Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Desarrollo de un sistema de detección de fake news en español y aplicación en noticias

Grado en Ingeniería Mecánica

ALUMNO: Jose Ramón Medina Lorente

DIRECTORES: Milagros Fernández Gavilanes

CURSO ACADÉMICO: 2019-2020

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Desarrollo de un sistema de detección de fake news en español y aplicación en noticias

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

Universidade Vigo

RESUMEN

El controvertido ascenso de Donald Trump hacia la Casa Blanca ha despertado un mal que permanecía oculto para toda la sociedad, las *fake news*. Este fenómeno ha sido capaz de poner en jaque la credibilidad del periodismo en general y ha propiciado la búsqueda de diferentes mecanismos para detectar e identificar este tipo de noticias. El método tradicional y más empleado pasa por el procesado manual de las noticias mediante la comprobación del autor, la fuente o el contenido (*fact-checking*).

En este TFG se busca un sistema más sofisticado basado en la utilización de técnicas de procesamiento del lenguaje natural mediante *machine learning* para el desarrollo en lenguaje Python de un sistema, en el cual el usuario sea un mero observador y el programa pueda dirimir la veracidad de la noticia. En función de un conjunto de datos textuales proporcionados, en el cual encontramos las noticias clasificadas como verdaderas y falsas, se crea un modelo a partir de diferentes algoritmos de aprendizaje como *Long Short-Term Memory* (LSTM) para discernir si la noticia es verdadera o falsa. Antes de poder hacer que el texto sea comprendido por el modelo creado precisa pasar por diferentes procesos como es el *embedding*, en otras palabras, la transformación del corpus tal y como lo conocemos a un lenguaje numérico y que permite un óptimo funcionamiento de los diferentes algoritmos empleados.

PALABRAS CLAVE

Fake News, Python, Dataset, Procesamiento del Lenguaje Natural, *Machine learning*, *Embedding*.

AGRADECIMIENTOS

Durante el desarrollo del TFG conté con el apoyo incondicional de mucha gente que me apoyó en los momentos de alegría y en los muchos momentos de estrés.

A mi tutora, por no rendirse ni desesperarse con mis múltiples preguntas y correos desde el primer día y a cualquier hora.

A mi camareta, X3, por los momentos de apoyo y de distracción cuando las ganas de tirar el ordenador por la ventana superaban incluso a las de respirar.

A Orteguita, por las interminables horas de conspiración y lucha contra Python. Por ganarle y ser capaces de conseguir nuestros objetivos, incluso cuando al principio parecía imposible.

Al CTE IM Francisco Gómez Conde por su apoyo y colaboración en el desarrollo del TFG.

Por ti, Luisete.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Introducción.

Jean Cocteau afirmó que no se debe confundir la verdad, con la opinión de la mayoría. Esta frase se ha mantenido inquebrantable durante años debido a la actitud de rechazo de la sociedad humana ante la verdad.

Uno de los mayores peligros a los que debe hacer frente el ser humano en su día a día, es discernir entre lo veraz y el fraude. Esta distinción que antes se antojaba sencilla, se está convirtiendo en una ardua tarea debido a la actual facilidad en la difusión de la información, tal y como le ocurre a la sociedad estadounidense [1]. No hay que mirar muy atrás para comprobar como el Brexit o las elecciones norteamericanas se vieron influenciadas por campañas de propaganda subvencionadas por terceros con el fin de buscar un resultado deseado.

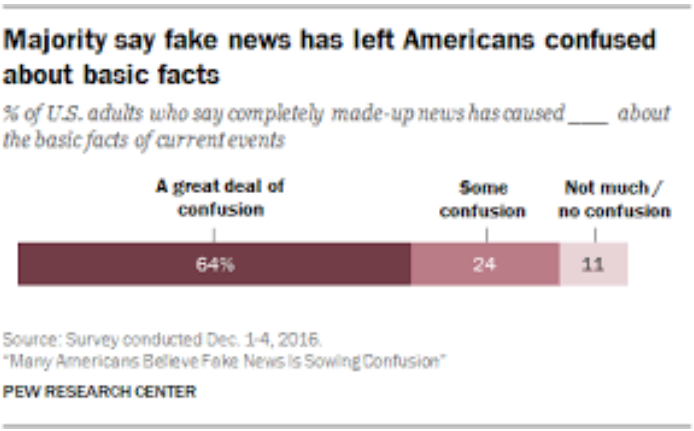

Figura 1-1 Impacto de las *fake news* en la sociedad norteamericana [1]

El decisivo papel de estas campañas con fuentes no contrastadas permite prever la dificultad del ciudadano medio para encontrar el camino hacia la verdad. Esta dificultad es en gran parte derivada por un descuido, cada vez mayor, de la educación y a la convivencia en una sociedad cada vez más polarizada.

Según afirman diferentes sociólogos, e incluso la justicia y congreso norteamericano, las intromisiones, según se sospecha, rusas han sido determinantes para la elección del nuevo presidente [2]. Es decir, actualmente, la actividad de ciertos usuarios en la red es decisiva para la opinión de todo un país. Es por motivos como este, que surge la necesidad de crear una herramienta que permita a los usuarios de la red verificar que las fuentes de información, así como la noticia que acaban de leer se ajustan a la realidad.

La vigilancia y, posterior corrección de este tipo de información se antoja complicado teniendo en cuenta las pocas herramientas existentes, así como la velocidad de propagación que supone Internet en cuanto a la transmisión de información se refiere. Antaño, este trabajo se realizaba de manera manual dado que, ya en tiempos de Felipe II, existía gente a disposición del rey para enterarse de lo que hablaba el pueblo y cortar de raíz cualquier información falsa o que no le interesara a la corona.

Aunque este método parezca anticuado, la detección de las noticias falsas en potencias mundiales como puede ser Rusia, todavía se realiza de manera manual, es decir, se buscan esas diferentes noticias y se contrastan con fuentes fidedignas y de prestigio.

En este trabajo se propone una manera más rápida y eficaz de discernir entre las *fake news* y aquella información veraz que ha de tenerse en cuenta y, por ejemplo, si debe influir en la decisión de los votantes.

El método propuesto para esta identificación de noticias falsas se hace con la utilización de técnicas de *procesamiento de lenguaje natural* (PLN) mediante *machine learning*. La utilización de algoritmos que hagan uso de estas técnicas permite diferenciar entre una noticia verdadera de otra falsa gracias a la experiencia adquirida previamente por el sistema desarrollado. Es decir, gracias a la existencia de unos datos previos correctamente etiquetados según la veracidad o falsedad de su contenido, se pueden extraer unos patrones que permitirán comprobar el grado de falsedad de una noticia presentada en base a la experiencia adquirida por el sistema.

Estas técnicas de PLN mediante *machine learning* serán implementadas mediante un código en el lenguaje de programación denominado Python. Este lenguaje permite la implementación de código tal que recibiendo una serie de ejemplos de noticias convenientemente clasificados en verdadero o falso sea luego capaz de verificar la veracidad de la noticia.

1.2 Objetivos

Con este trabajo se busca tratar de concienciar al lector de la importancia que tienen las *fake news* en la sociedad actual y de la capacidad de las mismas de influir en asuntos de envergadura tal como puede ser la elección del presidente de la mayor potencia económica del mundo [2]

Además de intentar sembrar conciencia, se desea informar acerca de conceptos, hasta ahora desconocidos en la sociedad, como son la *posverdad* o el PLN mediante *machine learning*. En cualquier caso, estos conceptos son fundamentales a la hora de querer combatir las noticias falsas y lo que estas conllevan.

La utilización de técnicas de PLN mediante *machine learning* y sus correspondientes algoritmos abren una nueva línea de investigación, sobretodo en la sociedad española. El empleo de este tipo de técnicas para la detección de *fake news* no se remonta hasta 2016, año en el que saltaron todas las alarmas con la injerencia rusa en asuntos de estado norteamericano. Si bien esta línea de investigación está en fase de desarrollo en lengua inglesa [3], en lengua española se podría decir que todavía está en pañales, ya que a día de hoy todavía existen pocas empresas dedicadas a la lucha contra la desinformación y, la mayoría, se basan en el *fact-checking*.

Finalmente, el objetivo *persé*, es la implementación de un sistema que haga uso de técnicas de PLN y algoritmos de *machine learning* con una única finalidad, ser capaz de decir al usuario de un modo automático, si la noticia o información que está consultando es veraz o si, por lo contrario, no debe fiarse de la misma.

También se busca analizar el contenido de la noticia simplemente con la lectura del titular de la misma, haciendo así, que la aplicación no consuma demasiado tiempo en obtener una respuesta

Este sistema será diseñado e implementado en el lenguaje Python debido a que es un lenguaje con una sintaxis sencilla de entender y de emplear, y además dispone de una vasta biblioteca de herramientas

en materia de *machine learning*. El principal motivo por el que se ha optado por este lenguaje es en gran medida debido a que la demanda de profesionales de Python aumenta cada año como recoge la web *Stackoverflow* como se puede apreciar fácilmente en la gráfica recogida en la Figura 1-2.

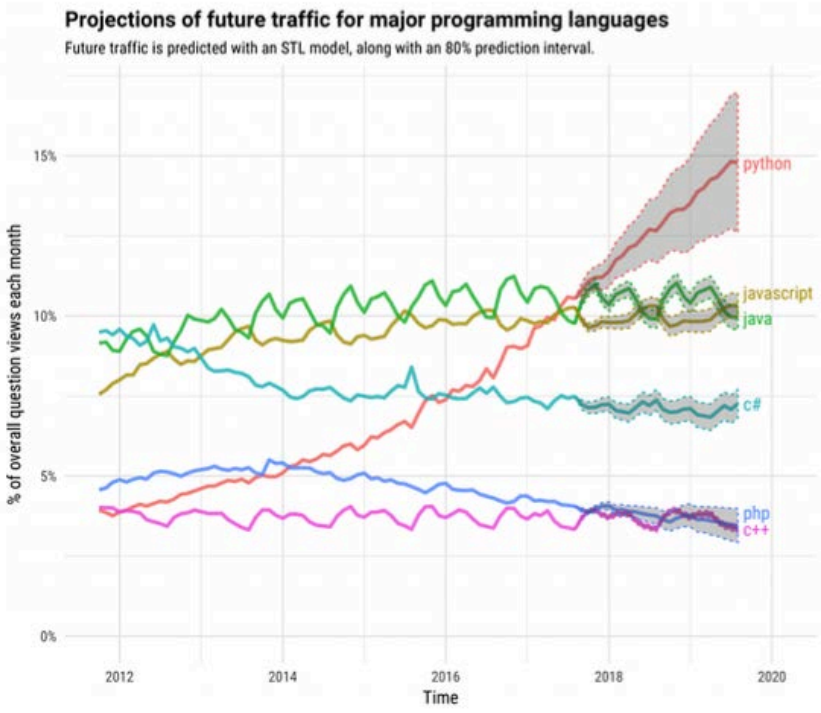

Figura 1-2 Predicción del uso de los diferentes lenguajes de programación [4]

1.3 Metodología

El procedimiento a seguir para el correcto desarrollo de este TFG consistió en un primer momento en una búsqueda exhaustiva de información ya existente en el campo a trabajar, así como un importante filtrado de aquella información que no permita la correcta elaboración de un trabajo veraz, objetivo y académico. Se fomentó la búsqueda de información en páginas de información convenientemente contrastadas, así como en artículos de periódicos que mantengan una credibilidad en el panorama internacional incuestionables.

Para el correcto desarrollo del sistema de aprendizaje se consultó las diferentes páginas que el propio lenguaje Python recomienda para intentar optar siempre a la mejor solución posible con el fin de conseguir que la implementación, además de cumplir con el objetivo marcado desde el principio del proyecto, sea la óptima para el correcto funcionamiento del programa.

En la medida de lo posible se usaron gráficos, así como otras herramientas visuales para acercar al lector al resultado final y posibilitar un mayor y más rápido entendimiento de lo que se intenta explicar.

Para poder realizar una implementación del código de la manera más eficiente posible, y con la intención de realizar continuas mejoras en él, se decidió buscar un dataset adecuado en lengua inglesa debido a la mayor facilidad de encontrar este tipo de datos y a que a la existencia de un mayor rodaje de los diferentes sistemas de aprendizaje en este idioma cuando estos se aplican al lenguaje natural.

Una vez que se ha alcanzado el objetivo en lengua inglesa se acomoda el código a la lengua hispana en busca de alcanzar unos resultados similares.

1.4 Organización de la memoria

La memoria se va a dividir en diferentes capítulos, empezando por el presente. En este, se ha presentado la forma en la que se va a desarrollar el TFG. También se ha hecho hincapié en la importancia al nivel mundial del tema en cuestión, así como la intención de concienciar a la sociedad de la importancia entre discernir entre información falsa y la veraz.

En el capítulo 2, se procede a explicar los antecedentes del proyecto al igual que algunos términos clave para la comprensión del mismo. Además, en este capítulo se hablará, de manera general, del lenguaje de programación Python y los beneficios que este supone en nuestro proyecto.

El capítulo 3, al contrario que en el anterior, ya se explica de manera detallada la forma en la cual se emplea Python, explicando el desarrollo del código de la misma manera que los múltiples problemas que hubo que hacer frente para la correcta consecución del objetivo. En este capítulo también se buscará explicar en mayor detalle el funcionamiento del lenguaje escogido. También se hará referencia a la elección de un IDE (entorno de desarrollo integrado) y el empleo de programas secundarios que facilitan la implementación del código, como puede ser Anaconda. En este capítulo también se pone de relieve el proceso de creación del código que permite la consecución de los objetivos de este proyecto; se hablarán de conceptos como es el *embedding* o diferentes modelos como son LSTM o SVM.

En el capítulo 4 se hablará acerca de los resultados obtenidos con los sistemas creados a partir de los diferentes algoritmos. Estos resultados serán clave para comprender si se han alcanzados los objetivos dictaminados o si, por el contrario, el TFG debería haberse planteado de otra manera. Durante este punto se hablará de medidas fundamentales de evaluación para evaluar un modelo como correcto o no en base a su precisión, su exactitud, su exhaustividad o su F1.

El capítulo de conclusiones y líneas futuras supone el fin de la memoria. En este capítulo se hará un pequeño resumen por todo el trabajo realizado y se propondrá como seguir con el TFG en los años venideros dado que este tema está en auge y es de gran utilidad tanto a la armada española como para la sociedad en general.

Por último, se propondrán una serie de anexos en los que se pondrán de manifiesto documentos de interés que no han sido por completo expuestos en la memoria.

2 ESTADO DEL ARTE

2.1 Concepto de posverdad

Antes de que se popularizara el concepto de las *fake news* o bulos, se hablaba de un término también relacionado con el tema que nos ocupa: *la posverdad*.

El término posverdad se emplea por primera vez en 1992 por el dramaturgo Steve Tesich como referencia al escándalo político durante la presidencia de Ronald Reagan en relación a la venta de armas a Irán y a la guerra del Golfo Pérsico [5]. El autor del artículo alude a que en esta época la sociedad americana no buscaba saber la verdad, sino que esperaba que fuese el gobierno quien les protegiera de ella tomando las acciones que resultasen necesarias.

En boca del que fuera director del periódico ABC, Jose Antonio Zarzalejos, la posverdad "*describe una situación en la cual (…) los hechos objetivos tienen menos influencia que las apelaciones a emociones y creencias personales*" [6].

Los términos sobre los que versa la posverdad tienen como fin la confusión de la realidad o el recelo hacia ciertos sectores de la población. Estos conceptos alcanzaron su punto álgido durante el siglo XX con el nazismo y el stalinismo.

Viendo estos ejemplos podemos observar cómo realmente este término no hace referencia a algo novedoso, a pesar de ser una de las palabras más empleadas en el 2016. Su propia definición: "*Distorsión deliberada de la realidad cuya intención es influir en la opinión de la sociedad" desplazando a vocablos, quizás, más malsonantes, tales como pueden ser los siguientes: mentira, estafa, bulo…*" [7].

A pesar de lo dicho con anterioridad, parece que las mentiras del hoy tienen mayor impacto en la sociedad y, sobretodo, alcanzan un mayor número de víctimas. Esto se debe a que las redes sociales han servido como catapulta a aquellas personas que con solo un ordenador son capaces de cambiar la opinión de un público ciertamente cada vez más y más susceptible. Parafraseando al famoso escritor Umberto Eco: *"Las redes sociales han dado derecho a hablar a legiones de idiotas"*.

A pesar de lo afirmado por el escritor italiano, la gravedad se manifiesta en estadísticas tales como que solo el 38% (año 2016) de la sociedad se fía de la información revelada por los medios [8] y, esta información contrasta por lo manifestado por periodistas tales como Susan Glasser que afirma que los hechos objetivos han pasado a un plano secundario prevaleciendo los llamamientos emocionales y las creencias personales [9]. A pesar de que los trabajos de investigación actuales son de mayor rigor que en el pasado, estos se han visto ensombrecidos por aquellas informaciones repetitivas y con motivos sensacionalistas que tanto abundan en redes sociales y que, ya, están acaparando portadas en periódicos de rigor. Un ejemplo sería el ilustrado en la Figura 2-1, donde tanto la portada como los artículos de

opinión de El Periódico del día 01/03/2018 (izquierda) eran una noticia *fake* y donde, el día 09/01/2019, el diario tuvo que realizar una rectificación a dos columnas en la página 15 del mismo medio.

Figura 2-1: Ejemplo de portada fake y su rectificación en dos columnas [10]

En los estudios realizados por el *"Pew Research Center"* en diciembre de 2016 se observa cómo, en las últimas elecciones de Estados Unidos, un 35% de jóvenes de entre 18-29 años afirman haber seguido las elecciones y haberse informado antes de votar por las redes sociales mientras que solo un 18% recurrió a medios periodísticos [1].

El recurrente empleo de estas redes sociales no permite asegurar la credibilidad de las fuentes. Se estima que, en los tres últimos meses de campaña de las elecciones presidenciales norteamericanas, las noticias falsas generaron más reacciones en Facebook que las noticias principales del periódico *The New York Times* como se puede comprobar en la Figura 2-2 [11].

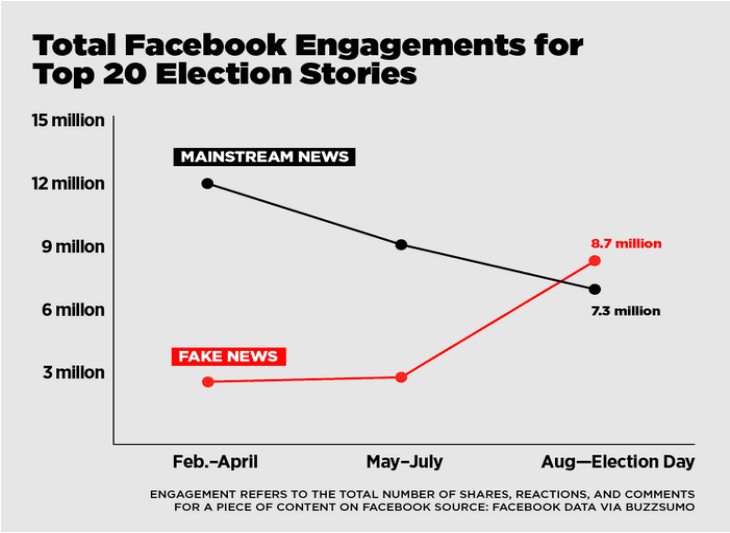

Figura 2-2 *Fake News* vs *News* en las elecciones norteamericanas [11]

Resumiendo, estamos en la era de la posverdad, donde las creencias personales han superado a los hechos y a la lógica para desconcierto de la opinión pública. La herramienta más sencilla de propagación de esta era son las *fake news*, que emplean las mentiras y persuasión de siempre pero ahora magnificadas gracias a las redes sociales que ponen en jaque las opiniones documentadas de periodistas debido a oleadas de opiniones y creencias que se difunden diariamente en la red [12].

2.2 *Fake news*

La popularidad del término *fake news*, es más bien reciente. Es el año 2016, año en el cual Donald Trump resultó presidente electo de los Estados Unidos o año en el cual se produjo la votación del "Brexit" cuando se empezó a acuñar este término. Noticias de este estilo, fueron rápidamente difundidas por diferentes medios de comunicación y twitteros haciendo referencia hechos tales como el comercio de armas por parte de la otra candidata, Hillary Clinton [13], al ISIS (Figura 2-3); o el apoyo del Papa Francisco a Donald Trump[1] (Figura 2-4) [14].

Figura 2-3: Noticia falsa acerca del comercio de armas por parte de Hillary Clinton al ISIS [13]

Figura 2-4: Desmentido de la noticia falsa acerca del apoyo del Papa a Donald Trump [14]

Según afirma el propio Donald Trump, el 2 de octubre de 2019 en una reunión con el presidente del gobierno finlandés [15], este término fue acuñado por él mismo a pesar de que ahora prefiere llamarla noticias corruptas para ser más crítico con este tipo de divulgaciones. Sin embargo, este término ya se

[1] Desmentido de la noticia del Papa apoyando a Trump disponible en El País en https://verne.elpais.com/verne/2016/11/17/articulo/1479386366_405746.html [Ultimo acceso el 25/02/2020]

empezó a popularizar en 2014 por Craig Silverman [16] cuando este realizaba un proyecto de investigación acerca de rumores e historias sin contrastar.

2.2.1 Definición y características

Durante el año 2016 hubo una combinación de palabras que aumentó su uso en más de un 365%. Esta combinación era *fake news*. Este hecho motivó al prestigioso diccionario Oxford a nombrarla palabra del año [17] y a recogerla en su diccionario. La definición que ahí se recoge es: "*falso relato de un evento recogido y escrito en páginas web*". Esta definición ya deja entrever lo esencial que es Internet para la difusión de este tipo de noticias.

Otros entendidos de la materia tales como el periodista, guionista y director de diferentes programas de televisión, Marc Amorós no duda en complementar esta definición y apuesta a que las *fake news* son noticias falsas que se difunden con una voluntad deliberada de engañar [18]. Para él estas noticias tienen que parecer reales y, además, tener un objetivo claro.

Está claro que estas noticias cumplen con las dos características mencionadas anteriormente. De hecho, según un estudio llevado a cabo por la Universidad Complutense de Madrid [19] en el que participaron más de 2.000 encuestados, un 60% de ellos se creía capaz de reconocer una noticia falsa, mientras que, a la hora de la verdad, solo un 15% fue capaz de distinguir entre un titular veraz y otro falso.

Otro de los resultados que se extrajo de este estudio es la verdadera desinformación puesto que inicialmente solo un 4% de los informados ha reconocido distribuir o generar este tipo de noticias. Entre los que han reconocido ser partícipes de este fenómeno, el motivo más repetido es el de la diversión. Este motivo difiere fuertemente con el de los originadores de tales noticias que se suelen decantar por motivos políticos o económicos.

2.2.2 Evolución de las noticias falsas

El ser humano es por naturaleza sensacionalista. A lo largo de la historia se pueden encontrar ejemplos de noticias falsas con motivos políticos o económicos de fondo.

Un ejemplo de ello es el acontecido en 1475 en Trento (Italia) cuando, tras la desaparición de un niño de 2 años, un cura franciscano, Bernardino da Feltre, acusó al pueblo judío de su asesinato y de beber su sangre para celebrar la Pascua. Para poner fin a tal noticia fue necesaria la intervención del mismísimo Papa [20].

Otro de los ejemplos a los que se puede hacer referencia es el que ocupó al país norteamericano, Estados Unidos, con España a colación de la explosión que se produjo en el buque acorazado el 15 de febrero de 1898, muriendo en total unas 256 personas. En aquel momento los periódicos americanos no dudaron en culpar en sus titulares al pueblo español, tal y como se muestra en la Figura 2-5. Este hecho derivó, como ya sabemos, en una declaración de guerra contra España por parte del congreso de los Estados Unidos auspiciada por el magnate de la prensa de la época William Randolph [21].

Otro ejemplo del peligro de este tipo de noticias es el ocurrido recientemente en Estados Unidos en el caso conocido como *Pizzagate*. Este famoso acontecimiento ocurrido en 2016 provocó que Edward Welch, de 28 años, entrara en la pizzería *Comet Ping Pong* armado después de que la ultraderecha americana difundiera que este local, donde tenía lugar un evento para recaudar fondos para Clinton, era en realidad una tapadera de una red de violaciones a niños que luego eran sacrificados en nombre de Satanás [22].

Como se observa en estos tres ejemplos se refuerza la teoría de que las noticias falsas buscan un objetivo, ya sea político o económico, y tienden a disfrazarse de noticias que a primera vista pueden ser verídicas. La principal diferencia que encontramos es la difusión en los diferentes casos. A medida que aumentan los medios de comunicación aumenta la facilidad de diseminación de estas noticias, llegando

a un número ingente de personas, algunas incapaces de discernir por si mismas la veracidad de lo que leen.

Esta facilidad en la difusión es la principal diferencia entre las *fake news*, caso del *Pizzagate*, y sus precursoras, los otros dos ejemplos.

Figura 2-5 Titular acerca de la destrucción del buque Maine [21]

2.2.3 Panorama legal ante las fake news

Como se ha podido observar con los ejemplos del Brexit y las elecciones norteamericanas los gobiernos se muestran especialmente vulnerables ante los efectos propiciados por las noticias falsas, esto ha derivado en que sean muchos los gobiernos que han comenzado a legislar para intentar poner fin a estos casos de manipulación informativa, uno de los cuantos titulares que ponen de manifiesto estas iniciativas, en este caso en el parlamento de Singapur, son los que se aprecian en la Figura 2-6.

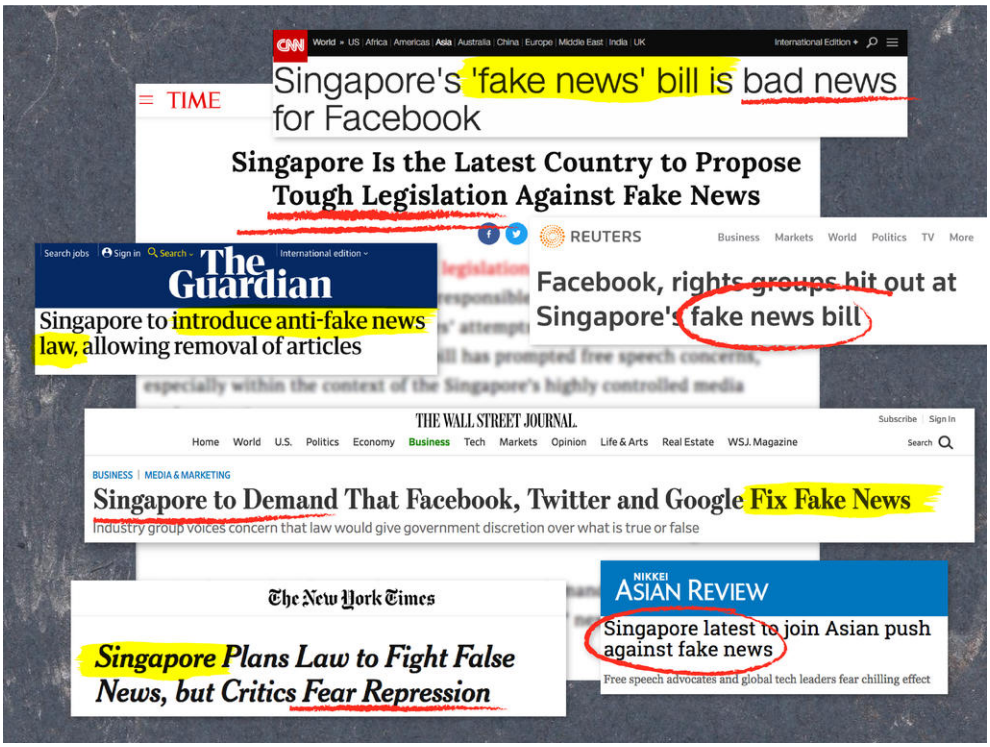

Figura 2-6 Titulares acerca de las *fake news* [23]

Si comparamos las medidas adoptadas hoy en día a este respecto, uno de los pioneros en esta batalla fue el gobierno germano donde se pueden alcanzar penas de hasta 50 millones de euros en el caso de cometer delitos en caso de incurrir en mentiras o medias verdades.

En cuanto el ejecutivo francés, con una ley aprobada el 20 de noviembre de 2018 [24], el gobierno de Macron puso especial énfasis en el cumplimiento de la ley durante las campañas electorales, con el fin de evitar intromisiones de otros países. La ley se focaliza en conseguir una transparencia total por parte de las plataformas informantes, así como la potestad por parte de jueces para detener de manera inmediata la difusión de noticias falsas en función de la definición que se hace de estas en la ley de libertad de prensa de 1881.

Otro de los parlamentos que abogó por poner veda a esta difusión indiscriminada fue la cámara de los Lores donde se recoge la posibilidad por parte del gobierno a eliminar aquel contenido que lleve información falsa o que no sea exacto con el fin de obtener beneficio con la mal información de los informados [25].

En España se han impulsado múltiples propuestas por parte del gobierno liderado por el anterior presidente, D. Mariano Rajoy, pero ninguna llegó a buen puerto. La última propuesta llegó por el entonces candidato a la presidencia, D. Pedro Sánchez, de impulsar una estrategia nacional de lucha contra la desinformación, así como desarrollar un plan de ciberseguridad. En boca del presidente del gobierno, "las *fake news* son uno de los fenómenos más dañinos para la confianza" [26].

La lucha contra este fenómeno también ha llegado a las redes sociales como se puede observar en el caso del gigante de las redes sociales, Facebook. La plataforma creada por Mark Zuckerberg está en el ojo del huracán al haber sido parte fundamental de la difusión de estas noticias falsas durante la carrera electoral que coronó a Donald Trump. La plataforma estadounidense ha anunciado ligeras modificaciones que pasan por eliminar cuentas falsas y otras pequeñas modificaciones que en ningún caso pueden asegurar la transmisión veraz de información de sus usuarios [27].

2.2.4 Defensa y las fake news

Desde 2017 se empieza a considerar la importancia de este asunto en el Gobierno español cuando el Consejo de Ministros decide aprobar la Estrategia de Seguridad Nacional del año 2017 donde se incluye como amenaza, por primera vez, la difusión de *fake news*. La exministra de Defensa, Dª María Dolores de Cospedal no ha dudado en catalogar las noticias falsas como "*uno de los mayores retos para los sistemas de defensa de las democracias*" [28].

En los Cuadernos de Estrategia 197 del Instituto Español de Estudios Estratégicos, dependiente del Ministerio de Defensa, también se pone de relieve la importancia de este fenómeno. En estos artículos se hace hincapié en que los principales motivos de las noticias falsas o bulos van desde la manipulación o la generación de ruido para encubrir informaciones de mayor enjundia hasta la búsqueda de fines económicos. En un artículo de Federico Aznar Fernández-Montesinos se hace referencia a que el fin último de estos bulos es "*provocar la desconfianza en el conjunto de la sociedad y afectar a su cohesión*" [29].

El CNI tampoco ha perdido la ocasión de sumarse a esta lucha contra los bulos dado que, en la propia toma de posesión de la nueva Directora del Centro Nacional de Inteligencia, Paz Esteban López, esta ha recalcado la lucha contra las *fake news* una de sus tres prioridades haciendo referencia al problema actual de Cataluña y la cantidad de perfiles digitales falsos existentes que distribuían noticias falsas o criticaban la actitud del gobierno. La directora fue más allá poniendo incluso ejemplos de cuentas falsas, como la cuenta de Twitter `@ivan226622`, la cual demostró no ser una persona como afirmaba, sino un boot[2] [30].

[2] Se trata de un sistema informático automatizado que actúa como usuario fantasma con el objetivo de cumplir funciones como puede ser realizar determinadas publicaciones o seguir a ciertos usuarios.

En esta línea el CNI ha decidido publicar una guía para luchar con la desinformación en el ciberespacio y lo ha hecho ofreciendo un decálogo de seguridad para los consumidores digitales. Algunas de estas recomendaciones son verificar la fuente de la información o no fiarse de perfiles anónimos en las redes sociales [31].

2.3 Inteligencia artificial

Para poder procesar una herramienta que sea capaz de analizar las *fake news* antes se debe de mencionar la rama de las ciencias de la Computación dedicada al desarrollo de agentes racionales no vivos, llamada *inteligencia artificial* [32].

En este sentido, se entiende por agente cualquier cosa capaz de percibir su entorno, procesar esas percepciones y actuar sobre ese entorno. En este caso, la racionalidad viene determinada como una capacidad humana que permite pensar, evaluar y actuar conforme a ciertos principios con el fin de cumplir algún objetivo o finalidad. Por tanto, y ya de manera mucho más específica, la inteligencia artificial es la disciplina que se encarga de construir procesos que al ser ejecutados sobre una arquitectura física producen acciones o resultados que maximizan una medida de rendimiento determinada, basándose en la secuencia de entradas percibidas y en el conocimiento almacenado en tal arquitectura [32].

Se puede entender este concepto como la combinación de algoritmos para crear máquinas con capacidades similares a los seres humanos. Este concepto se basa en la capacidad de análisis por parte de las máquinas de gran cantidad de datos y de realizar predicciones en función de estos [33].

La inteligencia artificial se considera un campo fundamental en el día a día de la sociedad actual. Estando presente en temas tales como el aprendizaje máquina a través de la repetición, empleo de anuncios en función de nuestras búsquedas recientes, así como muchas otras cosas.

El desarrollo de la inteligencia artificial tiene sus inicios en 1943 con el intento de crear un sistema que simulase comportamientos humanos. El objetivo de este sistema era crear un ordenador capaz de solucionar los problemas de manera similar a la que lo haría un ser humano.

El siguiente paso en la evolución de la inteligencia artificial fue el *machine learning*, base fundamental en la cual se desarrolla este trabajo. Este concepto se basa en que los sistemas pueden aprender de los datos proporcionados con una intervención humana mínima. Se explicará con más detalle el apartado 2.3.1.

El sistema neuronal, propiamente dicho, tiene sus inicios en 1986 con Rumelhart y McClelland. Este sistema recibe su nombre debido a que están conformadas por un conjunto de neuronas artificiales interconectadas entre sí. Las neuronas artificiales se caracterizan por tener un conjunto de elementos que se denominan entradas para dar lugar a una salida [34].

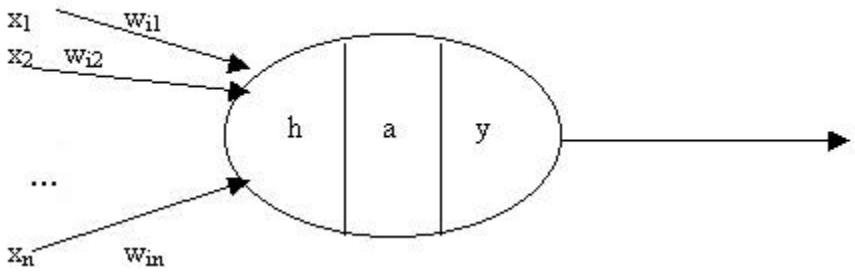

Figura 2-7 Esquema funcionamiento red neuronal

En general, las neuronas se agrupan en capas. De la cuales encontramos tres tipos diferenciados:

1. De entrada: reciben los datos.
2. De salida: proporcionan la respuesta de la red.
3. Ocultas: ni reciben ni suministran información.

Por último, destacar la existencia del Deep Learning, estos algoritmos tratan de imitar el funcionamiento del cerebro por lo que se dice que funciona por capas o unidades neuronales. En esta técnica la primera capa procesa la información y arroja un resultado probabilístico, las sucesivas utilizan su propio juicio con el resultado de la capa anterior aumentando la precisión de manera notable. La principal novedad es que la red neuronal es capaz de modificar el modelo en el momento que se produce un error, algo que no sucede con el resto de técnicas de *machine learning* [35]. Uno de los empleos más novedosos y eficientes en los que impera está técnica de aprendizaje es el fraude bancario. Se analizan el número de transacciones, fecha y hora, cantidad de dinero que se ha movido para arrojar un resultado concluyente.

2.3.1 Machine learning

Definimos *machine learning* o Aprendizaje Automático como la capacidad de una máquina de aprender de la experiencia [36]. Esta herramienta de inteligencia artificial extrae la mayor cantidad de información posible de los datos y realiza una tarea basándose en los conocimientos adquiridos.

A la hora de aplicar algoritmos de *machine learning* se tiene que distinguir entre dos clasificaciones. Los primeros consisten en algoritmos que requieren supervisión por parte de un humano para alcanzar los conocimientos suficientes; y en el segundo caso, algoritmos que no requieren de supervisión.

Por un lado, los algoritmos con supervisión son los más utilizados en lo que a aprendizaje se refiere. Partiendo de una entrada (x), se obtiene una salida representada por otra variable (y). El objetivo de este tipo de algoritmos es aproximar una función tal que, dándole al sistema una variable x, pueda predecir la y. Se llama a este sistema supervisado dado que el algoritmo aprende a partir de unos datos de prueba como si le estuviera supervisando un profesor. Algunos ejemplos de algoritmos de *machine learning* con supervisión son los siguientes:

a) *Support Vector Machine* (SVM): método basado el aprendizaje para la resolución de problemas de clasificación y regresión. En este algoritmo se dibujan los datos en una gráfica separando cada variable con su característica, posteriormente se busca el hiperplano que se ajusta mejor para diferenciar las dos clases con claridad [37].
b) *Naive Bayes Classifier* (NBC): Es una implementación del teorema de Bayes el cual dice: "Siendo un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos sea distinta de cero. Sea B un suceso cualquiera del que se conocen las probabilidades condicionales, entonces la probabilidad a posteriori será (Figura 2-8)" [38].

$$P(\mathrm{A}|\mathrm{B}) = \frac{P(\mathrm{B}|A) \cdot P(A)}{P(B)}$$

Figura 2-8 Probabilidad Gaussiana [38]

Hablando ya de la técnica de clasificación de aprendizaje de máquina, este clasificador asume que la presencia de una característica en concreto es independiente del resto de características que existan en el sistema.

El primer paso de este algoritmo es crear una tabla de frecuencia, después crea una tabla independiente para cada característica suponiendo que no existe relación entre ellas y que cada una contribuye de una manera diferente y, por último, calcula su probabilidad [39].

c) *Random Forest:* este se define como un método versátil de *machine learning*, capaz de actuar tanto en tareas de regresión como de clasificación. Para clasificar un nuevo objeto se basa en una serie de atributos y, cada árbol, es el que vota para elegir la clase. En el caso de la regresión se realiza una media de las salidas de los diferentes árboles como se aprecia en la Figura 2-9 [40].

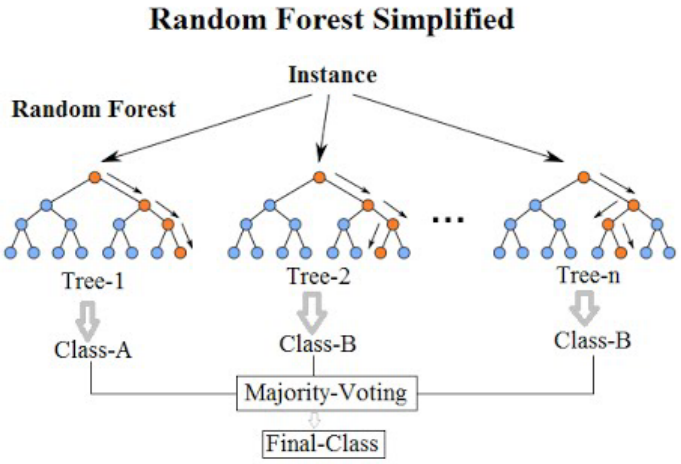

Figura 2-9 Esquema Random Forest [41]

Para un correcto funcionamiento de los sistemas supervisados es fundamental tener suficientes datos etiquetados, es decir, disponer de un dataset lo suficientemente grande y variado. Esto como se verá en el desarrollo del TFG es uno de los principales problemas a los que se tiene que hacer frente por la escasez de datos existentes en general para los diferentes ámbitos de trabajo y en especial para el caso del español debido a lo largo y tedioso del proceso. Este problema se ve acentuado cuando se trata de datos relacionados con la detección de *fake news*, cuya investigación es relativamente reciente.

Por otro lado, encontramos los algoritmos sin supervisión. Estos se basan simplemente en poseer datos de entrada los cuales no sufren ningún tipo de entrenamiento por lo que disponer *a priori* de la información de salida no es determinante para que el sistema funcione. El objetivo de estos es modelar a estructura o distribución subyacentes en los datos para aprender más sobre estos. Los algoritmos son dejados a su voluntad para encontrar la estructura adecuada de los diferentes datos. Si bien es cierto que no requieren entrenamiento, los resultados obtenidos por ellos suelen ser inferiores a los obtenidos en el caso de la utilización de sistemas supervisados. Algunos ejemplos de este tipo de algoritmo son:

a) *K-Medias:* este algoritmo se emplea principalmente en problemas de *clustering*. Su funcionamiento se puede entender como un intento de agrupar las muestras en un número predefinido de grupos en función de un centroide cercano. Este algoritmo busca minimizar la varianza total del sistema.

b) Algoritmo de expectativa de maximización (EM): presenta una técnica iterativa general con el fin de realizar una estimación de máxima verosimilitud de parámetros de problemas que existen en ciertos datos [42].

2.3.2 Deep learning

Por último, este último concepto basado en las redes neuronales se está empezando a implementar también a la hora de localizar las noticias falsas. Este uso se debe principalmente a la capacidad de reconocimiento de lenguaje. En este algoritmo el usuario le otorga a la computadora un modelo para que evalúe los ejemplos y a partir de ellos ser capaz de extraer los patrones. Una de las fórmulas más comunes

para implementar este sistema y, la que se llevará a cabo en nuestro TFG, es simular un sistema de redes neuronales artificiales haciendo el símil que se representa en la Figura 2-10.

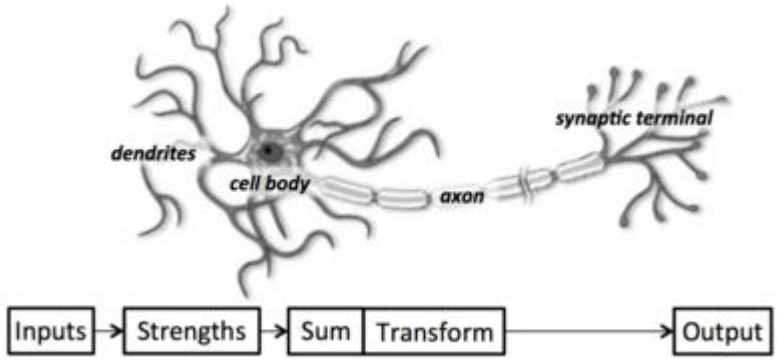

Figura 2-10 Red neuronal [43]

Uno de los algoritmos basados en este sistema son las *redes neuronales recurrentes* (RNN). Este sistema se basa en la alimentación del sistema teniendo en cuenta las decisiones tomadas con anterioridad, es decir, este tipo de red neuronal basa su funcionamiento en dos entradas, en la presente y en la pasada.

La información secuencial que es procesada pasa a un estado oculto para volver a ser recirculada y de nuevo procesada para interactuar con las reacciones futuras y de esta manera influir en el desarrollo del proceso. Los franceses llaman a este tipo de funcionamiento “Le passé qui ne passe pas” es decir, el pasado que no pasa [44].

Otro de los tipos y el que, probablemente, cuente con más aceptación en el panorama científico es el denominado *Long Short Term Memory* (LSTM) [45]. La base de este sistema es que no busca el error que se produce en las sucesivas iteraciones, sino que guarda este error obligando así al sistema a realizar un mayor número de pasos y vincular diferentes sucesos de manera remota. Los LSTMs contienen información fuera del flujo normal da la red recurrente en una celda cerrada. Es esta celda la que toma la decisión y decide cuando la información que esta guarda puede ser empleada para leerse, sobrescribirse o borrarse.

Con el fin de visualizar la dificultad de funcionamiento de este segundo concepto y las diferencias que existen con las RNN se aporta la Figura 2-11:

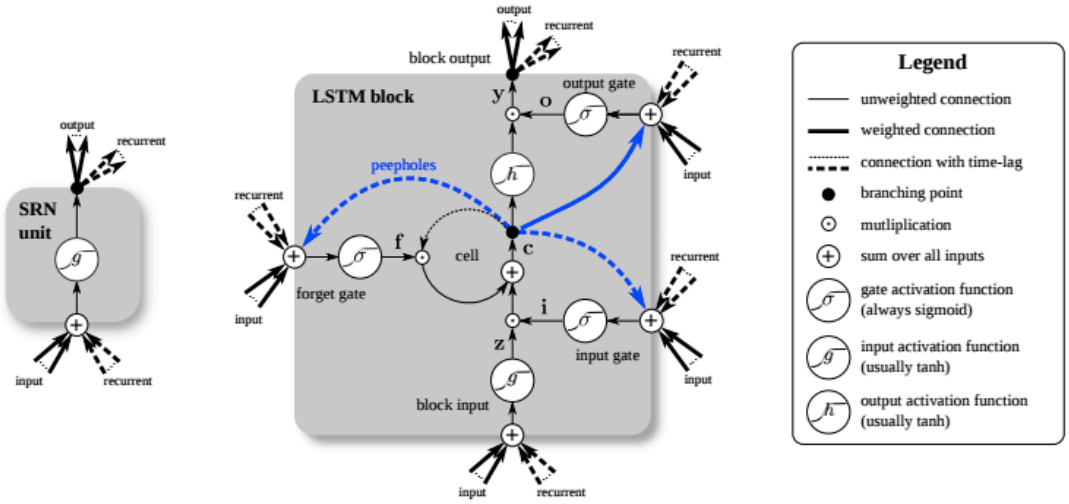

Figura 2-11 RNN (izq.) vs LSTM (dcha.) [44]

2.4 Procesamiento de lenguaje natural (PLN)

Cuando se habla de procesamiento de lenguaje natural se habla de la tecnología empleada en ayudar a los ordenadores a ayudar el lenguaje natural humano. Una interacción entre humanos y máquinas que emplean este tipo de tecnología suele seguir una serie de pasos determinados [46]:

1. Humano habla a la máquina.
2. La máquina recoge el audio.
3. Se convierte el audio en texto.
4. Se procesa la información del texto.
5. Se transforma esta información a audio.
6. La máquina responde al humano reproduciendo el archivo de audio.

La dificultad de esta tecnología reside, sobretodo, en la interacción humana. El ser humano cuando interacciona no recurre a usos sencillos de la fraseología, emplea técnicas tales como el hipérbaton o el sarcasmo que todavía son difícilmente interpretables por una máquina. En otras palabras, la ambigüedad y la imprecisión son las características que provocan que PLN sea tan difícil de implementar [47].

Las técnicas más empleadas en el procesamiento de lenguaje natural son:

1. Sintaxis: en este punto se hace referencia a la disposición de las palabras en una oración de tal manera que tengan sentido gramatical. Diferentes algoritmos son empleados para analizar un grupo de palabras y comprobar la relación entre ellas.
   Algunas de estas técnicas son la segmentación de palabras, lematización o la derivación.
2. Semántica: Implica el significado que transmite el texto. Se trata de aplicar algoritmos informáticos para comprender el significado y la interpretación de las palabras. Dentro de la semántica encontramos el NER o el uso de base de datos para analizar valores semánticos.

2.5 Aproximación para la detección de *fake news*

Empresas de comunicación de todo el mundo han focalizado sus esfuerzos en la lucha contra las noticias falsas, para ello se acude a diferentes reglas básicas de verificación. Estas medidas pasan desde análisis manual del contenido, autor y contexto del artículo hasta métodos más sofisticados de análisis más complejos que se basan en técnicas de PLN.

2.5.1 Enfoque práctico: Fact-checking

El término inglés *fact-checking* se puede traducir literalmente como comprobación o verificación de determinados hechos. Su objetivo es proporcionar cierta precisión e imparcialidad acerca de las afirmaciones que personajes públicos realizan a la sociedad para poder corregir posibles malentendidos y aumentar el conocimiento de la población.

Históricamente las empresas buscaban personal cualificado para revisar la forma de la noticia, dándole importancia a la forma y el cómo sería recibida por el público, el giro de 180 grados se observa cuando ahora es más demandado aquellos expertos que no solo buscan perfeccionar la estética, sino el contenido para evitar falsedades o equivocaciones en la propia noticia.

Establecer si la noticia es verdadera o falsa de una forma rotunda no siempre es sencillo por lo que existen páginas webs como:

- `polifact.com` donde se encuentran 6 posibles divisiones en función del nivel de veracidad [48].
- `maldita.es` página web en español donde se tratan artículos de ferviente actualidad y de manera crítica para llegar a la verdad [49].
- `factcheck.org` que sigue un proceso de selección, investigación y edición acerca de noticias de actualidad o demandadas por los lectores [50].

Las diferentes páginas acerca de *fact-cheking explicadas con anterioridad,* funcionan siguiendo la siguiente estructura [51]:

1. Se escoge la noticia o afirmación que desea ser investigada.
2. Se procede a explicar el motivo y el contexto en el cual la noticia o afirmación fue hecha.
3. Se evalúa la afirmación: para este proceso se suele recurrir principalmente a información original o fehacientemente contrastada.
4. Identificación y explicación de los posibles malentendidos o hechos no corroborados de la noticia o afirmación.
5. Se establece si la noticia o afirmación era verdadera o falsa.

También se puede encontrar el *International Fact-Checking Network* (IFCN) que es una unidad del Instituto Poynter [52] que sirve como órganos internacional dedicado a reunir a aquellos periodistas que se dedican a la verificación de hechos. Este instituto redactó un código de buenas prácticas que deben de cumplir los medios pertenecientes a esta asociación para evitar la inclusión de *fake news* en sus artículos. Entre los aspectos que a tener en cuenta destacan el de no partidismo, así como la transparencia con las fuentes y la financiación.

En el panorama nacional solo encontramos un medio entre los 50 integrantes que es *El Objetivo*, programa de La Sexta presentado por Ana Pastor o la empresa Newtral con una zona de verificación de noticias que puede ser fácilmente empleada desde WhatsApp [53].

2.5.2 Enfoque lingüístico: análisis textual del contenido

Con el desarrollo de las técnicas de Deep learning en los últimos años, algoritmos tales como RNN son máquinas poderosas para la lucha contra las *fake news*. Para poder discernir bien entre las noticias realmente falsas, rumores o aquéllas fruto de la sátira. Los detectores optan por centrarse, cada vez más, en el análisis del contenido. En esta clasificación se pone de relieve la importancia del contenido físico (título, cuerpo del texto, imágenes o video) y el contenido no físico como pueden ser los sentimientos, temática o propósito.

Dentro de este enfoque todavía se encuentran dos formas de análisis. El primero de ellos se basa en un análisis más exhaustivo en referencia a la lingüística mientras que el segundo aboga por el análisis semántico.

En cuento al análisis lingüístico destacar los métodos de "*Bag of words*" y "*n-grams*", el primero de ellos analiza cada una de las palabras por separado por lo que no importa el orden de las palabras o la relación semántica lo que produce errores a la hora de interpretar el significado del texto o del contexto. El segundo método busca solucionar este fallo al analizar n ítems que pueden ser desde letras hasta palabras. Algunas de estas técnicas pueden ser el Word2vec o sequence-2-sequence.

Hablando del análisis semántico destacar que se refiere mezcla el uso de *n-gram* con el análisis sintáctico de las oraciones pudiendo analizar así la compatibilidad y consistencia. Por ejemplo, los creadores de noticias falsas tienden a emplear títulos exagerados para atraer la atención de los lectores o posee inconsistencias entre el titular o la noticia en sí. En otras palabas, el análisis semántico permite observar estas características propias de las *fake news* para poder así comprobar y catalogar las diferentes noticias [54].

2.5.3 Métricas de evaluación

Una vez aplicado cualquiera de estos algoritmos y creados los modelos es necesario el empleo de las denominadas, métricas de evaluación para determinar lo bueno o malo que funciona un algoritmo. A este respecto, una de las herramientas más usada es la matriz de confusión, que se emplea para ilustrar el desempeño de un algoritmo de manera más gráfica y simple para el usuario el resultado del sistema. Esta matriz nos muestra si se ha etiquetado incorrecta o correctamente una clase con respecto a otra.

Dentro de la matriz, cada fila representa los diferentes tipos de clase en los que se clasifica la muestra mientras que las columnas hacen referencia a las clases reales. Para explicarlo mejor se recurre a la Tabla 1, donde se cuenta con 10 personas alegres y diez tristes y el sistema va a intentar clasificarlos en alegres o tristes en función de lo que el sistema ha aprendido.

		Tipo de sentimiento real	
		Alegre	Triste
Tipo de sentimiento según algoritmo	Alegre	8 true positives (TP)	4 false positives (FP)
	Triste	2 false negatives (FN)	6 true negatives (TN)

Tabla 2-1 Ejemplo de matriz de confusión

Como se aprecia tenemos 8 de los 10 sentimientos de alegría clasificados correctamente mientras que 2 han sido clasificados incorrectamente. Esta información que se puede extraer de manera intuitiva se realiza tal y como sigue a continuación [55]:

- *Positives/negatives:* hace referencia a la predicción. Si el modelo predice 1 será positivo y si no, negativo.
- *False/true:* implica si la predicción que se ha hecho es correcta o no.
- *Precision (precisión):* Con esta fórmula (Figura 2-12) podemos medir la calidad del modelo de *machine learning* en cuanto a la clasificación se refiere.

$$precision = \frac{TP}{TP + FP}$$

Figura 2-12 Fórmula de la precisión (precision) [55]

- *Accuracy (exactitud):* mide el porcentaje de casos que el modelo ha acertado. Esta es una métrica muy peligrosa dado que suele indicar que un sistema trabaja mejor de lo que en realidad lo hace, la fórmula es la que se ve en la Figura 2-13.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Figura 2-13 Fórmula de la exactitud (accuracy) [55]

- *Recall (exhaustividad):* es la métrica que informa acerca de, habiendo el sistema proporcionado una salida, cual es la cantidad de esos elementos que el modelo es capaz de identificar correctamente. Se puede obtener como se aprecia en la Figura 2-14.

$$recall = \frac{TP}{TP + FN}$$

Figura 2-14 Fórmula de la exhaustividad (recall) [55]

- *F1:* se emplea para combinar las medidas de *recall* y *precision* en una sola. Esto permite una comparacion más directa del rendimiento del modelo, a continuación se indica como puede ser calculada (Figura 2-15).

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figura 2-15 Fórmula para F1 [55]

2.6 Herramientas utilizadas en el TFG

2.6.1 Python

Existen múltiples lenguajes de programación tales como Java, C++, Python o R, pero a día de hoy solo uno de ellos se corona como líder indiscutible del *machine learning*: Python. El punto fuerte de este lenguaje es básicamente el continuo crecimiento que ha provocado la gran comunidad de usuarios que está detrás tratando de mejorarlo. Además, el hecho de que esta plataforma sea abierta y tenga millones de usuarios detrás permite la existencia de miles de librerías con diferentes finalidades entre las que se encuentran aquellas centradas en análisis de datos. Además, su simplicidad lo convierte en un lenguaje ideal para la implementación de códigos.

Figura 2-16 Logo de Python [56]

Este lenguaje de programación fue diseñado por Guido van Rossum a finales de la década de los 80. Se caracteriza por ser un lenguaje multiparadigma dado que permite a los programadores emplear diferentes estilos de programación como son la programación imperativa o la funcional, la descarga se puede hacer fácil y gratuitamente a través de su página web [57].

Para un correcto y eficaz uso de las técnicas de PLN usando *machine learning*, Python ofrece sus propias librerías encaminadas a analizar diferentes datos y aprender de ellos para tomar una solución adecuada. Estas librerías se recogen dentro de *Machine Learning Python* [58]:

1. Amplia selección de librerías y frameworks[3].
2. Código conciso y legible.

[3] Frameworks: Estructura en capas que indica qué tipo de programas pueden o deben ser construidos y cómo se interrelacionan.

3. Agilidad.
4. Colaboración.
5. Código abierto.

2.6.2 R

Este lenguaje de programación se diferencia de Python en que este realiza un mayor análisis estadístico y posee herramientas enfocadas principalmente al cálculo, así como, a la representación de gráficos.

La historia de este lenguaje se remonta a la década de los 60 cuando IBM desarrolla un conjunto de subrutinas para el análisis estadístico denominado Fortran. Como este era un proceso muy tedioso se crea el lenguaje *S* que implementaba librerías de macros Fortran. Debido a los cambios de dueño de este lenguaje de programación la Universidad de Auckland crea en el 1991 *R* como subdialecto de *S* pasando en 1995 a ser software libre [59].

Este lenguaje es otro de los punteros a la hora de hablar de inteligencia artificial gracias a su gran capacidad de cálculo. Para poder aplicar *machine learning* mediante este lenguaje es necesario recurrir al algoritmo predefinido KNN que tiene un funcionamiento similar al que se explicará más adelante en Python.

Se precisa alimentar al sistema con unos datos preprogramados que le indica al programa cómo va a ser su patrón de comportamiento.

Las desventajas que posee este sistema y que, han decantado la balanza, son el lenguaje menos intuitivo que su rival directo o la velocidad de compilación [60].

2.6.3 Pycharm

Como ya se avanzó anteriormente, el lenguaje de programación elegido es Python. Este lenguaje se caracteriza por su simplicidad en la manera de expresar las instrucciones por lo que resulta bastante sencillo e intuitivo para el usuario corriente, en especial para aquellos que comienzan su aventura en el mundo de la programación.

Para facilitar el trabajo de los desarrolladores se suele recurrir a un entorno de desarrollo integrado (IDE) dado que nos permite construir un código de manera más eficaz y sencilla.

El IDE seleccionado para el desarrollo del proyecto es Pycharm. Tras haber analizado otros, tales como Jupyter Notebook, se ha llegado a la conclusión de que el entorno de desarrollo que más se ajustaba a nuestra necesidad era este. Esto se debe a la capacidad que ofrece de buscar módulos entre las diferentes líneas de código y a la facilidad proporcionada a la hora de resolver los diferentes problemas de compilación que puedan surgir, etc.

Figura 2-17 Logo de Pycharm [61]

Pycharm es un IDE desarrollado por la compañía Jetbrains basado en otro IDE de la misma compañía, pero enfocado para Java, *IntelliJ IDEA.* A pesar de ser un programa pesado ofrece numerosas ventajas como son el autocompletado, integración de lenguajes o gran compatibilidad.

La interfaz gráfica que muestra es sencilla de emplear, como se puede apreciar en la siguiente figura, consta de diferentes ventanas desplegables que sirven para facilitar al usuario la búsqueda de ciertos comandos o, simplemente para variar la configuración deseada. Entre sus diferentes opciones también

encontramos diferentes maneras de ejecutar el código implementado como pueden ser lanzarlo la consola de Windows/Linux, la consola de Python o directamente sobre la consola de Pycharm.

La captura de pantalla de la página principal (Figura 2-18), muestra la interfaz que se abre una vez iniciamos la aplicación. La parte principal es la ventana blanca de mayor tamaño, ahí es donde el usuario ha de escribir el código que va a implementar. En el recuadro situado inmediatamente debajo de esta, es la ventana de consola. En esta se permite ver el código corriendo, ver los resultados obtenidos, así como los errores que han de ser corregidos.

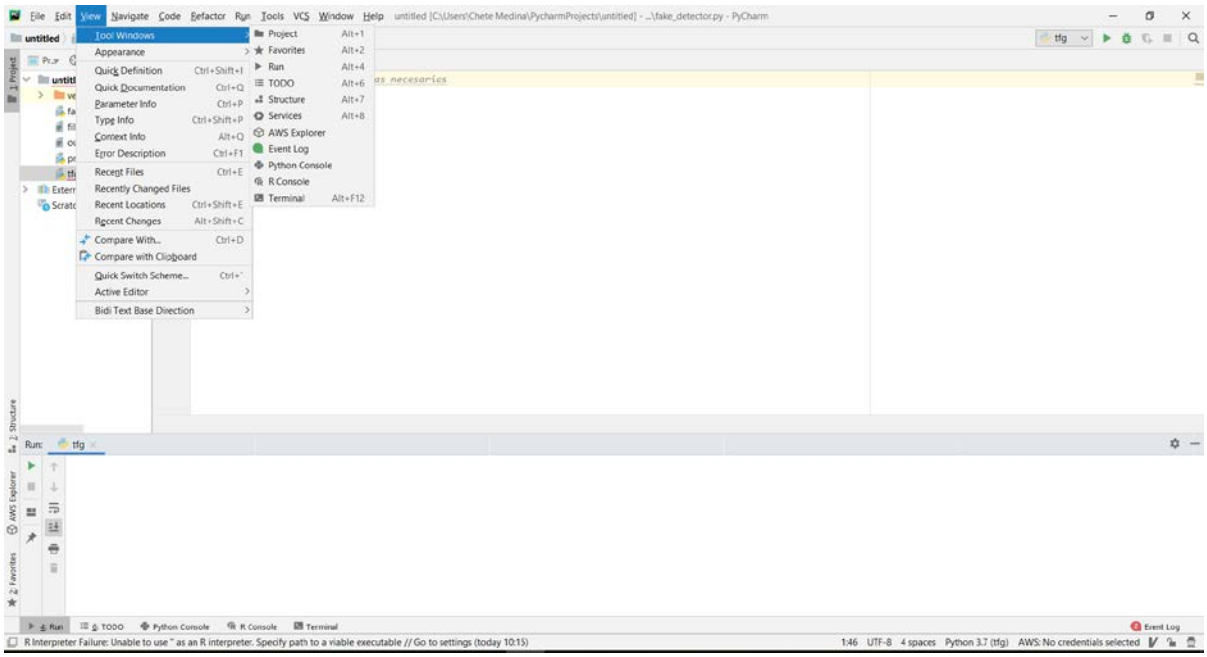

Figura 2-18 Interfaz gráfica de Pycharm (propia)

Una de las configuraciones que es recomendable realizar antes de empezar cualquier proyecto es el intérprete. Esto es un programa que nos permite ejecutar un código sin que exista la necesidad de compilarlo. Estos intérpretes son muy útiles para lanzar códigos y pequeños programas. Otra de las características que hace que un intérprete sea tan útil es que los *scripts*[4] pueden ser fácilmente editados sin necesidad de volver a compilarlos.

El que se ha decido implementar para el desarrollo del programa es el desarrollado por Anaconda.

2.6.1 Anaconda

Python, tal y como muchos otros lenguajes de programación precisa de la descarga y almacenamiento de librerías o módulos. Esto puede ser entendido como una ventaja a la hora de programar, pero también tiene su contrapartida.

Por defecto, todos los proyectos que creemos en nuestro sistema va a utilizar el mismo directorio para buscar estos datos descargados por lo que puede ser un problema cuando estamos realizando varios proyectos de manera simultánea y con diferentes versiones de la librería.

Es por esto que surgen los denominados *enviroments*. Estos buscan dar independencia a cada proyecto y no mezclar las diferentes librerías. Para este proyecto en concreto se ha creado un entorno llamado "`tfg`".

[4] *Script:* Documento donde se encuentra el código de programación.

Figura 2-19 Logo de Anaconda [62]

Enlazando con el uso de las librerías, surge la necesidad de encontrar una fuente de libre distribución para acceder a las diferentes librerías que nos van a permitir desarrollar adecuadamente nuestro código.

Además, estas librerías, necesitan una arquitectura determinada, es decir, para ser correctamente utilizadas se deben instalar siguiendo un orden determinado que no altere la funcionalidad del sistema. Por ello se ha decidido contar con el programa *Anaconda*.

Anaconda es una distribución libre y abierta de los lenguajes de Python y R para códigos relacionados principalmente con lenguaje automático, es decir, *machine learning*. Este sistema de gestión de paquetes fue lanzado por primera vez en 2012.

Está orientado a simplificar la administración y uso de los paquetes de software mediante el sistema de gestión de paquetes *conda*. Se caracteriza por su sencillez a la hora de instalar, correr y actualizar los diferentes softwares, destacando los empleados en el código como pueden ser *Scikit-team*, *TensorFlow* y *SciPy*.

La instalación es verdaderamente intuitiva dado que se puede realizar desde la propia página web del sistema, donde ofrece diferentes opciones según el sistema operativo del usuario [63].

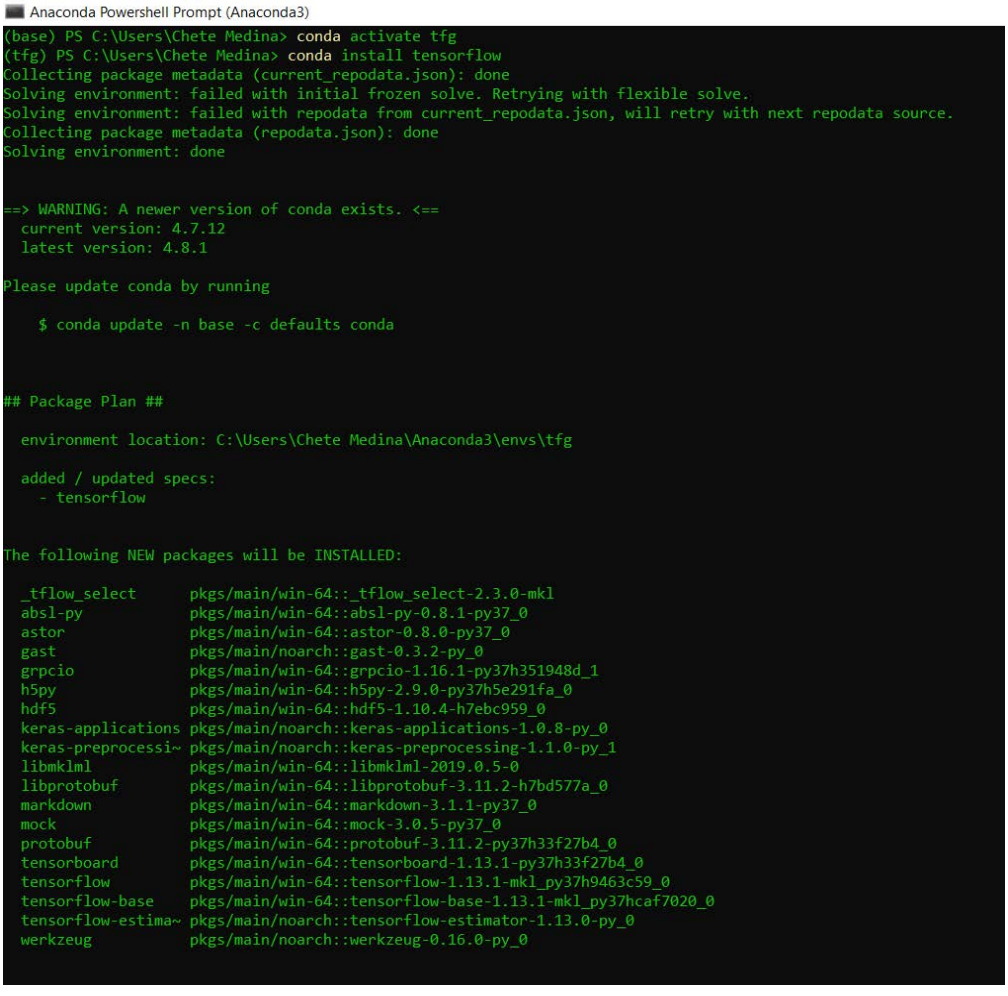

Figura 2-20 Interfaz de Anaconda (propia)

Como se ve en la Figura 2-20, el sistema de instalación de librerías es bastante intuitivo, pero posee su propio lenguaje claramente diferenciado del de Python. Mientras que en el primero se recurre a "`conda install`" el segundo se realiza el mismo proceso a través de "`pip install`". Sin embargo, esto no debe de suponer ningún problema puesto que Anaconda nos permite el uso de ambos sistemas para acomodar al usuario al lenguaje de programación original.

Siguiendo con la descripción de la ilustración se aprecia como para empezar a desarrollar el código antes vamos a precisar descargar ciertas librerías. Para una correcta descarga, Anaconda nos proporciona todos aquellos módulos que nos van a ser necesarios simplificando así el trabajo de tener que ir buscando los módulos uno por uno y asegurando así la descarga en el orden indicado.

3 DESARROLLO DEL TFG

3.1 Word Embedding

Antes de explicar a fondo el desarrollo del TFG es importante hablar de *embedding*, es decir, de qué se trata, qué significa su uso y para qué sirven, dado que este término será de vital importancia para el posterior funcionamiento de los diferentes modelos creados a partir de los algoritmos utilizados.

Podemos definir el *word embedding* como una técnica de modelado de lenguaje que vincula vectores de números reales con las palabras existentes en un texto. Se suelen emplear principalmente en problemas de aprendizaje neuronal debido a que representa el texto en una matriz multidimensional. Esta sustitución se puede hacer en función de la frecuencia y repetición de las palabras en un texto. Otra forma sería hacer esta sustitución en función de la predicción [64].

Durante el desarrollo del trabajo, se han empleado dos formas diferentes de *word embedding*. Se pueden distinguir entre dos: Glove y Word2Vec. El primero de ellos se empleó para implementar el código en lengua inglesa y en los inicios de la construcción del código en lengua española mientras que el último fue el que se impuso en el código en lengua española que analiza las noticias por completo.

El concepto de *word embedding* no sólo es aplicable a palabras de forma individual, sino que también permite vincularse con la formación de lo que se denomina *n-gramas.* Esto es, se toman n palabras agrupadas de manera consecutiva, obteniendo bigramas cuando la agrupación se realiza de dos en dos; trigramas cuando la agrupación en el texto se hace de tres en tres, etc.; y por cada una de estas agrupaciones se vincula con su correspondiente vector.

3.1.1 Glove vs Word2Vec

El primero de ellos es Glove que se caracteriza por ser un algoritmo de aprendizaje sin supervisión para la creación del *embedding* en función de la concurrencia de las palabras de un corpus. El resultado de la aplicación de este algoritmo generad vectores globales para la representación de palabras. En este caso se empleó un vector de 600 dimensiones ya existente para la lengua inglesa [65] y uno de 971, también ya existente, para la lengua hispana [66].

En la Figura 3-1,se aprecia gráficamente la relación semántica entre las diferentes palabras dado que en la misma línea se sitúan palabras afines como pueden ser nombres relacionados con países como son estadounidense, Europa o Afganistán.

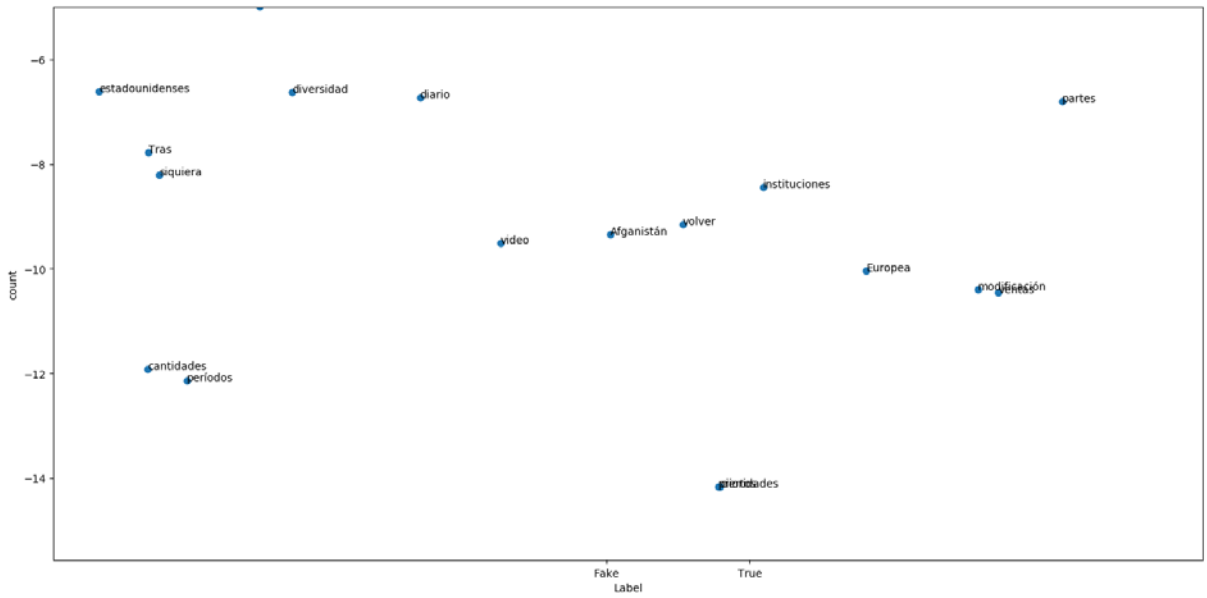

Figura 3-1 Distribución de palabras Glove (propia)

Para la implementación del código usando el archivo en formato Glove en formato `.txt` debido a la mejor lectura por parte del código creado. Una vez que el sistema era capaz de comprender el archivo este se empleaba para sustituir las diferentes palabras que aparecían en el corpus por su equivalencia numérica. Se decidió que el archivo fuera en formato `.txt` para posibilitar la comprobación por parte del usuario de la calidad del archivo y que no existieran palabras inexistentes o mal escritas en lengua española.

Este paso se realiza para que modelos creados a partir de algoritmos como el LSTM vean su precisión incrementada dado que se ofrece mayor relación semántica entre las palabras. Cabe destacar que este paso suele generar problemas debido a la complejidad del mismo y al formato de los archivos Glove. En lengua inglesa el archivo proporcionado por el repositorio *GitHub*[5] es un archivo fuertemente contrastado y que no dispone, en principio, de fallos de estructura y que las palabras se han elegido cuidadosamente para que la relación semántica entre ellas sea sólida, pero en el archivo de lengua española esto no se cumple. Para encontrar el archivo en lengua española se recurrió al mismo repositorio que al empleado en lengua inglesa, pero la oferta de archivos en la lengua de Cervantes es mucho menor por lo que hubo que recurrir al único archivo que se ofrecía en dicha página.

Destacar que GitHub se define como una plataforma de desarrollo colaborativo en el cual se almacenan desde códigos, subidos a la red por usuarios, hasta diferentes datasets [67]

Este archivo tuvo que sufrir una serie de modificaciones manuales tales como eliminar líneas adicionales sin contenido o verificar si las palabras que conformaban el archivo estaban correctamente escritas dado que el poco uso de archivos Glove en lengua española provoca que estos sean publicados por usuarios anónimos y no garantiza una alta fiabilidad pudiendo ser que, el propio archivo, sea una traducción literal del existente en inglés.

La forma de implementar el código, así como la visualización por pantalla de la relación semántica de las palabras es la que se muestra en la Figura 3-2.

[5] Para acceder a este repositorio: https://github.com/

```
###Word embedding
embedding_dict = {}
with open("spanishvector.txt", 'r', encoding="utf_8") as f:
    for line in f:
        values = line.split()
        word = values[0]
        vector = np.asarray(values[1:], "float32")
        embedding_dict[word] = vector

#Graficamos los diferentes vectores
from sklearn.manifold import TSNE

tsne = TSNE(n_components= 2, random_state=0)

words = list(embedding_dict.keys())
vectors = [embedding_dict[word] for word in words]

Y = tsne.fit_transform(vectors[1:3000])

plt.scatter(Y[:,0], Y[:,1])

for label, x, y in zip(words, Y[:, 0], Y[:, 1]):
    plt.annotate(label, xy=(x, y), xytext=(0, 0), textcoords="offset points")
plt.show()
```

Figura 3-2 Glove y visualización por pantalla (propia)

Para el desarrollo de la arquitectura final se recurrió a *Word2Vec*. Este algoritmo es el más adecuado para códigos relacionados con NLTK (librería que se ha empleado para construir el contexto lingüístico de las palabras y que permite hacer un óptimo uso de los algoritmos de *machine learning*). Word2Vec utiliza dos tipos de arquitectura diferente [68]:

- CBOW (*continuous bag of words*): se predice la palabra actual teniendo en cuenta el contexto de la frase, para ello recurre a una ventana de un tamaño preestablecido. Con el tamaño de esta ventana analiza un número n de palabras para analizar el contexto y poder predecir eficientemente la siguiente palabra.
- Skip Gram: a diferencia del caso anterior, este algoritmo se basa en la determinación de las palabras que pueden seguir a una palabra dada.

Una vez probados los métodos solo quedaba decantarse por una de las dos opciones, para ello nos hicimos eco de un estudio llevado a cabo por la Escuela Nacional de Ciencias Informáticas de Túnez [69]. En esta conferencia se compara a fondo los dos modelos explicados con anterioridad, para ello empleaban un corpus de artículos científicos en lengua inglesa obtenidos en la librería digital ACM llegando a la conclusión de que Word2Vec es el algoritmo óptimo para datasets con una cierta relación semántica, en otras palabras, con temas comunes; y el indicado para ser empleado con datasets de gran tamaño.

En cuanto a nuestro código, finalmente es implementado con el artículo de periódico entero y el archivo de entrenamiento nos proporciona los diferentes temas acerca de lo que versan las noticias, esto indica que los temas tratados son limitados y permite que el sistema aprenda con rapidez acerca de economía, política o noticias de sociedad. Es por todo esto que el modelo elegido al tratar el texto de las noticias (considerada de un tamaño suficiente) es el Word2Vec mientras que cuando nos limitamos a analizar simplemente titulares nos quedamos con Glove.

3.1.2 Embedding según el algoritmo utilizado

Antes de explicar los diferentes algoritmos empleados es necesario hablar de una serie de restricciones que surgen a la hora de emplear uno u otro. No es lo mismo utilizar la librería *Keras* que la librería *Sklearn*. En este punto se hablará acerca del tipo de *embedding* que se ha de llevar a cabo en función de la librería y del modelo utilizado.

El modelo usado a partir de la utilización de LSTM se construye con diferentes capas, siendo una de ellas la que se encarga de llevar a cabo el *embedding*. La capa que se emplea (*Embedding Layer*) es muy similar a la función que puede realizar Glove o el Word2Vec.

Por lo tanto, el primer paso es asegurarse de que todos los vectores que acceden a la capa son de la misma longitud. Para ello en el caso de que un vector contenga menos palabras que el número que se le indique, se rellenará con ceros al final del mismo. Para esto se suele recurrir a la función *pad_sequences*.

A esta capa se le han de pasar una cierta cantidad de datos [70]:

- `Embedding`: clase que permite que la formación de la capa *embedding*. Simplemente se les asigna un cierto valor a los diferentes elementos de la final para conseguir recorrer todas las palabras únicas de nuestro corpus.
- `Input_dim`: argumento para especificar el número de filas que posee la matriz.
- `Output_dim`: para especificar el número de columnas de la matriz.
- `Input_length`: máxima longitud de todas las filas.

En la Figura 3-2 se puede apreciar en las primeras líneas como se forma el *embedding*, tal y como se explicó detalladamente en este apartado.

El modelo creado a partir de RNN no sigue exactamente el mismo proceso que el que se acaba de explicar dado que la aplicación del proceso de *embedding* se ha decidido realizar ajeno a las capas que se van a construir y sobre las cuales se cimienta el modelo. El peso del *embedding* se le deja a una función aparte como sucede con las librerías de *Sklearn* y que se explicarán en los párrafos venideros.

En el caso de *Sklearn*, para aplicar el *embedding* hubo que decidir entre Glove y Word2Vec como se explicó en el punto anterior. Una vez tomada la decisión, se procede a diseñar la arquitectura para el funcionamiento de la función.

En el fondo, el mecanismo de *embedding* en la librería *Sklearn* es similar al explicado para los modelos de *Keras*. Solo destacar que, en esta primera librería, al no existir las diferentes capas que facilitan que se recorra todo el corpus se ha de recurrir a bucles que recorrerán la matriz y sustituirán la palabra por el valor numérico correspondiente.

Para que el proceso de *embedding* se complete sin mayor novedad, no solo se ha de prestar atención al corpus del archivo que se está empleando, también es necesaria una revisión del documento Word2Vec, el cual posee el valor que ha de adoptar el vocabulario. El proceso completo de *embedding* se ve reflejado en la Figura 3-3.

```
    text_model = Doc2Vec(min_count=1, window=5, vector_size=vector_dimension, sample=1e-4, negative=5, workers=7, epochs=epoch_num,seed=42)
    text_model.build_vocab(x)
    text_model.train(x, total_examples=text_model.corpus_count, epochs=text_model.iter)

    train_size = int(0.8 * len(x))
    test_size = len(x) - train_size

    text_train_arrays = np.zeros((train_size, vector_dimension))
    text_test_arrays = np.zeros((test_size, vector_dimension))
    train_labels = np.zeros(train_size)
    test_labels = np.zeros(test_size)

    for i in range(train_size):
        text_train_arrays[i] = text_model.docvecs['Text_' + str(i)]
        train_labels[i] = y[i]

    j = 0
    for i in range(train_size, train_size + test_size):
        text_test_arrays[j] = text_model.docvecs['Text_' + str(i)]
        test_labels[j] = y[i]
        j = j + 1

    #print(text_train_arrays)
    return text_train_arrays, text_test_arrays, train_labels, test_labels
```

Figura 3-3 Embedding Sklearn (propia)

3.2 Sistemática

A continuación, se va a proceder a explicar el procedimiento seguido para desarrollar el sistema de detección de *fake news* para español, diferenciando las etapas seguidas (véase Figura 3-4). Simultáneamente, se tratará de poner de relieve los diferentes problemas encontrados, así como la solución que se ha adoptado. A continuación, se explicarán cada una de estas etapas con más detalle.

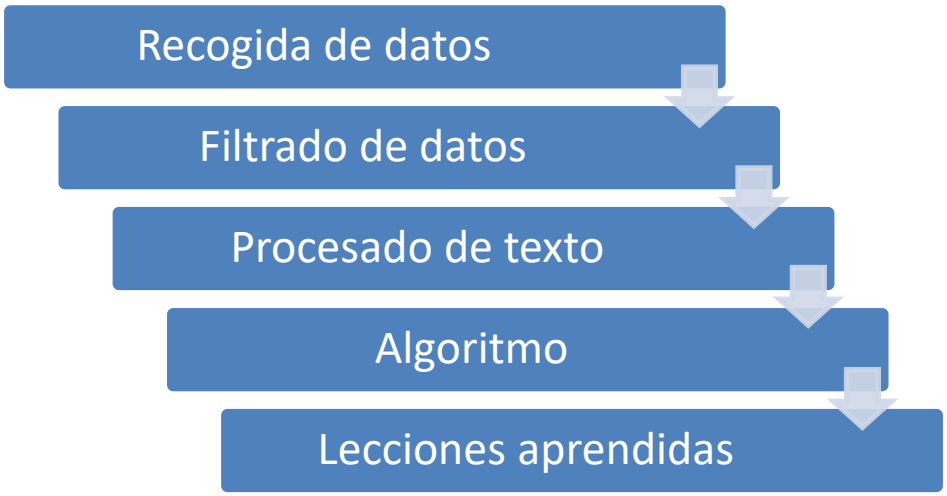

Figura 3-4 Esquema del proceso (propia)

3.2.1 Recogida y filtrado de datos

El primer paso a la hora de llevar a cabo este trabajo, paso fundamental en la consecución del objetivo, es la captura de datos.

Como bien se expuso anteriormente a la hora de explicar cómo funcionan los algoritmos supervisados de *machine learning*, estos precisan de unos datos que van a servir para alimentar su aprendizaje. Esta primera etapa del proceso consiste en realizar una búsqueda exhaustiva de este tipo de datos. Concretamente en el caso del tema que nos ocupa, de artículos de noticias o titulares que estuvieran verazmente documentados y acreditados como noticias falsas o verdaderas.

Llegados a este punto, en un primer momento se plantea un problema. Se trata de la escasa disponibilidad de datos existentes acerca de este tema en lengua española. Este hecho hacer necesario en un inicio proceder a desarrollar y testear los algoritmos diseñados con datos en lengua inglesa.

En este sentido, para encontrar los datos necesarios se ha acudido a repositorios de fuente libre para localizar noticias ya clasificadas. La fuente más empleada para este fin es *GitHub*. Concretamente, se ha hecho uso inicialmente del dataset proporcionado por el usuario `Tariq60` [71]. Los datos ahí almacenados y que se han empleado consisten en 2550 titulares de noticias que han sido clasificados por personas con anterioridad como "TRUE" o "FALSE". Estos datos se encuentran almacenados en formato `.csv` para facilitar la lectura de las dos columnas (*statment*, *label*) y poder trabajar con ellas de manera independiente. Se dará más detalles en la fase de validación.

Cabe destacar la mínima difusión de datos clasificados para esta temática en lengua española, por lo que inicialmente el programa fue implementado en lengua inglesa.

Antes de pasar a la fase de preprocesado, es necesario realizar un paso previo de los textos categorizados que componen el dataset para una correcta separación entre las noticias falsas de las verdaderas. Para ello se siguió un procedimiento bastante sencillo y rudimentario.

Lo primero que se decidió hacer es pasar estos archivos `.csv` -delimitados por comas- a archivos `.txt`. Este paso se llevó a cabo dado que es más fácil manipular ficheros con este tipo de extensiones. Una vez convertido a formato de texto se decidió, mediante el bloc de notas, eliminar todas las comas y comillas para que así, la única ',' existente fuera aquella que separaba el *statment* del *label*, es decir, la que separaba el titular de la noticia de la etiqueta de verdadero o falsa.

Says the Annies List political group supports third-trimester abortions on demand.	FALSE
When did the decline of coal start? It started when natural gas took off that started to begin in (President George W.) Bushs administration.	TRUE
Hillary Clinton agrees with John McCain by voting to give George Bush the benefit of the doubt on Iran.	TRUE
Health care reform legislation is likely to mandate free sex change surgeries.	FALSE
The economic turnaround started at the end of my term.	TRUE
The Chicago Bears have had more starting quarterbacks in the last 10 years than the total number of tenured (UW) faculty fired during the last two decades.	TRUE
Jim Dunnam has not lived in the district he represents for years now.	FALSE
I'm the only person on this stage who has worked actively just last year passing along with Russ Feingold some of the toughest ethics reform since Watergat	TRUE
However it took $19.5 million in Oregon Lottery funds for the Port of Newport to eventually land the new NOAA Marine Operations Center-Pacific.	TRUE
Says GOP primary opponents Glenn Grothman and Joe Leibham cast a compromise vote that cost $788 million in higher electricity costs.	TRUE
For the first time in history the share of the national popular vote margin is smaller than the Latino vote margin.	TRUE
Since 2000 nearly 12 million Americans have slipped out of the middle class and into poverty.	TRUE
When Mitt Romney was governor of Massachusetts we didnt just slow the rate of growth of our government we actually cut it.	FALSE
The economy bled $24 billion due to the government shutdown.	TRUE
Most of the (Affordable Care Act) has already in some sense been waived or otherwise suspended.	FALSE
In this last election in November ... 63 percent of the American people chose not to vote ... 80 percent of young people (and) 75 percent of low-income wor	TRUE
McCain opposed a requirement that the government buy American-made motorcycles. And he said all buy-American provisions were quote 'disgraceful.'	TRUE
U.S. Rep. Ron Kind D-Wis. and his fellow Democrats went on a spending spree and now their credit card is maxed out	FALSE
Water rates in Manila Philippines were raised up to 845 percent when a subsidiary of the World Bank became a partial owner.	TRUE
Almost 100000 people left Puerto Rico last year.	TRUE
Women and men both are making less when you adjust for inflation than when John Kitzhaber was first elected governor.	FALSE
The United States has the highest corporate tax rate in the free world.	TRUE
We just had the best year for the auto industry in America in history.	TRUE
Says Scott Walker favors cutting up to 350000 families and children off health care.	TRUE
Says Mitt Romney wants to get rid of Planned Parenthood.	FALSE
I dont know who (Jonathan Gruber) is.	FALSE
Hate crimes against American Muslims and mosques have tripled after Paris and San Bernardino.	TRUE
Rick Perry has never lost an election and remains the only person to have won the Texas governorship three times in landslide elections	TRUE

Figura 3-5 Muestra del dataset en inglés con formato `.csv` (propia)

Una vez que tenemos claramente diferenciados las dos columnas el archivo debería de ser leído sin ningún tipo de problema.

Cabe destacar, como se aprecia en la Figura 3-5, las frases que vamos a utilizar como texto son sentencias de carácter corto que fácilmente se pueden asociar a titulares de noticia, no hace falta recurrir a toda la noticia para conseguir una clasificación efectiva.

Uno de los pasos más comunes del preprocesado es la eliminación de las palabras de parada (*stopwords*), palabras tales como preposiciones, conjunciones o determinantes que no aportan valor semántico al texto. Debido a la escasa longitud del texto y a que el algoritmo LSTM funciona bien sin

eliminar este tipo de palabras se ha decidido obviar este paso y, así, no arriesgarse a que disminuya la precisión del modelo.

Cabe destacar que uno de los primeros problemas que se encontró al intentar cargar todo el texto es el formato y las dimensiones del archivo. A pesar de ser un archivo guardado en formato `.csv` como el analizado en lengua inglesa, la división en columnas no seguía un patrón claro por lo que se recurrió a otro programa de apoyo: *EmEditor*. Este editor de texto nos permitió abrir el archivo sin aparentemente ningún problema, así como reconvertirlo a formato `UTF-8`. Una vez que el texto era perfectamente legible para Python solo quedaba restructurar las columnas para seguir el formato que se seguía en lengua española.

Para este último paso se recurrió a las herramientas de este programa para eliminar las comas que se encontraban a lo largo del texto y sustituirlas por punto y coma. Esto nos permitió que las únicas comas fueran aquellas que separaban la columna de las etiquetas de la columna del texto.

Una vez que se comprobó la viabilidad del proyecto en lengua inglesa, analizando tan solo los titulares de las noticias, se decidió probar con un dataset en lengua española. Para ello, y tras una profunda búsqueda en internet, se encontró un congreso, donde una de las actividades que llevan a cabo consiste en una competición internacional para la detección de noticias falsas para el idioma español. Gracias a la existencia de esta competición se dispone de un dataset en lengua hispana de diferentes noticias previamente clasificadas según sean fake o no fake. Como bien se desarrollará en los siguientes puntos, se hizo la prueba con este dataset analizando en una primera instancia simplemente los titulares de las noticias y una vez alcanzando un resultado aceptable se pasó a implementarlo con la noticia entera [72].

Una vez comprobado que los sistemas anteriores funcionaban de manera adecuada, se decide implementar un código más complejo que sea capaz de leer el archivo proporcionado por *MEX-A3T*, de manera directa y no solamente el titular como se explicó en párrafos anteriores. Para ello se va a emplear el archivo entero diferenciando entre las columnas que nos proporcionan tanto la etiqueta de noticia verdadera o falsa, el titular de la noticia y, por último, la noticia entera. A diferencia de cómo se trataron estos datos con anterioridad, tanto tildes como otros caracteres que no está recogidos en el formato `UTF-8` serán tratados directamente con el código.

Toda vez que se ha obtenido el formato deseado, similar al que tenemos en la figura, es el momento de pasar al preprocesado de texto en sí mediante el empleo de código en Python.

3.2.2 Preprocesado de texto

En este punto del TFG, se procede a hacer una limpieza del texto en cuestión y dejarlo preparado para que el sistema de detección que se desarrolle pueda leer el archivo fácilmente sin que haya posibilidad de error.

A diferencia de cómo se explicó en el apartado 3.2.1, esta limpieza se tratará de hacer de forma sistemática y mediante código y no de manera manual como se hizo previamente a través del bloc de notas.

Para que este proceso sea posible, antes de nada, hay que descargar algunas librerías que van a facilitar el proceso de implementación [56]:

a) *Pandas*: librería de Python destinada al análisis de datos, proporciona estructuras de datos flexibles. Esta se emplea desde el principio del código para la lectura del archivo que contiene las noticias. El uso de esta librería se aprecia en la Figura 3-6, en la cual se llama al archivo `.csv` y se informa de que el archivo está escrito con caracteres legibles en código `UTF-8.`

```
52
53  def getEmbeddings(path,vector_dimension=300):
54      data = pd.read_csv(path, encoding="utf8")
55
```

Figura 3-6 Uso de Pandas en el código desarrollado (propia)

b) *Numpy*: librería que se emplea para el cálculo científico a través de Python o para el correcto uso de matrices de N dimensiones. Un ejemplo de su uso es la Figura 3-7, para gestionar el tamaño de las matrices compuestas por los datos de test y los de entrenamiento.

```
76      text_train_arrays = np.zeros((train_size, vector_dimension))
77      text_test_arrays = np.zeros((test_size, vector_dimension))
78      train_labels = np.zeros(train_size)
79      test_labels = np.zeros(test_size)
```

Figura 3-7 Uso de Numpy en el código desarrollado (propia)

c) *Re*: esta librería hace referencia a aquellas expresiones regulares, es decir, permite comprobar si una cadena de texto (*string*) se corresponde con otra expresión. Una buena forma de entenderlo, como explica Python en su propia página web, es que, si A es una cadena de texto conocida, y B es otra cadena de texto, entonces, AB también será una cadena de texto. Esta librería también permite el uso de comandos tales como "\n", "( )" u otros que se nos puedan ocurrir. Además de los casos comentados con anterioridad, en nuestro texto también utilizamos esta librería en forma de *re.sub*. Este se utiliza, como en la Figura 3-8, con el fin de obtener otra cadena de texto habiendo eliminado los caracteres que se le indiquen.

```
30  def textClean(text):
31      text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)
32      text = text.lower().split()
```

Figura 3-8 Uso de Re en el código desarrollado (propia)

d) *Keras*: esta librería juega un papel fundamental al encontrarse tanto en el preprocesado como en el algoritmo en sí. En cuanto al preprocesado destacar que se emplea en la realización del tokenizado. Este módulo funciona como un escáner léxico que se encarga de dividir las cadenas de texto iniciales en unidades más pequeñas haciendo posible que estas se moldeen a gusto del usuario. Una de las funciones atribuibles a este módulo es contar el número de palabras diferentes que vamos a encontrar en todo el texto.
Otro de los módulos dentro de la librería de Keras relacionado con el preprocesado es *pad_sequences*. La función de este reside en transformar una lista, en una matriz de dos dimensiones con un número de filas y columnas predeterminado; esto nos permite comparar a posteriori los vectores de texto y el de las etiquetas y trabajar con cada uno de ellos por separado.

e) *Gensim:* librería de código de texto que se emplea para análisis de textos con la implementación, como en este caso, de Word2vec. Esta librería será la encargada de llevar a cabo el *embedding*.

```
68
69          text_model = Doc2Vec(min_count=1, window=5, vector_size=vector_dimension,
70                               sample=1e-4, negative=5, workers=7, epochs=epoch_num,seed=42)
71          text_model.build_vocab(x)
72          text_model.train(x, total_examples=text_model.corpus_count, epochs=text_model.iter)
73
```

Figura 3-9 Uso de Gemsim en el código desarrollado (propia)

En la Figura 3-9, se aprecia cómo se emplea el *Gemsim* con la función Doc2Vec con el fin de realizar el *embedding*. Es recomendable pasarle una serie de factores tales como *Window* para marcar la distancia máxima entre la palabra actual y la predicha o *workers* para el entrenamiento del modelo.

Una vez explicadas cuales han sido las herramientas necesarias, se pasa a explicar los pasos de preprocesado llevados a cabo.

```
11  my_stop_words = [...]
28
29  def textClean(text):
30      text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)
31      text = text.lower().split()
32      # stops = set(stopwords.words("spanish"))
33      stops = set(my_stop_words)
34      text = [w for w in text if not w in stops]
35      text = " ".join(text)
36      return (text)
37
38
39  def cleanup(text):
40      text = textClean(text)
41      text = text.translate(str.maketrans("", "", string.punctuation))
42      return text
43
44
```

Figura 3-10 Preprocesado (Parte 1) (propia)

a) Función *textClean*: en la Figura 3-10 se muestra la primera función del preprocesado, en la primera línea de esta función se sustituye por un espacio todos aquellos caracteres diferentes a los universales como pueden ser, por ejemplo, la "Ç". Posteriormente se convierte el texto a minúscula para finalmente, en las cuatro últimas líneas de la función, recorrer el texto y eliminar aquellas palabras que aparecen en el vector *my_stop_words* .
El paso de las *stopwords* no es determinante a la hora de aplicar el algoritmo dado que existen algunos como LSTM que funcionan igual con estas palabras o sin ellas, pero se ha decidido implementar debido a su utilización con varios algoritmos y a que se disminuye el tiempo total de compilación del programa. Las *stopwords* que se han decidido eliminar son las que se pueden observar en la Tabla 3-1.

```
my_stop_words =
['a','al','algo','algunas','algunos','ante','antes','como','con','contra','cual',
'cuando','de','del','desde','donde','durante','e','el','ella','ellas','ellos','en
',

'entre','era','erais','eran','eras','eres','es','esa','esas','ese','eso','esos','
esta','estaba','estabais','estaban','estabas','estad','estada','estadas','estado'
,

'estados','estamos','estando','estar','estaremos','estará','estarán','estarás','e
staré','estaréis','estaría','estaríais','estaríamos','estarían','estarías','estas
```

```
',

'este','estemos','esto','estos','estoy','estuve','estuviera','estuvierais','estuv
ieran','estuvieras','estuvieron','estuviese','estuvieseis','estuviesen',

'estuvieses','estuvimos','estuviste','estuvisteis','estuviéramos','estuviésemos',
'estuvo','está','estábamos','estáis','están','estás','esté','estéis','estén',

'estés','fue','fuera','fuerais','fueran','fueras','fueron','fuese','fueseis','fue
sen','fueses','fui','fuimos','fuiste','fuisteis','fuéramos','fuésemos','ha',

'habida','habidas','habido','habidos','habiendo','habremos','habrá','habrán','hab
rás','habré','habréis','habría','habríais','habríamos','habrían','habrías',

'habéis','había','habíais','habíamos','habían','habías','han','has','hasta','hay'
,'haya','hayamos','hayan','hayas','hayáis','he','hemos','hube','hubiera',

'hubierais','hubieran','hubieras','hubieron','hubiese','hubieseis','hubiesen','hu
bieses','hubimos','hubiste','hubisteis','hubiéramos','hubiésemos','hubo','la','la
s',

'le','les','lo','los','me','mi','mis','mucho','muchos','muy','más','mí','mía','mí
as','mío','míos','nada','ni','no','nos','nosotras','nosotros','nuestra','nuestras
',

'nuestro','nuestros','o','os','otra','otras','otro','otros','para','pero','poco',
'por','porque','que','quien','quienes','qué','se','sea','seamos','sean',

'seas','seremos','será','serán','serás','seré','seréis','sería','seríais','seríam
os','serían','serías','seáis','sido','siendo','sin','sobre','sois','somos','son',

'soy','su','sus','suya','suyas','suyo','suyos','sí','también','tanto','te','tendr
emos','tendrá','tendrán','tendrás','tendré','tendréis','tendría','tendríais',

'tendríamos','tendrían','tendrías','tened','tenemos','tenga','tengamos','tengan',
'tengas','tengo','tengáis','tenida','tenidas','tenido','tenidos','teniendo',

'tenéis','tenía','teníais','teníamos','tenían','tenías','ti','tiene','tienen','ti
enes','todo','todos','tu','tus','tuve','tuviera','tuvierais','tuvieran',

'tuvieras','tuvieron','tuviese','tuvieseis','tuviesen','tuvieses','tuvimos','tuvi
ste','tuvisteis','tuviéramos','tuviésemos','tuvo','tuya','tuyas','tuyo','tuyos',

'tú','un','una','uno','unos','vosotras','vosotros','vuestra','vuestras','vuestro'
,'vuestros','y','ya','yo','él','éramos']
```

Tabla 3-1 Stopwords (propia)

b) Función *cleanup*: esta función se emplea para eliminar los signos de puntuación del texto.
c) Función *constructLabeledSentences*: nos permite construir las diferentes filas de la matriz y pasar el texto a Unicode, Figura 3-11.

```
46      def constructLabeledSentences(data):
47          sentences = []
48          for index, row in data.iteritems():
49              sentences.append(LabeledSentence(utils.to_unicode(row).split(), ['Text' + '_%s' % str(index)]))
50          return sentences
51
```

Figura 3-11 Preprocesado parte 2 (propia)

d) Función *GetEmbeddings*: la última función del preprocesado, también es la más larga debido a que, sobre ella, recae el peso del embedding y del correcto etiquetado y su transformación

```
53  def getEmbeddings(path,vector_dimension=300):
54      data = pd.read_csv(path, encoding="utf8")
55
56      missing_rows = []
57      for i in range(len(data)):
58          if data.loc[i, 'text'] != data.loc[i, 'text']:
59              missing_rows.append(i)
60      data = data.drop(missing_rows).reset_index().drop(['index','id'],axis=1)
61
62      for i in range(len(data)):
63          data.loc[i, 'text'] = cleanup(data.loc[i, 'text'])
```

Figura 3-12 Preprocesado parte 3 (propia)

En estas primeras líneas de la función, Figura 3-12, se recorre las filas del archivo comprobando si existe alguna fila sin contenido para eliminarla y que no afecte al funcionamiento del algoritmo, además, posteriormente se llama a la función *Cleanup* para hacer la limpieza del texto.
En la Figura 3-13, se recorre la etiqueta de cada uno de los artículos para clasificarlos como 1 si la noticia es veraz y con cero si estamos ante una *fake news*.

```
65      x = constructLabeledSentences(data['text'])
66      y = data['category'].values
67      y = np.array(list(map(lambda x: 1 if x == "True" or x == 1 else 0, y)))
68
```

Figura 3-13 Preprocesado parte 4 (propia)

Las siguientes líneas de código de la función hacen referencia al *embedding* en sí, como se aprecia en la Figura 3-14 y Figura 3-15, sustituyendo las diferentes palabras por un valor numérico para poder entrenar los modelos posteriormente de manera correcta. Destacar que este paso no se ha realizado con el modelo obtenido a partir de LSTM (ver apartado 3.1).

```
69      text_model = Doc2Vec(min_count=1, window=5, vector_size=vector_dimension,
70                           sample=1e-4, negative=5, workers=7, epochs=epoch_num,seed=42)
71      text_model.build_vocab(x)
72      text_model.train(x, total_examples=text_model.corpus_count, epochs=text_model.iter)
73
```

Figura 3-14 Preprocesado parte 5 (propia)

El último paso de la función es dividir los datos que tenemos en datos de test y de entrenamiento. Pero una vez hecha la distribución se ha de comprobar las dimensiones de ambas matrices para comprobar que no existe disparidad entre las matrices que contienen el texto de las noticias y las que contienen su corriente etiquetado.

```
77      text_train_arrays = np.zeros((train_size, vector_dimension))
78      text_test_arrays = np.zeros((test_size, vector_dimension))
79      train_labels = np.zeros(train_size)
80      test_labels = np.zeros(test_size)
81
82      for i in range(train_size):
83          text_train_arrays[i] = text_model.docvecs['Text_' + str(i)]
84          train_labels[i] = y[i]
85
86      j = 0
87      for i in range(train_size, train_size + test_size):
88          text_test_arrays[j] = text_model.docvecs['Text_' + str(i)]
89          test_labels[j] = y[i]
90          j = j + 1
91
92      #print(text_train_arrays)
93      return text_train_arrays, text_test_arrays, train_labels, test_labels
94
```

Figura 3-15 Preprocesado parte 6 (propia)

3.2.3 Algoritmos empleados

Una vez realizados los pasos previos, se procede a implementar la segunda parte del código en el lenguaje de Python. En esta parte se analizará el algoritmo que clasifica las noticias en fake o no fake, como se entrenan los modelos.

Para seguir una estructura similar a la que se ha seguido en el apartado 3.2.2, el primer paso es proceder a definir las diferentes librerías empleadas, así como el motivo por el cual se han usado:

a) *Sklearn o Scikit-learn:* es una librería que provee de una cantidad ingente de algoritmos con y sin supervisión a Python. Esta librería está estrechamente relacionada con otras que han de ser instaladas con anterioridad tales como "*Numpy*" o "*Pandas*". Esta librería permite realizar acciones relacionadas con la regresión, clasificación, clustering o el preprocesado. En este código hemos usado esta librería para implementar los modelos a partir de *Naive Bayes* y *SVM*.

b) *Keras:* es una librería que incluye algoritmos de alto nivel de redes neuronales que se utiliza para correr encima de otras tales como *TensorFlow*. Esta librería nos permite definir los diferentes modelos de predicción que van a ser realizados, alimentarlos con los datos de entrenamiento y, por último, realizar la predicción.

c) *Matplotlib:* esta librería se usará para realizar las diferentes gráficas de las matrices de confusión de los diferentes modelos obtenidos.

Una vez definidas todas las librerías empleadas, el sistema creado se puede subdividir en función de los algoritmos utilizados. A continuación, se procede a explicar los diferentes modelos obtenidos a partir de los algoritmos en función de la precisión que han arrojado:

- LSTM: para poder implementar correctamente este modelo se han seguido unos primeros pasos encaminados a la legibilidad del mismo. El primer paso a realizar es cargar los ficheros que se guardaron al lanzar el *GetEmbeddings* y que contiene las matrices de test y de entrenamiento con el preprocesado ya hecho y que nos permiten lanzar al programa con mayor rapidez, en el caso que estos no existieran se lanzaría el programa desde el principio. Es importante la existencia de estos ficheros, como se analizará posteriormente en los resultados, debido a que permiten que los diferentes algoritmos se entrenen con los mismos

datos de train y por lo tanto la comparación para saber cuál de los modelos es más eficiente se puede realizar de manera exacta.
Las siguientes líneas del código (Figura 3-16), se emplean para el proceso de codificar los datos de entrenamiento y test, así como asegurar que todas las noticias están correctamente etiquetadas. El último paso es la creación del modelo en sí.

```
101  # Create the model
102  embedding_vector_length = 32
103  model = Sequential()
104  model.add(Embedding(top_words+2, embedding_vector_length, input_length=max_review_length))
105  model.add(LSTM(100))
106  model.add(Dense(1, activation='sigmoid'))
107  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
108  print(model.summary())
109
110  #from keras.utils.vis_utils import plot_model
111  #plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
112
113  model.fit(X_train, y_train, validation_data=(X_test, y_test),
114            epochs=epoch_num, batch_size=batch_size)
115
116  # Final evaluation of the model
117  scores = model.evaluate(X_test, y_test, verbose=0)
118  print("Accuracy= %.2f%%" % (scores[1]*100))
119
120  # Draw the confusion matrix
121  y_pred = model.predict_classes(X_test)
122  plot_cmat(y_test, y_pred)
```

Figura 3-16 Modelo LSTM (propia)

Para la creación del modelo ha sido necesario definirlo como *sequential*, es decir, el modelo va a estar compuesto por una serie de capas:

1. La primera capa es la encargada de hacer el *embedding* al modelo, para ello es necesario tener en cuenta diferentes entradas como son el número máximo de palabras, el tamaño de los vectores en el cual las palabras han sufrido el *embedding* y el tamaño del vocabulario que se encuentra en el texto.
2. La segunda capa indica el número de unidades en las celdas LSTM, es decir, una matriz horizontal de esas unidades. En esencia, la capa contiene múltiples unidades LSTM paralelas, estructuralmente idénticas, pero cada una se encarga de "aprender a recordar" algo diferente [73].
3. La tercera capa, *Dense*, que se emplea para cambiar las dimensiones del vector. En este caso le indicamos la forma de salida es 1 y que la activación se realizará mediante *sigmoid* dado que la salida devuelve un valor entre 0 y 1.

Una vez creadas las diferentes capas es necesario compilar el sistema. Esta función define la función de pérdida[6], el optimizador de los pesos de las conexiones y las métricas que queremos obtener.
El último paso para crear el modelo, es alimentarlo. Para ello se introducen los datos de entrenamiento, tanto la noticia como su etiqueta, y se le indica al sistema el número de iteraciones, así como el número de muestras que será propagadas por la red.

- Naive-Bayes: La arquitectura de este modelo es más sencilla que en el modelo anterior dado que no precisa de la creación de diferentes capas (Figura 3-17). El proceso de *embedding* se realiza fuera de lo que es el modelo.

[6] Función de pérdida: es una función usada para determinar el error entre la salida de nuestro algoritmo y el valor dado. Según la definición de Layman, hace referencia a lo lejos que está nuestra máquina del resultado real [72]

El primer paso es llamar a la función *GaussianNB( )* que será la encargada de crear el clasificador Naive Bayes.

```
25  from sklearn.metrics import accuracy_score
26  gnb = GaussianNB()
27  gnb.fit(xtr,ytr)
28  y_pred = gnb.predict(xte)
29
30  print(accuracy_score(yte, y_pred))
31
32  plot_cmat(yte, y_pred)
33
```

Figura 3-17 Modelo Naive-Bayes (propia)

Después de crear el modelo es necesario alimentarlo con los datos de entrenamiento para, finalmente, realizar la predicción gracias a la función *gnb.predict*.

- SVM: este modelo es muy similar al Naive-Bayes, simplemente se diferencia en que, en este caso, la librería empleada nos proporciona el modelo en cuestión (Figura 3-18).

```
24  from sklearn.metrics import accuracy_score
25  clf = SVC()
26  clf.fit(xtr, ytr)
27  y_pred = clf.predict(xte)
28
29  print(accuracy_score(yte, y_pred))
30  plot_cmat(yte, y_pred)
```

Figura 3-18 Modelo SVM (propia)

- Neural Network: en este último caso se ha seguido una estructura similar a la que se siguió con el LSTM, pero el número de capas creadas es mayor. Esta estructura consiste en la creación de unas capas determinadas para, posteriormente, compilar el sistema y alimentarlo con los datos de test y de entrenamiento. A continuación, se procede a definir las diferentes capas creadas y cuál es su uso dentro del modelo, para ello primero se define el modelo como secuencial (Figura 3-19):
    1. La primera capa hace referencia a *Dense*, como ya sabemos, se emplea para cambiar la dimensión del vector, en este caso emplearemos 256 neuronas. Los otros datos que se pueden apreciar en esta primera capa son la dimensión de entrada, así como los métodos de activación de inicialización del *kernel*.
       El modelo de redes neuronales necesita ciertos pesos y luego los itera para obtener mejores resultados, este *kernel_initializer* es una forma de inicializar estos valores dado que creará números de una distribución estadística como pesos iniciales.
    2. La siguiente capa recurre al *Dropout*, método en el cual se desactiva un cierto número de neuronas aleatorio en cada iteración obligando así que no se dependa tanto de las conexiones cercanas.
    3. Las diferentes capas hasta llegar a la capa de compilación siguen la misma estructura que las dos anteriores, una de ellas se emplea para *Dense* y otra para *Dropout*.

Al contrario que en el LSTM en este modelo se decidió crear un propio optimizador para la compilación con el fin de mejorar el rendimiento.

Una vez definida la arquitectura del modelo, antes de alimentarlo, es preciso realizar el *Label_Encoder* que se emplea para codificar etiquetas de una característica categórica en valores numéricos entre 1 y 0.

La siguiente función que realizamos es la creación de *one-hot vector* gracias a la función *np.utils.to_categorical*. Este paso es necesario debido a que los datos categóricos no son aplicables en este tipo de modelos por lo que se precisa convertir las etiquetas a un valor binario.

```
    '''Neural network with 3 hidden layers'''
    model = Sequential()
    model.add(Dense(256, input_dim=300, activation='relu', kernel_initializer='normal'))
    model.add(Dropout(0.3))
    model.add(Dense(256, activation='relu', kernel_initializer='normal'))
    model.add(Dropout(0.5))
    model.add(Dense(80, activation='relu', kernel_initializer='normal'))
    model.add(Dense(2, activation="softmax", kernel_initializer='normal'))

    # gradient descent
    sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

    # configure the learning process of the model
    model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
    return model


model = baseline_model()
model.summary()
x_train, x_test, y_train, y_test = train_test_split(xtr, ytr, test_size=0.2, random_state=42)
label_encoder = LabelEncoder()
label_encoder.fit(y_train)
encoded_y = np_utils.to_categorical((label_encoder.transform(y_train)))
label_encoder.fit(y_test)
encoded_y_test = np_utils.to_categorical((label_encoder.transform(y_test)))
estimator = model.fit(x_train, encoded_y, epochs=epoch_num, batch_size=batch_size, verbose=1,validation_split=0.001)
print("Model Trained!")
score = model.evaluate(x_test, encoded_y_test)
print("")
print("Accuracy = " + format(score[1]*100, '.2f') + "%")
```

Figura 3-19 Modelo Neuronal Network (propia)

4 RESULTADOS EXPERIMENTALES

4.1 Dataset Github

4.1.1 Origen del dataset

El dataset empleado para probar el algoritmo LSTM únicamente en titulares se ha descargado de la página web Github gracias al usuario `Tariq60` [71]. Este dataset está compuesto de 2550 titulares de noticias con la etiqueta de "TRUE" o "FAKE".

Este dataset que se encuentra guardado en formato `.csv` está balanceado de tal manera que el 51% de los titulares son "TRUE" con una temática variada que abarca desde temas de política hasta economía pasando por otros tales como deporte.

4.1.1 Estructura del corpus

Estos datos se encuentran almacenados en dos columnas *statment* y *label*. *Label* hace referencia a la etiqueta de la noticia, es decir, si es fake o no lo es y el *statment* que es el titular de la noticia en sí como se ve en la Figura 3-5.

4.2 Dataset congreso MEX-A3T

4.2.1 Origen del dataset

Como se mencionó en el capítulo 3, el dataset que se empleó para la implementación final del código fue proporcionado por la organización de la edición de 2019 del congreso MEX-A3T, sobre detección de noticias falsas en español al rellenar la inscripción [72].

El dataset proporcionado es una recopilación de noticias procedentes de diferentes fuentes de internet tales como periódicos de renombre, páginas web que se dedican a la comprobación de las noticias y páginas que normalmente publican noticias falsas. Todas las noticias proceden de fuentes mejicanas.

El corpus original tiene 971 noticias recolectadas desde 2018, aunque debido a que la inscripción se ha realizado antes de las fechas permitidas solo podemos contar con 686 noticias que han sido clasificadas como verdaderas o falsas siguiendo un proceso de clasificación manual:

- La noticia es verdadera si hay evidencias de que la noticia ha sido publicada en fuentes fiables.
- La noticia es falsa si proviene de fuentes que se dedican a la divulgación de noticias falsas o si una página web especializada en detección de noticias falsas ha afirmado que tal noticia no es verdadera.

- Se han agrupado las noticias en parejas con el fin de que un mismo tema tenga tanto noticias falsas como verdaderas.

Para evitar que el dataset verse sobre un solo tema y permitir un correcto entrenamiento del sistema el corpus cubre noticias de 9 temas diferentes tales como ciencia, deportes, economía, educación, entretenimiento, política, salud, seguridad y sociedad.

Con el mismo fin, pero para evitar que el sistema está balanceado se realiza una división similar entre noticias verdaderas y falsas resultando 343 noticias de cada tipo.

4.2.2 Estructura del corpus

El corpus que se ha empleado posee la siguiente información y estructura, tal y como se aprecia en la Figura 4-1:

- ID: número de la noticia en la que estamos
- Categoría: no fake/ Fake
- Tema: uno de los mencionados en el punto anterior
- Titular: el titular de la noticia en cuestión
- Texto: la noticia en cuestión
- Link: URL de la noticia

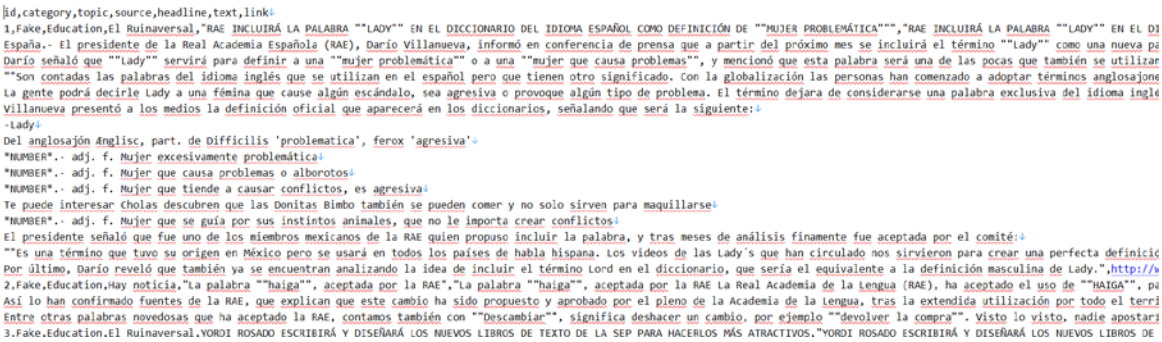

Figura 4-1 Dataset utilizado (propia)

4.3 Compilación

Una vez finalizada la implementación del sistema, se procede a compilar el código.

Lo primero es definir qué significa compilar. Este término informático se puede definir como el proceso de traducir las líneas de código que se han escrito en un lenguaje de programación concreto. Esta compilación precisa de traducir las instrucciones contenidas en el texto a lenguaje binario [74].

Los compiladores se dividen en dos partes, el *Front End* parte encargada de analizar y verificar el código fuente y el *Back End* que es la parte en la cual es generado el código máquina [75].

El código se ha dividido según los diferentes modelos para facilitar la realización y posterior comprensión por parte del usuario. A continuación, se procede a explicar los resultados obtenidos en función del modelo.

Destacar que cada uno de los datos de entrenamiento y de test de cada compilación será guardado en un fichero para poder reutilizarlo y que la comparación entre los diferentes modelos sea lo más realista posible.

4.3.1 Modelo LSTM

La primera vez que se procede a compilar este modelo, nos obliga a llamar a la función *getEmbedding2*. Esta función, como ya se explicó con anterioridad con *getEmbedding*, es la encargada

de realizar el preprocesado para los modelos de *Keras*, a saber, LSTM y RNN. Posteriormente, se procede a dividir las diferentes noticias del corpus entre datos de entrenamiento y de test.

Una vez que se dividen las noticias se decide guardarlas en diferentes ficheros, Figura 4-2, para poder acceder a ellas siempre que se precise. Por lo tanto, se van a crear 4 ficheros diferentes dos para los de entrenamiento (uno para el texto y otro para el etiquetado) y otros dos para el test.

```
77      xtr = X_train
78      xte = X_test
79      ytr = y_train
80      yte = y_test
81
82      np.save('xtr_shuffled.npy',xtr)
83      np.save('xte_shuffled.npy',xte)
84      np.save('ytr_shuffled.npy',ytr)
85      np.save('yte_shuffled.npy',yte)
```

Figura 4-2 Guardado de datos en ficheros (propia)

Una vez completado este paso se procede a compilar el modelo en sí. Una vez que comienza la compilación, el primer resultado que aparece por pantalla es el resumen del modelo que se va a proceder a entrenar.

Este resumen nos muestra las diferentes capas que conforman el modelo, con la forma de las diferentes matrices. Finalmente, nos muestra los diferentes parámetros con los que cuenta el modelo, como se puede apreciar en la Figura 4-3.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 32)           160064
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 213,365
Trainable params: 213,365
Non-trainable params: 0
_________________________________________________________________
None
```

Figura 4-3 model.summary LSTM (propia)

Los diferentes parámetros que podemos encontrar en el modelo son:

- Nombre y tipo de todas las capas existentes en el modelo.
- Las dimensiones de salida de cada capa.
- Número de parámetros ponderados para capa.
- El número total de parámetros entrenables y no entrenables del modelo.

Destacar que el *None* que se aprecia en la dimensión de salida de las diferentes capas hace referencia a que el modelo espera un tamaño de entrada que puede ser flexible gracias a este valor nulo.

La siguiente función en ser compilada es la alimentación del modelo con los datos de entrenamiento y luego verificar si el modelo ha aprendido bien con los datos de test. Para que esto se lleve a cabo se le ha de indicar el número de iteraciones que deseamos que el modelo realice (*epochs*) en nuestro caso hemos decidido que sean 15 y, además, el número de muestras, 64, que serán propagadas por la red (*batch_size*).

Si el programa no contiene errores debería aparecer por pantalla lo que se aprecia en la Figura 4-4, donde se ve el estado en la iteración número 3. También se puede observar que el número de datos recogido para realizar la iteración es de 64 por cada vez hasta llegar a los 540 que completan el número de datos que se guardan en la muestra de entrenamiento.

En cada iteración el sistema va a ir ofreciendo un valor relacionado con la función de perdida y la precisión que tiene en cada iteración e, incluso, cada muestreo.

```
Epoch 3/15

 64/540 [==>...........................] - ETA: 3s - loss: 0.6751 - acc: 0.8281
128/540 [======>.......................] - ETA: 3s - loss: 0.6734 - acc: 0.8672
192/540 [=========>....................] - ETA: 2s - loss: 0.6701 - acc: 0.8438
256/540 [=============>................] - ETA: 2s - loss: 0.6674 - acc: 0.8164
320/540 [================>.............] - ETA: 1s - loss: 0.6625 - acc: 0.8094
384/540 [====================>.........] - ETA: 1s - loss: 0.6554 - acc: 0.7943
448/540 [=======================>......] - ETA: 0s - loss: 0.6317 - acc: 0.8013
512/540 [===========================>..] - ETA: 0s - loss: 0.6238 - acc: 0.7793
```

Figura 4-4 Entrenamiento del modelo (propia)

4.3.2 Modelo Naive-Bayes y SVM

Ambos modelos poseen un proceso de compilación más rápido que el que se aprecia en el modelo anterior debido a la inexistencia de las diferentes capas.

En estos no se produce una iteración visible entre las muestras de entrenamiento ya que la propia función que nos ofrece la librería *Sklearn* está especializada en procesos de aprendizaje de máquinas y se encarga de realizar todo el proceso de iteraciones por sí mismo. Para entenderlo mejor, se puede entender que estas funciones son como una caja negra a través de las cuales no podemos observar que es lo que pasa, pero sí que podemos obtener el resultado del proceso.

4.3.3 Modelo RNN

El proceso de compilación de este modelo se desarrolla de manera similar al de cómo el modelo LSTM se ha realizado. El primer paso es la impresión por pantalla de un resumen del modelo como se ve en la Figura 4-5.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 2000)              602000
_________________________________________________________________
dropout_1 (Dropout)          (None, 2000)              0
_________________________________________________________________
dense_2 (Dense)              (None, 2000)              4002000
_________________________________________________________________
dropout_2 (Dropout)          (None, 2000)              0
_________________________________________________________________
dense_3 (Dense)              (None, 2000)              4002000
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 4002
=================================================================
Total params: 8,610,002
Trainable params: 8,610,002
Non-trainable params: 0
_________________________________________________________________
```

Figura 4-5 model.summary RNN (propia)

En este resumen se aprecian las mismas características que en el modelo LSTM, pero con un número mayor de capas como consecuencia directa de esto, el número de parámetros asciende considerablemente.

Tras el resumen se procede a alimentar el modelo con los datos de entrenamiento para después realizar la comprobación con los de test. En este modelo se realizan con 25 iteraciones y 64 números de muestras alimentar el modelo en los diferentes muestreos.

Por último, a los 4 modelos se les pedirá el valor de los parámetros de *Accuracy, Precision, Recall y F1* que proceden de la librería *Sklearn* como se puede apreciar en la Figura 4-6.

```
127  # accuracy: (tp + tn) / (p + n)
128  accuracy = accuracy_score(y_test, y_pred)
129  print('Accuracy: %f' % accuracy)
130  # precision tp / (tp + fp)
131  precision = precision_score(y_test, y_pred)
132  print('Precision: %f' % precision)
133  # recall: tp / (tp + fn)
134  recall = recall_score(y_test, y_pred)
135  print('Recall: %f' % recall)
136  # f1: 2 tp / (2 tp + fp + fn)
137  f1 = f1_score(y_test, y_pred)
138  print('F1 score: %f' % f1)
```

Figura 4-6 Análisis de resultados (propia)

4.1 Ejecución

4.1.1 Resultados del dataset del congreso MEX-A3T

Para analizar los resultados se van a estudiar una serie de características de los 4 modelos empleados con el dataset proporcionado por el congreso MEX-A3T, es decir, sobre el dataset completo. Para poder comparar cómo funcionan y determinar cuál de ellos se comporta mejor y por qué. Además, para una mejor comprensión, se recurrirá a visualizar gráficamente las respectivas matrices de confusión.

4.1.1.1 Métricas

Antes de comenzar con el análisis de los resultados obtenidos (Tabla 4-1), se recomienda la lectura del apartado 2.3.1 en él se definen las métricas que aquí se muestran.

Modelo	Precision	Accuracy	Recall	F1
LSTM	81.81%	69.85%	52.17%	63.71%
Naive-Bayes	66.67%	63%	42.42%	51.85%
SVM	61.76%	61.76%	39.39%	50%
RNN	52%	*55.57%*	*68.7*%	57.8%

Tabla 4-1 Resultados obtenidos sobre el dataset en español

La exactitud (*accuracy*) es el método más directo de comprobar cómo trabaja un sistema. En este caso el que se sitúa a la cabeza de los modelos es LSTM con un 69.85% frente al 63% del obtenido mediante Naive-Bayes.

La precisión se calcula de manera similar a la exactitud, pero solo teniendo en cuenta los verdaderos positivos y falsos positivos. Esta medida es útil en campos como la detección de spam dado que un falso positivo es peor que un falso negativo, es decir, eliminar un email importante es peor que dejar pasar una noticia spam. En este campo LSTM vuelve a dominar con solvencia dado que parece distinguir fácilmente cuando una noticia es verdadera.

La exhaustividad (*recall*) se calcula teniendo en cuenta únicamente los verdaderos positivos y los falsos negativos. Esta medida da una estimación de la cobertura que se está dando en los resultados. En este campo parece que el RNN es el que mejor cobertura logra con las noticias *fake* debido a que posee el mayor *recall*.

Por último, destacar el caso de *F1* [76]. Esta medida indica como de correcto es el sistema teniendo en cuenta los valores de precisión y de exhaustividad. En esta medida el LSTM también vuelve a ser el claro ganador por lo que se puede argumentar que este es el mejor modelo para comprobar la veracidad o falsedad de una noticia.

Destacar que, a pesar de hacer todas estas medidas con el texto completo, se podría implementar solo con el titular de la noticia y se obtendrían resultados similares para el modelo LSTM debido a que este modelo está pensado para textos cortos; en cambio, el resto de modelos vería enormemente afectada su precisión dado que precisan de una gran cantidad de información para poder hacer un correcto entrenamiento y poder digerir los datos que están analizando.

4.1.1.2 Matrices de confusión

Una vez analizados los resultados matemáticos se emplea la matriz de confusión para ver cómo realmente tratan los diferentes modelos las noticias de test. Con esta imagen se puede analizar fácilmente como está clasificando las noticias en función de si son *fake* o no.

En las siguientes gráficas, las filas representan las etiquetas reales donde 0 representa una noticia no fake y el 1 que hace referencia a las fake; en cuanto a las columnas, es la etiqueta que asigna el sistema donde 0 vuelve a ser no fake y 1 fake,

La matriz de confusión asociada al modelo LSTM se puede ver en la Figura 4-7. Este modelo funciona correctamente a la hora de clasificar la mayoría de las noticias fake . En cambio, a la hora de tratar con las noticias que son fake se aprecia cómo tiende a catalogarlas como de manera correcta algo más que el 50% de las veces.

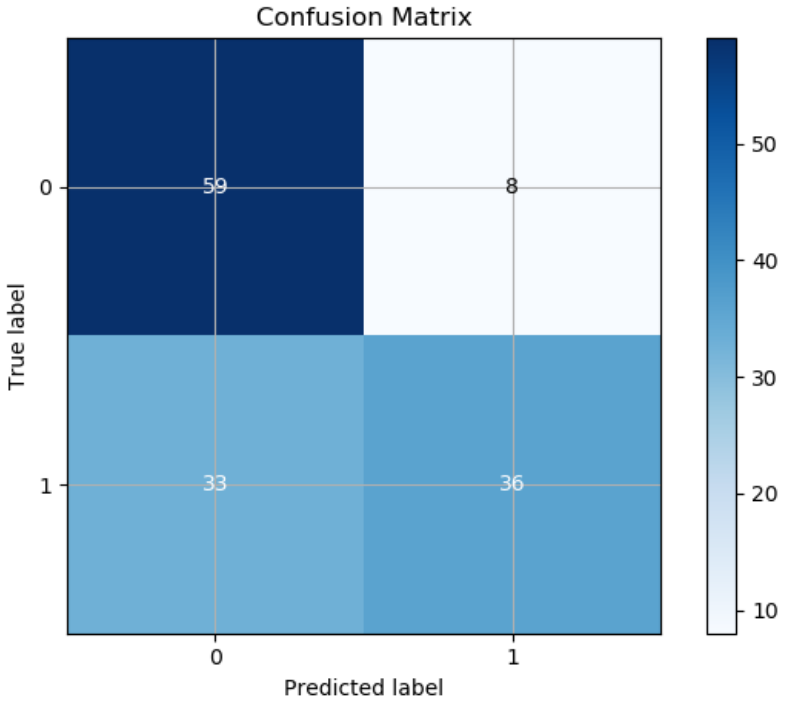

Figura 4-7 Matriz de confusión LSTM (propia)

La siguiente matriz de confusión obtenida es la que procede del modelo Naive-Bayes, ilustrada en la Figura 4-8. En ella se aprecia que la clasificación de las noticias no fake funciona ligeramente diferente al del anterior modelo y que la tendencia de clasificar las noticias fake como no fake sigue al alza. Este porcentaje además es mayor que en el anterior dado que solo encontramos 26 noticias que realmente son *fake*.

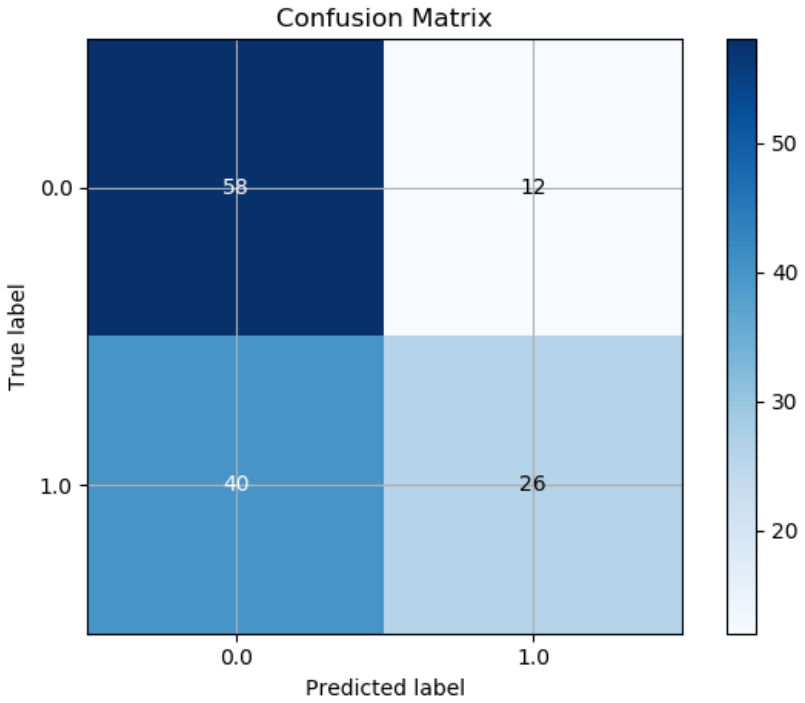

Figura 4-8 Matriz de confusión Naive-Bayes (propia)

La matriz de confusión de la Figura 4-9 corresponde con el del SVM y que arroja unos resultados similares a los que ofrecía el anterior modelo. No se consigue solucionar el problema de catalogar los *fake* correctamente dado que la mayoría de las noticias *fake* son catalogadas incorrectamente.

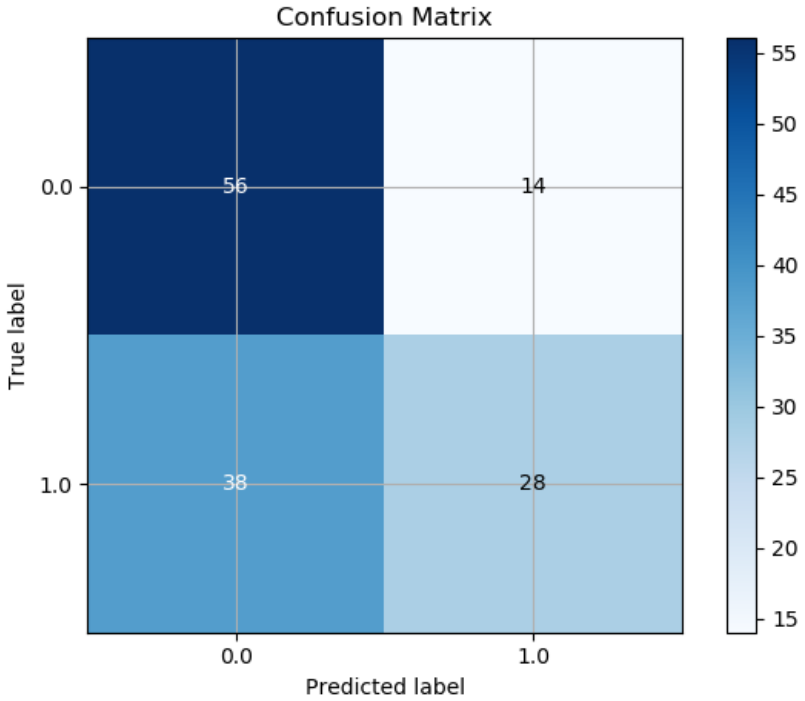

Figura 4-9 Matriz de confusión SVM (propia)

Por último, se procede a analizar los resultados que nos ofrece el modelo RNN en la Figura 4-10. Este es el modelo que peor resultados muestra, en cuanto a las noticias no fake tiende al 50% de acierto en la clasificación, no siendo determinante para saber si la noticia es fake o no fake. Con las noticias

fake muestra mejores registros que los tres anteriores dado que el porcentaje de estas noticias que cataloga correctamente supera al 50% por primera vez.

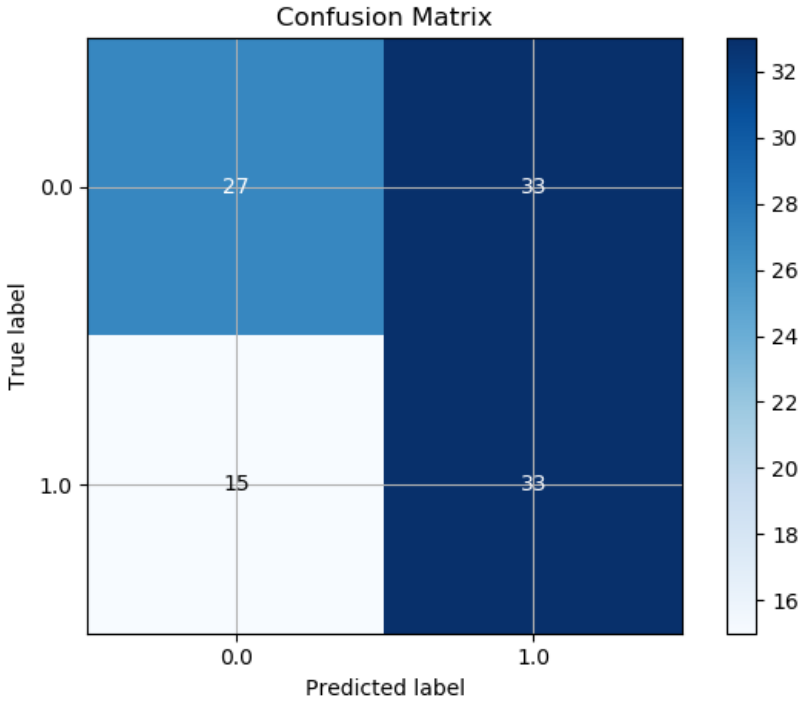

Figura 4-10 Matriz de confusión RNN (propia)

4.1.2 Resultados del dataset de GitHub

Una de las implementaciones que se ha llevado a cabo es permitir que el usuario introduzca una noticia y que sea el sistema el que establezca si la noticia es fake o no fake.

Para realizar esta función se ha precisado realizar todos los pasos que se han realizado con anterioridad para que el modelo pudiera después clasificarlo. Uno de los principales problemas que esto suponía era que las dimensiones de las matrices no concordaban dado que tenía que seguir exactamente el formato de las noticias con las que se ha adiestrado el modelo.

La solución pasa por la creación de un modelo que funcione de manera diferente de cómo funciona el modelo que se ha explicado con anterioridad. A pesar de que el funcionamiento es similar, el preprocesado del texto se hace de manera ligeramente diferente. En esta implementación se realiza gracias al *tokenizado*. Esto se realiza con el comando *Tokenize* (Figura 4-11). Otra de las funciones que se necesita emplear para que las diferentes matrices sean uniformes y tengan el mismo tamaño es el *pad_sequences*.

Esta primera función se emplea como un analizador léxico para Python en el cual se separa el texto en las diferentes palabras para posteriormente hacer el *embedding* a las mismas. En este proceso se ha decidido utilizar Glove en lugar de Word2Vec para comprobar si, usando otro diccionario, el porcentaje de precisión variaría sustancialmente.

Después de este proceso se crea el modelo LSTM, como ya se vio en el punto 4.3.1, alimentándolo con los diferentes datos de entrenamiento y de test para luego obtener la precisión del sistema y entender cómo está funcionando este modelo y comprender si este parámetro es válido para posteriormente clasificar un texto.

El primer problema que arroja este sistema, comparándolo con el que se realizó y se explicó desde el principio, es el tiempo de compilación. Mientras que para realizar la compilación de los 4 modelos

iniciales llega con 5 minutos, simplemente para compilar el modelo de LSTM con esta nueva forma de procesado nos alargamos hasta la hora y veinte minutos en el mejor de los casos.

```
56  from keras.preprocessing.text import Tokenizer
57  tokenizer = Tokenizer(num_words=max_words)
58  tokenizer.fit_on_texts(X)
59  sequences = tokenizer.texts_to_sequences(X)
60  word_index = tokenizer.word_index
61  #print('Vocabulary size:', len(word_index))
62
63  ###Padding para hacerlo uniforme
64  from keras.preprocessing.sequence import pad_sequences
65
66  data = pad_sequences(sequences, padding= 'post',
67                       maxlen=max_length_sentence)
68  print('Shape of data tensor', data.shape)
69  print('Shape of label tensor', Y.shape)
70
```

Figura 4-11 Tokenizado (propia)

Sin embargo, los resultados obtenidos por este modelo se antojan ligeramente peores a los obtenidos con el modelo inicial (Tabla 4-2). Además de estos resultados en cuanto a precisión, exactitud, *recall* y F1 es importante destacar que, al modelo en cuestión, LSTM, se le ha proporcionado un titular de una noticia.

Modelo	Precision	Accuracy	Recall	F1
LSTM	80.76%	75.73%	64.62%	71.79%

Tabla 4-2 Resultados modelo LSTM

En este caso se le ha proporcionado el titular, como por ejemplo “Obama ha decidido lanzar una bomba nuclear esta semana”. El sistema va a proporcionar un valor entre 0 y 1 que hace referencia al porcentaje de veracidad de la noticia. Siendo un 0 que la noticia es completamente falsa y un 1 si la noticia se clasifica como veraz. En el caso del ejemplo el sistema imprime por pantalla un 0.2 es decir, es falsa en un 80%.

Como en los modelos anteriores, también se visualiza la matriz de confusión, Figura 4-12, que nos permite observar rápidamente como está funcionando el modelo. Gracias a esta imagen se puede observar como este modelo tiene mejor rendimiento tanto a la hora de clasificar las noticias verdaderas y la fake.

Destacar que este modelo no ha sido entrenado con el mismo dataset con el que fue entrenado el modelo que se explica en el punto 4.3.1. Este dataset consta simplemente con cerca de 2000 titulares de noticias clasificados como fake o true. Este dataset también está compensado buscando que existiera un número similar entre las noticias veraces y las falaces

El punto negativo de este dataset es que no está organizado según la temática, es decir, el modelo puede estar mal entrenado al tratar de temas políticos y no poder clasificar con exactitud una noticia que sea, por ejemplo, de biología.

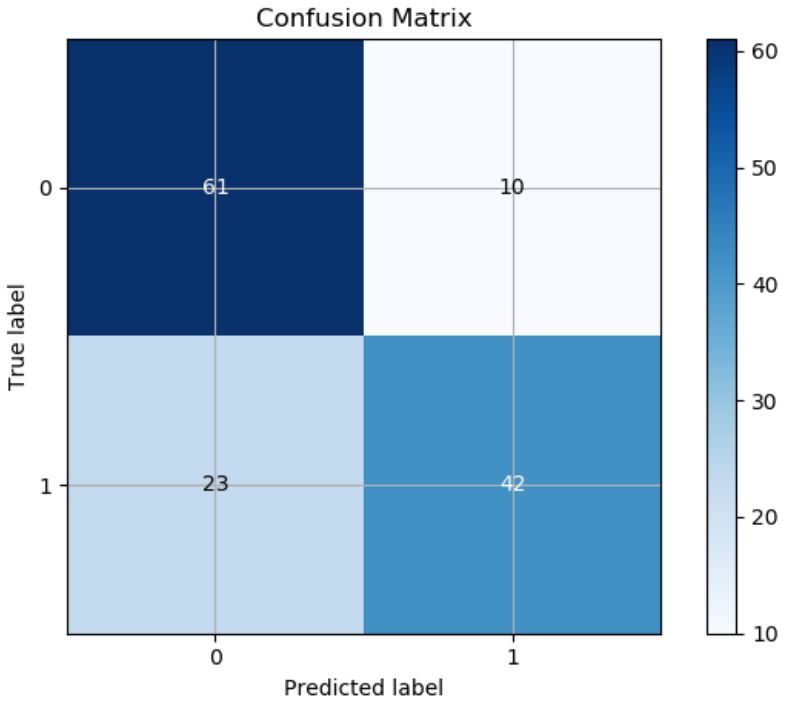

Figura 4-12 Matriz de confusión LSTM' (propia)

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

El objetivo principal propuesto al inicio del TFG consistía en la creación de un sistema capaz de detectar *fake news* en noticias en español. Derivados de este objetivo surgen otros tales como la implementación de un código eficaz y optimizado usando un lenguaje de programación lo más adecuado posible que nos ayude a cumplimentar el objetivo mencionado. Estos objetivos han sido detallados en los capítulos 3 y 4 de la memoria conformando el sistema que se pretendía implementar al comienzo de este TFG. Por tanto, se pueden considerar cumplidos a la finalización de este proyecto.

Gracias a la investigación previa a la realización del desarrollo del código, se ha podido advertir la práctica inexistencia de información acerca de esta materia en lengua española. Las investigaciones llevadas a cabo en el ámbito de la detección de *fake news* usando técnicas de PLN mediante *machine learning* se limitan en su mayoría a su aplicación para la lengua inglesa. A diferencia de otros ámbitos de trabajo, como por ejemplo en el caso del análisis de sentimiento, la detección de *fake news* aplicado al español está en sus inicios,

Es por tanto que este TFG se puede considerar en cierta medida novedoso en la lucha de la detección de noticias falsas gracias a la utilización de técnicas de PLN usando *machine learning*. Los modelos que se han empleado son LSTM, Naive-Bayes, SVM y RNN siendo el primero de estos el que mejores resultados ha proporcionado.

Uno de los principales problemas que se encontraron a la hora de desarrollar el código fue encontrar un dataset adecuado en lengua española. La mayoría de los que se encontraron en esta lengua eran muy pocas noticias como para entrenar un modelo de *machine learning* o no estaban correctamente clasificados. Otro de los problemas a los que hubo que hacer frente durante la realización de este trabajo fue la aplicación correcta del uso del *embedding* debido a que, en muchos de los casos, no se producía el aprendizaje de los modelos como debiera, tendiendo a clasificar todas las noticias como fake.

La interpretación de los diferentes modelos ha permitido comprobar que los diferentes sistemas de análisis tienden a un mejor comportamiento con las noticias verdaderas mientras que las *fake news* son más difíciles de clasificar como tale. Esto se ha verificado gracias a la matriz de confusión de los diferentes modelos.

5.2 Líneas futuras

A pesar de la consecución de los objetivos propuestos, este TFG tiene capacidad de mejora. Una de las posibles líneas de mejora es el tiempo de compilación y ejecución de la detección de una noticia proporcionada por el usuario. Esta implementación debe comenzar por permitir que esta función se

ajuste al código dado y que no se precise que el usuario siga un formato determinado para posibilitar una detección rápida y eficaz.

Además de esta posibilidad es factible adoptar este modelo como una aplicación para un teléfono móvil para que un usuario pueda comprobar en línea la veracidad de una noticia que está consultando al momento. Esta implementación permitiría luchar contra la desinformación reinante en la sociedad actual y a la continua caída de la credibilidad de los artículos de noticias.

Existe también la posibilidad de implementar este código para noticias, este puede ser modificado para trabajar en otros campos de *machine learning* para encaminarlo hacia el campo de la Defensa. Este código puede ser empleado para analizar diferentes factores que puedan ser clasificados como booleanos como son los del TFG, es decir, verdadero o falso, bueno o malo…

6 BIBLIOGRAFÍA

En esta sección figurarán todas las referencias, sean recursos web, libros, artículos, etc., incluyendo la información de autores, título de la obra, nombre de la publicación, año, edición y enlace más fecha de último acceso en el caso de referencias a recursos online.

[1] M. Barthel, «Pew Research Center,» 15 Diciembre 2016. [En línea]. Available: https://www.journalism.org/2016/12/15/many-americans-believe-fake-news-is-sowing-confusion/. [Último acceso: 2 Enero 2020].

[2] REUTERES, «El Departamento de Justicia de EEUU abre una investigación criminal sobre la pesquisa de la trama rusa,» Europapress, 20 Octubre 2019.

[3] J. Benítez, «Cazadores de 'fake news': así funciona la tecnología que evitará que te manipulen,» El Mundo, 12 Septiembre 2018.

[4] D. Robinson, «Stackoverflow,» 6 Septiembre 2017. [En línea]. Available: https://stackoverflow.blog/2017/09/06/incredible-growth-python/. [Último acceso: 11 Enero 2020].

[5] S. Tesich, «A Government of Lies,» The Nation, Enero 1992.

[6] J. A. Zarzalejos, «Comunicación, periodismo y fast-checking,» Revista uno, nº 27, p. 11, 2017.

[7] F. Berckemeyer, «La mentira de la posverdad,» Revista uno, nº 27, pp. 26-27, 2017.

[8] A. Cortazar, «El Boletín,» 2 Diciembre 2016. [En línea]. Available: https://www.elboletin.com/nacional/142815/mayoria-espanoles-no-creen-periodismo.html.

[9] S. B. Glasser, «brookings.edu,» 2 Diciembre 2016. [En línea]. Available: https://www.brookings.edu/essay/covering-politics-in-a-post-truth-america/.

[10] [En línea]. Available: elperiodico.com.

[11] C. Silverman, «BuzzFeed.News,» 16 Noviembre 2016. [En línea]. Available: https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook.

[12] J. E. Baidez Guillen, Fake News. Evolución, ámbitos de desarrollo y repercusión en cibermedios nacionales., Universidad Complutense de Madrid, 2018.

[13] K. Lacapria, «WikiLeaks Confirms Hillary Clinton Sold Weapons to ISIS?,» Snope, 13 Octubre 2016.

[14] J. R. Hancock, «El País,» 17 Noviembre 2016. [En línea]. Available: https://verne.elpais.com/verne/2016/11/17/articulo/1479386366_405746.html. [Último acceso: 19 Febrero 2020].

[15] A. Beaujon, «Washingtonian,» 2 Octubre 2019. [En línea]. Available: https://www.washingtonian.com/2019/10/02/trump-claims-he-invented-the-term-fake-news-an-interview-with-the-guy-who-actually-helped-popularize-it/. [Último acceso: 22 Enero 2020].

[16] A. Beujon, «https://www.washingtonian.com,» 2 Octubre 2019. [En línea]. Available: https://www.washingtonian.com/2019/10/02/trump-claims-he-invented-the-term-fake-news-an-interview-with-the-guy-who-actually-helped-popularize-it/.

[17] C. Rodríguez, «El Mundo,» 3 Noviembre 2017. [En línea]. Available: https://www.elmundo.es/cultura/cine/2017/11/03/59fc80f4468aebd1508b46a0.html.

[18] L. Blanco, «La Vanguardia,» 01 Abril 2018. [En línea]. Available: https://www.lavanguardia.com/vida/20180401/442106640262/marc-amoros-las-fake-news-estan-de-moda-pero-no-son-solo-una-moda.html. [Último acceso: 9 Enero 2020].

[19] U. C. d. Madrid, «I Estudio sobre el impacto de las fake news en España,» 2017. [En línea]. Available: https://d3vjcwm65af87t.cloudfront.net/novacdn/EstudioPescanova.pdf.

[20] J. Soll, «The long and brutal history of fake news,» 18 Diciembre 2018. [En línea]. Available: https://www.politico.com/magazine/story/2016/12/fake-news-history-long-violent-214535.

[21] I. Imprescindibles, «ideasimprescindibles.es,» 15 Junio 2018. [En línea]. Available: https://ideasimprescindibles.es/fake-news-noticias-falsas/.

[22] P. Pardo, «El Mundo,» 12 Diciembre 2016. [En línea]. Available: https://www.elmundo.es/cronica/2016/12/12/584bf995e5fdea39528b463b.html.

[23] N. R. Wei, «Today Singapure,» 3 Abril 2019. [En línea]. Available: https://www.todayonline.com/singapore/look-what-world-saying-about-singapores-bill-online-falsehoods. [Último acceso: 22 Febrero 2020].

[24] G. Francés, «gouverment.fr,» 21 Noviembre 2018. [En línea]. Available: https://www.gouvernement.fr/en/against-information-manipulation.

[25] C. Feikert-Ahalt, «Gobierno de Reino Unido,» [En línea]. Available: https://www.loc.gov/law/help/fake-news/uk.php.

[26] cope.es, «cope.es,» 04 Enero 2020. [En línea]. Available: https://www.cope.es/actualidad/espana/noticias/propuesta-sanchez-luchar-contra-las-fake-news-mentira-desata-las-carcajadas-del-congreso-20200104_586207.

[27] D. Lu, «newscientist.com,» 21 Octubre 2019. [En línea]. Available: https://www.newscientist.com/article/2221963-facebook-has-a-plan-to-tackle-fake-news-heres-why-it-wont-work/.

[28] R. TO, «The Objetive,» 1 Diciembre 2017. [En línea]. Available: https://theobjective.com/la-estrategia-de-seguridad-nacional-incluye-por-primera-vez-las-fake-news-como-amenaza/.

[29] F. A. Fernández-Montesinos, Cuadernos de Estrategia 197. La posverdad. Seguridad y defensa., Instituo Español de Estudios Estratégicos, 2018.

[30] F. H. Valls, «La Información,» 10 Julio 2019. [En línea]. Available: https://www.lainformacion.com/espana/cni-paz-esteban-fake-news-ciberataques/6506175/.

[31] C. C. Nacional, 19 Febrero 2019. [En línea]. Available: https://www.ccn-cert.cni.es/comunicacion-eventos/comunicados-ccn-cert/7680-como-actuar-frente-a-las-campanas-de-desinformacion-en-el-ciberespacio-uno-de-los-mayores-retos-de-seguridad-del-pais.html. [Último acceso: 9 Febrero 2020].

[32] «Ciencias de la Computación,» [En línea]. Available: https://sites.google.com/site/cienciasdelacompucion/inteligencia-artificial.

[33] Salesforce, «Salesforce,» 22 Junio 2017. [En línea]. Available: https://www.salesforce.com/mx/blog/2017/6/Que-es-la-inteligencia-artificial.html.

[34] Avellanos. [En línea]. Available: http://avellano.fis.usal.es/~lalonso/RNA/index.htm.

[35] P. G. Bejerano, «Blogthinkbig,» 8 Febrero 2017. [En línea]. Available: https://blogthinkbig.com/diferencias-entre-machine-learning-y-deep-learning. [Último acceso: 27 Febrero 2020].

[36] E. paths, «Telefónica,» Telefónica Cyber Security Unit, Junio 2019. [En línea]. Available: https://www.elevenpaths.com/wp-content/uploads/2019/06/whitepaper-la-inteligencia-artificial-aplicabilidad-de-gans-autoencoders-ciberseguridad.pdf.

[37] J. Álvarez, «analiticaweb,» 22 Diciembre 2016. [En línea]. Available: https://www.analiticaweb.es/machine-learning-y-support-vector-machines-porque-el-tiempo-es-dinero-2/.

[38] P. Prathvikumar, «Towards data Science,» 7 Octubre 2019. [En línea]. Available: https://towardsdatascience.com/intro-to-bayesian-statistics-5056b43d248d. [Último acceso: 14 Febrero 2020].

[39] A. Venkataraman, «Floydhub,» 8 Noviembre 2019. [En línea].

[40] W. Koehrsen, «TowardsDataScience,» 30 Agosto 2018. [En línea]. Available: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76. [Último acceso: 15 Enero 2020].

[41] W. Koehrsen, «Medium,» Diciembre 27 2017. [En línea]. Available: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d. [Último acceso: 24 Febrero 2020].

[42] J. J. A. Jiménez, «Combinación del aprendizaje multitarea y del algoritmo EM en problemas de clasificación con datos incompletos,» ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN UNIVERSIDAD POLITECNICA DE CARTAGENA, 2006.

[43] N. Buduma, «O'Reilly,» [En línea]. Available: https://www.oreilly.com/library/view/fundamentals-of-deep/9781491925607/ch01.html. [Último acceso: 30 Enero 2020].

[44] C. Nicholson, «pathmind,» [En línea]. Available: https://pathmind.com/wiki/lstm. [Último acceso: 02 Febrero 2020].

[45] C. Olah, 27 Agosto 2015. [En línea]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Último acceso: 1 Febrero 2020].

[46] D. M. J. Garbade, «Millenium,» 15 Octubre 2018. [En línea]. Available: https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32. [Último acceso: 5 Marzo 2020].

[47] J. Abad, 8 Enero 2019. [En línea]. Available: https://www.dail.es/aplicaciones-del-procesamiento-del-lenguaje-natural/. [Último acceso: 5 Marzo 2020].

[48] [En línea]. Available: https://www.politifact.com/article/2018/feb/12/principles-truth-o-meter-politifacts-methodology-i/.

[49] [En línea]. Available: https://maldita.es/.

[50] [En línea]. Available: https://www.factcheck.org/our-process/.

[51] A. D. Holan, «Polifact,» 12 Febrero 2018. [En línea]. Available: https://www.politifact.com/truth-o-meter/article/2018/feb/12/principles-truth-o-meter-politifacts-methodology-i/.

[52] «Poynter,» [En línea]. Available: https://www.poynter.org/ifcn/.

[53] [En línea]. Available: http://www.newtral.es/.

[54] A. A. G. Xichen Zhang, «An overview of online fake news: Characterization, detection and discussion,» Elsevier, 2019.

[55] J. M. Heras. [En línea]. Available: https://iartificial.net/precision-recall-f1-accuracy-en-clasificacion/. [Último acceso: 29 Febrero 2020].

[56] «Python,» [En línea]. Available: python.org.

[57] [En línea]. Available: https://www.python.org/.

[58] E. d. e. d. l. u. d. Valencia, «https://www.universidadviu.es,» 1 Abril 2019. [En línea]. Available: https://www.universidadviu.es/machine-learning-python-el-lenguaje-de-los-negocios-del-futuro/.

[59] Phobos, 31 Octubre 2015. [En línea]. Available: http://rstadistica.blogspot.com/2015/10/historiaR.html. [Último acceso: 27 Febrero 2020].

[60] BBVAOPEN4U, «BBVA,» 9 Agosto 2016. [En línea]. Available: https://bbvaopen4u.com/es/actualidad/ventajas-e-inconvenientes-de-python-y-r-para-la-ciencia-de-datos.

[61] «Pycharm,» [En línea]. Available: https://www.jetbrains.com/es-es/pycharm/.

[62] «Anaconda,» [En línea]. Available: https://www.anaconda.com/.

[63] [En línea]. Available: https://www.anaconda.com/why-anaconda/.

[64] NSS, «Analytics Vidya,» 4 Junio 2017. [En línea]. Available: https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/. [Último acceso: 19 Febrero 2020].

[65] [En línea]. Available: https://github.com/KaiDMML/FakeNewsNet.

[66] [En línea]. Available: https://github.com/jpposadas/FakeNewsCorpusSpanish.

[67] «Github,» [En línea]. Available: www.github.com.

[68] S. Kadam, «GeeksforGeeks,» [En línea]. Available: https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/. [Último acceso: 19 Febrero 2020].

[69] A. H. C. H. H. Marwa Naili, «Comparative study of word embedding methods in topic segmentation,» de International Conference on Knowledge Based and Intelligent Information and Engineering, Francia, 2017.

[70] H. Ranvir, «Medium,» 2019 Enero 20. [En línea]. Available: https://medium.com/@hemantranvir/spam-detection-using-rnn-simplernn-lstm-with-step-by-step-explanation-530367608071. [Último acceso: 25 Febrero 2020].

[71] [En línea]. Available: https://github.com/Tariq60/LIAR-PLUS/tree/master/dataset.

[72] «MEX-A3T,» [En línea]. Available: https://sites.google.com/view/mex-a3t/data-and-evaluation. [Último acceso: 12 Febrero 2020].

[73] «StackExchange,» 24 Julio 2016. [En línea]. Available: https://datascience.stackexchange.com/questions/12964/what-is-the-meaning-of-the-number-of-units-in-the-lstm-cell. [Último acceso: 23 Febrero 2020].

[74] «TecnologiaFacil,» [En línea]. Available: https://tecnologia-facil.com/que-es/que-es-compilar/. [Último acceso: 28 Febrero 2020].

[75] «Tecnología-Fácil,» [En línea]. Available: https://tecnologia-facil.com/que-es/que-es-compilar/.

[76] P. Kiely, «Floydhub,» 31 Octubre 2918. [En línea]. Available: https://blog.floydhub.com/a-pirates-guide-to-accuracy-precision-recall-and-other-scores/. [Último acceso: 29 Febrero 2020].

[77] J. Rodríguez y V. Fernández, Cómo redactar el estado del arte de un trabajo, Editorial Genios, 2010.

[78] 3 Diciembre 2015. [En línea]. Available: http://www.slothslab.com/python/2015/12/03/clasificador-bayesiano-ingenuo-python.html.

[79] M. Hargrave, «Investopedia,» 30 Abril 2019. [En línea]. Available: https://www.investopedia.com/terms/d/deep-learning.asp.

[80] T. Rodríguez, «xataka,» 26 Septiembre 2018. [En línea]. Available: https://www.xataka.com/robotica-e-ia/machine-learning-y-deep-learning-como-entender-las-claves-del-presente-y-futuro-de-la-inteligencia-artificial.

[81] [En línea]. Available: https://www.kaggle.com/saketchaturvedi/fake-news-six-way-classification-part-2/data.

[82] «DeepAI,» [En línea]. Available: https://deepai.org/machine-learning-glossary-and-terms/loss-function. [Último acceso: Febrero 28 2020].

[83] [En línea]. Available: https://www.python.org/.

ANEXO I: DATOS DE ENTRENAMIENTO Y TEST (REDUCIDO)

Aquí se incluye una muestra del dataset utilizado para entrenar los cuatro modelos iniciales:

id	category	topic	headline	
1	Fake	Education	RAE INCLUIRÁ LA PALABRA "LADY" EN EL DICCIONARIO DEL IDIOMA ESPAÑOL COMO DEFINICIÓN DE "MUJER PROBLEMÁTICA"	PALABRA "LADY" EN EL DICCIONARIO DEL IDIOMA ESPAÑOL COMO DEFINICIÓN DE "MUJER PROBLEMÁTICA" España.- El presidente de la Real Academia Española (RAE), Darío Villanueva, informó en conferencia de prensa
2	Fake	Education	La palabra "haiga", aceptada por la RAE	aceptada por la RAE La Real Academia de la Lengua (RAE), ha aceptado el uso de
3	Fake	Education	YORDI ROSADO ESCRIBIRÁ Y DISEÑARÁ LOS NUEVOS LIBROS DE TEXTO DE LA SEP PARA HACERLOS MÁS ATRACTIVOS	ESCRIBIRÁ Y DISEÑARÁ LOS NUEVOS LIBROS DE TEXTO DE LA SEP PARA HACERLOS MÁS ATRACTIVOS México.- El director de la Secretaría de Educación Pública, Aurelio Nuño, informó que el dramaturgo y conductor Yordi Rosado
4	True	Education	UNAM capacitará a maestros para aprobar prueba Pisa	maestros para aprobar prueba Pisa La máxima casa de estudios y la SEP firmaron cinco
5	Fake	Education	pretenden aprobar libros escolares con contenido sesgado sobre el conflicto armado interno	aprobar libros escolares con contenido sesgado sobre el conflicto armado interno Sigue la campaña negacionista del fujimorismo. Otra vez

Tabla Anexo1-1 Datos 4 modelos principales

ANEXO II: DATASET DE LSTM

Label	Statement
Fake	ESPAnhOL COMO DEFINICIoN DE MUJER PROBLEMaTICAEspanha.- El presidente de la Real Academia Espanhola (RAE) Dario Villanueva informo en conferencia de prensa que a partir del proximo mes se incluira el termino Lady como una nueva palabra en el diccionario del idioma
Fake	(RAE) ha aceptado el uso de HAIGA para su utilizacion en las tres personas del singular del presente del subjuntivo del verbo hacer aunque asegura que la forma mas recomendable en la lengua culta para este tiempo sigue siendo haya.Asi lo han confirmado fuentes de la RAE que explican que
Fake	DE LA SEP PARA HACERLOS MaS ATRACTIVOSMexico.- El director de la Secretaria de Educacion Publica Aurelio Nunho informo que el dramaturgo y conductor Yordi Rosado sera el encargado de redactar los nuevos libros de texto que se reparten en todas las escuelas del pais y que
True	estudios y la SEP firmaron cinco convenios para que las facultades de Ciencias y Quimica asi como el Instituto de Matematicas de ensenhen a los profesores estrategias para impartir estas disciplinas a los alumnos de preescolar primaria secundariaLa Universidad Nacional Autonoma de
Fake	el conflicto armado internoSigue la campanha negacionista del fujimorismo. Otra vez desde el Congreso la bancada Fuerza Popular (FP) presiono para que se revisen los textos escolares y sus contenidos sobre la epoca del terrorismo segun denuncia la Coordinadora Nacional de
True	universitarios de seis universidades europeas participan este cuatrimestre en la ultima fase de pruebas de un programa informatico que permitira certificar la identidad y autoria de los estudiantes cuando realizan actividades 'online' como examenes trabajos u otras pruebas.El proyecto
Fake	SOCIOLOGiA Y FILOSOFiA; EXPULSARaN A LOS QUE DEN POSITIVOMexico.- El rector de la Universidad Autonoma de Mexico Enrique Graue Wiechers revelo en conferencia de prensa que a partir del proximo ciclo escolar estudiantes de las carreras de Sociologia y Filosofia

Tabla Anexo2-1 Datos LSTM

ANEXO III: MODELO LSTM'

```
    ###El primer paso es leer el archivo .csv

import pandas as pd
import numpy as np

fake_news = pd.read_csv('textoespañol.csv', encoding='unicode_escape', delimiter=',',
engine='python')
fake_news.isnull().values.any()
print(fake_news.shape)
print(fake_news.head())

###Graficamos cuantos son verdaderos y cuantos son falsos
import seaborn as sns
import matplotlib.pyplot as plt

#vf = sns.countplot(x='Label', data=fake_news)
#plt.show()

### Limpieza de las palabras y caracteres que nos resultan innecesarias as'i como poner
todo en minuscula
import re
from unicodedata import normalize
def preprocesado(sen):
    #quitamos los caracteres que apareces solos
    sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sen)
    #Quitamos los m'ultiples espacios en blanco
    sentence =  re.sub(r'\s+', ' ', sentence)
    #Quitamos url
    sentence = re.sub(r"http[\s:/\.]+",' ',sen)
    #Quitamos acentos
    sentence = re.sub(r"([^n\u0300-\u036f]|n(?!\u0303(?![\u0300-\u036f])))[\u0300-
\u036f]+", r"\1",
        normalize( "NFD", sentence), 0, re.I
    )
    sentence = normalize('NFC', sentence)
    #Ponemos todo en minuscula
    sentence = sentence.lower()
    return sentence

###Guardamos los titulares en la variable X
X = []
sentences = list(fake_news["Statement"])
for sen in sentences:
    X.append(preprocesado(sen))
print(X[4])

###Guardamos las label en variable Y pero primero las pasamos a 0 y 1(verdadero)
Y = fake_news['Label']
Y = np.array(list(map(lambda x: 1 if x== 'True' else 0, Y)))
#print(Y)

#######Tokenizamos
##Primero es necesario definir una serie de parametros
max_words = 100000
max_length_sentence = 10000
EMBEDDING_DIM = 300
```

```
GLOVE_DIR = 'SBW-vectors-300-min5.txt'+str(EMBEDDING_DIM)

from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(X)
sequences = tokenizer.texts_to_sequences(X)
word_index = tokenizer.word_index
#print('Vocabulary size:', len(word_index))

###Padding para hacerlo uniforme
from keras.preprocessing.sequence import pad_sequences

data = pad_sequences(sequences, padding= 'post',
                     maxlen=max_length_sentence)
print('Shape of data tensor', data.shape)
print('Shape of label tensor', Y.shape)

###Split en traing and test
from sklearn.model_selection import train_test_split

print(data, '\n',Y)

x_train,x_test, y_train, y_test = train_test_split(data,Y, test_size=0.2,
random_state=0)


###Word embedding
embedding_dict = {}
with open("spanishvector.txt", 'r', encoding="utf_8") as f:
    for line in f:
        values = line.split()
        word = values[0]
        vector = np.asarray(values[1:], "float32")
        embedding_dict[word] = vector

#Graficamos los diferentes vectores
from sklearn.manifold import TSNE

tsne = TSNE(n_components= 2, random_state=0)

words = list(embedding_dict.keys())
vectors = [embedding_dict[word] for word in words]

Y = tsne.fit_transform(vectors[1:3000])

plt.scatter(Y[:,0], Y[:,1])

for label, x, y in zip(words, Y[:, 0], Y[:, 1]):
    plt.annotate(label, xy=(x, y), xytext=(0, 0), textcoords="offset points")
plt.show()

###Representamos los datos en la matriz de embedding

embedding_index = {}
fileembedding = open('spanishvector.txt', encoding='utf_8')
for line in fileembedding:
    values = line.split()
    word = values[0]
    embedding_index[word] = np.asarray(values[1:], dtype='float32')
fileembedding.close()
```

```
embedding_matriz = np.random.random((len(word_index)+1,
                                     EMBEDDING_DIM))
for word, i in word_index.items():
    embedding_vector = embedding_index.get(word)
    if embedding_vector is not None:
        embedding_matriz[i] = embedding_vector


###Primer modelo LSTM
from keras.models import Sequential
from keras.layers.recurrent import LSTM
from keras.layers.core import Dense, Dropout
from keras.layers import Input, GlobalMaxPool1D
from keras.layers.embeddings import Embedding



model = Sequential()
#model.add(Input(shape=(max_length_sentence,), dtype='int32'))
model.add(Embedding(len(word_index) + 1,
 EMBEDDING_DIM,
 weights = [embedding_matriz],
 input_length = max_length_sentence,
 trainable=False,
 name = 'embeddings'))
model.add(LSTM(60, return_sequences=True,name='lstm_layer'))
model.add(GlobalMaxPool1D())
model.add(Dropout(0.1))
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy',
              metrics=['accuracy'])
print(model.summary())
entrenando = model.fit(x_train,y_train, epochs=10, batch_size=50,
                       validation_data=(x_test, y_test))

resultado = model.evaluate(x_test, y_test, verbose=2)
print('Recall: ', resultado[0])
print('Precision de LSTM', resultado[1])

#Es el momento de intentar una prediccion en vacio

deteccion = ['Obama ha decidido lanzar una bomba nuclear esta semana']
deteccion_limpia = deteccion

deteccion_tokenizada = Tokenizer(num_words=max_words)
deteccion_tokenizada.fit_on_texts(deteccion_limpia)
deteccion_sequences = tokenizer.texts_to_sequences(deteccion_limpia)
word_index2 = deteccion_tokenizada.word_index

deteccion_padded = pad_sequences(deteccion_sequences, padding = 'post', maxlen =
max_length_sentence)

prediction = model.predict(deteccion_padded)
print(prediction)
import matplotlib.pyplot as plt
import scikitplot.plotters as skplt
```

```
top_words = 5000
epoch_num = 15
batch_size = 64

def plot_cmat(yte, ypred):
    '''Plotting confusion matrix'''
    skplt.plot_confusion_matrix(yte, ypred)
    plt.show()
# Draw the confusion matrix
y_pred = model.predict_classes(x_test)
plot_cmat(y_test, y_pred)

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy: %f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, y_pred)
print('Precision: %f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, y_pred)
print('Recall: %f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, y_pred)
print('F1 score: %f' % f1)
```

Tabla Anexo 3-1 Modelo LSTM'