Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA AGRONÓMICA Y BIOCIENCIAS

NEKAZARITZAKO INGENIARITZAKO ETA BIOZIENTZIETAKO GOI MAILAKO ESKOLA TEKNIKOA

TRABAJO DE FIN DE GRADO**:**

COMPARATIVA DE TÉCNICAS DE EXTRACCIÓN DE CARACTERÍSTICAS DE TEXTO PARA LA DETECCIÓN DE *FAKE NEWS*

Presentado por

Alejandro Amatriain García

Dirigido por

Mikel Sesma Sara

GRADO EN CIENCIA DE DATOS

Septiembre de 2022

RESUMEN

En la actualidad, la divulgación de noticias falsas es una estrategia de manipulación a la sociedad, que repercute en la visión y opinión que esta se forja respecto a diversos temas.

En el ámbito del procesamiento del lenguaje natural el proceso de distinción entre una noticia real y una falsa comenzaría por definir una estructura de cómo se caracteriza el contenido de un texto, para después, conforme a esa estructura, encontrar los patrones y/o tendencias que se dan en un tipo de noticia falsa y una real (tendencias que serán detectadas a partir de aplicar dicha estructura a un conjunto de fake news y a otro conjunto de noticias verdaderas que disponemos, conjuntos que llamaremos de entrenamiento o train).

A estas estructuras se les llama Técnicas de extracción de características. Forman parte del proceso de aprendizaje automático. El trabajo consistirá en aplicar y comprobar la eficiencia de varias de ellas.

En el presente trabajo abordaremos cuatro técnicas: Bag of Words TF-IDF, GloVe, Word2vec y FastText.

La primera (TF-IDF) se trata de una técnica supervisada ya que se comienza por definir un vocabulario. Su forma de caracterizar un texto es dar pesos a los términos presentes en el texto que forman parte del vocabulario.
Las otras tres pertenecen a la familia de Representación vectorial de palabras. Grosso modo, consisten en aprender un vector numérico para cada término del vocabulario para después agregar todos los vectores de los términos presentes. Dichos métodos aprenden por sí mismos el vector de características de cada palabra basándose en la idea de que los vectores de características de dos términos que aparecen juntos con mucha frecuencia, deben ser parecidos. Con estos métodos, el programador debe especificar únicamente el número de componentes del vector de características. Tras el proceso de entrenamiento, desconoceremos qué representa cada componente del vector, al contrario que con Bag of Words.

PALABRAS CLAVE

Fake new, Aprendizaje automático, Extracción de características, Clasificación, NLP

ABSTRACT

Nowdays, the dissemination of false news is a manipulation strategy for society, which damages the vision and opinion that people have about some issues.

On the field of natural language processing, the process of distinguishing between real and false news begins by defining a structure of how the content of a text is characterized, and then, according to that structure, finding patterns or tendencies that relate to fake news but not real news (trends will be detected by applying said structure to a set of fake news and another set of true news, sets that we will call “train”).

These structures are called feature extraction techniques. They are part of the machine learning process. This work will aims at applying and checking the efficiency of several of them.

In this work we will apply four techniques: Bag of Words, GloVe, Word2vec and FastText.

TF-IDF is a supervised technique since it begins by defining a vocabulary. Its way of characterizing a text is to give weights to the terms present in the text that are part of the vocabulary.
The other three belong to the Vector Representation word family. They consist of learning a numerical vector for each vocabulary term and then aggregating all the vectors of the present terms. These methods learn by themselves the feature vector of each word based on the idea that the feature vectors of two terms that appear together very frequently must be similar. The programmer must specify only the number of features of the feature vector. After the training process, unlike Bag of Wordswe will not know what each feature of the vector represents.

KEYWORDS

Fake new, Machine learning, Feature Extraction, Classification, NLP

ÍNDICE

1. INTRODUCCIÓN Y OBJETIVOS

Las nuevas tecnologías permiten que se potencie la distribución de las noticias falsas y que éstas, como afirma el diario La Tribuna (2018), se posesionen del espacio de las redes sociales, hasta el punto de "dominar la conversación con alto rédito para obtener el propósito que se ha dejado quien la difunde". Su poder es tan grande que, según un estudio realizado por la Asociación de Internautas (2018), el 70% de los españoles no sabe distinguir entre una noticia verdadera y un rumor o un bulo, y "ello es debido a que no existe la fórmula exacta para diferenciar entre la información cierta y la que no lo es" [1]

El conocimiento de la realidad es lo que nos permite a los receptores formarnos opinión sobre el mundo que nos rodea, sin embargo, el flujo permanente de información y las noticias falsas erosionan esa credibilidad generando desinformación, entendida ésta como "cualquier contenido informativo falso que haya sido creado y difundido de forma deliberada" (Wardle, 2018) [1]

La consultora Gartner sostiene en su último informe de "Predicciones tecnológicas para el 2018" que en el 2022 el público occidental consumirá más noticias falsas que verdaderas [1].
La necesidad de herramientas automáticas para la distinción entre una noticia real y una falsa es urgente.

El Aprendizaje Automático es un método científico que nos permite usar dispositivos con capacidad computacional para que aprendan a extraer los patrones y relaciones existentes en los datos por sí solos. El conocimiento de éstos patrones se podrá emplear después para predecir comportamientos, clasificar o tomar decisiones.

La problemática de la identificación de las *Fake News* unida a la herramienta que es el Aprendizaje Automático, da lugar al concepto de *Procesamiento del lenguaje natural* (NLP, Natural language processing). Un lenguaje natural es aquel que ha evolucionado a lo largo del tiempo con fines de comunicación humana. Estos lenguajes continúan su evolución sin considerar la gramática, es decir, cualquier regla se desarrolla después de sucedido el hecho. En contraste, los lenguajes formales (como el lenguaje que se emplea para plasmar razonamientos matemáticos, o un lenguaje de programación como Java, por ejemplo) están definidos por reglas preestablecidas, y por tanto, se rigen con todo rigor a ellas.[2]

El NLP consiste en el proceso que comprende la utilización de un lenguaje natural para comunicarnos con la computadora, debiendo ésta entender las oraciones que le son proporcionadas. El uso de estos lenguajes naturales facilita el desarrollo de programas que realicen tareas relacionadas con el lenguaje, o bien, desarrollar modelos que ayuden a comprender mecanismos humanos relacionados con el lenguaje. [2]

La arquitectura de un sistema NLP es la siguiente: [2]

- Nivel fonológico: Trata de cómo las palabras se relacionan con los sonidos que representan.

- Nivel morfológico: Trata de cómo las palabras se construyen a partir de unas unidades de significado más pequeñas llamadas morfemas.

- Nivel sintáctico: Trata de cómo las palabras pueden unirse para formar oraciones, fijando el papel estructural que cada palabra juega en la oración.

- Nivel semántico: Tarta del significado individual de las palabras, así de cómo los significados se unen para dar un significado global a la oración. Se trata del significado independiente del contexto.

- Nivel pragmático: Trata de cómo las oraciones se usan en distintas situaciones, así de cómo el significado de una oración se ve afectado por el significado de las oraciones inmediatamente anteriores.

El objetivo general de este trabajo es diseñar y entrenar modelos de clasificación de fake news partiendo de distintas técnicas de extracción de características.

Los objetivos específicos serán:

- Estudiar el método de extracción de características TF-IDF.
- Estudiar el método de extracción de características GloVe.
- Estudiar el método de extracción de características Word2vec.
- Estudiar el método de extracción de características FastText.
- Experimentar con distintas formas de agregar la información que aparece en un mismo texto (instancia).

2. PRELIMINARES

2.1 Aprendizaje automático

El aprendizaje automático aplica de manera sistemática algoritmos para lograr sintetizar relaciones subyacentes automáticamente dentro de un conjunto de datos que llamamos de entrenamiento, para ser considerados en un futuro para la ejecución de predicciones y en una mejora en la toma de decisiones. Según la naturaleza de dichos algoritmos clasificamos el aprendizaje automático en dos tipos [3]:

El aprendizaje supervisado es aquel en el que el entrenamiento se realiza a partir de una colección de experiencias y resultados correctos en base a los cuales el algoritmo de clasificación predice nuevos resultados.

El aprendizaje no supervisado es aquel en el cual se dispone de una gran colección de experiencias y atributos pero no se tienen datos de los resultados correctos. Este tipo de aprendizaje se basa en encontrar similitudes o patrones que permitan realizar agrupamientos.

2.2 Clasificación

El proceso de clasificación corresponde a usar la información capturada por el algoritmo (o clasificador) en los datos de entrenamiento para predecir resultados de nuevas instancias (experiencias). Se clasifica la experiencia para obtener su resultado correspondiente [3].

La calidad de la clasificación debe cuantificarse por medio de otro conjunto de datos de la misma naturaleza que los de entrenamiento que llamamos conjunto de test. Se clasifican las experiencias de test, de las que disponemos de antemano el resultado correcto. Hay diferentes formas de cuantificación. En el presente trabajo utilizaremos tres [3]:

- Accuracy: Se trata del porcentaje de experiencias del conjunto de test que se han clasificado correctamente.

- Precisión: Es el porcentaje de experiencias clasificadas como clase positiva (la clase positiva es el uno de los resultados posibles de las experiencias que abordamos en cada problema específico) que realmente son clase positiva.

- Recall: Corresponde al porcentaje de experiencias de la clase positiva que han sido detectadas como tal.

2.3 Métodos de clasificación

2.3.1 Regresión logística [3]

La regresión lineal es empleada cuando se tiene una variable dependiente de tipo continuo. No obstante, a menudo, esta variable es de tipo categórico.
Se tienen dos categorías (1 y 0) y una serie de variables predictoras (supongamos x_1 y x_2). El modelo busca obtener la probabilidad de que una instancia pertenezca a la clase 1. Una primera aproximación podría ser la simple combinación lineal de predictores:

$$\text{P(X)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

No obstante, no se puede asegurar que los valores que tome la fórmula anterior estén entre 0 y 1. Por ello, en su lugar se utiliza la función logística:

$$\text{P(X)} = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}$$

La estimación de los coeficientes puede hacerse por el método de máxima verosimilitud. Dicho método consiste en encontrar los valores de los parámetros que, de acuerdo con el modelo logístico, maximizan la probabilidad de la muestra observada.

Se busca maximizar la siguiente función (función de verosimilitud):

$$\text{l}(\beta_0, \beta_1, \beta_2) = \prod_{i\ ;\ y_i=1} P(x_i) \prod_{i\ ;\ y_i=0} (1 - P(x_i))$$

No obstante, esta expresión presenta un problema. Ante la presencia de algún factor cercano a cero, el resultado del productorio también rondará entorno a cero. Es decir, la incorrecta clasificación de una única instancia 'estropea' en sobremanera el valor de la verosimilitud de toda la muestra. Aplicamos la función logaritmo (pues al ser monótona, los valores que la maximizan son los mismos que maximizan la función original) a ambos lados para solucionar el problema:

$$\text{L}(\beta_0, \beta_1, \beta_2) = \sum_{i\ ;\ y_i=1} \ln(P(x_i)) + \sum_{i\ ;\ y_i=0} \ln(1 - P(x_i))$$

A esta función resultante de aplicar el logaritmo se le llama función log-lokelihood y es muy utilizada en ajuste de modelos.

Desarrollando:

$$\text{L}(\beta_0, \beta_1, \beta_2) = \sum_{i\ ;\ y_i=1} \ln\left(\frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}\right) + \sum_{i\ ;\ y_i=0} \ln\left(1 - \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}\right)$$

Para maximizar esta última función puede usarse el método de descenso del gradiente. Requiere calcular las derivadas parciales de L respecto de cada parámetro a estimar, y sumar el gradiente al valor actual de los parámetros. También pueden emplearse algoritmos heurísticos y genéticos.

Una vez se han estimado los coeficientes, ya se pueden realizar predicciones. Dada una instancia X, se calcula P(X). Éste valor representa la probabilidad de que la instancia X pertenezca a la categoría '1'.

2.3.2 Support Vector Machines [3]

Dado un conjunto de puntos, subconjunto de un espacio mayor, en el que cada uno de ellos pertenece a una de dos posibles categorías, este algoritmo pretende predecir a qué categoría corresponde un nuevo punto. El algoritmo SVM pretende encontrar un hiperplano óptimo (dado que existen infinitos) que separe ambas clases.
Se considera hiperplano óptimo aquel que guarda un margen lo más grande posible entre la instancia más cercana de cada clase y el propio hiperplano. Ésta es la consideración clave que se tiene en cuenta a la hora de diseñar la función de coste para aprender los coeficientes del hiperplano.
En la Ilustración 1 se muestra un ejemplo de dos hiperplanos de separación para los mismos datos.

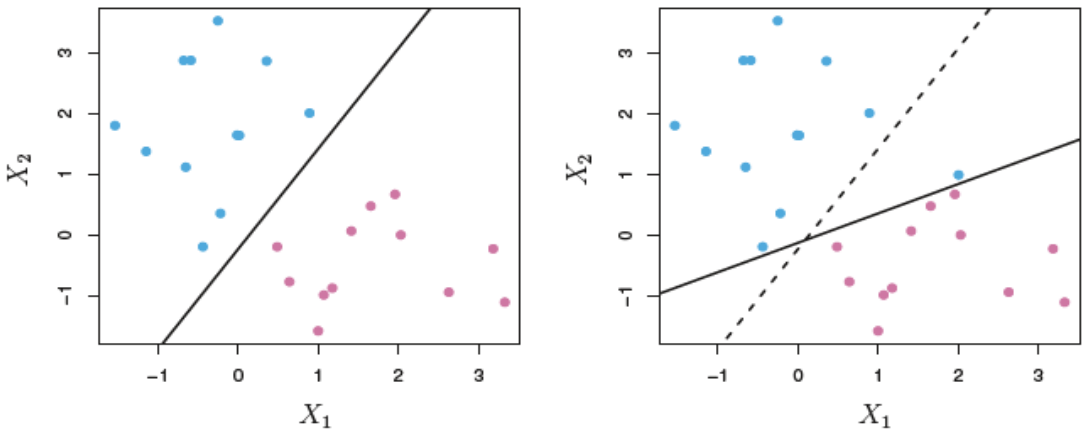

Ilustración 1: Hiperplanos SVM de separación

En el ejemplo plasmado en la ilustración 1, se observa que el hiperplano óptimo (de acuerdo con lo expuesto en el párrafo anterior) es el de la izquierda. También se tiene que los datos son linealmente separables en su espacio original. No obstante, es común que los datos no sean linealmente separables y haya que proyectarlos a un espacio de dimensiónalidad superior para que sí lo sean.
La función kernel es la que mapea la información original al espacio de dimensión superior, donde después se busca el hiperplano óptimo. Existen múltiples funciones kernel, pero en el presente trabajo hemos usado el kernel Gaussiano: Dado un punto x, calculamos su similitud con el resto de puntos l_i usando la siguiente función:

$$f_1 = similitud(x, l_1) = e^{\frac{-||x-l_1||^2}{2\sigma^2}}$$

Siendo σ un hiper-parámetro ajustable. Cuanto más grande, las similitudes disminuyen mas lentamente que con valores bajos de σ.
Entonces, para cada punto se tiene un vector con todas las similitudes al resto de puntos. Éste es el vector de características de cada instancia. Lo que el algoritmo debe aprender son los coeficientes que acompañan a cada característica:

- Predecir ‘1’ cuando $\theta_0 + \theta_1 f_1 + \cdots + \theta_n f_n \geq 0$

- Predecir ‘0’ cuando $\theta_0 + \theta_1 f_1 + \cdots + \theta_n f_n < 0$

2.3.3 Naive Bayes [3]

En el caso de clasificadores discriminativos como es el caso de la regresión logística o las máquinas de soporte vectorial, se clasifica a partir de un único modelo con una frontera de decisión. En los clasificadores generativos, como es el caso de Naive-Bayes, lo que se modeliza es la distribución de los diferentes predictores para cada clase por separado. Es decir, se tienen tantos modelos como clases existen en el problema a abordar. Se asume que la distribución de cada predictor obedece a una distribución Normal.

Supongámos que se quiere clasificar una instancia en una de un total de K categorías. Supóngase también que se cuenta con una única variable preidctora. Se denota como π_k
la probabilidad a priori de que, escogiendo una observación al azar, pertenezca a la clase K. Sea $f_k(X) \equiv P(X = x\,|Y = k)$ el valor que toma la función de densidad para el predictor X en la categoría k. Usando el teorema de Bayes, la probabilidad de que una nueva instancia pertenezca a la clase k es:

$$\mathrm{P(Y = k}|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

Es fácil estimar el valor de π_k a partir de una muestra aleatoria de la población. Se calcula el cociente del número de instancias de la muestra que pertenecen a la clase ‘k’ entre el tamaño de la muestra.

Para estimar el valor de $f_k(x)$ se utiliza una muestra de instancias de la categoría ‘k’ lo suficientemente grande como para determinar la media y desviación típica para el predictor en la categoría en cuestión. Una vez estimada la media y desviación típica, ya se puede obtener el valor $f_k(x)$ de acuerdo con la expresión de la función de densidad para una distribución aleatoria de tipo Normal:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2\sigma_k^2}(x-\mu_k)^2}$$

Si se atiende a más de un predictor para realizar la clasificación, se asume que las distintas variables predictoras cumplen una condición de independencia condicionada. Se tiene que, por ejemplo, la probabilidad de que una nueva instancia pertenezca a la clase ‘k’ es:

$$\mathrm{P(Y = k}|\boldsymbol{X} = (\mathrm{x_1}, \mathrm{x_2}, \ldots, \mathrm{x_n})) = \frac{\pi_k f_{1k}(x_1)}{\sum_{l=1}^{K} \pi_l f_{1l}(x_1)} \cdot \frac{\pi_k f_{2k}(x_2)}{\sum_{l=1}^{K} \pi_l f_{2l}(x_2)} \cdot \ldots \cdot \frac{\pi_k f_{nk}(x_n)}{\sum_{l=1}^{K} \pi_l f_{nl}(x_n)}$$

A cada una de estas probabilidades se le llama probabilidad a posteriori.
Por tanto, para clasificar una nueva instancia se debe hacer previamente el cómputo de $\mathrm{P(Y = k}|\boldsymbol{X} = (\mathrm{x_1}, \mathrm{x_2}, \ldots, \mathrm{x_n}))$ para tantos valores de k como categorías tenga el problema. Finalmente, clasificar la instancia en aquella categoría que ha obtenido el mayor valor de probabilidad a posteriori.

2.3.4 Random Forest [3]

Un árbol de decisión es un clasificador que divide el espacio de todas las variables predictoras en un número de regiones. Cada una de éstas regiones tendrá una predicción asociada para cualquier instancia que pertenezca a dicha región.
En el caso de problemas de clasificación, la predicción para una región determinada es la moda de las categorías correspondientes a las instancias de entrenamiento que recaen en esa misma región. Un ejemplo de ésta partición lo encontramos en la Ilustración 2.

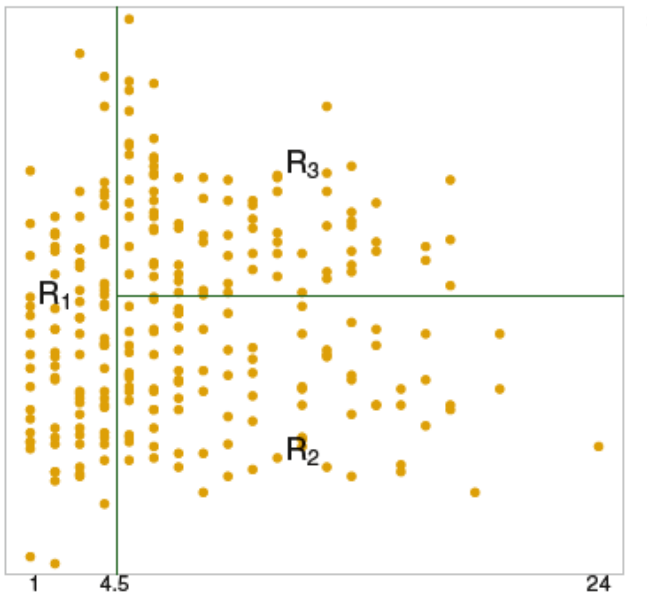

Ilustración 2: División del espacio de dos predictores en tres regiones

La partición final es producto de un procedimiento recursivo de particiones binarias. Ésta partición final la constituyen las hojas (nodos finales) del árbol de decisión compuesto por nodos internos. En cada nodo interno se evalúa una de las variables predictoras.

Se comienza por lo que se denomina nodo raíz. Se debe elegir una variable (predictor) en base a la cual hacer la primera partición binaria. Dicha partición binaria consistirá en establecer un umbral. Las instancias de entrenamiento que tengan en la variable elegida para ése nodo un valor por debajo del umbral, iran a la partición de la izquierda. En caso contrario a la derecha. De ésta forma, las instancias de entrenamiento van recorriendo verticalmente de arriba hacia abajo el árbol hasta acabar en alguno de los nodos finales u hojas.

En la Ilustración 3 se tiene un ejemplo de árbol de decisión. En la Ilustración 4 se plasma la partición final resultante correspondiente al árbol de la ilustración 3.

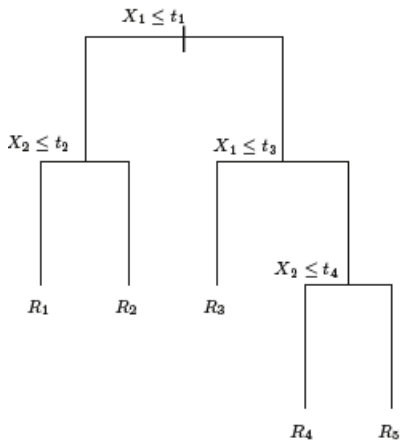

Ilustración 3: Árbol de decisión

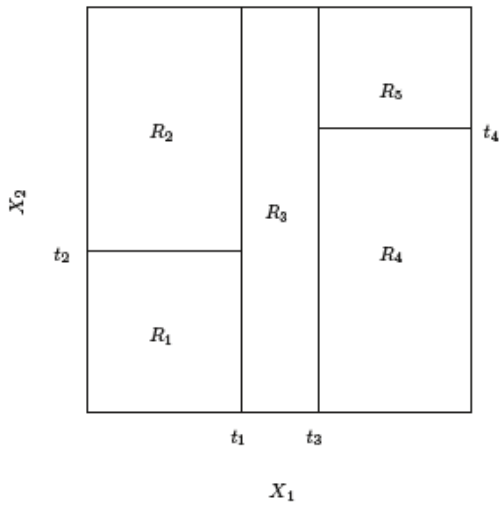

Ilustración 4: Partición del espacio de predictores asociada al árbol de decisión de la Ilustración 3

En cada nodo interno se debe elegir la variable que mejor permita "discriminar por clases" las instancias que han llegado hasta tal nodo. Para ello se necesita disponer de una función que mida el grado de "impureza" de un conjunto de instancias. Entendiendo que un conjunto de instancias alcanza su grado máximo de pureza cuando todas las instancias pertenecen a la misma categoría o clase. En cada nodo interno se busca que tras hacer la partición binaria, los dos conjuntos restantes sean lo más "puros" posible.

La función que mide la "impureza" del conjunto de instancias que han llegado hasta el nodo m es la función de entropía cruzada:

$$\mathrm{D} = -\sum_{k=1}^{K} p_{mk} \log p_{mk}$$

Siendo $p_{mk} \in [0,1]$ la proporción de instancias que han llegado al nodo m que pertenecen a la clase k.

Para evaluar el nivel de entropía o "impureza" de una partición binaria se debe tener en cuenta el número de instancias que van a cada partición y la entropía de cada una de ellas:

$$D = -p_{m,partIzq} \cdot \sum_{k=1}^{K} p_{partIzq,k} \log p_{partIzq,k} - p_{m,partDcha} \cdot \sum_{k=1}^{K} p_{partDcha,k} \log p_{partDcha,k}$$

Siendo $p_{m,partIzq}$ la proporción de instancias del nodo m que van a la partición de la izquierda, $p_{m,partDcha}$ la proporción de instancias que van a la partición de la derecha, $p_{partIzq,k}$ la proporción de instancias que pertenecen a la clase k de entre las que han ido a la partición de la izquierda y $p_{partDcha,k}$ la proporción de instancias que pertenecen a la clase k de entre las que han ido a la partición de la derecha.

Se busca una variable y un umbral para esa variable tal que la partición resultante tenga una entropía baja.

A continuación, explicaremos como se elige el mejor umbral para una variable dada:

Si la variable es categórica, las particiones deben ser los posibles valores que toma la variable. En el caso de variables numéricas se debe establecer un umbral. El procedimiento es el siguiente:

1. Se ordenan los valores en esa variable de todas las instancias que han llegado hasta el nodo en cuestión.
2. Se inspeccionan como posibles umbrales todos los valores para los que hay un cambio de categoría en la secuencia ordenada. Para cada uno de ellos, calcular la entropía de la hipotética partición según la última fórmula previamente expuesta.
3. Escoger como umbral aquel que ha dado un menor valor de entropía.

Se acaba de explicar cómo se elige el mejor umbral dada una variable y cómo medir la entropía de su partición resultante. Falta explicar cómo seleccionar las variables para hacer este estudio: Si se dispone de pocas variables puede hacerse una búsqueda exhaustiva y realizar el procedimiento explicado previamente con todas ellas. Si se dispone de muchas variables lo anteriormente mencionado es computacionalmente inviable y/o ineficiente. En ese caso se debe fijar de antemano un número de variables a examinar en cada nodo. Éstas serán escogidas al azar de entre todas las variables que hay. De entre las escogidas al azar se elegirá la que menor entropía presente conforme al procedimiento expuesto.

Se establece que un nodo no se va a volver a dividir y va a ser una hoja cuando se presente alguna de las siguientes situaciones:

- Todas las instancias que han llegado hasta el nodo pertenecen a la misma categoría.
- El número de instancias que han llegado hasta el nodo es menor a cierto valor prefijado de antemano.
- Se ha llegado al nivel máximo de profundidad del árbol establecido de antemano.

Un ensemble es un sistema con múltiples clasificadores. El sistema se compone de cierto número de clasificadores base y para cada uno de ellos se ha utilizado distintos datos para su entrenamiento. Por ello, la diversidad es un factor clave: cada clasificador base comete errores diferentes. Su combinación puede llevar a más aciertos. La cambinación puede hacerse mediante voto simple en el caso de clasificación. En el caso de regresión suele hacerse la media de los valores devueltos por cada clasificador base.
Random Forest es un ensemble en el cual los clasificadores base son árboles de decisión. Para entrenarlos, puede usarse un conjunto de datos distintos para cada uno. Esta técnica llevaría a cierto sobreaprendizaje de

cada árbol, resultando una varianza alta en el ensemble. Para evitar esta situación podría usarse boostrap: Si se tienen N instancias para entrenar T árboles, cada árbol debe entrenarse con N/T instancias, pero éstas instancias serán elegidas al azar de entre el total de instancias. Por lo tanto, una misma instancia puede haberse utilizado para entrenar varios de los clasificadores base.
A la hora de predecir la clase de una instancia, ésta debe ser clasificada por cada uno de los árboles que corresponden a los clasificadores base. La predicción final corresponderá a la moda de dichas clasificaciones.

2.3.5 Adaboost [3]

Adaboost se trata de otra forma de clasificación basada en ensemble. Los clasificadores base también son árboles de decisión, con la particularidad de que son de un único nivel de profundidad (también llamados Decision Stumps).
Al contrario que Random Forest, Adaboost aprende cada árbol de decisión utilizando todas las instancias de entrenamiento. Además, el entrenamiento de los clasificadores base se hace de forma secuencial, esto es cuando se ha terminado de entrenar el árbol previo. De ésta manera, los errores cometidos por clasificadores anteriores son tenidos en cuenta a la hora de entrenar nuevos árboles.

La idea clave es asignar pesos a las instancias de entrenamiento. El error de cada clasificador base corresponderá a la suma de los pesos de las instancias mal clasificadas por dicho clasificador.
Asimismo, también se asignarán pesos a los clasificadores base para hacer el voto ponderado final del ensemble a la hora de hacer predicciones. Dicho peso asignado a cada clasificador base, tendrá una relación de proporcionalidad inversa con el error del clasificador base.

De forma esquemática, el procedimiento es el siguiente:

Se dispone de un conjunto de instancias de entrenamiento $\{x_i, y_i\},\ i = 1, \dots, N\ , y_i \epsilon \{-1, +1\}$. Supóngase un número de clasificadores base T. Los pesos para cada clasificador base son $\alpha_t\ , t \epsilon \{1, \dots, T\}$. La salida (predicción) de cada clasificador base para la instancia x es $h_t(x)$.

Inicialmente se asignan pesos a las instancias para el entrenamiento del primer clasificador base: $D_1(i) = \frac{1}{N}$, $i = \{1, \dots, N\}$. Dichos pesos irán modificándose tras el entrenamiento de cada árbol base.

Para cada clasificador $t\ , t = 1, \dots, T$:

1. Entrenar el árbol de decisión de un único nivel usando los pesos actuales de las instancias. Al usar un único atributo para clasificar, debe elegirse el atributo que mejor separe las clases de acuerdo con el peso actual de cada instancia, dado que interesa acertar las instancias con gran peso.

   Por lo tanto, para cada atributo j, debe encontrarse el punto de corte θcon menor error cuyo valor se calcula mediante la siguiente fórmula:

$$\mathrm{J}(\theta, \mathrm{j}) = \frac{1}{N} \cdot \sum_{i=1}^{N} [D_t(i) \cdot I(h_t(x_i) \neq y_i)]$$

   Para encontrar el punto de corte θ óptimo se ordenan los valores de el atributo j de las instancias de entrenamiento y se establecen como umbrales candidatos aquellos en los que hay un cambio

de clase en la secuencia ordenada. Para cada umbral candidato, calcular la suma de los pesos de las instancias mal clasificadas. El punto de corte óptimo θ para el atributo j es aquel cuya dicha suma es menor.

Finalmente, escoger el atributo cuya clasificación obtiene el menor error.

2. Calcular el error del clasificador base como la suma de los pesos de las instancias mal clasificadas:

$$\epsilon_t = \sum_{i,\ y_i \neq h_t(x_i)} D_t(i)$$

3. Si ϵ_t > 0.5 el clasificador es peor que uno aleatorio, por lo que se deshecha y se pasa a entrenar el siguiente.

4. Asignar peso al clasificador: $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$

5. Actualizar los pesos de las instancias: $D_{t+1}(i) = D_t(i) \cdot\ e^{-\alpha_t h_t(x_i) y_i}\ ,\ i = 1, \dots, N.$

6. A continuación, normalizar $D_{t+1}(i),\ i = 1, \dots, N$ para que estén en una distribución adecuada.

A la hora de hacer predicciones de una nueva instancia x, la clasificación será la correspondiente a aplicar la siguiente expresión:

$$sign(\sum_{t=1}^{T} \alpha_t h_t(x)\ )$$

2.4 NLP. Fase de preprocesamiento general previa a la extracción de características [4]

El NLP consiste en el proceso que comprende la utilización de un lenguaje natural para comunicarnos con la computadora, debiendo ésta entender las oraciones que le son proporcionadas. En el presente trabajo nos centraremos concretamente en la minería de texto la cual consiste en la extracción de información automática a partir de texto no estructurado. La información extraída sí que debe de ser estructurada, de tal forma que posibilite analizarla para los fines específicos que dicte la aplicación de minería de texto que se desea construir. Las aplicaciones son diversas: clasificación de documentos, resúmenes, sistemas de recomendación…
Con la información extraída y estructurada se realiza aprendizaje automático, por lo que se necesita un conjunto de entrenamiento: Cada instancia es un documento y cada documento está caracterizado por las palabras que contiene, más bien por las palabras relevantes que contiene.

Se llama corpus al conjunto de documentos de texto de los que se dispone y que se usarán para realizar el entrenamiento de cierto modelo. Por ejemplo, un conjunto de textos correspondientes a e-mails etiquetados como spam/no-spam. Antes de comenzar dicho entrenamiento se han de representar todos los textos en forma vectorial. Para ello es necesario una fase previa de preprocesamiento. La idea de esta fase es obtener los diferentes términos que mejor representan a todos los documentos del corpus.

Antes de comenzar con la extracción de características de un texto se debe decidir qué términos del total de los que incluye el texto se han de tener en cuenta. Por ello se debe realizar una fase de preprocesamiento.
Los principales pasos en el preprocesamiento son la eliminación de los términos que no aportan información alguna, llamadas stop-words y la normalización de los términos que sí aportan información.

Eliminación de Stop Words:

Los términos más comunes correspondientes a esta categoría son artículos, preposiciones y pronombres. Por ejemplo “me”, “se”, “hasta”,“por”. El motivo por el que se desea eliminar estos términos es que aumentan de forma considerable la cantidad de palabras a tener en cuenta, sin guardar relación alguna con las palabras clave de las aplicaciones concretas de la minería de textos. Es más, al tenerlas en cuenta se diluye el contenido relevante del texto. Por ejemplo, si queremos discriminar entre textos de tipología política y de tipología médica interesan términos como “presidente”, “congreso”, “presupuestos” o “gastritis”, “tratamiento”, “cáncer”. El primer grupo de palabras sería más probable de encontrar en un texto o noticia política que médica. Con el segundo sería más razonable encontrarlos en un texto clínico. Sin embargo, palabras como “me”, “te” no resultan útiles para discriminar la naturaleza o tipología de un texto.
También se consideran stop-words los términos “raros” que aparecen una única o pocas veces en todo el corpus. Al aparecer en pocas ocasiones, no se tiene una muestra suficiente de contextos con los que el término pueda estar asociado. Por ese motivo también se decide eliminarlos.

Para llevar a cabo esta eliminación existen varios métodos:

- Eliminar los términos que se encuentran incluidos en listas elaboradas por terceros. En estas listas están incluidas palabras que aparecen con frecuencia de forma genérica en todo tipo de contextos.
  Eligiendo esta opción, no se haría un estudio personalizado de la frecuencia de aparición de cada término en el ámbito que interesa (el corpus del que se nutre cada aplicación particular de la minería de textos).

- Hacer un estudio estadístico de la frecuencia de aparición de todos los términos en el cuerpo de textos de la aplicación específica. Se deben establecer umbrales mínimo y máximo para que los términos que no lleguen al mínimo o sobrepasen el máximo sean eliminados.

- Método de la información mutua. Se trata de un método supervisado que consiste en etiquetar cada término en función de la clase a la que pertenece el texto en donde se encuentra dicho término. Conservar los términos que tienen más predilección por una clase. Eliminar los términos que se encuentran distribuidos de una forma más o menos homogénea en las clases del problema.

Normalización de los términos:

Se debe tener en cuenta que los términos que sí aportan información pueden presentarse de múltiples formas: mayúsculas/minúsculas o con tildes/sin tildes. Han de eliminarse las tildes y pasar todo el texto a minúsculas.

También han de tenerse en cuenta los morfemas que forman derivados de género, persona, número y modo verbal. Es decir, la idea es unificar una base gramatical con todos sus posibles derivados en un único término pues se considera que el contenido semántico es eminentemente el mismo. Existen diferentes métodos para realizar este procedimiento donde muchos son derivados de otros, pero el método principal es el Stemmer de Lovins:

Se elaboran listas de sufijos para un cierto idioma. Esta lista está ordenada, lo que significa que para normalizar un término se busca el primer sufijo de la lista que coincida con la parte final del término en cuestión. También se elabora una lista de "términos base", éstas son palabras normalizadas que forman el vocabulario al que deberán mapearse todos los términos del texto que se pretende normalizar. Cuando se ha eliminado el sufijo, queda una palabra que no tiene porqué ser uno de los "términos base". Se calcula la distancia (número de caracteres distintos) a cada palabra de la lista de "términos base". Se establece como término normalizado aquella que ha obtenido una distancia menor.

No obstante, si se dispone de la capacidad computacional suficiente, junto con una gran muestra de documentos que forman el corpus, se pueden considerar derivados de la misma base gramatical y semántica como términos distintos (por ejemplo, incluir en el vocabulario los términos "escrito" y "escritura" como términos diferentes). Esto último depende mucho del método de extracción de características que se vaya a usar, métodos que expondremos a continuación.

3. MÉTODOS DE EXTRACCIÓN DE CARACTERÍSTICAS DE TEXTO

Una vez se ha realizado la fase de preprocesamiento explicada previamente, ya se tiene un vocabulario definido. A continuación, se pretende encontrar el vector de características que definirá el contenido de los textos del corpus. Para ello se expondrán a continuación diferentes técnicas con las que se conseguirá obtener la representación vectorial de los textos del corpus.

3.1 Método Bag Of Words TF-IDF [5]

Bag Of Words TF-IDF (term frequency and inverse of document frequency) pretende representar un texto empleando un vector numérico con N componentes. Cada una de estas N componentes representa a cada uno de los N términos normalizados que se han considerado para definir el vocabulario durante la fase de preprocesamiento.
Se basa en la idea de que las características de un texto las definen la presencia o ausencia de los términos que forman el vocabulario, sin entrar en la semántica de los mismos (al contrario que los siguientes métodos de extracción de características que se explicarán en el presente trabajo, los cuales sí procuran hacer una descomposición semántica de los términos del vocabulario).

La cuestión es cómo asignar pesos a cada una de éstas componentes: La idea inicial de Bag Of Words es asignar pesos binarios (1:presencia, 0:ausencia) de tal forma que se le da la misma importancia a todos los términos del vocabulario. TF-IDF es una variante la cual consiste en valorar los términos que no son muy comunes en el corpus pero que tienen un nivel de frecuencia de aparición dentro de un mismo texto relativamente alto.

Cuando se está codificando la información con la que se pretende representar un texto : Para el término codificado en la componente t, su peso asignado será:

$$TF(t) \cdot \log\left(\frac{N}{df(t)}\right)$$

Siendo:

$TF(t)$ la proporción entre el total de términos considerados (tras eliminar stop words y normalizar) en el documento correspondientes al término t.

$df(t)$ el número de documentos del corpus en los que el término t aparece.

N el número total de documentos que forman el corpus.

3.2 Método GloVe [6]

Con este método introducimos la noción de representación en espacios vectoriales de palabras. El objetivo compartido de esta familia de métodos es capturar componentes semánticos de las palabras, donde cada componente semántico es representado por una componente del vector. Se busca capturar analogías del tipo

rey – reina = hombre – mujer donde el vector resultante de ambas restas sea más o menos el mismo. De esta forma nos aseguramos de que la componente semántica "género" está codificada de alguna forma en la representación vectorial. Otro ejemplo de analogía podría ser bibliotecario – biblioteca = panadero – panadería, en donde si se cumple la igualdad nos aseguraríamos de que la componente "profesión" queda implícita en la representación vectorial. Dentro de esta filosofía se encuentran dos grandes familias de métodos:

Métodos basados en factorización de matrices: Caracterizados por aprender los vectores utilizando la información estadística de todo el corpus. El funcionamiento se basa en que dos términos que aparecen juntos en el mismo contexto a menudo (entendiéndose por contexto a una cierta longitud de términos a la derecha e izquierda de la palabra en cuestión) deben tener una representación vectorial similar.
Para ello se hace uso de una matriz de co-ocurrencia de términos en donde se va sumando una unidad a a_{ij} por cada vez que el término j-ésimo aparece en el contexto del término i-ésimo en algún documento del corpus. En la práctica estas matrices son *sparse* o huecas, esto es que la gran mayoría de elementos son cero o muy bajos, por lo que el entrenamiento se centra en "encajar" las representaciones vectoriales de los términos cuya co-ocurrencia es relativamente alta.
Se agiliza el entrenamiento al centrarse únicamente en las relaciones notables entre pares de términos. La desventaja de esta familia de métodos es que los términos muy frecuentes en todo tipo de contextos "desproporcionan" la medida de similitud entre pares.

Métodos basados en ventanas del contexto local: Estos métodos tratan de aprender la representación vectorial de los términos realizando predicciones donde los términos contiguos (términos pertenecientes al contexto de otro dado) actúan como inputs para predecir el término en cuestión que actúa como output en una red neuronal. Tanto el input como el output de estas redes neuronales se codifican con one-hot encoding. El entrenamiento se realiza extrayendo como instancia cada elemento del corpus (un mismo término se extrae tantas veces como veces aparezca en todo el corpus, que puede verse como un único documento) junto con los términos que aparecen en su contexto. Estos métodos tienen la desventaja de que desaprovechan el poder resumir globalmente la información: Muchas de las instancias que servirán para entrenar la red neuronal serán repetidas o al menos serán muy parecidas. Como ventaja, estos métodos han demostrado aprender mejor las analogías del tipo rey – reina = hombre – mujer. El método GloVe (Global Vectors) pertenece al grupo de los métodos basados en factorización de matrices. El entrenamiento comienza por contabilizar la co-ocurrencia de los términos dos a dos, para ello haremos uso de una matriz X de tamaño N x N siendo N el número de términos del vocabulario que ha sido definido tras la primera fase de preprocesamiento. Las entradas X_{ij} corresponden al número de veces que el término j-ésimo ha aparecido en el contexto del término i-ésimo. No obstante, se tiene en cuenta si un término de contexto a otro dado se presenta inmediatamente contiguo o si está a unos cuantos términos de distancia.

Supongamos que se define una ventana de tamaño 3 a derecha e izquierda. Supóngase la frase "Las enfermedades genéticas son estudiadas por investigadores de diferentes disciplinas de la medicina". Tras eliminar stop-words la frase quedaría "enfermedades genéticas estudiadas investigadores diferentes disciplinas medicina". Si fijamos la palabra "investigadores" como término i-ésimo y "estudiadas" y "medicina" como términos j-ésimo y k-ésimo respectivamente (a distancias 1 y 3 de "investigadores"), tendremos que actualizar la matriz de co-ocurrencias con las siguientes operaciones:

$$X_{ij} \mathrel{+}= 1$$
$$X_{ik} \mathrel{+}= 1/3$$

De esta forma se van recorriendo todos los documentos del corpus término por término (asociando a cada término su correspondiente contexto) realizando la actualización de la matriz de co-ocurrencias de la misma forma que hemos ilustrado en el ejemplo previo.

Cuando se ha terminado este proceso de cuantificación de las co-ocurrencias ya se tiene la matriz X, la cual será tratada como el "target" de la relación entre pares de palabras.

Se dice que GloVe se basa en la factorización de matrices porque se usan dos matrices para representar los términos en forma vectorial de tal forma que el producto de ambas resulte en otra matriz, idealmente lo más parecida posible a la matriz X. Esto lo respalda el hecho de que dos vectores son más parecidos cuanto mayor es su producto escalar. Cada matriz codifica la representación vectorial que buscamos para los términos del vocabulario, solo que una lo hace por filas y otra por columnas.

Antes de comenzar el entrenamiento, el programador debe decidir el número de componentes que van a tener los vectores. Supongamos que se decide que sean 200. La relación entre pares que hace este método se modeliza matricialmente de la siguiente forma:

$$WU + B + S \approx X$$

W es una matriz N x 200. Cada fila corresponde a la representación vectorial de cada término del vocabulario.

U es una matriz 200 x N. Cada columna corresponde a la representación vectorial de cada término del vocabulario.

B es una matriz N x N. Sus parámetros corresponden al bias o sesgo del primer término (el codificado en la matriz W) por lo que todos los elementos de una misma fila son iguales.

S es una matriz N x N. Sus parámetros corresponden al bias o sesgo del segundo término (el codificado en la matriz U) por lo que todos los elementos de una misma columna son iguales.

X es la matriz N x N de co-ocurrencias que hemos obtenido.

Por tanto, los valores de las matrices W, U, B y S son los parámetros que se deben aprender. Al comenzar el entrenamiento se asignan valores aleatorios.

Nótese que se está aprendiendo dos representaciones vectoriales para cada término (las de la matriz W y las de la matriz U). Se escogerá como representación definitiva las de una o las de otra, de forma arbitraria.

Para relajar el efecto que tienen los términos con alta frecuencia de aparición se aplica el logaritmo sobre los elementos de X. Como puede haber elementos de esta matriz cuyo valor sea cero, antes de aplicar el logaritmo se suma una unidad a todos los elementos. De esta forma, la relación entre los términos i-ésimo y k-ésimo del vocabulario quedan modelizados de la siguiente forma:

$$\mathrm{w}_{i*}^{T} u_{*k} + b_{i*} + s_{*k} \approx \log(1 + X_{ik})$$

Donde $w_{i*}^{T} u_{*k}$ es el producto escalar de las representaciones vectoriales de los términos i-ésimo y k-ésimo, las cuales iran modificándose durante las iteraciones en el entrenamiento.

Para dar más importancia durante el entrenamiento a los pares de términos que aparecen juntos en el mismo contexto más a menudo (es lo que interesa acertar), se asigna un peso a cada par mediante cierta función del valor correspondiente x_{ij} que cumpla lo siguiente:

- $f(0) = 0$. Esto se exige porque si los términos i-ésimo y j-ésimo no han aparecido en el mismo contexto nunca en todo el corpus no podemos cuantificar su similitud. No se busca acertar la relación de estos pares durante el entrenamiento.

- $f(x_{ij})$ debe ser no-decreciente. Los pares para los que se ha cuantificado una relación menor, no pueden tener más peso en la función de coste que los pares para los que se ha cuantificado una relación mayor.

- $f(x_{ij})$ debe estar acotada superiormente tomando un valor constante a partir de cierto valor. Esta condición se exige para no asignar un peso descomunalmente grande a los pares que aparecen juntos con alta frecuencia.

Una familia de funciones que cumplen estas características y que ha demostrado funcionar eficazmente es:

$$\mathrm{f(x)} = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & if \;\; x < x_{max} \\ 1 & otherwise \end{cases}$$

En la Ilustración 5 se muestra una gráfica de una función perteneciente a esta familia, con $\alpha = 3/4$:

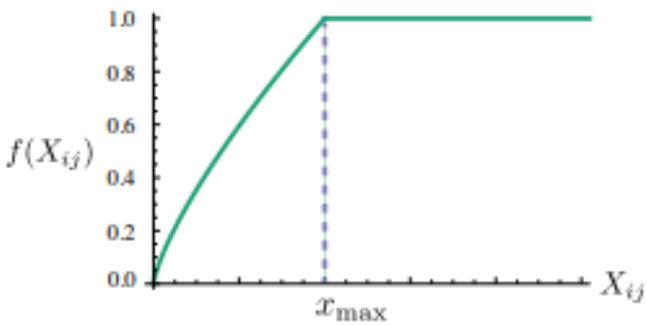

Ilustración 5: Gráfica de la función que asigna pesos a las parejas de ítems en función de la cuantificación de su co-ocurrencia

Por tanto, la función de coste la cual deberá minimizarse durante el entrenamiento es:

$$\mathrm{J} = \sum_{i,j=1}^{N} f(X_{ij})[w_i^T u_j + b_{i*} + b_{*j} - \log(X_{ij} + 1)]^2$$

3.3 Método Word2vec [7]

Este método pertenece a la familia de métodos de ventana de contexto local que explicamos en la sección anterior. Se comienza por definir un número de términos que conformarán la vecindad de otro dado, estos términos se toman por igual a derecha e izquierda del término en cuestión en cierto corpus. El entrenamiento se realiza extrayendo como instancia cada elemento del corpus (un mismo término se extrae tantas veces como veces aparezca en todo el corpus, que puede verse como un único documento) junto con los términos que aparecen en su contexto.

Dentro de las técnicas word2vec detallaremos la técnica CBOW (continuous bag of words). Esta consiste en aprender las representaciones vectoriales por medio de una red neuronal de una única capa oculta. La arquitectura se muestra en la Ilustración 6.
Supongamos que el vocabulario queda definido en V términos, se decide que las representaciones vectoriales tengan N componentes y se establece un contexto de tamaño 3. El procedimiento que realiza esta red es el siguiente:

- Tanto la entrada como la salida están codificadas con one-hot-encoding. La capa de entrada consta de 3 vectores de tamaño 1xV. La salida es un vector de tamaño 1xV. Al representar cada componente del vector salida la probabilidad de un término del vocabulario, los valores de las componentes deben ser normalizados mediante la función Softmax. El valor para la componente j-ésima que devuelve esta función es:

$$\frac{e^{z_j}}{\sum_{k=1}^{V} e^{z_k}}$$

- Se tienen dos matrices de pesos. Una entre la capa de entrada y la capa oculta y la otra entre la capa oculta y la capa de salida. La primera es de tamaño VxN y la segunda de tamaño NxV.

- No hay ninguna función de activación entre ninguna de las capas.

- Cada uno de los tres vectores input es multiplicado por la primera matriz de pesos (resultando una copia de la fila i-ésima de la matriz si el vector input tiene su único elemento no nulo en la posición i-ésima). Los tres vectores resultantes se agregan usando la media. Al resultado de ésta operación se le conoce como estado oculto.

- Se multiplica el estado oculto por la segunda matriz de pesos. El resultado se entrega como output de la red neuronal.

- Calcular el error entre el output y el target, que recordamos que es un vector 1xV en one-hot-encoding (todo ceros salvo un uno). Usar backpropagation para reajustar los pesos de las dos matrices de la red.

- Los pesos de la segunda matriz (columnas) se toman como las representaciones vectoriales de los V términos. La columna i-ésima de esta matriz corresponde a la representación vectorial del término i-ésimo del vocabulario.

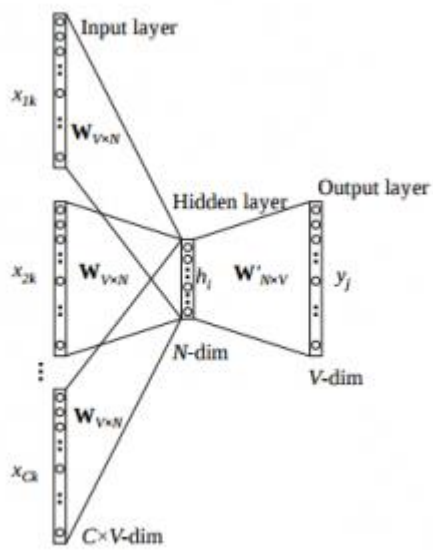

Ilustración 6: Arquitectura red neuronal modelo CBOW de Word2vec

La otra implementación incluida en Word2vec es Skipgram. La diferencia entre ésta y CBOW es que la arquitectura de una y de otras son inversas. Esto quiere decir que con Skipgram el input es el término "target" y el output son los términos del contexto. Por tanto, en la arquitectura Skipgram el input es un vector 1 x V codificado también con one-hot encoding. La matriz de pesos entre el input y el estado oculto es de la forma V x N. El estado oculto es de la forma 1 x N. La segunda matriz de pesos es de la forma N x V. De esta manera el output es de la forma 1 x V. Las componentes del vector salida deben normalizarse haciendo uso de la función Softmax dado que se interpretan como probabilidades de que el término codificado en cada componente pertenezca al contexto de la palabra input. Los pesos de la primera matriz (filas) se toman como las representaciones vectoriales de los términos. En la Ilustración 7 se muestra la topología de Skipgram.

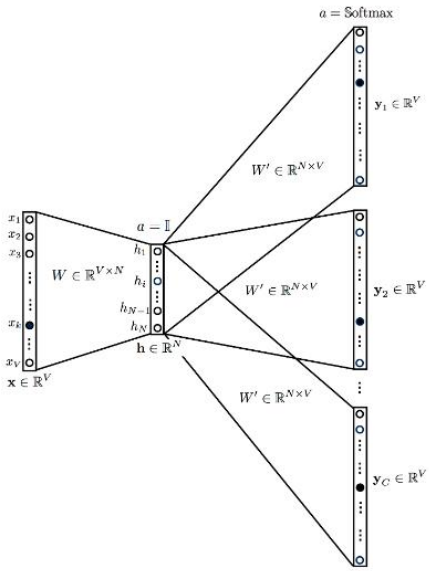

Ilustración 7: Arquitectura de la implementación Skipgram en Word2vec

3.4 FastText [8]

Los métodos anteriores trataban cada término del vocabulario como una unidad de significado y no tenían en cuenta la estructura interna de las palabras. En la práctica las palabras se componen de unidades más simples como raíz, sufijo o prefijo. Pueden existir muchas palabras derivadas de otra dada. El tener este hecho en cuenta cobra especial importancia cuando en el corpus de entrenamiento para la representación vectorial de los términos del vocabulario hay palabras que aparecen pocas veces. Si un término aparece poco se tiene una muestra muy pequeña de los contextos en los que aparece por lo que es difícil establecer una acertada representación vectorial del mismo. No obstante, es más probable que al dividir esa palabra en partes más pequeñas, alguna de ellas aparezca con relativa frecuencia en el corpus. Ilustrémoslo con un ejemplo: Puede darse el caso de que en un corpus la palabra "gastronomía" aparezca poco. Si la atomizamos en "gastro" y"nomia" quizá sea más probable encontrar estos tokens en el corpus, pese a que pertenezcan a términos distintos como "gastroenteritis" o "astronomía".

FastText puede verse como un derivado del método Word2vec Skipgram en donde el input, en lugar de ser el término cuyo contexto ha de predecirse se tienen como tal los n-gramas de dicho término. Por ello, una vez se tiene definido el vocabulario (los términos "enteros" de los que se quiere averiguar su representación vectorial) se ha de representar cada uno de ellos como un conjunto de n-gramas a los que llamamos tokens. Para ello el usuario define unas longitudes mínima y máxima. Por ejemplo, si definimos un mínimo de 3 y máximo de 4, para el término "hablar" el conjunto de n-gramas sería:

{"<ha" , "hab", "abl", "bla","lar", "ar>", "<hab", "habl", "abla", "blar", "lar>", "<hablar>"}

Nótese que se emplean los símbolos "<" y ">" para denotar el inicio y final, respectivamente. También debe incluirse siempre el término completo dentro del conjunto, pues al final es del que nos interesa conocer su representación vectorial.

Se debe crear este conjunto para todos los términos. Durante el entrenamiento se aprenderá la representación vectorial de todos los tokens presentes en el corpus, así como de los términos.

Volvamos a la diferencia con el método Word2vec. En este caso, el estado oculto de la red neuronal lo forma la suma de los tokens (parámetros que están aprendiéndose) de la palabra input. Supongámoslo como un vector de tamaño 1 x N.
Si el vocabulario está formado por V términos (no tokens), la matriz de pesos entre el estado oculto y el output es de la forma N x V. De esta manera el output será de tamaño 1 x V. Cada una de estas componentes del output corresponde al producto escalar entre el estado oculto y la representación vectorial de cada término (no token) aplicándole después la función sigmoide para obtener un valor entre 0 y 1. Se buscan valores cercanos a 1 para los términos que pertenecen al contexto de la palabra input.

En la práctica, entre el estado oculto y el output no se hace el producto escalar con todos los términos del vocabulario. Los productos escalares que se conservan son con los términos que aparecen en su contexto (positive sampling) así como 5 términos elegidos al azar del vocabulario que no aparecen en el contexto por cada término que sí aparece (si el contexto es de tamaño 4, deberán escogerse al azar 20 términos que no aparecen en ese contexto). A esto último se le conoce como negative sampling. La selección de un término durante el negative sampling obedece a una probabilidad, la cual es directamente proporcional a la frecuencia de su aparición en los documentos del corpus. Esta característica de no realizar el producto escalar con todos los términos del vocabulario es lo que le da nombre al método: "Fast", ya que agiliza muchísimo el entrenamiento.

El error para cada término input cuyo contexto ha de predecirse se calcula restando el target en formato one-hot encoding (las palabras de su contexto) a la salida de la red neuronal. Aplicar backpropagation para actualizar los parámetros.

Para poder llevar el entrenamiento ha de almacenarse para cada término su conjunto de n-gramas, así como un index que represente la posición de este término entre el total de términos (no tokens) del vocabulario.

Mostremos de forma esquemática el proceso con un ejemplo:

Se tiene la frase "I am eating food". Se toma como término central la palabra "eating" y como términos de contexto "am" y "food". El input será "eating" (más concretamente los n-gramas que lo forman).

El conjunto de n-gramas de "eating", así como el cálculo de su estado oculto se muestran en Ilustración 8.

Mantenemos los vectores de "am" y "food" para hacer el producto escalar con el estado oculto. Buscamos también términos pertenecientes al vocabulario que no se encuentran en este contexto: "paris" y "earth". Pese a que en este ejemplo sólo escogemos 2, deberían escogerse 10 (ya que como hemos explicado previamente 2x5 = 10).

Se realiza el producto escalar entre el estado oculto y cada uno de los 4 términos mencionados. A los resultados se les aplica la función sigmoide. Este procedimiento queda ilustrado en Ilustración 9.

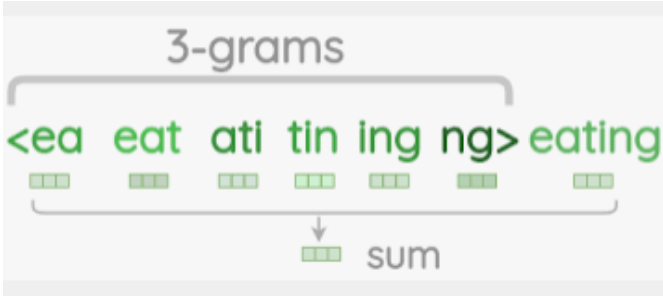

Ilustración 8: N-gramas y estado oculto en el método FastText

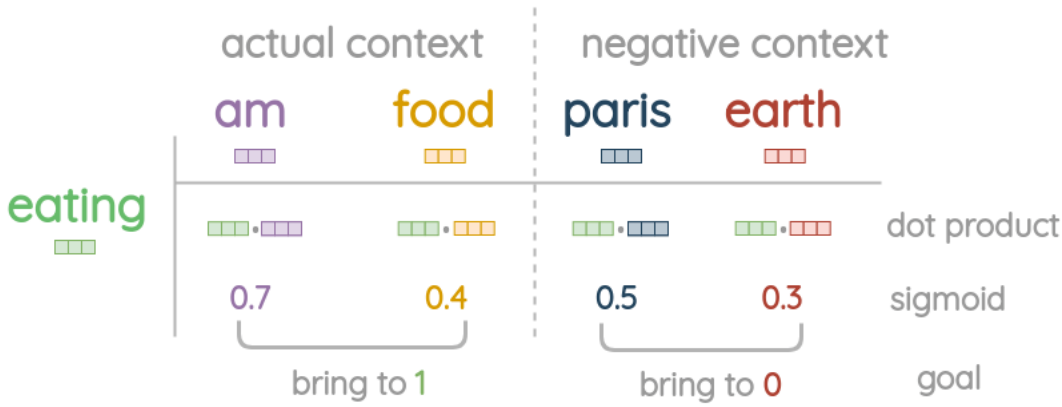

Ilustración 9: Procedimiento entre el estado oculto y el output en el método FastText

4. EXPERIMENTACIÓN Y RESULTADOS

4.1 Dataset

El dataset ha sido extraído del repositorio de Kaggle. Procede de un Trabajo de Fin de Máster en Ciberseguridad por la Universidad Politécnica de Madrid [9].

Se disponen de 538 instancias para train y 60 para test. Cada instancia consta de un texto en castellano correspondiente al cuerpo de una noticia y su clase, la cual toma dos valores posibles: 'fake' o 'real'.

Tanto las instancias de train como las de test están balanceadas, lo cual significa que la cantidad de instancias cuya clase es 'fake' es aproximadamente la misma que la de 'real'.

4.2 Diseño de la experimentación

La experimentación la hemos dividido en tantas secciones como métodos de extracción de características se van a probar, es decir cuatro. La estructura de la experimentación en cada método difiere en algunos casos del resto a causa de la naturaleza del mismo y/o de hacer uso de información entrenada por terceros o por nosotros:

En el caso de Bag Of Words TF-IDF la experimentación consta de una primera fase de preprocesamiento en donde se establecen qué términos van a formar parte del vocabulario. A continuación, se analizan los resultados con cada uno de los cinco clasificadores que se van a usar.

En el caso de GloVe, hemos procedido a entrenar nosotros las representaciones vectoriales. La experimentación se centra en analizar los resultados con cada uno de los cinco tipos de clasificadores, donde cada clasificador es probado con cinco formas de agregar los términos de una misma instancia (noticia). También se ha incluido una tabla para ilustrar la relación entre términos que resulta del método, mostrando las similitudes entre ciertos pares de palabras.

Para el caso de Word2vec y FastText las representaciones vectoriales usadas han sido entrenadas por terceros. Las hemos descargado de un repositorio de GitHub [https://github.com/dccuchile/spanish-word-embeddings]. Se analizan los resultados con cada uno de los cinco tipos de clasificadores, donde cada clasificador es probado con cinco formas de agregar los términos de una misma instancia (noticia). También se ha incluido una sección para ilustrar la relación entre términos que resulta del método, mostrando las similitudes entre los mismos pares de términos que los expuestos en la sección de GloVe.
Para las cuatro técnicas, los valores que se prueban durante el ajuste de hiperparámetros de cada clasificador son los mismos, y se seleccionan haciendo uso de la validación cruzada (explicada a continuación) con cuatro particiones. Los errores reportados para cada uno de ellos son evaluaciones sobre el conjunto de test (60 instancias balanceadas).

Por último, se ha dedicado una última sección a la visualización, donde usando los vectores de características obtenidos con Word2Vec y FastText se pretende ilustrar la agrupación de términos de distintos ámbitos, así como la exploración de algunas analogías .No hemos considerado útiles para la visualización los vectores de características obtenidos con GloVe al haber tenido que establecer una vocabulario bastante reducido, como explicamos en la sección dedicada al mismo.

4.2.1 Técnica de validación cruzada [3]

Ésta técnica se emplea para seleccionar el mejor ajuste de parámetros. Dada una configuración de hiperparámetros, se dividen los datos pertenecientes al conjunto de entrenamiento en K particiones disjuntas con la misma proporción de instancias de cada clase en cada partición.
Se llevarán a cabo K procesos de entrenamiento que pueden hacerse de forma paralela pues son independientes. Para cada uno de estos entrenamientos, se usan K-1 particiones para entrenar y se reserva otra partición para validar. Con la partición de validación de obtiene un accuracy del modelo.
Para cada entrenamiento, la partición de validación es distinta. Finalmente, se reporta como accuracy del modelo con una cierta configuración de hiperparámetros, a la media de los accuracy en validación de los K entrenamientos. En la Ilustración 10 se ilustra este procedimiento:

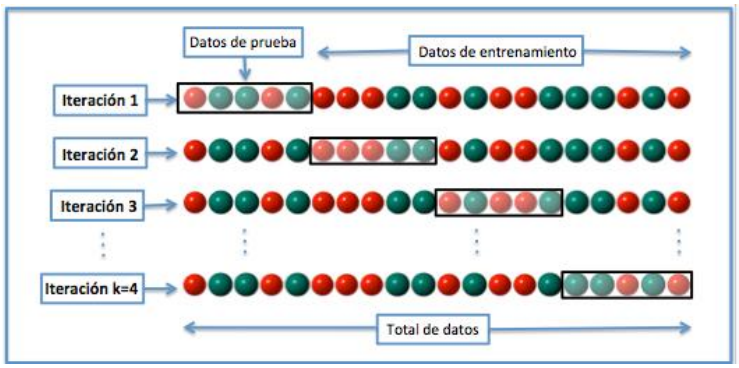

Ilustración 10: Técnica K-Fold

4.3 Bag Of Words TF-IDF

Tras normalizar todos los términos presentes en el corpus de train eliminando signos no alfabéticos y reduciendo cada palabra a su raíz aplicando stemming, el primer procedimiento ha sido establecer los umbrales máximo y mínimo de frecuencia de aparición de los términos en algún documento del corpus. Para seleccionar la combinación óptima se ha empleado validación cruzada con cuatro particiones usando regresión logística como clasificador y el accuracy como score. Los candidatos a umbral mínimo han sido 0.001, 0.01, 0.04, 0.07 y 0.09. Los candidatos a umbral máximo han sido 0.1, 0.3, 0.5, 0.6, 0.7 y 0.95. La combinación óptima ha resultado ser 0.04 como umbral mínimo y 0.6 como umbral máximo.

Por tanto, para obtener el vocabulario que definirá a cada uno de los textos, se escoge la combinación 0.6 (umbral máximo) y 0.04 (umbral mínimo). El número de términos que pertenecen a éste vocabulario es 838. El vocabulario resultante será el empleado para todos los clasificadores usados:

Para el clasificador Adaboost se ajusta como hiperparámetro el número de estimadores base. El ajuste se lleva a cabo por validación cruzada con cuatro particiones y accuracy como medida de scoring. Los siguientes reportes son sobre el conjunto de test usando la mejor configuración de hiperparámetros encontrada durante la validación cruzada:

Nº Estimadores	Accuracy	Precisión	Recall
700	90%	95.45%	80.76%

Tabla 1:Métricas de rendimiento para el clasificador Adaboost en función del número de estimadores base

Para la clasificación mediante regresión logística:

Accuracy	Precisión	Recall
91.66%	95.65%	84.61%

Tabla 2: Resultados del clasificador regresión logística para TF-IDF

Para el clasificador Random Forest se fija el número de estimadores base en 100 y se ajustan los parámetros de profundidad máxima y número de variables a examinar mediante validación cruzada. Los siguientes reportes son sobre el conjunto de test usando la mejor configuración de hiperparámetros encontrada durante la validación cruzada:

Nº vbles a examinar en cada nodo	Profundidad máxima	Accuracy	Precisión	Recall
30	30	93.33%	95.83%	88.46%

Tabla 3:Métricas de rendimiento para Random Forest en TF-IDF

Para el clasificador SVM se ajustan como hiperparámetros el valor C y el valor gamma. El ajuste se lleva a cabo por validación cruzada con cuatro particiones y accuracy como medida de scoring. Los siguientes reportes son sobre el conjunto de test usando la mejor configuración de hiperparámetros encontrada durante la validación cruzada:

C	gamma	Accuracy	Precisión	Recall
10	0.7	91.66%	95.65%	84.61%

Tabla 4: Métricas de rendimiento para SVM en TF-IDF

Para el clasificador Naive-Bayes:

Accuracy	Precisión	Recall
86.66%	87.5%	80.76%

Tabla 5: Métricas de rendimiento para Naive-Bayes en TF-IDF

4.3 GloVe

En éste método, obtenemos las representaciones vectoriales empleando las instancias de train de nuestro problema como corpus para el entrenamiento. Decidimos obtener representaciones vectoriales de 200 componentes. Como la capacidad computacional de nuestra máquina es limitada, necesitamos hacer una reducción de términos del vocabulario. Decidimos emplear el mismo vocabulario que el que se definió en la sección anterior (Bag Of Words TF-IDF) el cual consta de 838 términos. Tras el entrenamiento, decidimos cuantificar similitudes entre algunas de las representaciones obtenidas para ilustrar cómo de bien ha ido el entrenamiento. Usaremos estos mismos términos para ilustrar también los próximos métodos de extracción de características. Para ello usamos la similitud del coseno:

$$similitud(x,y) = \frac{< x,y >}{||x|| \cdot ||y||}$$

En la ilustración 10 se muestra la similitud del coseno (a mayor valor mayor relación entre los términos) de 8 términos dos a dos:

	Dinero (diner)	**Millones (millon)**	**Banco (banc)**	**Juez (juez)**	**Denuncia (denunci)**	**Muertos (muert)**	**Abogado (abog)**	**Asesinato (asesinat)**
Dinero (diner)	1	0.16	0.04	0.03	0.12	0.02	-0.09	-0.009
Millones (millon)	0.16	1	0.07	-0.16	0.03	0.05	-0.02	-0.05
Banco (banc)	0.04	0.07	1	0.08	-0.025	0.017	-0.03	0.019
Juez (juez)	0.04	-0.02	-0.06	1	-0.07	-0.03	0.11	0.11
Denuncia (denunci)	0.12	0.03	-0.025	-0.07	1	0.1	-0.009	-0.007
Muertos (muert)	0.02	0.05	0.017	-0.13	0.1	1	0.01	0.1
Abogado (abog)	-0.09	-0.02	-0.03	0.11	-0.009	0.01	1	0.03
Asesinato (asesinat)	-0.009	-0.05	0.019	0.006	-0.007	0.1	0.03	1

Tabla 6: Similitudes entre pares de términos empleando las representaciones vectoriales obtenidas por GloVe

Como ahora tenemos un vector por cada término perteneciente al vocabulario y cada instancia (texto) debe ser representada por un único vector, los términos pertenecientes a una misma instancia deben ser agregados. Para ello probaremos con cinco métodos:

- Máximo: Para la componente i-ésima nos quedamos con el mayor valor que toma la misma componente entre todas las componentes i-ésimas de los términos que aparecen en la instancia y pertenecen al vocabulario.

- Mínimo: Para la componente i-ésima nos quedamos con el menor valor que toma la misma componente entre todas las componentes i-ésimas de los términos que aparecen en la instancia y pertenecen al vocabulario.

- Suma probabilística: Si disponemos de N instancias en el texto, cada componente i del vector final tomará el valor $1 - (1 - x_{1i})(1 - x_{2i}) \cdot \ldots \cdot (1 - x_{Ni})$.

- Media aritmética: Cada componete i-ésima del vector tomará el valor de la media aritmética de todas las componentes i-ésimas de los términos que se incluyen en el texto y pertenecen al vocabulario.

- Media geométrica: Cada componente i-ésima del vector tomará el valor $(x_{1i} \cdot x_{2i} \cdot \ldots \cdot x_{Ni})^{\frac{1}{N}}$ siendo N el número de términos que se encuentran en el texto y pertenecen al vocabulario.

A continuación, mostraremos para cada clasificador y cada tipo de agregación los resultados obtenidos con la mejor configuración de hiperparámetros, en el caso de haberlos. La mejor configuración se obtiene mediante validación cruzada con cuatro particiones. Las métricas de rendimiento reportadas son evaluadas sobre el conjunto de test:

A continuación, se muestran los resultados para Regresión logística usando los cinco tipos de agregación:

Agregación	Accuracy	Precisión	Recall
Máximo	76.66%	83.33%	57.69%
Mínimo	73.33%	75%	57.69%
Suma probabilística	61.66%	100%	11.53%
Media aritmética	71.66%	100%	34.61%
Media geométrica	71.66%	90.9%	38.46%

Tabla 7: Métricas de evaluación del rendimiento de la clasificación mediante regresión logística para GloVe

Con el clasificador Random Forest se fija el número de clasificadores base en 100 y se buscan los mejores valores para el número de variables a examinar en cada nodo y la profundidad máxima del árbol:

Agregación	Profundidad máxima	Número de variables a examinar	Accuracy	Precisión	Recall
Máximo	30	30	81.66%	82.6%	73.07%
Mínimo	30	30	83.33%	90%	69.23%
Suma probabilística	30	30	61.66%	54.28%	73.07%
Media aritmética	10	30	76.66%	80%	61.53%
Media geométrica	30	30	73.33%	72.72%	61.53%

Tabla 8: Rendimiento para las mejores configuraciones de hiperparámetros para el clasificador Random Forest para GloVe

Para el clasificador Adaboost se ajusta el número de estimadores base:

Agregación	Número de estimadores	Accuracy	Precisión	Recall
Máximo	100	75%	73.91%	65.38%
Mínimo	700	75%	73.91%	65.38%
Suma probabilística	10	50%	44.44%	61.53%
Media aritmética	700	78.33%	78.26%	69.23%
Media geométrica	10	50%	41.66%	38.46%

Tabla 9: Rendimiento para las mejores configuraciones de hiperparámetros para el clasificador Adaboost para GloVe

Para el clasificador Naive Bayes no se hace ninguna búsqueda de hiperparámetros y los resultados obtenidos se muestran a continuación:

Agregación	Accuracy	Precisión	Recall
Máximo	61.66%	100%	11.53%
Mínimo	68.33%	81.81%	34.61%
Suma probabilística	61.66%	100%	11.53%
Media aritmética	61.66%	100%	11.53%
Media geométrica	61.66%	100%	11.53%

Tabla 10: Rendimiento para el clasificador Naive Bayes para GloVe

Con el clasificador Máquina de Soporte Vectorial se hace una búsqueda de hiperparámetros para los valores de C (que controla cuánto se centra en acertar todas las instancias de entrenamiento) y de gamma (que controla la similitud al resto de instancias cuando se hace el mapeo a dimensionalidad mayor). Los resultados obtenidos se muestran a continuación:

Agregación	C	gamma	Accuracy	Precisión	Recall
Máximo	10	0.7	76.66%	73.07%	73.07%
Mínimo	100	0.5	73.33%	70.83%	65.38%
Suma probabilística	100	0.2	60%	75%	11.53%
Media aritmética	100	0.2	73.33%	69.23%	69.23%
Media geométrica	100	0.2	68.33%	62.96%	65.38%

Tabla 11: Rendimiento para las mejores configuraciones de hiperparámetros con el clasificador SVM para GloVe

4.4 Word2vec

Las representaciones vectoriales obtenidas mediante este método las hemos descargado ya pre-entrenadas del repositorio de GitHub [https://github.com/dccuchile/spanish-word-embeddings]. Han sido obtenidas con la variante Skipgram empleando un corpus de mil millones de palabras resultando un vocabulario de 820.000 términos. Las representaciones vectoriales constan de 300 componentes.

A continuación, cuantificaremos la similitud por pares para los 8 términos con los que hicimos lo mismo en la sección anterior dedicada a la experimentación con el método GloVe. La única diferencia es que en la sección de GloVe estas comparaciones se hacían entre los términos reducidos a su raíz. Empleamos también la similitud del coseno:

	Dinero	Millones	Banco	Juez	Denuncia	Muertos	Abogado	Asesinato
Dinero	1	0.45	0.36	0.3	0.37	0.22	0.31	0.28
Millones	0.45	1	0.3	0.23	0.26	0.29	0.23	0.13
Banco	0.36	0.3	1	0.21	0.16	0.15	0.25	0.16
Juez	0.3	0.23	0.21	1	0.49	0.19	0.61	0.41
Denuncia	0.37	0.26	0.16	0.49	1	0.24	0.44	0.39
Muertos	0.22	0.29	0.15	0.19	0.24	1	0.14	0.37
Abogado	0.31	0.23	0.25	0.61	0.44	0.14	1	0.4
Asesinato	0.28	0.13	0.16	0.41	0.39	0.37	0.4	1

Tabla 12: Similitudes entre pares de términos usando Word2vec

A continuación, mostraremos para cada clasificador y cada tipo de agregación los resultados obtenidos con la mejor configuración de hiperparámetros. Esta se encuentra mediante validación cruzada.

A continuación, se muestran los resultados para Regresión logística usando los cinco tipos de agregación:

Agregación	Accuracy	Recall	Precisión
Máximo	83.33%	69.23%	90%
Mínimo	86.66%	80.76%	87.5%
Suma probabilística	58.33%	3.84%	100%
Media aritmética	91.66%	88.46%	92%
Media geométrica	91.66%	92.3%	88.88%

Tabla 13: Rendimiento del clasificador Regresión Logística para Word2vec

Con el clasificador Random Forest se fija el número de clasificadores base en 100 y se buscan los mejores valores para el número de variables a examinar en cada nodo y la profundidad máxima del árbol mediante validación cruzada:

Agregación	Profundidad máxima	Nº variables a examinar	Accuracy	Recall	Precisión
Máximo	30	30	93.33%	84.61%	100%
Mínimo	10	10	88.33%	80.76%	91.3%
Suma probabilística	5	5	61.66%	11.53%	100%
Media aritmética	10	30	90%	88.46%	88.46%
Media geométrica	30	30	91.66%	92.3%	88.88%

Tabla 14: Rendimiento del clasificador Random Forest para Word2vec

Con el clasificador Adaboost se hace una búsqueda para el número de clasificadores base. A continuación se muestran los rendimientos para este clasificador:

Agregación	Nº estimadores	Accuracy	Recall	Precisión
Máximo	700	86.66%	84.61%	84.61%
Mínimo	700	86.66%	84.61%	84.61%
Suma probabilística	10	61.66%	11.53%	100%
Media aritmética	700	93.33%	92.3%	92.3%
Media geométrica	700	93.33%	96.15%	89.28%

Tabla 15: Rendimientos del clasificador Adaboost para word2vec

Para el clasificador Naive Bayes no se hace ninguna búsqueda de hiperparámetros y los resultados obtenidos se muestran a continuación:

Agregación	Accuracy	Recall	Precisión
Máximo	58.33%	19.23%	55.55%
Mínimo	61.66%	19.23%	71.4%
Suma probabilística	58.33%	3.8%	100%
Media aritmética	90%	88.46%	88.46%
Media geométrica	45%	96.15%	43.85%

Tabla 16: Rendimientos del clasificador Naive Bayes para word2vec

Con el clasificador Máquina de Soporte Vectorial se hace una búsqueda de hiperparámetros para los valores de C (que controla cuánto se centra en acertar todas las instancias de entrenamiento) y de gamma (que controla la similitud al resto de instancias cuando se hace el mapeo a dimensionalidad mayor). Los resultados obtenidos se muestran a continuación:

Agregación	C	gamma	Accuracy	Precisión	Recall
Máximo	10	0.7	86.66%	87.5%	80.76%
Mínimo	10	0.5	86.66%	84.61%	84.61%
Suma probabilística	1000	0.7	58.33%	100%	3.8%
Media aritmética	100	0.2	91.66%	92%	88.46%
Media geométrica	100	0.7	96.66%	92.85%	100%

Tabla 17: Rendimientos del clasificador SVM para Word2vec

4.5 FastText

Las representaciones vectoriales obtenidas mediante este método las hemos descargado ya pre-entrenadas del repositorio de GitHub [https://github.com/dccuchile/spanish-word-embeddings]. Han sido obtenidas empleando un corpus de mil millones de palabras resultando un vocabulario de 855.000 términos. Las representaciones vectoriales constan de 300 componentes.
A continuación, cuantificaremos la similitud por pares para los 8 términos con los que hicimos lo mismo en las secciones anteriores. Empleamos también la similitud del coseno:

	Dinero	Millones	Banco	Juez	Denuncia	Muertos	Abogado	Asesinato
Dinero	1	0.38	0.35	0.23	0.26	0.14	0.22	0.17
Millones	0.38	1	0.29	0.16	0.17	0.31	0.14	0.08
Banco	0.35	0.29	1	0.27	0.16	0.12	0.25	0.18
Juez	0.23	0.16	0.27	1	0.41	0.16	0.58	0.36
Denuncia	0.26	0.17	0.16	0.41	1	0.2	0.38	0.36
Muertos	0.14	0.31	0.12	0.16	0.2	1	0.07	0.38
Abogado	0.22	0.14	0.25	0.58	0.38	0.07	1	0.32
Asesinato	0.17	0.08	0.18	0.36	0.36	0.38	0.32	1

Tabla 18: Similitudes por pares de términos empleando las representaciones ontenidas mediante FastText

A continuación, mostraremos para cada clasificador y cada tipo de agregación los resultados obtenidos con la mejor configuración de hiperparámetros. Esta se encuentra mediante validación cruzada.

A continuación, se muestran los resultados para Regresión logística usando los cinco tipos de agregación:

Agregación	Accuracy	Recall	Precisión
Máximo	81.66%	73.07%	82.6%
Mínimo	83.33%	73.07%	86.36%
Suma probabilística	58.33%	3.84%	100%
Media aritmética	80%	53.84%	100%
Media geométrica	80%	53.84%	100%

Tabla 19: Rendimiento clasificador Regresión Logística para el método FastText

Con el clasificador Random Forest se fija el número de clasificadores base en 100 y se buscan los mejores valores para el número de variables a examinar en cada nodo y la profundidad máxima del árbol mediante validación cruzada:

Agregación	Profundidad máxima	Nº variables a examinar	Accuracy	Recall	Precisión
Máximo	30	30	75%	65.38%	73.9%
Mínimo	30	5	90%	76.9%	100%
Suma probabilística	5	5	61.66%	11.53%	100%
Media aritmética	30	10	96.66%	96.15%	96.15%
Media geométrica	10	10	96.66%	96.15%	96.15%

Tabla 20: Rendimiento del clasificador RandomForest en el método FastText

Con el clasificador Adaboost se hace una búsqueda para el número de clasificadores base. A continuación se muestran los rendimientos para éste clasificador:

Agregación	Nº estimadores	Accuracy	Recall	Precisión
Máximo	700	83.33%	88.46%	76.66%
Mínimo	700	93.33%	92.3%	92.3%
Suma probabilística	700	61.66%	11.53%	100%
Media aritmética	700	90%	84.6%	91.6%
Media geométrica	700	88.33%	84.61%	88%

Tabla 21: Rendimiento Clasificador Adaboost en el método FastText

Para el clasificador Naive Bayes no se hace ninguna búsqueda de hiperparámetros y los resultados obtenidos se muestran a continuación:

Agregación	Accuracy	Recall	Precisión
Máximo	65%	30.7%	72.72%
Mínimo	70%	46.15%	75%
Suma probabilística	58.33%	3.8%	100%
Media aritmética	93.33%	88.46%	95.8%
Media geométrica	41.66%	96.15%	42.3%

Tabla 22: Rendimiento clasificador Naive Bayes en el método FastText

Con el clasificador Máquina de Soporte Vectorial se hace una búsqueda de hiperparámetros para los valores de C (que controla cuánto se centra en acertar todas las instancias de entrenamiento) y de gamma (que controla la similitud al resto de instancias cuando se hace el mapeo a dimensionalidad mayor):

Agregación	C	gamma	Accuracy	Precisión	Recall
Máximo	10	0.2	86.66%	82.14%	88.46%
Mínimo	10	0.2	88.33%	91.3%	80.7%
Suma probabilística	100	0.5	58.33%	100%	3.8%
Media aritmética	1000	0.5	96.66%	96.15%	96.15%
Media geométrica	100	0.7	96.66%	96.15%	96.15%

Tabla 23: Rendimiento clasificador SVM en el método FastText

4.6 Visualización de representaciones vectoriales y exploración de analogías

A continuación, procedemos a presentar una visualización de las representaciones vectoriales de algunas palabras obtenidas mediante las técnicas Word2vec y FastText. Para ello, se ha hecho una reducción de dimensionalidad aplicando la técnica de análisis de componentes principales (PCA), lo cual supone pasar de 300 dimensiones a 2. Estas dos dimensiones con las que visualizamos las instancias (palabras) son las de mayor variabilidad posible de las obtenidas a partir de combinaciones lineales de las dimensiones originales.

Las dimensiones seleccionadas para representar con Word2vec explican un 3,3% (eje horizontal) y 2,9% (eje vertical) de la variabilidad total:

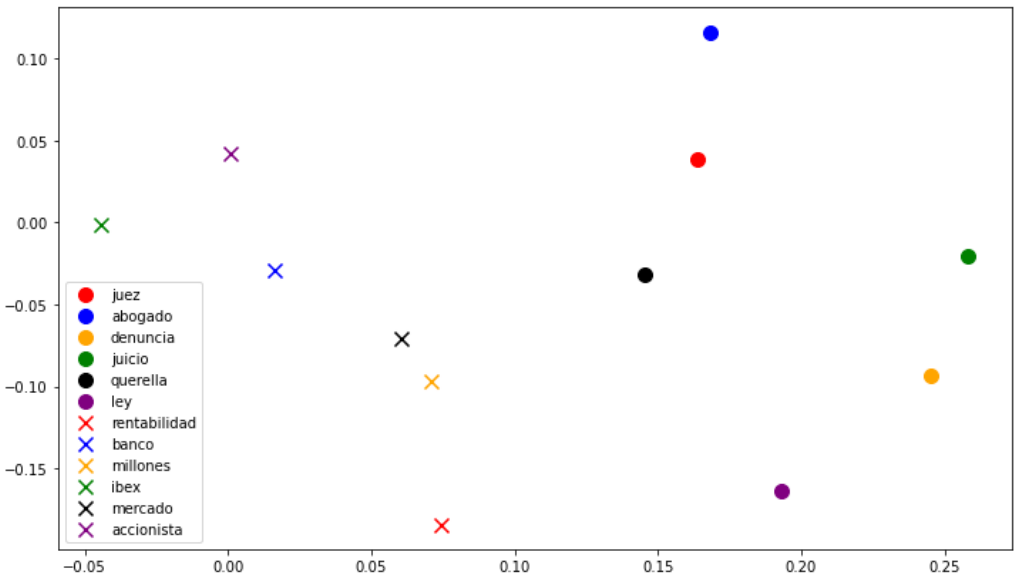

Ilustración 11: Visualización de la representación de algunos términos obtenidos con la técnica Word2vec

Las dimensiones seleccionadas para representar con FastText explican un 3,5% (eje horizontal) y 2,8% (eje vertical) de la variabilidad total:

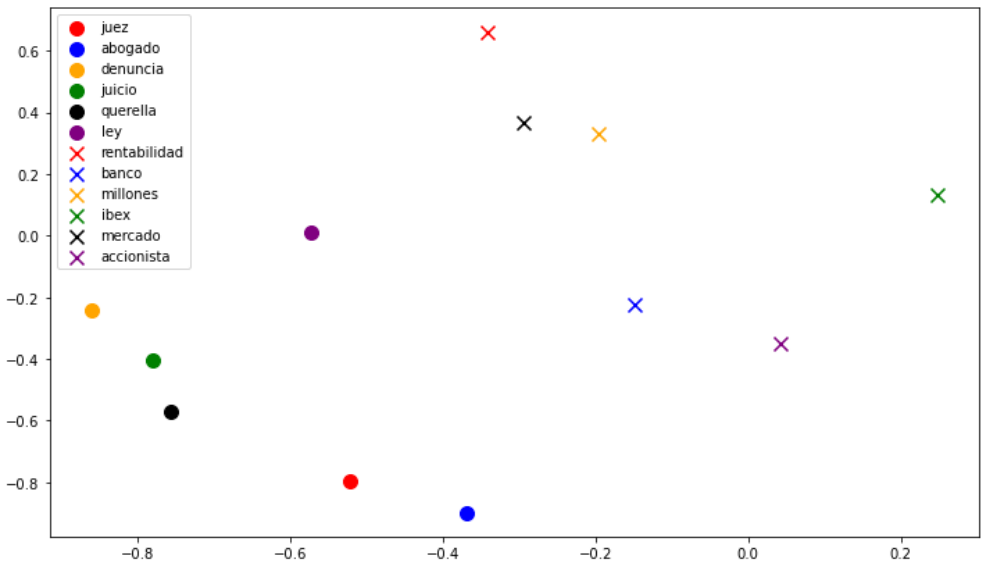

Ilustración 12: Visualización de la representación de algunos términos obtenidos con la técnica FastText

En ambas ilustraciones (ilustraciones número 11 y 12) se aprecia cómo los seis términos relativos al ámbito jurídico (juez, abogado, denuncia, juicio, querella, ley) tienden a agruparse por un lado, así como los relativos al ámbito de las finanzas (rentabilidad, banco, millones, Ibex, mercado, accionista) se agrupan por otro lado.

Se ha explicado previamente que en los métodos de representación vectorial de palabras se busca que se dé el fenómeno de las analogías. De este modo se codifican en las representaciones vectoriales conceptos genéricos como "natal de", "dedicado a", "estudio de", sustantivaciones…etc. Por ejemplo, si dadas las representaciones vectoriales de los términos España, español, Francia y francés se cumple que español – España = francés – Francia sabemos que el vector resultante a ambos lados de la ecuación codifica el concepto abstracto "natal de".

De los tres métodos de extracción de características de texto que hemos tratado en este trabajo, con el primero de ellos (GloVe) no podemos pretender comprobar la presencia de analogías pues el vocabulario tuvo que ser reducido a la raíz de los términos con la técnica de stemming. Con los otros dos (Word2vec y FastText) sí vamos a hacer un estudio exploratorio en la presente sección de algunas analogías.

La experimentación con las analogías la hemos diseñado de la siguiente manera: cada experimento consta de 4 términos a,b,c y d de forma que se busca el grado de cumplimiento de a-b = c-d. Este grado de cumplimiento se mide mediante la similitud del coseno (al igual que hicimos en las secciones previas con la similitud de términos dos a dos).
Lo primero que hacemos es establecer dos experimentos de control. Esto son términos en los que no tiene sentido buscar ninguna analogía por lo que la similitud de los vectores (a-b) y (c-d) debe ser baja (recordamos que el máximo valor de similitud usando la similitud coseno es 1). Los experimentos de control corresponden a las dos primeras filas de las siguientes tablas correspondientes a Word2vec y FastText, respectivamente:

ANALOGÍA	**SIMILITUD DEL COSENO PARA LOS VECTORES RESULTANTES DE LA RESTA A CADA LADO DE LA ECUACIÓN**
español – camarero = hombre – rey (CONTROL)	-0.05
gato – rey = gen – proteína (CONTROL)	0.06
rey – hombre = reina - mujer	0.74
camarero – hombre = camarera - mujer	0.66
gato – macho = gata – hembra	0.45
español – España = alemán - Alemania	0.71
aragonés – Aragón = valenciano - Valencia	0.46
irlandés – Irlanda = francés - Francia	0.65
genoma – gen = proteoma – proteína	0.4
biblioteca – libro = hemeroteca - noticia	0.26
maldad – malo = bondad – bueno	0.69
bióloga – biología = ingeniera - ingeniería	0.57
filósofa – filosofía = historiadora – historia	0.39

Tabla 24: Analogías entre términos usando las representaciones vectoriales aportadas por Word2vec

ANALOGÍA	SIMILITUD DEL COSENO PARA LOS VECTORES RESULTANTES DE LA RESTA A CADA LADO DE LA ECUACIÓN
español – camarero = hombre – rey (CONTROL)	-0.02
gato – rey = gen – proteína (CONTROL)	-0.01
rey – hombre = reina - mujer	0.64
camarero – hombre = camarera - mujer	0.62
gato – macho = gata – hembra	0.59
español – España = alemán - Alemania	0.7
aragonés – Aragón = valenciano - Valencia	0.34
irlandés – Irlanda = francés - Francia	0.62
genoma – gen = proteoma – proteína	0.33
biblioteca – libro = hemeroteca - noticia	0.41
maldad – malo = bondad – bueno	0.62
bióloga – biología = ingeniera - ingeniería	0.55
filósofa – filosofía = historiadora – historia	0.52

Tabla 25: Analogías entre términos empleando las representaciones vectoriales obtenidas mediante FastText

Hemos comprobado con los ejemplos expuestos que las representaciones vectoriales obtenidas con ambos métodos codifican conceptos como “género”, ”lugar de procedencia”, “conjunto de” , “lugar destinado a”, sustantivación a partir de adjetivos o “profesión”.

También hemos tratado de representar algunas de las analogías para ambos métodos presentadas en las tablas previas haciendo uso del mismo procedimiento de PCA explicado previamente:

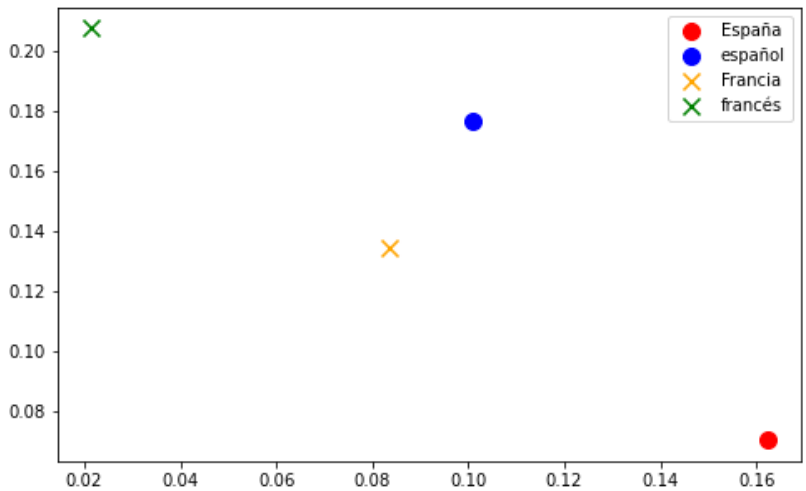

Ilustración 13: Visualización de la analogía español - España = francés - Francia usando Word2vec

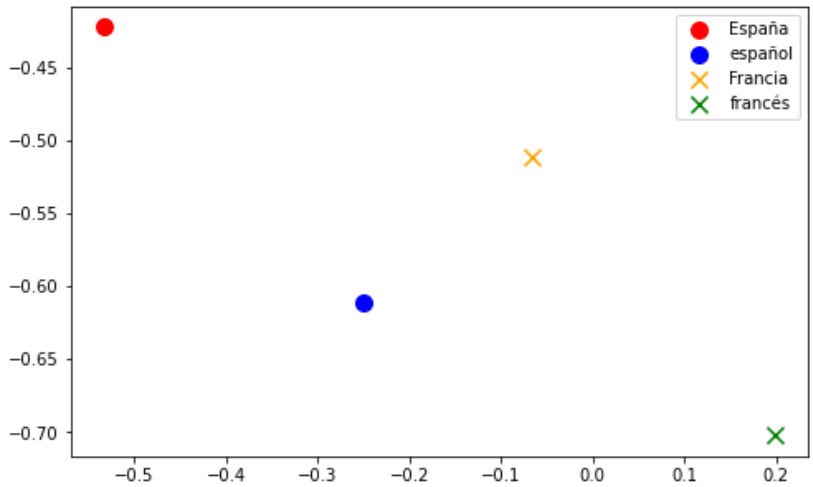

Ilustración 14: Visualización de la analogía español - España = francés - Francia usando FastText

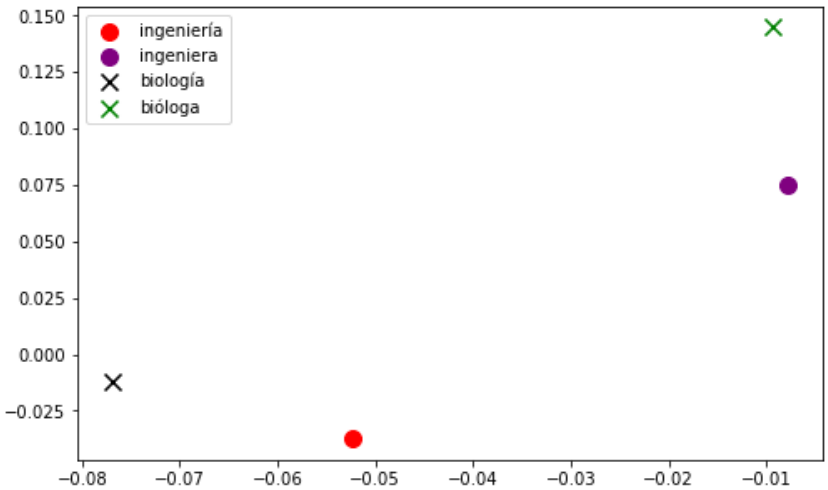

Ilustración 15: Visualización de la analogía ingeniera - ingeniería = bióloga - biología usando Word2vec

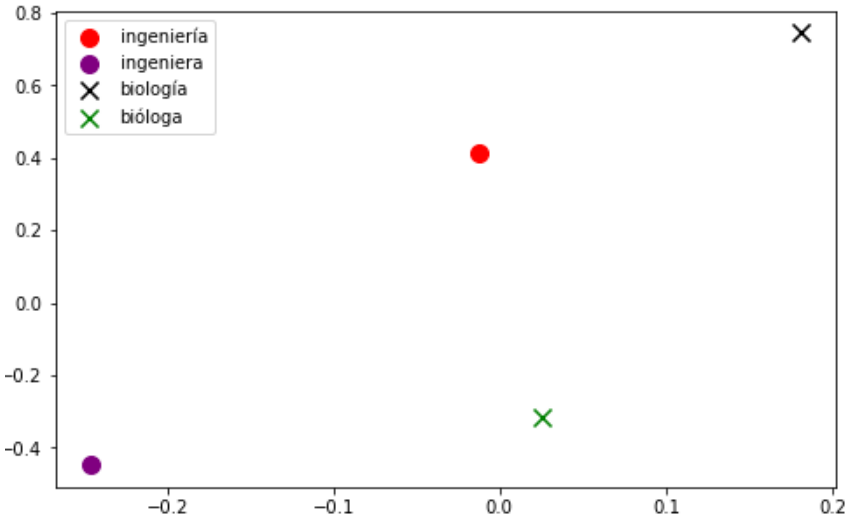

Ilustración 16: Visualización de la analogía ingeniera - ingeniería = bióloga - biología usando FastText

En las previas ilustraciones dedicadas a la visualización de analogías se puede observar que se tienen dos puntos en el espacio de características que representan términos distintos (por ejemplo “ingeniería” y “biología”). Al añadir a ambos un mismo concepto extra como por ejemplo “mujer dedicada a” se tiene que “ingeniería” + “mujer dedicada a” = “ingeniera”. Por otro lado, “biología” + “mujer dedicada a” = “bióloga”. Este concepto extra queda codificado como un vector, y gráficamente se presenta como una traslación.

5.CONCLUSIONES Y TRABAJO FUTURO

En general, hemos conseguido lograr porcentajes de acierto muy buenos sobre test (el mejor de 96.66%). Este resultado es bueno teniendo en cuenta que el conjunto de test es balanceado.

Se tiene que para todos los métodos y para todos los clasificadores la precisión es más alta que el recall (en algunos pocos casos vemos que es similar). Esto nos hace pensar que hay instancias "difíciles" de fake news, pero que las instancias de categoría "real" son más sencillas de detectar.

De los cuatro métodos de extracción de características probados el que peores resultados obtiene es GloVe. Seguramente este hecho se deba a que es el único de los tres que pertenecen a la categoría de representación vectorial de palabras que hemos entrenado nosotros mismos. Al tener una capacidad computacional bastante limitada, tuvimos que:

- Reducir el número de términos del vocabulario a considerar reduciendo palabras a su raíz mediante stemming lo cual hace perder semántica a los mismos. También eliminamos términos con frecuencia relativamente alta o baja (los umbrales fueron seleccionados mediante validación cruzada).
- Entrenar dichas representaciones vectoriales sobre un corpus "reducido". Se trata del mismo corpus empleado para entrenar los clasificadores que cuenta con 538 instancias de texto.

Para los métodos de representación vectorial de palabras, la agregación que peor funciona (detecta casi todas las instancias como noticias reales) es la suma probabilística. Las otras cuatro funcionan bien.

También, para los métodos de representación vectorial de palabras, hemos estudiado la presencia de analogías en las representaciones vectoriales, observando que se codifican conceptos abstractos en forma de direcciones vectoriales.

Asimismo, al hacer el estudio de similitud entre pares de términos de las 8 palabras escogidas como ejemplo observamos que por ejemplo el término "juez" es muy similar a "abogado" y muy distinto a "millones". O el término "asesinato" es muy similar a "muertos" pero muy distinto a "banco". En la experimentación con GloVe observamos que estas comparaciones por pares no muestran buenos resultados (en comparación con los otros dos métodos en los que hemos hecho las comparaciones).

La línea de trabajo futura (no la hemos llevado a cabo por falta de recursos computacionales) podría ser concatenar las características procedentes de distintos métodos (por ejemplo, glove + word2vec) y estudiar si podría mejorar el accuracy. Esta concatenación podría llevarse a cabo a nivel de palabra (para todas las palabras presentes en un texto, obtener su representación vectorial concatenando el vector que da un método y el que da otro) para posteriormente agregar todos. O también podría llevarse a cabo a nivel de texto (tras haber agregado todas las representaciones vectoriales con un método por una parte y con otro por otra parte, concatenar).

6. BIBLIOGRAFÍA

[1] **M. Alonso González, "Fake News: desinformación en la era de la sociedad de la información.," *Ámbitos. Rev. Int. Comun.*, no. 45, pp. 29–52, 2019.**

[2] **A. C. Vásquez, H. V. Huerta, J. P. Quispe, and A. M. Huayna, "Procesamiento de lenguaje natural robusto," *Rev. Ing. Sist. e Informática*, vol. 6, no. 2, pp. 45–54, 2009.**

[3] **G. (Gareth M. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning : with applications in R*. .**

[4] **M. N. Dr. S. Vijayarani, Ms. J. Ilamathi, "Preprocessing Techniques for Text Mining Preprocessing Techniques for Text Mining," *Int. J. Comput. Sci. Commun. Networks*, vol. 5, no. October 2014, pp. 7–16, 2015.**

[5] **Z. Qi, "The Text Classification of Theft Crime Based on TF-IDF and XGBoost Model," *Proc. 2020 IEEE Int. Conf. Artif. Intell. Comput. Appl. ICAICA 2020*, pp. 1241–1246, 2020.**

[6] **P. M. Brennan, J. J. M. Loan, N. Watson, P. M. Bhatt, and P. A. Bodkin, "Pre-operative obesity does not predict poorer symptom control and quality of life after lumbar disc surgery," *Br. J. Neurosurg.*, vol. 31, no. 6, pp. 682–687, 2017.**

[7] **T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st Int. Conf. Learn. Represent. ICLR 2013 - Work. Track Proc.*, pp. 1–12, 2013.**

[8] **P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, 2017.**

[9] **T. F. I. N. D. E. Máster, "Fabricio Andrés Zules Acosta," 2019.**