semántico (https://www.nltk.org).
- **Gensim**: Librería para el procesamiento de Lenguaje Natural (PLN) y recuperación de Información (RI). Utilizada frecuentemente para el uso de word embeddings (word2vec, Glove, FastText, etc.) (https://pypi.org/project/gensim).
- **Scikit-Learn**: es un conjunto de librerías para aprendizaje automático, presenta algoritmos para clasificación, regresión y clustering. Técnicas de clasificación como:

SVM, Random Forest (RF), Gradient Boosting (GB) y Logistic Regression (LR) son incluidas (http://scikitlearn.org).

- **Tensorflow**: Framework desarrollado originalmente por investigadores e ingenieros del equipo Brain de Google dentro de la organización de investigación en Inteligencia Artificial de Google para investigaciones y desarrollo de aprendizaje automático y redes neuronales profundas. (https://www.tensorflow.org).
- **Guardian Open Plataform**: es un conjunto de servicios web que hace posibles usuarios poder consultar artículos de la base de datos de *The Guardian (guardian.co.uk)*. La plataforma actualmente incluye la *Content API*. Mediante esta API, es posible consultar artículos de su base de datos y recuperarlos en formatos orientados a la integración con otras aplicaciones (por ejemplo, JSON). Los métodos habilitados son mediante *contenido, etiquetas, secciones* y otros *elementos individuales* (https://open-platform.theguardian.com/).

3. METODOLOGÍA

En esta sección se detallan todo el análisis y proceso de implementación que se ha llevado a cabo en el presente trabajo. En la Fig. 5 se muestra el diagrama del procesamiento efectuado en el presente trabajo.

En primer lugar, se hace un análisis del flujo de tweets obtenido por el API de Twitter. Estos tweets son pre-procesados para un mejor procesamiento con posterioridad. A continuación, se extraen características que nos ayuden a discernir las clases planteadas (en este caso tweets noticiosos o no). Luego se etiquetan el conjunto de datos que se tiene en *Positivo* y *Unlabeled*, para pasar al proceso de PU-Learning, con el que obtendremos el conjunto de *positivos*, negativos y el resto seguiría como no etiquetado (*unlabeled).* Como se dió el problema de que en el proceso de etiquetado se obtuvo un conjunto positivo menor del 5% del conjunto *unlabeled* se toma la decisión de crear muestras sintéticas (*aumentación de datos*) para el periodo de aprendizaje. La aumentación de datos es utilizada tanto para el PU-Learning como el modelo propuesto.

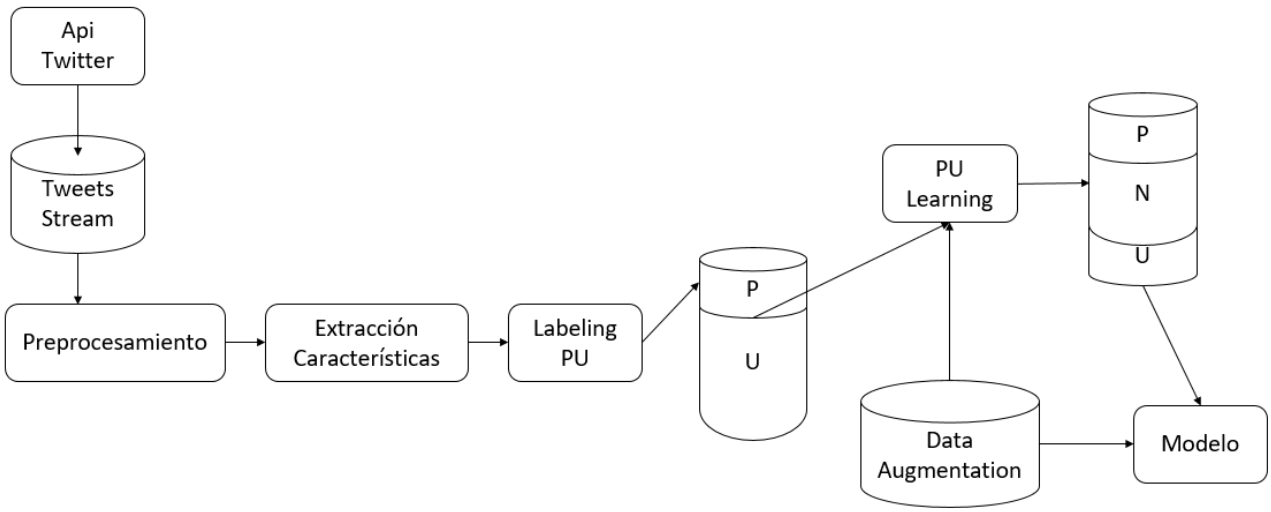


Fig. 5 Metodología para el procesamiento.

Datos obtenidos del flujo:

El conjunto de datos de partida obtenido consta de un flujo de tweets obtenido desde el año 2012 hasta 2018, el cual posee 1733162 de tweets en el dominio de automóviles. De este conjunto se proveen diferentes métricas del tweets publicado y del usuario, estas son descritas en Tabla 1.

Tabla 1 Métricas obtenida del flujo de tweets.

Tweets	Usuario
• **tweet_id**: ID del tweets. • **replies**: Si el tweet tiene respuestas. • **retweets**: Si ha sido re-twitteado. • **favourites**: Cantidad de favoritos. • **fecha**: Fecha de publicación del tweet. • **idioma**: Idioma en que se ha creado el tweet • **texto**: El texto publicado en el tweet • **coherencia**: Indica el nivel de coherencia del tweet (cuanto menor más coherencia)	• **user_id**: ID del perfil • **statuses**: El estatus del usuario • **listed**: Veces que ha sido listado el usuario • **friends**: Son los amigos que el usuario sigue. • **followers**: Seguidores del usuario • **tweets_usuario_stream**: Cantidad de veces que ha publicado un tweet en el conjunto de datos que se posee. • **verificado**: Si el usuario esta verificado por twitter • **coherencia**: Es la coherencia que tiene el usuario (cuanto menor más coherencia posee) • **idioma**: Idioma del perfil. • **lugar**: Donde está ubicado el usuario. • **screen_name**: El screen_name del usuario en twitter (ejemplo @ford) • **fecha**: Fecha de registro del usuario en twitter

3.1. Pre-procesamiento y extracción de características

Se realiza el pre-procesamiento de los tweets con el objetivo de lograr una limpieza de los datos y un óptimo procesamiento con posterioridad. Para esto se tiene en cuenta la naturaleza y características de los *tweets*.

Una serie de elementos característicos de los tweets son sustituidos por etiquetas. En la Tabla 2 se muestran algunos ejemplos antes y después da las sustituciones. A continuación, se muestran algunas de las trasformaciones:

- *emoticonos (ej: {:loudly_crying_face:}) -> emojins*
- *hashtag (#federacion)* -> hashtag
- *Las menciones ejemplo (@ford)* -> mentions
- *Los email (xx@uji.es)* -> email

Tabla 2 Ejemplos de tweet plano antes (columna izquierda) y después (columna derecha) de las sustituciones por las etiquetas.

#atencion la nueva kangoo ya se puede probar en #federacion {ret} {ret} estará en la avenida comercial durante toda la {lnk}	*{hashtag} la nueva kangoo ya se puede probar en {hashtag} {ret} {ret} estará en la avenida comercial durante toda la {lnk}*
dad {qmark} s getting a nissan pathfinder {:loudly_crying_face:} {:loudly_crying_face:} {:face_with_tears_of_joy:}	*dad {qmark} s getting a nissan pathfinder {emoji} {emoji} {emoji}*

Posteriormente, se pasa a extraer *características del contenido del tweet y del usuario*, que no se tenían del conjunto de datos original. Del texto original del tweet se extraen características como: cantidad de *caracteres*, *signos de exclamación e interrogación, emojins, urls,* palabras de *sentimientos positivos y negativos.* Además, se transforma el texto del *tweet plano* (no el original) en *word embeddings.* Del usuario se extraen otras características derivadas de la que contamos como: el tiempo que tiene el usuario registrado, medido en meses.

En Tabla 3 se muestran todas las características que nos quedaremos para todo el proceso posterior de etiquetado y aprendizaje, donde no se tienen en cuentas algunas que se poseen en el conjunto original (ejemplo: fecha).

Tabla 3 Características para el proceso de aprendizaje

	Características	Descripción
Contenido	char	# caracteres
	words	# palabras
	exclamation_marks	# signos de exclamación
	question_marks	# signos de pregunta
	emojins	# emojins
	urls	# urls
	sentimiento_pos	# palabras sentimiento positivo
	sentimiento_neg	# palabras sentimiento negativo

	hashtags	# número de hashtag
	lexical_diversity	Proporción de palabras por tweets
	users_mentions	# Menciones a usuarios en el tweets
	uppercasses	# Numero de caracteres en mayúscula
	coherencia_tweet	Coherencia del tweets, a menor valor más coherencia.
	links	Número de links que posee el tweets
	reply	Es una respuesta o no
	EMBEDDINGS	Embedding del tweets
Usuario	u_month_registration	# meses que lleva registrado en twitter
	u_statuses	El estatus del usuario
	u_friends	# de amigos
	u_followers	# de seguidores
	u_bal_soc	u_followers / (u_followers + u_friends)
	u_coherencia_user	

3.2. Etiquetado

El flujo de tweets que se posee carece de un etiquetado. Por lo tanto, se plantea realizar un etiquetado semi-supervisado teniendo en cuenta el objetivo planteado, en nuestro caso crear un modelo para la detección de *tweets noticiosos*.

Para este objetivo después de realizado un estudio de los datos y las características típicas de las noticias se define unas serie de reglas para obtener del flujo un grupo de tweets noticioso (P) y un grupo no etiquetados (*unlabeled* U). A continuación, se describen como se seleccionan las muestras:

Positivos (P):

Las muestras tomadas como positivos son los tweets del flujo que cumplen todas las condiciones siguientes.

- Tweets con usuarios que cumplan:
    - Al menos 10000 seguidores.
    - Verificados.
    - Cuentas creadas antes de enero de 2012.
- Tweets que en el flujo este repetido al menos 10 veces.
- Tweets que no sean respuesta.

En Tabla 4 se muestra ejemplos de tweets etiquetados como positivos teniendo en cuenta las consideraciones anteriores descritas.

Tabla 4 Ejemplos de etiquetados Positivos

Tweets	Usuario
'the ford focus is dead in the us because of trade war {lnk} {lnk}'	verge
'los autos toyota hilux son los favoritos del estado islamico - {lnk} {lnk}'	noticia24
'#rav4hybrid is now available from full price and spec details are available here : {lnk} {lnk}'	ToyotaGB

Unlabeled (U):

Las muestras tomadas como *unlabeled* son los tweets que no cumplieron las condiciones para ser positivos y cumplen al menos una de las siguientes:

- Tweets de usuarios verificados.
- Tweets con usuarios con más de 10000 seguidores (*followers*).
- Tweets que en el flujo este repetido al menos 50 veces.

En la Tabla 5 se muestran ejemplos de tweets definidos como *unlabeled*. Se muestra como el tweet escrito por el perfil '*@kblock43'* el cual cuenta con una gran suma de seguidores y verificado (481k) no se tuvo en cuenta como noticioso por la razón de estar solamente repetido 1 vez en el flujo de datos. En los casos de los tweets escritos por *'@e7_radio'* y '*@Diariocadiz'* se descartaron como positivo por no estar verificado por twitter. Señalar que estos tweets posteriormente en el proceso de PU-Learning pueden no ser seleccionados como negativos.

Tabla 5 Ejemplos *Unlabeled*

Tweets	Usuario	Repetición Tweet en el flujo
'qualifying is done here in canada sitting in p2 overall and {mention} is p4 . that means pole position {lnk}'	kblock43 (Verificado) *481 K seguidores*	1
'ford presenta el nuevo focus con inteligencia artificial y tecnología audiovisual {lnk}'	e7_radio (No Verificado)	21
'the {mention} hilux still dominates sales in sa and toyota is the best-selling car brand in all the land . more {lnk}'	*CarsSouthAfrica (No verificado)*	1
'2013 nissan pathfinder - jersey city , nj : {lnk} via {mention}'	*Njautoauction (No verificado)*	1
'{hashtag} {bar} nissan duplicara en 2018 las ventas de la e-nv200 , la furgoneta electrica fabricada en españa {lnk}'	*Diariocadiz (No verificado)*	9
'vendo opel corsa'	*MeDicenIsma Followers: 381 No verificado*	310

Negativos:

Las muestras tomadas como *negativas* son los tweets que se mostraron no creíble como noticiosos. Estos tweets se caracterizan en general por:

- Tener diferentes contenidos del tweet y que tienen forma de noticia.
- Usuarios diferentes creados en un periodo de tiempo corto
- Introducen en el texto de las publicaciones *hashtag (*ej. #automotive, #topseed, etc.*).*

3.3. Aumentación de datos

Se ha mostrado la carencia de muestras positivas y la proporción con relación a los casos negativos es de menos del 10% por lo que evidencia un claro desbalanceo de las clases. Por lo expuesto se toma la decisión de crear muestras *sintéticas*.

Para la tarea de la *aumentación de datos positivos sintéticos* se tiene en cuenta:

- Las *características de usuario* pueden ser creadas tomando aleatoriamente características de los usuarios de su clase (*muestras positivas*).
- Los tweets sintéticos se construyen mediante el uso de la *The Guardian Open Plataform*.

Mediante el uso de la plataforma *The Guardian Open Plataform* se recuperan noticias haciendo uso de *hashtag*, *palabras* y *bi-gramas* más repetidas en las clases positiva.

En la Fig. 6 se muestra el *WordCloud de Hashtag* de la clase positiva, donde observamos los *hashtag* más repetidos que se tendrán en cuenta para la recuperación de noticias (ejemplo de *hashtag* más repetidos: *#ford, #citroen, #seat, '#renault*). En la Fig. 7 se observa las palabras más repetidas sin tener en cuenta los hashtag, por ejemplo: *ford focus, nuevo opel*, *land rover*.

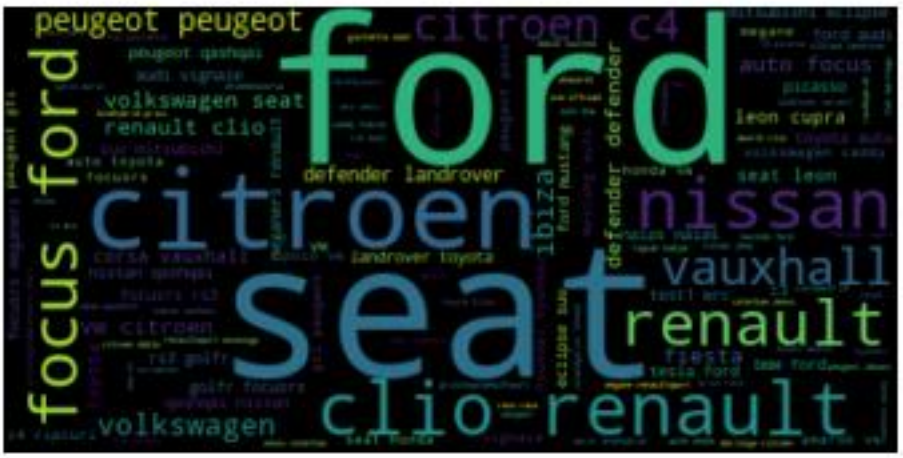


Fig. 6 *WordCloud* de *Hashtag* en la clase positiva.

Fig. 7 *WordCloud* de *palabras* en la clase positiva

Luego cada titular de noticia es procesado para transformarlo en los tweets, teniendo en cuenta los siguientes aspectos: se les agregan a entidades # o @ con (por ejemplo, *'ford' -> '#ford'* o *'citroen' -> '@citroen'*), además se le agrega la etiqueta '*lnk'* a los tweets. Para estas transformaciones se explora el comportamiento y probabilidades en cada caso a cómo se comportan sus frecuencias en los tweets ya etiquetado como positivos Se muestran algunos ejemplos de tweets sintéticos creados por esta aproximación en la Tabla 6.

Tabla 6 Tweets sintéticos creados a través de noticias recuperadas por *The Guardian Open Plataform*

Titulares de noticias recuperadas	Transformación en tweets sintéticos
'Fiat Chrysler proposes merger with Renault to reshape car industry'	*#fiat_chrysler proposes merger with @renault to reshape car industry*
Vauxhall says it won't shy away from the 'dark side' in no-deal Brexit	*'#vauxhall says it won't shy away from the 'dark side' in no-deal Brexit {lnk}'*
BMW moves some engine production out of UK over Brexit fears	*@BMW moves some engine production out of UK over Brexit fears {lnk}*
Mandelson fights threat to Vauxhall jobs	*Mandelson fights threat to Vauxhall jobs {lnk}*

3.4. PU-Learning

En este caso se propone a través del PU-Learning construir la clase negativa y a la vez crear una clase de positivos relativos (PR). Se implementan las dos aproximaciones descritas en la sección 2.2.

En *Algoritmo* 3 se muestra el pseudocódigo de la implementación hecha para la aproximación de PU-Learning Bagging. Se tiene en cuenta que son introducidos dos umbrales α y β para definir el conjunto *negativos*, que se define por las muestras de no etiquetados que su clasificación promedio tuvo una probabilidad menor o igual ***α***, en el caso de *positivos relativos* se tiene en cuenta que sea mayor o igual a ***β***. Finalmente se construye el clasificador final con los datos *positivos* y *negativos.*

Algoritmo 3 PU-Learning Bagging.

Entrada: P, U, K , T = # de bootstrap, α y β umbrales para definir N y PR.

Salida: P, N, U, PR, clasificador_final

for t= 1 **to** T **do**

Extraer subconjunto U_t de tamaño K desde U
Entrenar clasificador f_t (P,U_t)
$prob_t$= Clasificar(U)

end for

$$f = \frac{1}{T}\sum_{t=1}^{T} prob_t$$

$N = f \leq \alpha$

$PR = f \geq \beta$

clasificador_final = Entrenar clasificador f_t con P, N y PR

En PU-Learning iterativo se describe su proceso en pseudocódigo en el Algoritmo 4. En este caso, igualmente, tiene en cuenta que son introducidos dos umbrales α y β para definir los conjuntos. Los *negativos* son las muestras del conjunto no etiquetado que en cada iteración su clasificación tiene una probabilidad menor o igual ***α***. En el caso de *positivos relativos* se tiene en cuenta que sea mayor o igual a ***β*** al final de la última iteración. Finalmente se construye el clasificador final con los datos *positivos* y *negativos.*

Algoritmo 4 PU-Learning iterativo.

Entrada: P, U, α y β umbrales para definir N y PR.

Salida: P, N, U

$i = 1$

$C_i = \text{Generar\_Clasificador}(P,U)$

$Q_i = Extraer\_Negativos(U_i^L)$

$RN_i = Q_i$

$U_i = U - Q_i$

while $Q_i > \emptyset$ **do**

$i = i + 1$
$C_i = \text{Generar\_Clasificador}(P,U)$
$U_i^L = C_i(U_{i-1})$
$Q_i = Extraer\_Negativos(U_i^L, \alpha)$
$U_i = U_{i-1} - Q_i$
$RN_i = RN_{i-1} + Q_i$

end for

$N = RN_i$

$PR = C_i(U_i^L) \geq \beta$

Entrenar **clasificador_final** f_t con P, N y PR

3.5. Modelo

Se propone un modelo basado en aprendizaje profundo para la detección de tweets noticiosos, en este se utiliza los conjuntos de datos obtenidos por el PU-Learning (negativos) y obtenidos por el proceso de etiquetado (positivo).

Los modelos de *word embeddings* pre-entrenados son de mucha utilidad y tienen una óptima representación de frases (BERT) o palabras (Glove o FastText). En cambio, muchos de estos modelos son pre-entrenados sobre un dominio específico y carecen de las peculiaridades de la tarea que se vaya a desarrollar. Por lo tanto, en este trabajo, se propone un modelo de aprendizaje profundo capaz de utilizar estas aproximaciones y adaptarla al objetivo propuesto.

En la Fig. 8 se observa la arquitectura del modelo propuesto. Este consta de dos etapas:

1. Se entrena el modelo de aprendizaje profundo (parte superior) solo con los textos de los tweets etiquetados como positivos y negativos en base al objetivo especio, en este caso detección de tweets noticiosos, el cual llamaremos '*CNN+LSTM+DNN'*
2. Se entrena el clasificador final (parte inferior) con la salida de la capa LSTM previamente entrenado para cada tweet las otras características del tweet que se poseen (ejemplo, métricas del usuario), el cual llamaremos '*CNN+LSTM+RF'*, por el uso del clasificador Random Forest.

Teniendo en cuenta esta propuesta se puede obtener una predicción a través de solo los tweets utilizando solo el modelo de aprendizaje profundo pero no sería tan eficaz como el uso de toda la característica adicional al texto, como se comprobara en la sección 4.4.

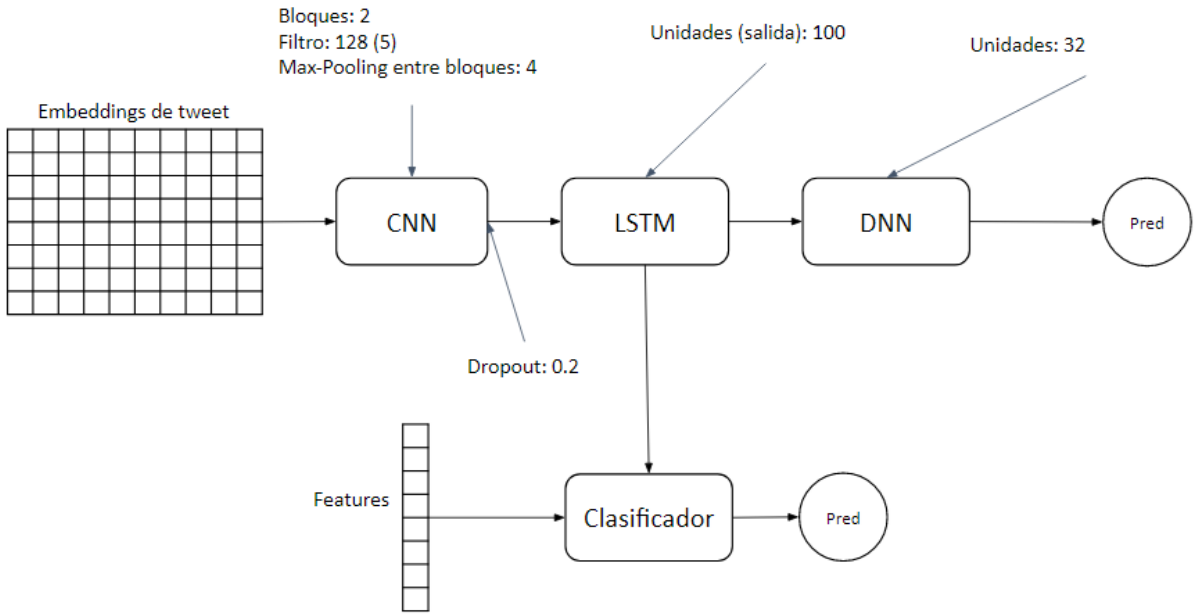


Fig. 8 Modelo propuesto.

El modelo propuesto consta de las siguientes características.

Arquitectura:

Capas convolucionales:

- Números de bloques: 2.
- Números de filtro: 128.
- Longitud (Kernel): 5

MaxPooling1D (capa ubicada entre bloques convolucionales):

- Longitud: 4

Dropout (capa ubicada entre CNN y LSTM):

- dropout_rate: 0.2

LSTM:

- Número de unidades (salida): 100

Entrenamiento:

- Épocas = 10
- Batch sizes=32
- Funcion de perdida (loss): ‘binary_crossentropy'
- Optimizador = 'adam'

4. RESULTADOS

En esta sección se muestran y analizan los resultados del presente trabajo. En primer lugar, se hacen las validaciones correspondientes para el PU-Learning con las aproximaciones desarrolladas, parámetros y algoritmos de clasificación. Posteriormente, con los conjuntos de datos positivos y negativos obtenidos por el PU-Learning se hace un estudio de la importancia de grupos de características utilizadas. Finalmente se describen y comparan los resultados del modelo propuesto (sección 4.4) para la clasificación final.

4.1. Configuración de los experimentos.

En el proceso de validación se ha realizado experimentos en diferentes entornos. A continuación, se describen cada configuración experimentada.

PU-Learning

En la Tabla 7 se muestran el conjunto de datos para el proceso de PU-Learning. El conjunto de *positivo*, *negativo* y *no etiquetado* son obtenidos a través del proceso de etiquetado descrito en la sección 3.2, además se tiene el conjunto de datos sintéticos creados para el proceso de entrenamiento.

Tabla 7 Conjunto de datos para el PU-Learning.

	Positivos (P)		**Negativos (N)**	**No etiquetado (U)**
		Aumentación de datos (sintéticos)		
Entrenamiento	2273	28752		148284
Validación	758		25536	

En la Tabla 8 se especifican las distintas configuraciones para la validación del proceso de PU-Learning. Como se observa son probados los mismos clasificadores y la aumentación de datos para ambas aproximaciones de PU-Learning.

Tabla 8 Configuraciones para PU-Learning

	Clasificadores	**Parámetros probados**
Mordelet	Regression logística (LR), Random Forest (RF), xGBoost (XGBoost)	T = 10, 20, 50, 100 K = longitud de conjunto positivos Aumentación de datos: *si* y *no*
PU-Learning iterativo		i = 10 Aumentación de datos: *si* y *no*

En los experimentos siguientes al PU-Learning se utilizada el conjunto negativo resultante por la mejor aproximación. Como se demostró en la sección 3.4, el PU-Learning Bagging fue el de mejor resultado, por lo se utiliza su conjunto de datos negativos resultantes.

Estudios de características

Se realiza un estudio de los grupos de características en el modelo con mejor resultados en el PU-Learning y los conjuntos obtenidos. En este caso son probadas las siguientes combinaciones:

- Embeddings.
- Características de usuario.
- Características de contexto (sin *embeddings*) + usuarios.
- *Embeddings* + usuarios.
- Todas las características.

Modelo propuesto

Es validado el modelo propuesto con el conjunto de datos obtenido por el proceso de PU-Learning. El conjunto de datos con que se validaran son: positivos *etiquetados* y *sintéticos*, y los obtenidos como *negativos*.

4.2. PU-Learning

Se realizan experimentaciones con el conjunto de datos previamente etiquetados como positivo (*noticiosos*) y no etiquetados. Primeramente, sin aumentación de datos y posteriormente con aumentación de datos (*sintéticos*). Estos son realizados en las dos

aproximaciones *PU-Learning Iterativo* y *Bagging*, en el caso de este último se realizan experimentos con varios valores para T (número de de *bootstrap*).

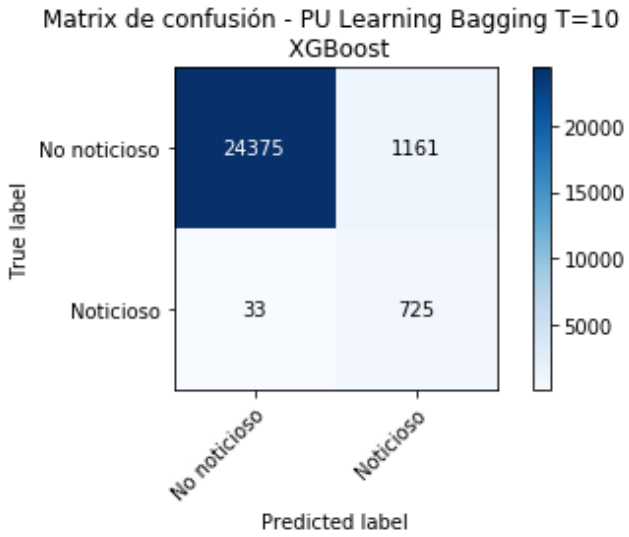
Experimentos sin aumentación de datos

En la Tabla 9 se muestran todos los resultados obtenidos para el PU-Learning Bagging sin aumentación de datos. Como se evidencia de manera general los resultados son bastante malos. Con *precisión* y *F1 score* muy baja, y *recall* y *accuracy* alto, lo que demuestra que se clasifica la mayoría del conjunto de validación en la clase mayoritaria. En estos resultado solo *es superior pero con muy bajo rendimiento* cuando se utiliza el clasificador XGBoost donde el *F1 score* sube hasta 0.55 cuando $T = 10$ y *precisión* 0.38. En la Fig. 9 se visualiza la matriz de confusión para este último caso, en el que se verifica que un mayor de muestras negativos es clasificado como positivos (FP), esto trae como consecuencia baja precisión.

Tabla 9 PU-Learning Bagging sin aumentación de datos.

Modelo	**T**	**Precisión**	**Recall**	**F1 score**	**Accuracy**
LR	10	0.03	0.99	0.07	0.19
	20	0.03	0.99	0.07	0.19
	50	0.03	0.99	0.07	0.19
	100	0.03	0.99	0.07	0.19
RF	10	0.04	0.98	0.09	0.39
	20	0.04	0.98	0.07	0.28
	50	0.04	0.98	0.07	0.25
	100	0.04	0.98	0.07	0.26
xGBoost	10	**0.38**	**0.96**	**0.55**	**0.95**
	20	0.35	0.95	0.51	**0.95**
	50	0.33	**0.96**	0.49	0.94
	100	0.32	**0.96**	0.48	0.94

Fig. 9 Matriz de confusión para PU-Learning Bagging – XGBoost – T=10

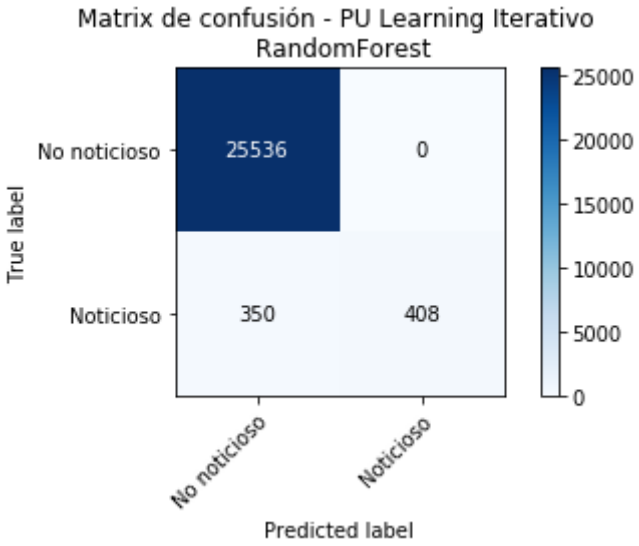


En *PU-Learning Iterativo* sin aumentación de datos se obtiene nuevamente resultados (Tabla 10) bajos. En este caso, las *precisiones* son bastante alta, hasta llegar al 100 % utilizando como clasificador RF y XGBoost, pero al contrario decrecen los valores de *recall*, esto se debe a que las muestras negativas se clasifican en su caso todas o casi todas correctamente (falso positivos bajos) y más de la mitad de positivas son predichas como negativos. EL algoritmo de clasificación de Random Forest es el de mejor resultado llegando alcanzar un 0.7 de F1-score, lo que es superior a los experimentos anteriores, pero la precisión es de solo 0.43. Este último caso es mostrado su matriz de confusión en la Fig. 10, donde se demuestra que no existe falso positivos, pero más de la mitad de positivos son clasificados como negativos (Falso Negativo), por lo que se obtiene un bajo *recall*.

Tabla 10 Resultados de PU-Learning Iterativo sin aumentación de datos.

	Precisión	**Recall**	**F1 score**	**Accuracy**
XGBoost	**1**	0.34	0.50	0.98
RF	**1**	**0.54**	**0.70**	**0.99**
LR	0.86	0.03	0.06	0.97

Fig. 10 Matriz de confusión para PU-Learning Iterativo - RF



Experimentos con aumentación de datos

Por el bajo rendimiento que se obtuvo por el desbalanceo, donde las muestras *positivas* son menores al 5 % de las *no etiquetadas* se realiza la aumentación de datos explicada en la sección 3.3.

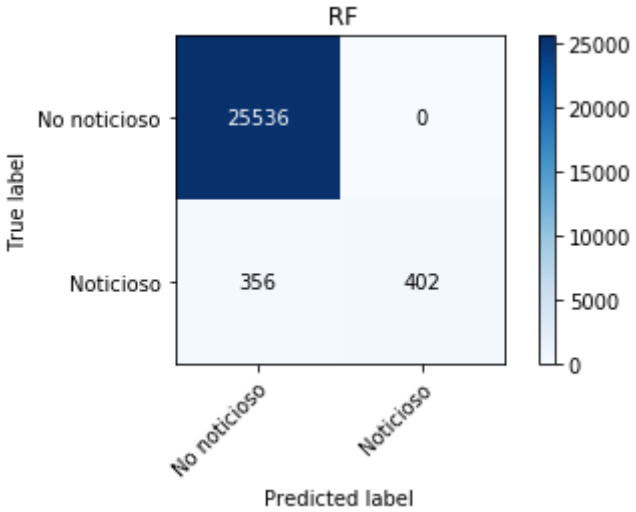
En la Tabla 11 se especifican los resultados para PU-Learning Iterativo **con** aumentación de datos. Los resultados en general no son buenos, al igual que este mismo experimento, pero sin aumentación de datos tiende a tener una *precisión alta* pero un *recall* bajo excepto para el LR. El PU-Learning Iterativo con Random Forest obtiene los mejores resultados, en este caso la *precisión* y *F1-score* son semejante al mejor resultado que sin aumentación de datos. Por lo tanto, para el PU-Learning Iterativo con aumentación de datos no existe mejora con respecto a los anteriores.

Tabla 11 Resultados de PU-Learning Iterativo CON aumentación de datos.

	Precisión	**Recall**	**F1 score**	**Accuracy**
XGBoost	**1**	0.35	0.52	0.98

RF	**1**	**0.53**	**0.69**	0.99
LR	0.03	0.98	0.06	0.19

Fig. 11 Matriz de confusión para PU-Learning Iterativo- Aumentación de datos - RF



El PU-Learning Bagging se experimenta igualmente **con** aumentación de datos, en la Tabla 11 se especifican sus resultados. Como se evidencia existe una mejora con respeto a los experimentos anteriores excepto para el caso de Regresión Logística, que siguen siendo bajos los resultados Para los casos donde se utiliza Random Forests y XGBoost, en cualquiera de sus configuraciones, la medida *f1-score* ronda los 0.81 y 0.80 respectivamente y en el *recall* de 0.67, estos son resultados superiores a los obtenidos anteriormente. En la Fig. 14 se muestra la matriz de confusión para el caso con RF, donde no existen muestras de falsos positivos, por lo que se tiene una buena del 100%; en cambio, el caso de falsos negativos aun es un porciento significativo de los positivos por lo que afecta el *recall*.

Fig. 12 PU-Learning Bagging con aumentación de datos.

Modelos	**T**	**Precisión**	**Recall**	**F1 score**	**Accuracy**
LR	10	0.03	0.99	0.07	0.19
	20	0.03	0.99	0.7	0.19
	50	0.03	0.99	0.7	0.19
RF	10	**1**	**0.68**	**0.81**	**0.99**

	20	**1**	**0.68**	**0.81**	**0.99**
	50	1	0.67	0.80	0.99
XGBoost	10	0.99	0.67	0.80	0.99
	20	0.99	0.67	0.80	0.99
	50	0.98	0.68	0.80	0.99

Fig. 13 PU-Learning Bagging con aumentación de datos – XGBoost

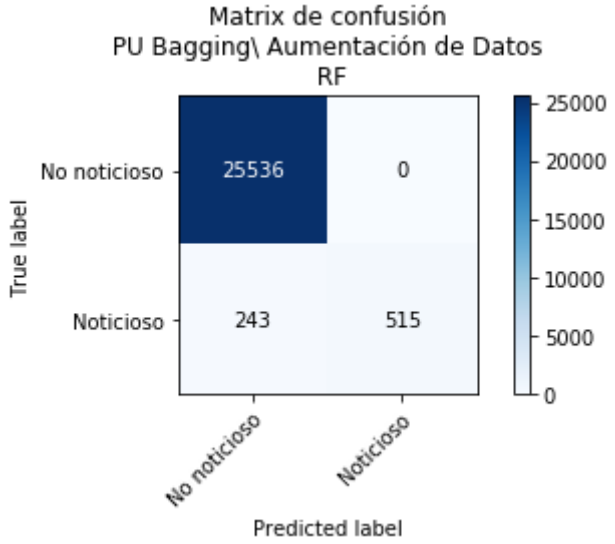


Fig. 14 PU-Learning Bagging con RF y aumentación de datos.

Comparación de las aproximaciones

En la Fig. 15 se realiza una comparación de los métodos con mejores resultados en cada aproximación y algoritmos de clasificación (RF y XGBoost) utilizados, con respeto a la métrica *F1-score*. En el caso del PU-Learning iterativo con Random Forest obtuvo los mejores resultados sin aumentación de datos alcanzando un 0.69, pero en ningún caso de los iterativos mejoro cuando fueron incluidos los datos sintéticos, manteniéndose con resultado similares. En cambio, es destacable, en el *PU bagging*, se mejoran sustancialmente los resultados al incluir muestras sintéticas, llegando hasta un 0.81 y 0.80, para los clasificadores Random Forest y XGBoost respectivamente.

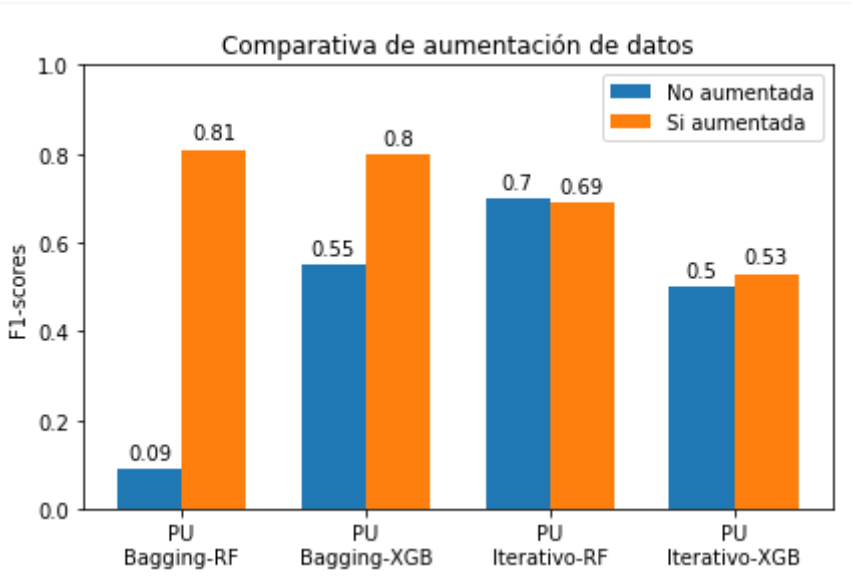


Fig. 15 Comparación con y sin datos sintéticos.

4.3. Estudios de características

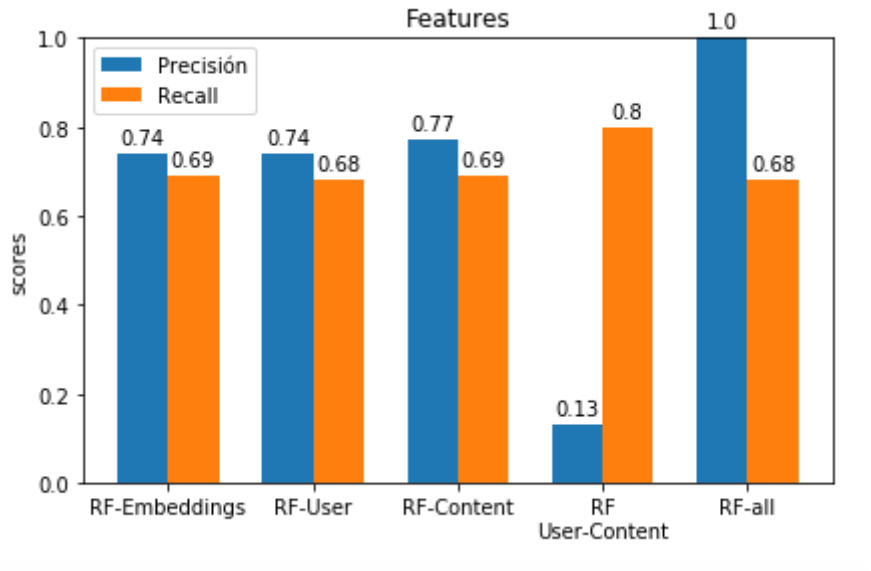
Se realiza una comparativa de diferentes grupos de características para ver su importancia en el clasificador final. El clasificador utilizado para este experimento es el Random Forest y se validan los grupos siguientes:

- Solo *embeddings (*en este caso BERT*)*
- Métricas del usuario.
- Métricas del contenido del tweet (*coherencia del tweet*) y características extraídas (ejemplo: # *palabras*)
- Métricas del usuario y contenido del tweet (*sin embeddings*)
- Todas.

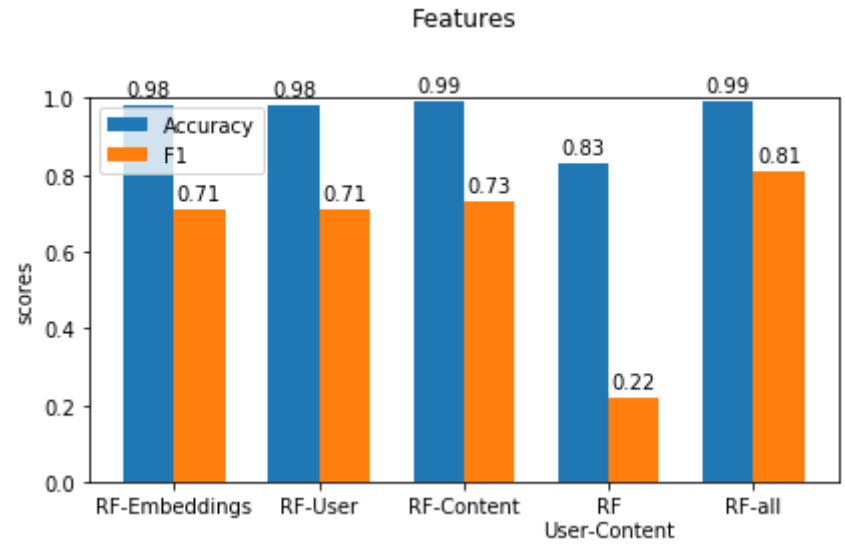
En la Fig. 16 se comparan los diferentes grupos de características con el conjunto de validación. En la Fig. 16 a) se evidencia una *precisión* entre los 0.74 y 0.80 en todos los grupos, excepto cuando se utiliza características de usuario y contenido que baja hasta los 0.13. En cambio, el clasificador solo tiene una alta precisión cuando son utilizadas todas las características llegando alcanzar un 100% y el *recall* se mantiene entre los 0.69 y 0.80 en todos los casos.

En Fig. 16, b) se compara el *accuracy* donde todos lo mantienen alto por la gran cantidad de muestras negativas y pocas positivas para la validación, excepto nuevamente cuando se utiliza las características de usuario y contenido donde baja hasta un 0.8. En el caso del *f1-*

score se mantiene sobre los 0.71 a 0.73 y solo se obtiene un 0.81 cuando se utiliza todas las características en su conjunto y baja 0.22 con características de usuario y contenido.



a)



b)

Fig. 16 Comparativa de Random Forest con diferentes grupos de características, a) precisión y recall, b) accuracy y F1-scrore.

4.4. Modelo propuesto

En esta sección se valida el modelo propuesto en la sección 3.5. Se realiza los siguientes experimentos.

- Word embeddings con red convolucional (CNN), red neuronal recurrente (LSTM) y red neuronal densa (DNN). En esta aproximación solo es utilizado el texto del tweet. Se experimenta con los modelos de Word embeddings: Glove y FastText
- Se sustituye la capa DNN por el clasificador Random Forest y se utilizan adicionalmente las otras características adicionales al texto del tweet.

En la Tabla 12 se muestran los resultados obtenido por cada configuración. Como se observa, cuando solo es utilizado el modelo '*CNN (Glove)+LSTM+DNN'* con solamente el texto de los tweets se obtienen resultados de 0.94 *precisión* y 0.55 de *recall*, los cuales son *superiores* a los anteriores (sección 4.3, cuando no se utilizaba todas las características) y solo *inferior* cuando se utilizaban todos los grupos caracterizas en conjunto con el clasificador RF o XGBoost. Además, se destaca que cuando se usa *Glove* se obtienen mejores resultados que *FastText,* en cuanto a *precisión*, *recall* y *F1 score*.

En cuanto al modelo propuesto con '*CNN+LSTM+RF'* se obtienen resultados superiores a los obtenidos al caso de utilizar todas las características (incluido *embeddings* BERT) con RF, ver Tabla 12. En '*CNN+LSTM+RF' se llega hasta una precisión* de 0.98*, recall* 0.76 y *F1-score* de 0.86 el cual mejora en 0.5 al RF. La matriz de confusión para '*CNN+LSTM+RF'* se muestra en Fig. 17, donde se evidencia que pocas muestras negativas son clasificadas como positivo (falso positivo bajo), aunque aún una cantidad considerable de muestras positivas son clasificadas negativas (falsos negativos), pero sustancialmente mejor que RF, lo que conlleva a lograr un *recall* de 0.76.

Tabla 12 Resultados de aproximaciones con Word embeddings, CNN y LSTM y RF.

Modelos	**Características**	**Precisión**	**Recall**	**F1 score**	**Accuracy**
CNN(Glove)+LSTM+DNN	*Solo texto del tweets*	0.94	0.55	0.70	**0.99**
CNN(FastText)+LSTM+DNN	*Solo texto del tweets*	0.83	0.51	0.63	0.98
CNN+LSTM+RF	*Texto del tweets y otras*	0.98	**0.76**	**0.86**	**0.99**

Embeddings (BERT) - RF	*Solo texto del tweets*	0.74	0.69	0.71	0.98
RF	*Todas*	**1**	0.68	0.81	**0.99**

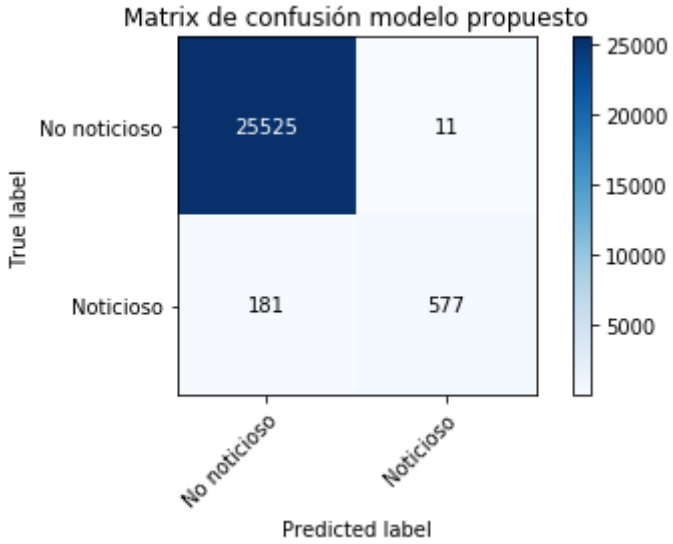


Fig. 17 Matriz de confusión para Word embeddings (Glove) + CNN + LSTM + RF.

5. CONCLUSIONES Y TRABAJO FUTURO

En el presente trabajo se desarrolló un método para la detección de tweets noticiosos dado un flujo de datos obtenidos de la plataforma Twitter. Para la solución del objetivo se realizó primeramente un pre-procesamiento y extracción de características teniendo en cuanta cuales son más importante para discernir las clases, donde se refleja que tanto las características del contenido como del contexto son importantes para la tarea planteada.

Se hace un etiquetado de solo el conjunto de tweets noticiosos y un conjunto de no etiquetado, este último incluye tweets que pueden ser noticiosos y los negativos (no noticioso). Para extraer los tweets negativos se aplicó un proceso de PU-Learning con el objetivo de extraer la clase negativa desde el conjunto no etiquetado. Se probó dos aproximaciones tanto por PU-Learning bagging como iterativa, además de diferentes clasificadores durante el proceso. El desbalanceo de los positivos con respecto a los no etiquetado era menor que el 5%, por lo que el desempeño fue muy bajo. Por lo expuesto, se realiza un aumento de los datos positivos mediante la creación de tweets sintéticos, extrayendo titulares de noticias mediante *The Guardian Open Plataform* construyendo tweets con ellos mediante transformaciones como: inclusión de *hashtag*, *menciones* y *link*. Con esta aumentación de datos se obtuvo mejores resultados significativamente en el caso del PU-Learning Bagging, no así en el caso de la aproximación iterativa.

Con los conjuntos negativos y positivos de tweets (noticiosos o no) obtenidos, se propuso un modelo mediante aproximaciones de aprendizaje profundo. Este se realiza mediante la creación de un modelo entrenado con solo texto para el objetivo planteado. Se hace uso en el modelo de word embeddings de palabras, capas convolucionales y *LSTM*. Para el modelo propuesto se obtiene una precisión de 0.98 y F1-score de 0.86. Esto corrobora que las aproximaciones de aprendizaje profundo tienen resultados mejores y significativos en diferentes áreas.

Teniendo en cuenta estos resultados se plantean varias ideas para realizar a trabajo futuro, estas son descritas a continuación:

- Unos de los inconvenientes en estos problemas son los bajos conjuntos de datos que se poseen para aplicar una aproximación de aprendizaje profundo. Por lo tanto, se plantea estudiar y desarrollar métodos de generación de texto automáticos, entrenándolo y aplicándolo al dominio especifico.
- Toda la experimentación del presente trabajo se desarrolló sobre un mismo dominio de noticias de la industria automovilística, por lo que se propone una experimentación

del modelo en otros dominios (política, deporte, etc.), así como probar la factibilidad de que sea capaz de trabajar en varios y no bajar su rendimiento sustancialmente.

- Los resultados del modelo pueden ser mejorables, observándose que el recall fue de solo 0.76, lo cual aún deja de detectar tweets noticiosos. Se plantea el estudio y desarrollo de aplicar un *fine tunning* con nuestro dominio y experimentar en word embeddings como BERT, y/o proponer modelos que nos caractericen mejor el contenido de los tweets. Además, se plantea el estudio de características de propagación lo cual puede ayudar a una mejor detección.
- Como un trabajo posterior se plantea el estudio y pruebas del modelo sobre *fake news* y hacer las modificaciones y diseño correspondientes para este problema, el cual es una tarea de una complejidad superior.

REFERENCIAS

[1] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang y Hua Liu, «Fake News Detection on Social Media: A Data Mining Perspective,» *arXiv:1708.01967,* 2017.

[2] M. Viviani y G. Pasi , «Credibility in social media: Opinions, news, and health information-a survey,» *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 7 (5). doi:10.1002/widm.1209,* 2017.

[3] Z. Xichen y G. Ali A., «An overview of online fake news: Characterization, detection, and discussion,» *Information Processing & Management,* 2019.

[4] S. Sikdar, S., B. Kang, J. Odonovan y T. Hollerer, «Understanding Information Credibility on Twitter,» *International Conference on Social Computing. doi:10.1109/socialcom.2013.9,* 2013.

[5] C. Castillo, M. Mendoza y B. Poblete, «Information credibility on twitter,» *Proceedings of the 20th international conference on World wide web, WWW 11. ACM,* pp. 675-684, 2011.

[6] M. Mendoza, C. Castillo y B. Poblete, «Predicting information credibility in time-sensitive social media,» *Internet Research ,* vol. 23, nº 5, 2013.

[7] F. Mordeleta y J. Vertbcd, «A bagging SVM to learn from positive and unlabeled examples,» *Pattern Recognition Latters,* vol. 37, nº 1, pp. 201-209, 2014.

[8] B. Liu, Y. Dai, X. Li, W. Lee y P. Yu, «Building text classifiers using positive and unlabeled examples,» *Proceedings of the Third IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA (2003),* pp. 179-186, 2003.

[9] D. Hernández, M. Montes, P. Rosso y R. Guzmán, «Detecting positive and negative deceptive opinions using PU-Learning,» *Information, Processing and Management,* vol. 51, nº 4, pp. 433-443, 2015.

[10] J. Bekker y J. Davis, «Learning From Positive and Unlabeled Data: A Survey,» *arXiv:1811.04820,* 2018.

[11] B. Wang, A. Wang, F. Chen, Y. Wang y J. Kuo, «Evaluating Word Embedding Models: Methods and Experimental Results,» *arXiv:1901.09785,* 2019.

[12] J. Devlin, M.-W. Chang, K. Lee y K. Toutanova, «Bert: Pre-training of deep bidirectional transformersfor language understanding,» *arXiv preprint arXiv:1810.04805,* 2018.

[13] P. Bojanowski, E. Grave, A. Joulin y T. Mikolov, «Enriching word vectors with subword information,» *arXiv preprint arXiv:1607.04606,* 2016.

[14] J. Pennington, R. Socher y C. Manning, «Glove: Global vectors for word representation,» *Proceedings ofthe 2014 conference on empirical methods in natural language processing (EMNLP),* p. 1532–1543, 2014.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado y J. Dean, «Distributed representations of words and phrases and their compositionality,» *Advances in neural information processing systems,* 2013.

[16] J. Devlin, M. Chang, K. Lee y K. Toutanova, «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.,» *arxiv.org/abs/1810.04805,* 2018.

[17] K. O'Shea y R. Nash, «An Introduction to Convolutional Neural Networks,» *arXiv:1511.08458,* 2015.

[18] Y. Kim, «Convolutional neural networks for sentence classification,» *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014,* October 25-29, 2014.

[19] S. T. Hsu, C. Moon, P. Jones y N. Samatova, «A Hybrid CNN-RNN Alignment Model for Phrase-Aware Sentence Classification.,» *Proceedings of the 15th Conference of the Volume 2, Short Papers,* vol. 2, 2017.

[20] S. Hochreiter y J. Schmidhuber, «Long Short-Term Memory,» *Neural Computation,* pp. 1735-1780, 1997.

[21] P. Zhou, W. Shi, J. Tian, Z. Qi y B. Li, «Attention-based bidirectional long short-term memory networks for relation classification,» *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,* 2016.