UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Detección de Noticias Falsas en español basado en aprendizaje no supervisado mediante el uso de agrupamiento espacial basado en densidad de aplicaciones con ruido (DBSCAN)

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

ACTUARIO

PRESENTA:
ÁNGEL EDUARDO CADENA BAUTISTA

DIRECTOR DE TESIS:
DR. PAUL ERICK MÉNDEZ MONROY
UNIDAD ACÁDEMICA DEL IIMAS EN EL ESTADO DE YUCATÁN

Ciudad Universitaria, Ciudad de México.
Octubre, 2023.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Universidad Nacional Autónoma de México
Este trabajo fue apoyado por los proyectos UNAM-PAPIIT IN105623
Agradezco el apoyo recibido a través de la beca UNAM-PAPIIT IA102620.

Capítulo 1

Introducción

Gracias a la gran infraestructura que se ha desarrollado en internet la diseminación y generación de información puede propagarse a gran escala y con la velocidad de alcanzar a un gran número de receptores. [2]

Esto es un arma de doble filo dada la enorme velocidad de propagación, la gran cantidad de información y la masa de receptores a los que esta información puede llegar, la diseminación de noticias falsas se vuelve un problema difícil de contener. Llevando esta propuesta a responder la pregunta ¿Es posible caracterizar y detectar una noticia falsa?.

Una noticia falsa es aquella que deliberadamente busca desinformar y engañar a una cantidad amplia de receptores [3]. Las noticias falsas generan múltiples consecuencias como daños a la reputación, sesgos políticos, pánico, desprestigio de personajes públicos u organizaciones.

El objetivo del trabajo se centra en abordar la problemática de detectar noticias falsas en la enorme cantidad de noticias que circulan en internet. Las preguntas que guían este trabajo son las siguientes, ¿Es posible clasificar mediante el uso de aprendizaje no supervisado noticias en falsas y verdaderas? y ¿Cómo se compara el rendimiento de este método de aprendizaje no supervisado con métodos al aprendizaje supervisado?.

Con el objetivo de contestar las preguntas anteriores se propone utilizar un proceso de aprendizaje no supervisado para la generación de un modelo para detectar una noticia falsa sin el conocimiento a priori de su clasificación. La metodología propuesta se centra en la detección de noticias falsas en español, una tarea que empieza a desarrollarse en el quehacer científico. El corpus que se utilizará para el entrenamiento y validación del modelo contempla 971 noticias de las cuales 491 son noti-

cias verdaderas y 480 noticias falsas.

El contenido del presente trabajo esta organizado como sigue:

En el capítulo 1 se hace una introducción breve de qué son las noticias falsas y que implicaciones tienen así como una revisión sobre el estado del arte de la clasificación de noticias falsas utilizando métodos de aprendizaje supervisado y no supervisado. En el capítulo 2 se presentan los antecedentes para la comprensión de la metodología propuesta del resto de la tesis.

El capítulo 3 presenta la metodología propuesta, iniciando con un corpus en español para entrenar e inferir el modelo no supervisado, a este se le realiza un preprocesamiento de las noticias donde se utiliza solo el cuerpo de las noticias representándolo en una bolsa de palabras, eliminando las palabras de paro, lematizando las palabras dejando la palabra raíz y quitando los sinónimos. Lo siguiente es realizar una reducción de dimensión de la bolsa de palabras resultante a través de Análisis de Componentes Principales (*PCA* por sus siglas en inglés) contemplando solo un número reducido de componentes. Finalmente, con la bolsa de palabras reducida se realiza el entrenamiento no supervisado con un algoritmo de agrupamiento espacial basado en densidad de aplicaciones con ruido (*DBSCAN* por sus siglas en ingles) que realiza una clasificación en dos grupos, noticia falsa o noticia verdadera. En el capítulo 4 y 5 se presentan los resultados así como las conclusiones del presente.

Capítulo 2

Antecedentes

2.1. Noticias Falsas, ¿Qué son?

Las redes sociales han permitido que el consumo de información sea de fácil acceso, de bajo costo y de rápida diseminación [3]. Esto puede ser efectivo y contraproducente al mismo tiempo ya que es posible que la información que se consulte y se disemine sea una noticia falsa es decir una noticia de baja calidad cuya intención es desinformar y diseminar información falsa. Podemos entrar en el debate de para quien resulta falsa dicha noticia, pero eso va más allá del alcance de este trabajo. Para nosotros una noticia falsa será una noticia cuyo contenido no puede ser verificable, es decir, no podemos encontrar información que refute o acepte la veracidad de la noticias en otros medios.
En cuanto a las consecuencias de las noticias falsas, como lo menciona Pérez en [4], "son palpables en discursos polarizados y la radicalización de las opiniones, la desinformación de la población e indirectamente en la menor confianza hacia la actividad periodística y los medios de comunicación producida por una doble dinámica: los ataques a los medios de comunicación y periodistas por parte de líderes políticos y la espiral de escepticismo que genera tanta desinformación hacia el crédito de la profesión periodística".

En cuanto a propuestas para la detección automática de noticias falsas existen múltiples soluciones. Están las basadas en conocimiento, algunas son mencionadas en Shu et. al. en [3], que incluyen la verificación hecha por humanos expertos que se dedican a recolectar documentos y datos para verificar la veracidad de dicha noticia. El *crowdsourcing*, que se basa

en recolectar información a través de contribuciones de un gran grupo de personas y se conoce como "el conocimiento de la parvada" para verificar y anotar si una noticia es falsa o verdadera. Y por último los métodos computacionales que se basan en tener un *corpus* previamente anotado y, a través de diversos algoritmos y la caracterización de las noticias, es posible realizar un entrenamiento y validar si dichos métodos son buenos para clasificar.
Para esta última solución podemos señalar diversas formas para caracterizar las noticias, vectores de características léxicas, conteo de palabras, etiquetado POS/partes de la oración, puntuación, uso de alguna forma verbal etc. Así como también diferentes algoritmos de clasificación como regresiones lineales, máquinas de soporte vectorial, bayes ingenuo, árboles de decisión, árboles aleatorios, vecinos más cercanos, entre otros. En cuanto a las métricas más comunes para medir el rendimiento de dichos modelos están la exactitud, respondiendo a la pregunta ¿en promedio cuantas veces la clasificación es correcta?, la precisión, que responde la pregunta ¿Cuantas veces la clasificación de la clase es correcta?, el *recall*, que responde a ¿Cuántas veces la clasificación identifica lo correcto? y la F1, que se define como la media armónica entre precisión y *recall*. En la siguiente sección se ahondará más en estás.

2.2. Estado del arte

Dentro de la literatura relacionada para la detección de noticias falsas encontramos diferentes artículos con diferentes acercamientos a la solución, se presentan los que fueron elegidos como más relevantes para el trabajo de tesis.

Shu et. al. en [3] hacen un escrito explorando el problema de las noticias falsas basándose en dos fases, la caracterización y la detección. En la primera fase revisan las características que se pueden extraer de las noticias falsas, tanto en medios tradicionales como en redes sociales. Dichas características pueden estar basadas en el contenido de la noticia o en el contenido del contexto social de la noticia, es decir, en como la noticia está impactando de acuerdo al usuario, el camino que sigue la noticia y las interacciones que provoca. En cuanto a la fase de detección

hacen una revisión de los métodos más utilizados para resolver el problema, métodos basados en conocimiento (recaen en hechos conocidos), basados en estilo (analizando el estilo de escritura), basados en el contexto (analizando la propagación en redes sociales). Además mencionan cuatro conjuntos de datos y las métricas de evaluación existentes, así como líneas de investigación relacionadas con la detección de noticias falsas.

En [5] Papalexakis y Hosseinimotlagh plantean una solución mediante la modelación tensorial del problema para identificar noticias falsas, donde buscan capturar las relaciones latentes entre los artículos y los términos, así como las relaciones espaciales/contextuales entre términos, buscando así, explotar el potencial del contenido. Explícitamente modelan el contexto de las palabras en un documento capturando la vecindad de cada palabra. En particular, modelan el conjunto de documentos, o corpus, como un tensor de tercer orden el cual simultáneamente modela la relación entre el articulo y el término, así como la relación espacial/-contextual entre términos. Así explotando ambos aspectos del corpus, y en particular la relación espacial ente palabras, es un factor determinante para identificar grupos coherentes de diferentes tipos de noticias falsas. Las contribuciones que presentan en su artículo son el explotar la información de orden mayor de la información del contexto. Modelan el corpus como un tensor de tres modos (artículo, término, término) que captura las relaciones espaciales entre términos y la relación artículo-término. Subsecuentemente usan una descomposición CP/PARAFAC para encontrar aquellos artículos que caen dentro de una categoría coherente de noticias falsas. Por otro lado, introducen un método de conjuntos que apalanca las múltiples descomposiciones del tensor (artículo, término, término), que permite refinar el descubrimiento de grupos por la descomposición y producir una categorización de los artículos. Usando un conjunto de datos reales el método propuesto produce grupos latentes con 65 % de coherencia en general y más del 80 % para la mayoría de las categorías, mientras que otros métodos no supervisados, en particular la factorización de matrices no negativas y la descomposición en valores singulares, obtienen hasta un 50 % en promedio de las 30 primeras noticias para cada factor.

Por otro lado Majbouri et. al. en [6] proponen una mejora a la detección de noticias falsas utilizando un método de selección de características con la integración de un clustering haciendo uso de k-medias y máquinas de soporte vectorial en 4 pasos. Primero, calculan las similitudes entre todas las características. En segundo lugar, las características son divididas en varios clústers usando k-medias de acuerdo a la similitud entre características. Después se elige para cada clúster un conjunto final de características basado en la de las características agrupadas en el segundo paso. Finalmente, después de especificar el conjunto final de características, se crea un conjunto de datos de dimensión reducida utilizando estos conjuntos de características y en la siguiente fase, se usa un clasificador de maquinas de soporte vectorial para predecir las noticias falsas. El rendimiento de la detección se mejoró en dos aspectos. En primer lugar, el proceso de detección en tiempo de ejecución decreció y, por otro lado, la precisión de la clasificación incremento por la eliminación de características redundantes y la reducción de la dimensión del conjunto de datos.

En [7] Posadas et. al. mencionan sus aportaciones que incluyen el desarrollo del primer corpus en español de noticias falsas y verdaderas extraídas de sitos web recopiladas entre enero y julio de 2018 y el proceso de anotación para la clasificación de las noticias y experimentos para la detección automática utilizando aprendizaje supervisado. Para el proceso de identificación utilizan tres diferentes representaciones de las noticias, la bolsa de palabras, n-gramas de caracteres y por último n-gramas de etiquetas POS/ (partes de la oración). En cuanto a los modelos utilizados para la clasificación automática utilizan máquinas de soporte vectorial, regresión logística, arboles aleatorios y *boosting*. Los mejores resultados obtenidos para este conjunto de datos los obtuvieron para la combinación de características entre la bolsa de palabras y los n-gramas de etiquetas POS/partes de la oración para un clasificador de bosques aleatorios obteniendo un 76.94 % de exactitud.

Capítulo 3

Procesamiento de Lenguaje Natural

3.1. Lenguaje natural

Para entender el análisis de textos y el procesamiento de lenguaje natural, debemos entender que es lo "natural" del lenguaje. En términos sencillos, el lenguaje natural es aquel que ha sido desarrollado y ha evolucionado por los humanos a través del uso natural y la comunicación, en lugar de aquellos construidos y creados artificialmente, como los lenguajes computacionales de programación.

Los lenguajes humanos como el Español, Inglés, Portugués, Sánscrito son lenguajes naturales. El lenguaje natural es aquel que puede ser comunicado en diferentes formas, incluyendo la escritura, el habla e incluso mediante señas.

Dimensiones del lenguaje natural

La lingüística se encarga del estudio e investigación del lenguaje, incluyendo las formas, la sintaxis del lenguaje, su significado, la semántica generada por el uso del lenguaje y además del contexto de uso. Se han creado diferentes disciplinas con objetivos específicos que ayudan en el entendimiento del lenguaje como mencionan en el capitulo introductorio de [8].

- Fonética: Es el estudio de las propiedades acústicas de los sonidos que son producidos por el ser humano a la hora de comunicarse. La unidad mínima del discurso humano es el fonema.
- Fonología: Es aquella que estudia los patrones de sonido que son interpretados por la mente humana y que se usa para distinguir

entre fonemas aquellos que son mas significantes. La fonología se complementa con los niveles morfológico, sintáctico y semántico.

- Morfología: El morfema es la unidad mínima del lenguaje que tiene un significado distintivo. Incluye las palabras, prefijos, sufijos, entre otras cosas que tienen su propio significado. La morfología explica la estructura interna de las palabras y el proceso de formación de palabras en un lenguaje.
- Sintaxis: Es el estudio de las reglas y principios que describen la estructura de las oraciones, frases, palabras. Esto incluye el cómo se combinan las palabras para formar frases y oraciones. El orden sintáctico en el cual se usan las palabras en una frase u oración importa ya que diferente orden puede cambiar totalmente el significado.
- Semántica: Es la disciplina que estudia el significado que se atribuye a las palabras. Esto es la relación del lenguaje con el mundo.
- Pragmática: Estudia cómo el contexto (escenario, audiencia, conocimiento previo, relaciones interpersonales, etcétera) puede influir y afectar el significado de una expresión. Esto incluye el tratar de inferir si hay un significado oculto o indirecto en la comunicación.
- Discurso: Analiza el lenguaje y el intercambio de información a través de las oraciones que se dan entre seres humanos. Estas conversaciones pueden ser habladas, escritas o incluso mediante señas.

A pesar de que estas son las áreas principales de estudio e investigación, la lingüística es un campo de investigación y estudio enorme con un alcance mucho más grande de lo que acabamos de mencionar. De cualquier manera, la sintaxis del lenguaje y la semántica son algunos de los conceptos más importantes que frecuentemente forman las bases para el procesamiento del lenguaje natural. Algunos de los problemas que encontramos en el procesamiento del lenguaje natural son la ambigüedad en la cual podemos destacar la ambigüedad sintáctica, la semántica y la pragmática [9] [8]. Otra es la multiplicidad de variantes, existen alrededor de 7000 lenguas en el mundo, hablando de México existen 364 variantes de lenguas. Otro problema es la evolución, el lenguaje es cambiante y no estático por lo que este tiende a cambiar. Otros problemas son la oscuridad, el uso slang, entre otros [7].

Conceptos básicos del lenguaje

Algunos de los conceptos básicos para el entendimiento y procesamiento del lenguaje son los lemas, los homónimos, los homófonos, la polisemia, los sinónimos, los antónimos, los hipónimos e hiperónimos, entre otros [10].

Lemas. Un lema es conocido como la forma canónica de un conjunto de palabras. El lema es usualmente la forma base de un conjunto de palabras, conocido como lexema en este contexto.

Homónimos y homófonos. Los homónimos son definidos como las palabras que se escriben o pronuncian igual pero tienen distintos significados. Los homófonos son palabras que se pronuncian de la misma manera pero tienen diferente significado y pueden, o no, escribirse de la misma manera.

Polisemia. La polisemia se presenta cuando una palabra que tienen la misma forma escrita u ortográfica y un muy diferente significado. A pesar de que puede confundirse con un homónimo, la diferencia subjetiva se encuentra que la polisemia depende del contexto, dado que estas palabras se complementan con las demás dentro del contexto.

Sinónimos y antónimos. Los sinónimos son palabras que se escriben y se pronuncian diferente, pero tienen el mismo significado en varios o en todos los contextos. Si dos palabras son sinónimos, éstas pueden ser sustituidas entre ellas sin que esto represente una pérdida de información pues tienen el mismo significado, en varios de los contextos. Sin embargo, esto puede presentar un problema al identificar aquellos sinónimos que no tienen el mismo sentido o significado en un contexto en específico.

Los antónimos son parejas de palabras que definen un sentido y significado completamente opuestas entre ellas. Los antónimos se clasifican en tres grandes grupos: graduales, complementarios y recíprocos. Los antónimos graduales son aquellos en los cuales ambas palabras se oponen de manera gradual, por ejemplo, frío y caliente o negro y blanco. Los antónimos complementarios son pares cuyo significado de uno es completamente lo contrario del otro o uno es la negación del otro, por ejemplo, legal e ilegal o dividir y unir. Finalmente los antónimos recíprocos designan una relación entre ellos y el antónimo surge en el contexto de la relación, por ejemplo, doctor y paciente o comprar y vender.

Hipónimos e hiperónimos. Los hipónimos son palabras que perte-

necen a una subclase de otra palabra. Los hipónimos son generalmente palabras con un sentido y contexto especifico relacionado con la palabra que es su superclase. Los hiperónimos son las palabras que actúan como superclase de los hipónimos y su significado es más bien general en comparación a los hipónimos. A manera de ejemplo la palabra fruta, que es un hiperónimos, y las palabras, mango, naranja y pera que serían posibles hipónimos.

3.2. Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural es un campo de las ciencias computacionales, la ingeniería y la inteligencia artificial con raíces en la lingüística computacional. Su principal objetivo es construir aplicaciones y sistemas que permitan la interacción entre máquinas y el lenguaje humano. La lingüística computacional no es más que la combinación entre la lingüística y las ciencias de la computación, es el estudio del lenguaje desde una perspectiva computacional, que busca crear modelos computacionales para distintos tipos de fenómenos lingüísticos, estos modelos pueden ser construidos manualmente, basados en conocimiento, o mediante aproximaciones empíricas, aquellos basados en datos. La lingüística computacional [9] pretende modelar el lenguaje de tal manera que este sea entendible para las computadoras, por lo que no solo se analiza el uso del lenguaje y el comportamiento humano, sino también se aplican métodos que formulen hipótesis y posterior verificación automática utilizando los datos lingüísticos, los corpus. El principal objetivo del procesamiento del lenguaje natural es permitir la comprensión, interpretación y generación de texto de las computadoras y que nos ayudarán con distintos recursos para el manejo de información y la generación de conocimiento. El procesamiento de lenguaje natural abarca una variedad de tareas y aplicaciones [8]:

- Lenguaje hablado
    - Reconocimiento de voz: Consiste en traducir el lenguaje hablado a texto.
    - Síntesis de habla: Generación de habla producida artificialmente a partir de textos o datos.

- Lenguaje escrito
  - Análisis de texto: Extrae información de alta calidad del texto.
  - Generación de texto: Generación de un texto producidos artificialmente a partir de otros textos o datos.

El reconocimiento de voz y el análisis de texto, forman parte de lo que se conoce como el entendimiento de lenguaje natural, mientras que la síntesis de habla y la generación de texto forman la generación de lenguaje natural.

3.3. Minería de Textos

La minería de textos es un campo de la inteligencia artificial y la lingüística computacional que intenta resolver la crisis de la sobrecarga de información buscando transformar el texto en información estructurada y significativa, esto a través de descubrir patrones, información relevante y conocimiento útil en grandes cantidades de datos textuales no estructurados.

A manera análoga de la minería de datos, la minería de textos busca extraer información útil de las fuentes de datos a través de la identificación y exploración de patrones interesantes. En la minería de textos las fuentes de datos son colecciones de documentos y los patrones interesantes no se encuentran entre los registros de bases de datos formalizadas, estos se encuentran en la información textual contenida en los documentos en estas colecciones.

La información contenida en los textos se considera como datos no estructurados, pero usualmente pertenecen a un lenguaje específico y tienen reglas sintácticas y semánticas específicas. Cualquier trozo de texto -una palabra, una oración o un documento- la mayoría del tiempo se relaciona de nuevo al lenguaje natural. Los sistemas de minería de textos se componen de una serie de tareas importantes de preprocesamiento las cuales tienen como objetivo identificar y extraer características que representen a los documentos contenidos en la colección. Estas operaciones de preprocesamiento son responsables de transformar los datos textuales contenidos en los documentos, que no tienen estructura, en datos medianamente estructurados.

La minería de textos explota técnicas y metodologías de las áreas de recuperación de información, extracción de la información y lingüística computacional.

El análisis de textos es la metodología y los procesos utilizados que derivan en conocimiento e información de calidad en los datos que se encuentran en texto. Estas técnicas utilizan procesamiento del lenguaje natural, recuperación de información así como técnicas de aprendizaje computacional para convertir los datos no estructurados del texto en algo de una forma estructurada que ayuden a encontrar patrones e información interesante en ellos. Estas técnicas buscan modelar y extraer información de los textos para análisis, que pueden ser exploratorios descriptivos o predictivos. Algunas de las técnicas en el análisis de textos como lo menciona Sarkar en [10]:

- Clasificación de textos
- Agrupación de textos
- Resumen de textos
- Análisis de similitud y relación textual
- Reconocimiento y extracción de entidades

Las principales aplicaciones en el análisis de textos son las listadas a continuación, en el presente trabajo nos enfocaremos en la detección de noticias falsas.

- Detección de Noticias Falsas
- Detección de *spam*
- Análisis de sentimientos
- Análisis y monitoreo de redes sociales
- Asistentes virtuales

Por definición para el análisis de textos se define un corpus, el corpus de un texto es una colección de documentos de un medio en particular de una o varias fuentes, por ejemplo un conjunto de noticias, un conjunto de libros, un conjunto de artículos, un conjunto de tweets, conforman un

corpus. Estos documentos pueden contener párrafos, oraciones, palabras, caracteres especiales, emoticones, entre otras estructuras.

Una vez establecido el corpus, se debe tomar la decisión de realizar tareas de preprocesamiento para la limpieza y adecuación de los textos contenidos en el corpus antes de aplicar técnicas de análisis de textos, con el fin de remover términos y datos que presenten información redundante, que no aporten información, mal escritos, etcétera.

3.3.1. Preprocesamiento

Debido a que las computadoras no pueden entender caracteres o palabras hay que convertir los datos textuales a un formato el cual permita que la computadora procese y analice la información. Dicha conversión busca estructurar y almacenar los contenidos extraídos en una representación intermedia como un modelo vectorial, un modelo relacional, una lista de palabras, entre otras.

Una de las maneras más comunes de obtener estas representaciones es mediante la caracterización de los textos, esto es extraer la información que se encuentra en el texto, dicha extracción de características pueden hacerse a diferentes granularidades que generan los llamados *tokens* [11]. Entre las características que se pueden extraer del texto están las oraciones, las frases, las palabras, los n-gramas, los caracteres, las categorías gramaticales, la estructura gramatical, agrupación de palabras de significado similar, entre otras [11].

Las palabras son la clase de tokens más empleado, a estas palabras se les puede aplicar diferentes tareas de preprocesamiento. Por ejemplo, tomar la decisión para trabajar en minúsculas, mayúsculas o tal y como vienen en el corpus, si se toman en cuenta los caracteres especiales, los signos de puntuación, los dígitos, palabras de paro o mejor conocidas como *stopwords*, raíces o lemas, entre otras.

El flujo de preprocesamiento de un corpus más común consiste en las tareas de transformar el texto a minúsculas, eliminar los dígitos, caracteres especiales, la segmentación del texto en tokens, eliminación de palabras cerradas, lematización, búsqueda de similitud, radicalización y finalmente el modelo de representación, esto se conoce como limpieza y adecuación del corpus [10].

Tokenización

Se conoce como tokenización al proceso de separar un texto en tokens, que son los componentes textuales mínimos e independientes, semántica y sintácticamente definidos. Un párrafo o un documento de texto puede ser fragmentado en componentes como oraciones, clausulas, frases, palabras o caracteres.

Las técnicas más populares separan un corpus de texto en oraciones y cada una de las oraciones en palabras, caracteres, subpalabras, etcétera.

La tokenización en oraciones es el proceso de separar un corpus de texto en oraciones, se conoce también como segmentación en oraciones. Existen diversas maneras de realizar esta tokenización. Las técnicas más básicas buscan delimitadores específicos entre oraciones, como el punto (.), o un carácter de nueva línea (\n), algunas veces se utiliza como delimitador el punto y coma (;) entre otros signos de puntuación.

La tokenización en palabras esencialmente es el proceso de separar las oraciones en listas de palabras que pueden ser utilizadas para reconstruir la oración. Esta es muy importante en varios procesos, especialmente en la limpieza y normalización del texto, pues permite el análisis de las palabras de manera individual.

La manera más común de procesar un corpus de texto es a nivel de tokens. Las técnicas que podemos aplicar a los tokens, dependiendo la granularidad y las características de los mismos, en el caso de las palabras, podemos buscar sus lemas, raíces e incluso si pertenecen a un conjunto definido de palabras de paro. El objetivo de estás es la reducción del tamaño del vocabulario, evitar información innecesaria, la extracción de características, la normalización u homogeneización de los tokens.

Palabras de paro

Las palabras de paro, conocidas en inglés como *stopwords* o *stop words*, son palabras que tienen poco o nulo significado, esto en cuanto a la información que pueden aportar al análisis. Usualmente son removidas durante el procesamiento de texto para retener aquellas palabras que tienen mayor significado y contexto. Las palabras de paro son usualmente palabras que ocurren con una alta frecuencia a lo largo de un corpus de texto una vez que se hace un resumen de la frecuencia de los tokens.

Palabras como *a*, *en*, *el*, *la*, *los*, entre otras, son palabras de paro. No existe una lista universal de palabras de paro. Cada dominio o lenguaje tienen su propio conjunto de palabras de paro.

Lematización

La lematización es el proceso en el cual se remueven los afijos de las palabras para quedarnos con el lema de la palabra. Esto es relacionar una palabra flexionada o derivada con su forma canónica o lema. Y un lema no es mas que la forma que tienen las palabras cuando se buscan en el diccionario. Por ejemplo, canto, cantas, canta, cantamos, cantan son distintas formas de un mismo verbo, en este caso cantar. Otro ejemplo puede ser que niño, niña, niñita, niños, niñotes, entre otras, son distintas formas del vocablo niño.

Búsqueda de similitud

Este proceso consiste en la búsqueda de los sinónimos de las palabras dentro del corpus, una vez recuperados los sinónimos se toma la decisión de cambiar aquellas palabras que se encuentran dentro del conjunto de una palabra, por el sinónimo que se encontró primero, con la finalidad de reducir el número de palabras en el corpus. Por ejemplo si dentro de nuestro corpus apareciera la palabra *alba* y el conjunto de sinónimos devuelto es *amanecer, aurora, madrugada*, si alguna de las palabras del conjunto de sinónimos esta dentro del corpus, se tomaría la decisión de cambiar a la palabra *alba* como palabra principal y así reducir el tamaño del vocabulario. Esto puede lograrse de distintas maneras, mediante la búsqueda de los sinónimos en una base de datos de palabras y sinónimos, o recuperar los sinónimos de las palabras de una fuente externa.

Radicalización

La radicalización, o *stemming* en inglés, es el procedimiento de convertir palabras en raíces. Estas raíces son la parte invariable de las palabras. A estas unidades se le añaden prefijos o sufijos que cambian o crean una palabra cuando se juntan. Las raíces son conocidas como la forma base de una palabra, pero estas no necesariamente son palabras de un idioma. Por ejemplo el algoritmo de radicalización puede decir que la raíz

de *amamos* no es *am* sino *amam*. Y que esto a su vez presenta problemas porque pueden existir palabras con la misma forma en su raíz pero significados completamente diferentes.

3.4. Modelos de representación textual

Como lo mencionamos anteriormente, una vez preprocesado el corpus necesitamos generar una representación que entienda la computadora, en este caso un modelo vectorial, entre las técnicas más utilizadas para esta transformación encontramos la bolsa de palabras, la matriz termino-documento, la matriz inversa termino-documento, entre otras [12].

El método de la bolsa de palabras es una representación en la cual se genera un conjunto que contiene todas las palabras individuales sin orden de los textos, algunas de estas palabras son relevantes para el contenido temático, pero otras palabras pueden aportar algo o nada de información. Este método nos ayuda a describir los textos tomando en cuenta la frecuencia de uso y la distribución de las palabras.

La matriz término-documento es una representación que se ayuda de la bolsa de palabras para formar el conjunto de vocabulario del corpus una vez normalizados, o no, los textos. Cada frecuencia de token individual se trata como una característica y el vector de todas las frecuencias de tokens para un documento dado se considera una muestra multivariada.

La matriz inversa termino-documento determina la frecuencia relativa de las palabras en un documento específico comparado con la proporción inversa de esa palabra en toda la colección, existen cuatro tipos de pesos en esta representación:

- Peso grande: Ocurre cuando un término tiene una alta frecuencia de aparición pero no se encuentra en la mayoría de los documentos analizados.
- Peso mediano: El número de apariciones tanto dentro de un documento como en los archivos que conforman el corpus no es baja ni alta.
- Peso bajo: Su aparición sucede cuando la frecuencia del término es baja y la construcción se encuentra en la mayoría de los documentos.

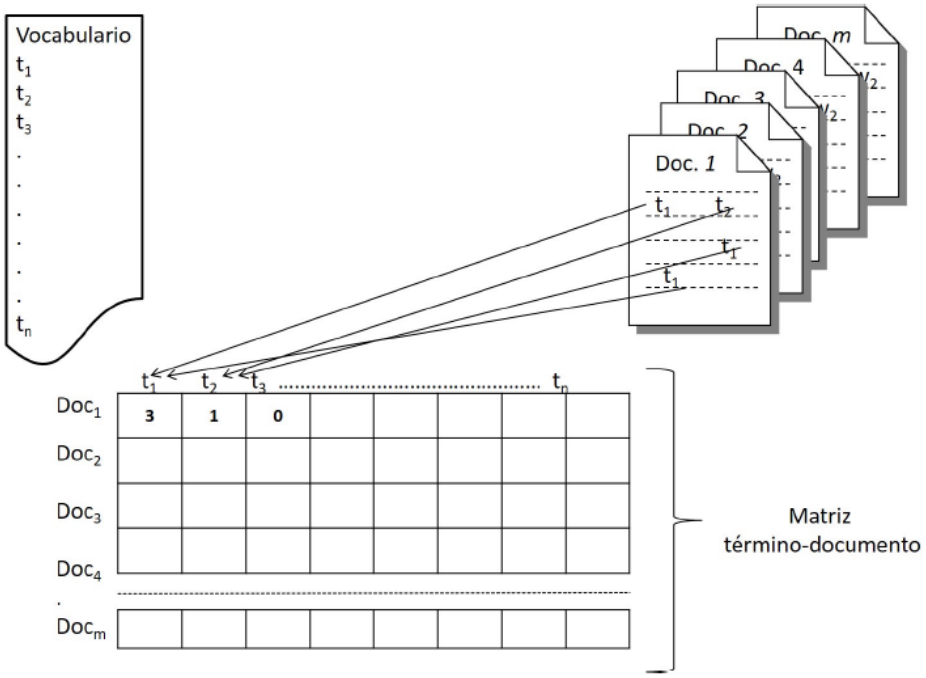

Figura 3.1: Visualización de la construcción de la matriz termino-documento

- Peso nulo: Acontece cuando la frecuencia de aparición de un término dentro de un documento es nula o cuando el término aparece en cada uno de los documentos que pertenecen al corpus analizado.

3.5. Herramientas de Procesamiento de Lenguaje Natural

Existen distintas herramientas que nos ayudan con el preprocesamiento, extracción de características y análisis de texto. En python existen distintas librerías que nos ayudan tanto para la extracción de características textuales así como para el procesamiento de lenguaje natural, entre las más utilizadas se encuentran NLTK, Scikit-learn y Spacy.

3.5.1. NLTK

NLTK [13] es una plataforma que permite el desarrollo de programas en Python para trabajar datos de lenguaje humano. Provee interfaces que permiten el acceso a diferentes corpus y recursos léxicos además de un conjunto de librerias para el preprocesamiento, clasificacion, toke-

nización, stemming entre otras tareas usuales en el procesamiento del lenguaje natural.

Palabras de paro

La clase *stopwords* de NLTK contiene el método words que devuelve el conjunto de las palabras de paro en un idioma predeterminado usando la siguiente sintaxis. Para el caso en español.

```
In [1]: from nltk.corpus import stopwords
In [2]: stopwords.words("spanish")
Out[3]: ["de", "la", "que", "el", "en", "y", "a", ...]
```

Tokenización

Para tokenizar un texto *NLTK* provee una clase con diversos métodos para realizar la tokenización de un texto. Entre los más utilizados encontramos el siguiente método, de la clase *tokenize*, que toma como entrada un texto y devuelve una lista con los tokens separando el texto con base en la puntuación y los espacios en blanco.

```
In [1]: from nltk.tokenize import word_tokenize
In [2]: s = "La manzana es roja"
In [3]: word_tokenize(s)
Out[4]: ["la", "manzana", "es", "roja"]
```

3.5.2. Scikit-learn

Scikit-learn [14] es una libreria de python para aprendizaje computacional, contiene distintos algoritmos de clasificación, regresión, agrupamiento entre otros. Entre sus clases encontramos la extracción de características de texto que reúne clases que permiten la construcción de los modelos de representación textual de documentos de texto. Aquí podemos encontrar las clases *CountVectorizer* y *TfidfVectorizer*.

Countvectorizer

Esta clase convierte una colección de documentos a una matriz de conteo de los tokens, es decir las columnas son los tokens contenidos en los textos, los renglones son los documentos de la colección y las celdas contienen la frecuencia del token en el documento. El conteo de la frecuencia de los tokens puede ser cambiado a la ocurrencia del token en el documento con el parámetro *binary*. Cuando este se establece como

verdadero, todos los conteos mayores a cero son fijados a 1, por defecto el valor del parámetro es falso. La sintaxis para el funcionamiento de la clase es la siguiente.

```
In [1]: from sklearn.feature_extraction.text import
   CountVectorizer
In [2]: corpus = [
...     'Este es el documento uno.',
...     'Este documento es el segundo documento.',
...     'Y este es el tercero.',
...     'Â¿Es este el tercer documento?',
... ]
In [3]: vectorizer = CountVectorizer()
In [4]: X = vectorizer.fit_transform(corpus)
In [5]: vectorizer.get_feature_names()
Out[6]: ['documento', 'el', 'es', 'este', 'segundo', 'tercer',
...      'tercero', 'uno']
In [7]: print(X.toarray())
Out[8]: [[1 1 1 1 0 0 0 1]
...      [2 1 1 1 1 0 0 0]
...      [0 1 1 1 0 0 1 0]
...      [1 1 1 1 0 1 0 0]]
```

TfidfVectorizer

Esta clase convierte una colección de documentos a la matriz inversa termino-documento. Esto es que las columnas son los tokens contenidos en los textos, los renglones son los documentos de la colección y las celdas contienen la frecuencia del token en el documento sobre la proporción del número de documentos que contienen el token en toda la colección. El parámetro *binary* al ser establecido en verdadero se asegura que aquellos conteos diferentes de 0 sean establecidos a 1, por defecto el valor del parámetro es falso. La sintaxis para el funcionamiento de la clase es la siguiente.

```
In [1]: from sklearn.feature_extraction.text import
   TfidfVectorizer
In [2]: corpus = [
...     'Este es el documento uno.',
...     'Este documento es el segundo documento.',
...     'Y este es el tercero.',
...     'Â¿Es este el tercer documento?',
... ]
In [3]: vectorizer = TfidfVectorizer()
In [4]: X = vectorizer.fit_transform(corpus)
```

```
In [5]: vectorizer.get_feature_names()
Out[6]: ['documento', 'el', 'es', 'este', 'segundo', 'tercer',
...      'tercero', 'uno']
In [7]: print(X.toarray())
Out[8]: [[0.4279 0.3498 0.3498 0.3498 0.0 0.0 0.0 0.6704]
         [0.6876 0.2810 0.2810 0.2810 0.5386 0.0 0.0 0.0]
         [0.0    0.3871 0.3871 0.3871 0.0 0.0 0.7418 0.0]
         [0.4279 0.3498 0.3498 0.3498 0.0 0.6704 0.0 0.0]]
```

3.5.3. Spacy

Spacy [15] es una librería gratuita y de código abierto para el procesamiento de lenguaje natural en python está diseñada específicamente para uso en producción y ayuda a construir aplicaciones que procesen y entiendan grandes volúmenes de texto.

Entre sus clases más utilizadas encontramos *Lemmatizer*, esta asigna a los tokens su forma base utilizando reglas basadas en etiquetas o tablas. En el caso del español utiliza una tabla donde busca el token y devuelve el lema. Una forma básica para utilizar *Lemmatizer* en español con Spacy es la siguiente:

```
In [1]: import spacy
In [2]: nlp = spacy.load("es_core_news_sm")
In [3]: doc = nlp("El kilo de manzanas subió")
In [4]: for token in doc:
In [5]:     print(token.lemma_)
Out[6]: ['el', 'kilo', 'de', 'manzana', 'subir']
```

Las funciones a destacar *load* que se utiliza para cargar el conjunto donde se buscan y recuperan los lemas de los tokens.

3.6. Algoritmos de clasificación

El enfoque dominante actualmente para el problema de categorización de textos se basa en técnicas de aprendizaje automático: se construye un clasificador mediante el aprendizaje de las características o atributos de las categorías contenidas en un conjunto de documentos previamente clasificados que sirven como conjunto de entrenamiento. Se trata por tanto del llamado aprendizaje supervisado. Entre los algoritmos supervisados

para la categorización de documentos encontramos, vecinos más cercanos (KNN), máquinas de soporte vectorial (SVM), árboles de decision, entre otros [16]. Estos algoritmos pueden ser entrenados para clasificar documentos dado un conjunto suficientemente grande de ejemplos de entrenamiento previamente etiquetado, por un especialista, con la categoría correspondiente. Esto puede presentar un problema debido a que la limitante de estos algoritmos es que precisan de una cantidad grande de datos de entrenamiento previamente etiquetados lo que requiere un esfuerzo enorme para la o los especialistas revisar cada uno de los documentos y clasificarlos. Esto ha dado pie a la creación de nuevos algoritmos denominados no supervisados o semi supervisados, los cuales se valen de pocos datos de entrenamiento de los cuales obtienen las características y clasifican, y mediante una serie de pasos pueden dar una categoría a datos no clasificados. Entre estos algoritmos encontramos DBSCAN.

3.7. Reducción de dimensión

3.7.1. Análisis de Componentes Principales

El análisis de componentes principales,*Principal Component Analisis* o PCA por sus siglas en inglés, es un método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conserva su información [17]. Supóngase que se tiene una muestra que contiene un número p de dimensiones, PCA permite encontrar un número de factores subyacentes z, $(z < p)$, que explican aproximadamente lo mismo que las variables originales. Ahora en lugar de tener p dimensiones para caracterizar las muestras, bastan solo z valores. Cada una de estas z nuevas variables recibe el nombre de componente principal [18].

El calculo de las componentes principales tiene sus bases en los conceptos de valores y vectores propios. Los vectores propios de una matriz son todos aquellos vectores que, al multiplicarlos por dicha matriz, resultan en el mismo vector o en un múltiplo del mismo. Los vectores propios tienen las siguientes propiedades:

- Si la matriz es de nxn el número de vectores propios que tendrá la

matriz es n.

- Si se escala un vector propio antes de multiplicarlo por la matriz, se obtiene un múltiplo del mismo vector propio.
- Todos los vector propio de una matriz son ortogonales entre si.

Los valores propios, son el número por el cual el vector propio esta multiplicado para obtener el vector original. A todo vector propio le corresponde un valor propio y viceversa.

En PCA, cada una de las componentes se corresponde con un vector propio y el orden de componente se establece por orden decreciente de valor propio. Entonces, la primera componente es el vector propio con el valor propio asociado más alto.

Existen diversas herramientas y paqueterías que permiten calcular de una manera ágil las componentes principales. Una de las más conocidas desarrollada en python es scikit-learn [14] mediante la clase *decomposition* que incluye algoritmos para la descomposición de matrices, entre ellos PCA. El algoritmo mediante la descomposicion en valores singulares permite la reducción de la dimensión de los datos proyectados a un espacio de dimensión menor.

La sintaxis para utilizar dicha clase es la siguiente.

```
In [1]: from sklearn.decomposition import PCA
In [2]: X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1],
...                   [2, 1], [3, 2]])
In [3]: pca = PCA()$
In [4]: pca.fit(X)
Out[5]: pca()
```

El siguiente método devuelve el porcentaje de varianza explicada por cada componente lo que permite elegir el número de componentes final.

```
In [6]: print(pca.explained_variance_ratio_)
Out[7]: [0.9924... 0.00755...]
```

Una vez analizado el porcentaje de varianza explicado por cada componente se elige el número de componentes a trabajar y se transforma la misma matriz indicando al algoritmo el número de componentes a trabajar.

```
In [1]: from sklearn.decomposition import PCA
In [2]: X = np.array([[-1, -1], [-2, -1], [-3, -2], Â´
...                   [1, 1], [2, 1], [3, 2]])
In [3]: pca = PCA(n_components = 2)
In [4]: pca.fit_transform(X)
Out[5]: PCA(n_components=2)
In [6]: print(pca.explained_variance_ratio_)
Out[7]: [0.9924... 0.0075...]
In [8]: print(pca.singular_values_)
Out[9]: [6.30061... 0.54980...]
```

Finalmente el algoritmo devuelve la matriz de dimensión reducida del tamaño de las componentes elegidas.

3.8. Aprendizaje no supervisado

3.8.1. DBSCAN

Agrupamiento espacial basado en densidad de aplicaciones con ruido, o en inglés conocido como *Density-based spatial clustering of applications with noise* [1], DBSCAN, es un algoritmo diseñado para descubrir conjuntos de datos con ruido en una base de datos espacial. Tomemos como ejemplo los siguientes conjuntos de datos:

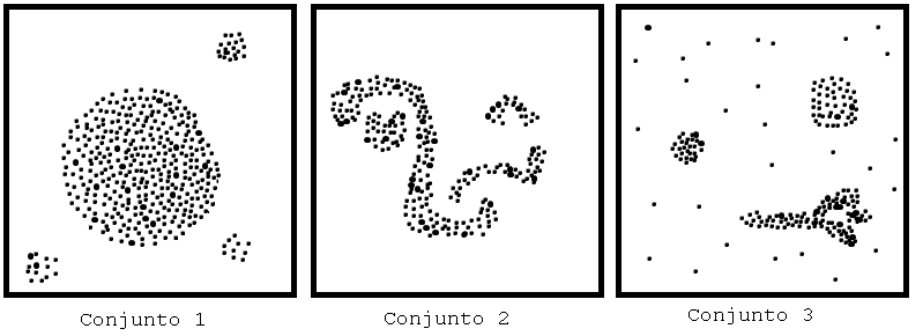

Figura 3.2: Conjuntos de datos de ejemplo, tomado de [1]

Es sencillo identificar aquellos puntos que pertenecen a un conjunto de datos, o clúster, y aquellos puntos que no pertenecen a ningún clúster. La principal razón por la cual se reconocen estos clústers es que existe una densidad alta de puntos que es considerablemente mayor que en las áreas fuera del clúster. La idea principal del algoritmo es que para cada punto en un clúster el vecindario del punto dado en un radio de distancia debe contener un mínimo de puntos, esto es, que la densidad en el vecindario debe superar cierto umbral.

Definición 1: Vecindario *epsilon* de un punto. El vecindario *epsilon* de un punto, denotado como $[N_{Eps}(p)]$ es el conjunto de puntos que se encuentran a una distancia menor al radio *epsilon* definido, esto es $N_{Eps}(p) = \{q \in D | dist(p,q) \leq Eps\}$, esta métrica de distancia también se conoce como métrica de similitud.

Existen dos tipos diferentes de puntos dentro de un clúster, los puntos dentro del clúster o puntos núcleo, y los puntos en la frontera o puntos frontera. En general el vecindario *epsilon* de un punto frontera contiene menos puntos que el de un punto núcleo. Se debe entonces definir un número mínimo de puntos relativamente bajo para incluir todos los

puntos que pertenecen a un mismo clúster. Por lo tanto necesitamos que para cada punto p en un clúster C haya un punto q en C de modo que p este dentro del vecindario *epsilon* de q y $[N_{Eps}(q)]$ contenga al menos el mínimo número de puntos dado, esto se define de la siguiente manera:

Definición 2: Directamente alcanzable por densidad. Un punto p es directamente alcanzable por densidad desde un punto q respecto a una *epsilon*, (Eps), y un mínimo número de puntos, $(MinPts)$, si

- 1) $[p \in N_{Eps}(q)]$ y
- 2) $[|N_{Eps}(q)| \geq MintPts]$ (condición de punto núcleo)

Ser directamente alcanzable por densidad es simétrico para pares de puntes núcleo, sin embargo, es asimétrico cuando se involucran un punto núcleo y un punto frontera. La figura siguiente muestra el caso asimétrico.

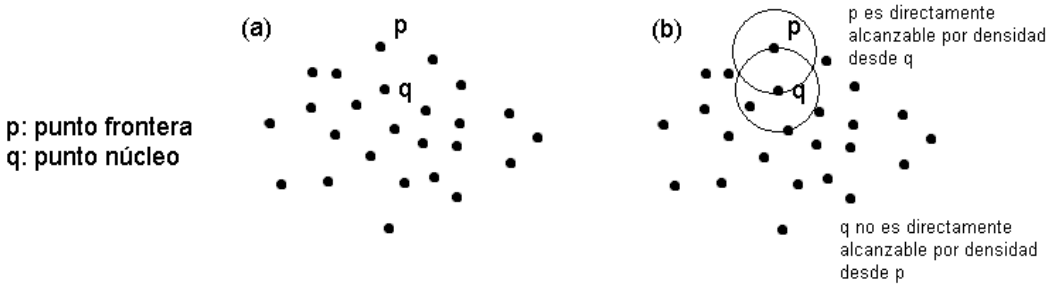

Figura 3.3: Puntos núcleo y puntos frontera, tomada de [1]

Definición 3: Densamente alcanzable. Un punto p es *densamente alcanzable* desde un punto q con respecto a un Eps y un $MinPts$ si existe una cadena de puntos $[p_1, ..., p_n]$, $[p_1 = q]$,$[p_n = p]$ tal que $[p_{i+1}]$ es directamente alcanzable por densidad desde $[p_i]$.

Densamente alcanzable es una extensión de directamente alcanzable por densidad, la relación es transitiva pero no es simétrica. La figura (a) ilustra el caso asimétrico.

Definición 4: Conectados densamente. Un punto p esta *densamente conectado* a un punto q con respecto a un Eps y un $MinPts$ si existe un punto o tal que ambos puntos p y q son densamente alcanzables desde o con respecto a un Eps y un $MinPts$.

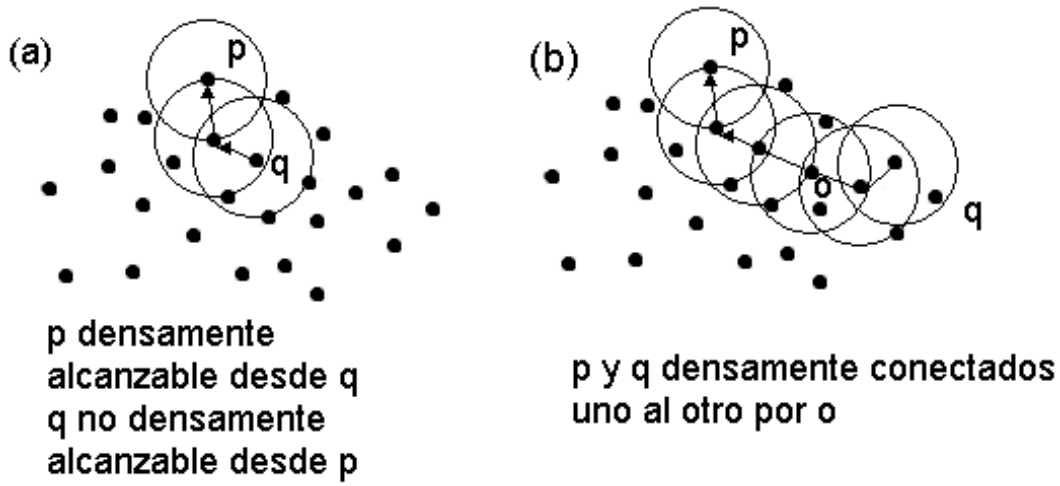

Figura 3.4: Densamente alcanzable y densamente conectado, tomada de [1]

La conectividad densa es una relación simétrica. Para puntos densamente alcanzables, la relación de conectividad densa es también reflexiva (figura (b)).

Ahora es posible definir a un clúster con base en la densidad. Intuitivamente, un clúster esta definido como el conjunto máximo de puntos densamente conectados. El ruido es simplemente el conjunto de puntos en D que no pertenecen a ningún clúster.

Definición 5: Clúster. Sea D una base de datos de puntos. Un *clúster* C dados Eps, y $MinPts$, es un conjunto no vacío de D que satisface las siguientes condiciones:

- 1)$\forall p, q$: Si $p \in C$ y q es directamente alcanzable por p dados Eps y $MinPt$, entonces $q \in C$.
- 2) $\forall p, q \in C$: p esta densamente conectado a q dados Eps y $MinPts$.

Definición 6: Ruido. Sean $[C_1, ..., C_k]$ los clústers del conjunto de datos D, con respecto a los parámetros dados $[Eps_i]$ y $[MinPts_i]$,$i = 1, \ldots, k$. Entonces definimos el ruido como el conjunto de puntos en la base de datos D que no pertenecen a ningún clúster $[C_i]$, es decir, $[p \in D | \forall i : p \notin C_i]$

Los siguientes lemas son importantes para validar que la clusterización se haya realizado de manera correcta.

Lema 1: Sea p un punto en D y $[|N_{Eps}(p)| \geq MinPts]$. Entonces el conjunto $[O = \{o | o \in D$ y o es densamente alcanzable desde p con respecto a Eps y $MinPts\ \}]$ es un clúster con respecto a un Eps y $MinPts$.

No es obvio que un clúster C dados Eps y $MinPts$, esta determinado por cualquiera de sus puntos núcleo. Sin embargo, cada punto de C es densamente alcanzable desde cualquier punto núcleo de C y, por tanto, C contiene exactamente los puntos que son densamente alcanzables desde cualquier punto núcleo de C.

Lema 2: Sea C un clúster dados Eps y $MinPts$ y sea p cualquier punto en C con $[|N_{Eps}(p)| \geq MinPts]$. Entonces C es igual al conjunto $[O = \{o|o$ es densamente alcanzable desde p dados Eps y $MinPts\}]$.

Idealmente nos gustaría conocer el valor apropiado para los parámetros *epsilon* y $MinPts$, no existe una manera sencilla de obtener esta información para todos los clústers de la base de datos. Existe así una heurística efectiva y simple para determinar los parámetros Eps y $MinPts$ del clúster "menos denso" en la base de datos. Al ser estos parámetros del clúster "menos denso" son buenos candidatos a ser parámetros globales pues consideran la densidad más baja que no es considerada ruido.

El Algoritmo

Para encontrar un clúster, DBSCAN comienza con un punto arbitrario p y recupera todos los puntos densamente alcanzables desde p dados los parámetros epsilon, eps, y mínimo número de puntos, $MinPts$. Sí el punto p es un punto núcleo este procedimiento devuelve un clúster dados Eps y $MinPts$, esto por el lema 2. Si p es un punto frontera, ningún punto es densamente alcanzable desde p por lo que DBSCAN visita el siguiente punto en la base de datos. Dado que los parámetros Eps y $MinPts$ son globales, DBSCAN puede combinar dos clústers de acuerdo a la definición 5, si dos clústers de diferente densidad se encuentran "cerca" el uno del otro. Para comprender mejor la combinación de clúster definamos la distancia entre dos clústers $[S_1]$ y $[S_2]$ como $[dist(S_1, S_2) = min\{dist(p, q)|p \in S_1, q \in S_2\}]$, entonces dos conjuntos de puntos van a estar separados el uno del otro solo si la distancia entre ambos clusters es mayor a Eps. En consecuencia, DBSCAN debe ser llamado recursivamente para detectar el clúster cuyo parámetro $MinPts$ sea el mayor. Esto no presenta una desventaja dado que la aplicación recursiva de DBSCAN entrega un algoritmo básico, elegante y muy eficiente. El siguiente esquema explica el flujo de trabajo del algoritmo de DBSCAN.

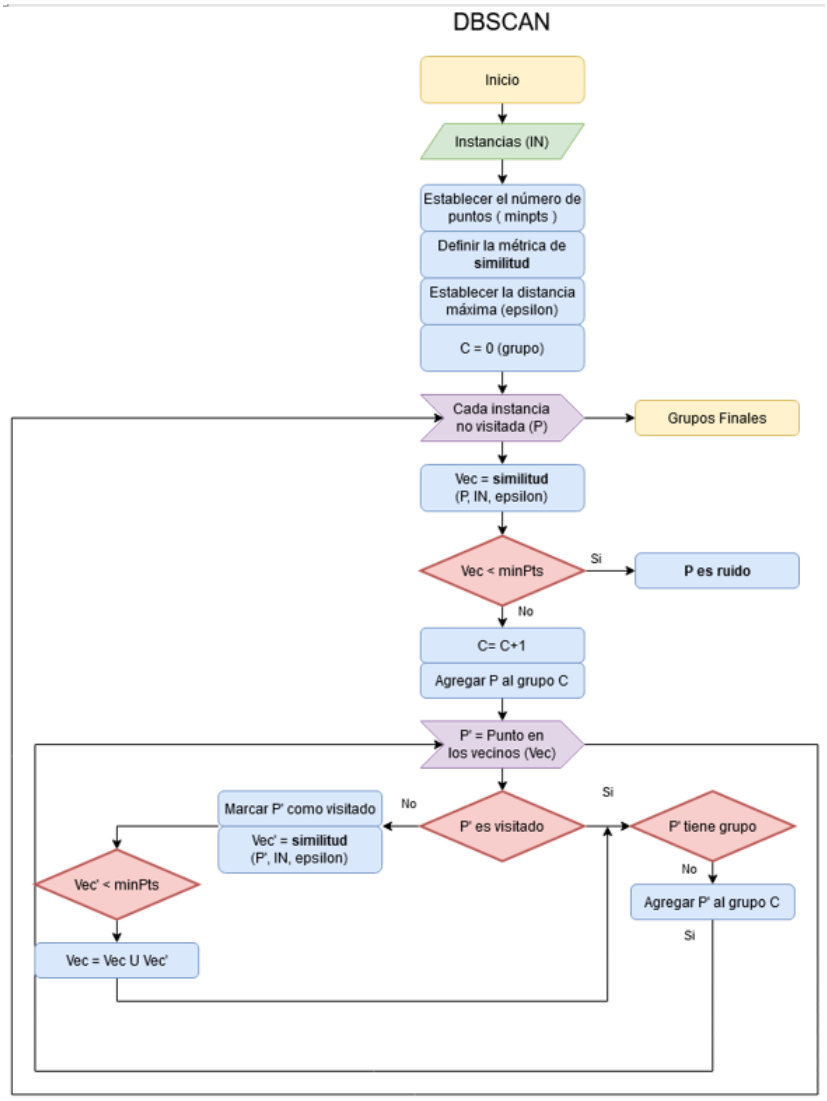

Figura 3.5: Diagrama de flujo del algoritmo de DBSCAN implementado para el proyecto

Siguiendo el diagrama de flujo 3.5 y con lo antes mencionado comenzamos con las instancias, o datos, que queremos agrupar. Establecemos los parámetros Eps y $MinPts$, así como la métrica de similitud. Todos los puntos son inicializados con una etiqueta de grupo para revisar si ese punto ya fue visitado, $C = 0$. Se toma un punto p aleatoriamente, si su etiqueta marca que no ha sido visitado, se genera su vecindad dado Eps, si el número de puntos en la vecindad es menor al parámetro $MinPts$ el punto p se marca como ruido, si no, la etiqueta de grupo C cambia y se agrega p al grupo. Se revisan así todos los puntos p' que pertenecen a la vecindad de p, si p' no ha sido visitado se marca como visitado y se calcula su vecindad, si la vecindad de p' es menor a $MinPts$, $[N_{Eps}(p') < MinPts]$, entonces la $[N_{Eps}(p) = N_{Eps}(p) \cup N_{Eps}(p')]$, esto es la expansión del clúster, y por lo tanto p', así como sus vecindario, se agregan al grupo de p. Por el contrario, si p' ya fue visitado, y tiene grupo, mantiene su etiqueta, si no, se agrega al grupo de p. El algoritmo termina cuando no queda ningún punto por etiquetar.

Capítulo 4

Metodología

La metodología la dividimos en las etapas siguientes:

- Adquisición de datos
- Pre-procesamiento
- Reducción de dimensión
- Clasificación
- Evaluación

Cabe resaltar que en cada una de las etapas se usó el lenguaje de programación python y se utilizaron las librerías descritas en la sección de antecedentes.

4.1. Adquisición de datos

4.1.1. Conjunto de datos

Se obtuvo el corpus del trabajo de Posadas et. al. [7] cuyo repositorio se encuentra en
https://github.com/jpposadas/FakeNewsCorpusSpanish, el corpus consta de 491 noticias falsas y 480 noticias verdaderas recopiladas de enero a julio de 2018, todas ellas escritas en español. La base de datos cuenta con 7 columnas con diversas categorías de la noticia, como se muestra en la tabla 4.1, y está dividida en conjunto de entrenamiento y validación.

Ambos conjuntos de datos contienen las columnas descritas en la tabla 4.2

Categoría	Entrenamiento		Validación	
	Verdaderas	Falsas	Verdaderas	Falsas
Ciencia	32	30	14	13
Deportes	45	41	21	17
Economía	18	12	6	7
Educación	6	9	4	3
Entretenimiento	48	55	22	23
Política	121	105	54	43
Salud	16	16	17	17
Seguridad	11	18	6	7
Sociedad	41	52	19	22
Total	338	338	153	142

Tabla 4.1: Distribución del Corpus

ID	Identificador de la noticia
Categoría	Clasificación de la noticia, verdadera o falsa
Tema	Tema principal de la noticia
Fuente	De donde se recuperó la noticia
Encabezado	Encabezado de la noticia
Texto	Texto de la noticia
Link	El hipervínculo a la noticia

Tabla 4.2: Estructura de las columnas presentes en el corpus para cada noticia

Id	**Cat**	**Tema**	**Fuente**	**Encabezado**	**Texto**	**Link**
1	Fake	Educ	El Ruina...	RAE INC...	RAE INC...	http://www.elruina...
2	Fake	Educ	Hay no...	La palabr...	La palabra...	https://haynoticia....
3	Fake	Educ	El Ruina..	YORDI...	YORDI R...	http://www.elruina...
4	True	Educ	EL UNI..	UNA...	UNAM cap...	http://www.elunive...
5	Fake	Educ	Lamula	pretend...	Alerta: prete...	https://redaccion.l...
6	True	Educ	Heraldo	Un paso ...	Un paso ...	https://www.heral...
7	Fake	Educ	El Ruina...	UNAM ...	UNAM R...	http://www.elruina...
8	True	Educ	Excelsi...	Niño de ...	Niño de ...	https://www.excels...
9	True	Educ	El país	*NUMBE...	*NUMBE...	https://elpais.com/e...
10	Fake	Educ	El Ruina...	LIMITAR ...	LIMITAR...	http://www.elruina...

Tabla 4.3: Ejemplo de algunas filas del corpus

Las columnas que se eligieron para el entrenamiento de nuestro modelo fueron 2: la columna “categoría” que establece si una noticia es falsa o verdadera y la columna “texto” que contiene el texto en crudo completo de la noticia.

4.2. Preprocesamiento

Como lo explican en [7] las noticias tienen un preprocesamiento previo, el cual consiste en la eliminación de elementos comunes en la estructura de una noticia como el nombre del autor o editor, fechas, pies de página y encabezados que hagan referencia al sitio web. Las referencias a fotos y vídeos, así como caracteres especiales son eliminados, la codificación fue el estándar *utf-8*. El proceso incluyó la sustitución de los siguientes elementos por un identificador en común.

1) Los números que representan cantidades, fechas o precios por *NUMBER*

2) Direcciones de correo electrónico por *EMAIL*

3) URLs de referencias por *URL*

4) Números de teléfono por *PHONE*

5) Los símbolos de dólar y euro por *DOL* y *EUR* respectivamente

Además del preprocesamiento de los autores mencionado anteriormente con el cual cuentan las noticias, se realizó un segundo preprocesamiento con el objetivo de eliminar todo aquello que no pueda aportar información, normalizar el texto y reducir el tamaño del vocabulario. Dicho preprocesamiento consistió en 6 etapas, manejar todas las palabras en minúsculas, eliminar palabras de paro, eliminar caracteres especiales, signos de puntuación y palabras en otro idioma, así como la búsqueda de lemas y sinónimos, con el objetivo de reducir el vocabulario. Para esta parte se crearon funciones que permitieron automatizar estas tareas y aplicarlas al conjunto de entrenamiento y de validación. Las funciones creadas que se utilizaron para el procesamiento son las siguientes:

$clean\_corpus(text\_docs)$
$bagofwords(corpus, CV\_TFIDF, binario, stopwords, n\_grams, analyzer)$
$lemmas(bag\_of\_words)$
$actbow(X, bag\_of\_words, sins)$

Tabla 4.4: Funciones de preprocesamiento

4.2.1. Limpieza de textos

La función $clean\_corpus(text\_docs)$ recibe el conjunto de textos y mediante el uso de expresiones regulares se eliminan los signos de puntuación, caracteres especiales, dígitos y palabras en otro idioma. Como salida se obtiene el conjunto de los textos con las eliminaciones realizadas.

4.2.2. Bolsas de palabras

Una vez obtenidos los textos después de la limpieza se procedió a realizar la extracción de características haciendo uso de la función $bagofwords(corpus, CV\_TFIDF, binario, stopwords, n\_grams, analyzer)$. Se establece con el uso de *CountVectorizer* el esquema de pesado de la matriz de frecuencias termino-documento, el analizador y el tamaño de n_grams en palabras y se eliminan las palabras de paro usando el conjunto de palabras de paro del corpus de español de *NLTK*. Esta es la primera etapa de reducción del vocabulario, el resultado es una bolsa

de palabras en forma de tabla con columnas de tamaño fijo /tokens) correspondiente al tamaño del vocabulario contenido en el corpus.

4.2.3. Lematización

Con la finalidad de reducir variantes morfológicas de la bolsa de palabras a raíces comunes o lexemas, procedemos a la lematización de los tokens en esta. Para cada palabra de la bolsa se aplico la lematización, es decir, si la palabra era un plural, estaba en femenino, o conjugada, se buscó su correspondiente lema. Si este era el caso de la palabra, se sustituía por su lema.

Aquellas palabras donde cuyos lemas fueron iguales, se sumaron a la primer ocurrencia en la matriz de ocurrencias. En caso de no ser una variante o no tener un lema, el token permanece sin cambios.

Aplicando la lematización a la bolsa de palabras se busca dejar únicamente lemas en la bolsa de palabras con el objetivo de eliminar la redundancia con el uso de variantes morfológicas.

Para esta tarea se usa la función *lemmas(bag_of_words)* que recibe la bolsa de palabras obtenida en la etapa anterior y la devuelve lematizada ayudándose del conjunto de lemas en español del módulo de *Spacy*. Para la actualización de las frecuencias en la matriz de ocurrencias, se emplea la función *actbow(X, bag_of_words, sins)* la cual recibe como parámetros la bolsa de palabras lematizada en la cual se hará la búsqueda de lemas iguales y la matriz de frecuencias donde se suman aquellas frecuencias de los lemas iguales a la primer ocurrencia encontrada. Esta función devuelve la bolsa de palabras con la reducción de vocabulario, así como la matriz de frecuencia actualizada con la suma de las columnas cuyos lemas eran iguales. De esta manera termina la segunda etapa de reducción de vocabulario.

4.2.4. Búsqueda de similitud

En la tercera parte de la reducción de vocabulario se realiza la búsqueda de similitud de las palabras, esto es la búsqueda de sinónimos. Se generó una función que recupera la información de búsqueda de sinónimos desde la página web *wordreference* devolviendo así los sinónimos de una palabra consultada.

El proceso recorre token a token de la bolsa de palabras lematizada anterior, recupera los sinónimos de cada palabra en *wordreference* y realiza una búsqueda en la bolsa de palabras de éste conjunto de sinónimos recuperado.

Si hay coincidencia de algún sinónimo en la bolsa de palabras, se almacena el índice del token y se marca como visitado. Una vez realizada la búsqueda de sinónimos para todos los tokens en la bolsa, se procede a la actualización de la matriz de frecuencias, haciendo uso nuevamente de la función *actbow(X, bag_of_words, sins)*, tomando en cuenta aquellos índices que se marcaron como sinónimos y sumando las frecuencias a la primer ocurrencia.

En la tabla 4.5 podemos observar como va cambiando el tamaño de la bolsa de palabras del conjunto de entrenamiento.

Bolsa de Palabras	Número de tokens	Reducción
Originalmente	24,792	-
Sin stopwords	24,567	.91 %
Con lemas	17,152	30.81 %
Con sinónimos	12,981	47.64 %

Tabla 4.5: Bolsas de palabras

Para ilustrar el proceso por el cual es sometido el texto de una noticia tomemos el siguiente fragmento extraído de una noticia.

NUMBER palabras que creíamos inaceptables y que la RAE (sorprendentemente) aprueba. La cuenta de Twitter de la Real Academia Española hierve de actividad. Sus respuestas a las consultas de los usuarios pocas veces dejan indiferente Çonsulta: cuál es la forma correcta de referirse a internet: la internet o el internet". Esta es una de las cientos de preguntas que los lectores lanzan a la RAE por Twitter. ...

Figura 4.1: Fragmento de una noticia

En el fragmento de la figura 4.2 en rojo se señalan las eliminaciones y los cambios se señalan en azul, sobre el fragmento de la figura 4.1, todo esto en la etapa de limpieza:

NUMBER palabras que creíamos inaceptables y que la rae (sorprendentemente) aprueba. la cuenta de twitter de la real academia española hierve de actividad. sus respuestas a las consultas de los usuarios pocas veces dejan indiferente“consulta: cuál es la forma correcta de referirse a internet: la internet o el internet". esta es una de las cientos de preguntas que los lectores lanzan a la rae por twitter. ...

Figura 4.2: Fragmento de una noticia después de la limpieza

Para la primer etapa de reducción de vocabulario se eliminan las palabras de paro, en la figura 4.3 se muestra un fragmento de la lista que se obtiene.

[‘abadesa’, ‘academia’, ‘académica’, ‘académico’, ‘académicos’, ‘aceptan’, ‘aceptar’, ‘aceptarlo’, ‘acepte’, ‘aclara’, ‘aclaran’, ‘acortamiento’, ‘acortamientos’, ‘actores’, ‘actual’, ‘actualidad’, ‘españa’, ‘español’, ‘española’, ‘españoles’, ‘estatuilla’, ‘estatuillas’,...]

Figura 4.3: Fragmento de la bolsa de palabras sin palabras de paro

En la segunda etapa lematizamos las palabras flexionadas y reducimos el vocabulario. Por ejemplo, si observamos la lista de la figura 4.3, las palabras “académica”, “académico”, “académicos” fueron reducidos a “académico” como se muestra en la figura 4.4.

[‘abadesar’, ‘academia’, ‘académico’, ‘aceptar’, ‘aceptar él’, ‘aclarar’, ‘acortamiento’, ‘activo’, ‘actividad’, ‘actor’, ‘españa’, ‘español’, ‘estatuilla’, ‘estatuil él’,...]

Figura 4.4: Fragmento de la bolsa de palabras aplicando lematización

Y finalmente con la búsqueda de sinónimos nuestra lista vuelve a cambiar en la figura 4.5 podemos ver un fragmento de la misma.

Para esta noticia en particular la evolución del tamaño de la bolsa de palabras se muestra en la figura 4.6.

['abadesar', 'abandonar', 'abarcar', 'abastecer', 'abogar', 'abordar', 'abortar', 'abran', 'abrupto', 'absoluto', 'aburrido', 'academia', 'académico', 'españa', 'español', 'estatuilla', 'estatuil él', 'etimológico', 'etiqueto',...]

Figura 4.5: Fragmento de la bolsa de palabras aplicando búsqueda de similitud

Bolsa de Palabras	Número de palabras
Original	2237
No stopwords y limpio	613
Lematizada	540
Con sinónimos	502

Tabla 4.6: Tamaño del vocabulario de la noticia de ejemplo

4.3. Reducción de dimensión

Una vez terminadas las etapas de limpieza y reducción de vocabulario sobre el conjunto de entrenamiento, obtuvimos una matriz de ocurrencias de tamaño 676 noticias por 12,981 tokens. A la matriz de ocurrencias resultante le aplicamos *PCA* para reducir la dimensión de nuestra matriz y experimentar con las primeras 10, 15 y 20 componentes que representen a la amyor diversidad de tokens posible.

La figura 4.6 muestra la varianza explicada por cada una de las componentes del conjunto de entrenamiento.

Y en la figura 4.7 se muestra la varianza acumulada de las componentes.

4.4. Clasificación

4.4.1. Entrenamiento

Con las matrices obtenidas para los primeros 10, 15 y 20 componentes obtenidos en la etapa de reducción de dimensión procedimos a realizar el entrenamiento del algoritmo que se programo basado en DBSCAN. Como se explicó en secciones anteriores, DBSCAN recibe 4 parámetros, el conjunto de datos, el mínimo número de puntos para que se genere un clúster *minPts*, epsilon *eps*, y la métrica de distancia, euclidiana o similitud coseno. Dado que no existen valores predeterminados para DBSCAN que devuelvan los mejores valores para los parámetros procedimos a ha-

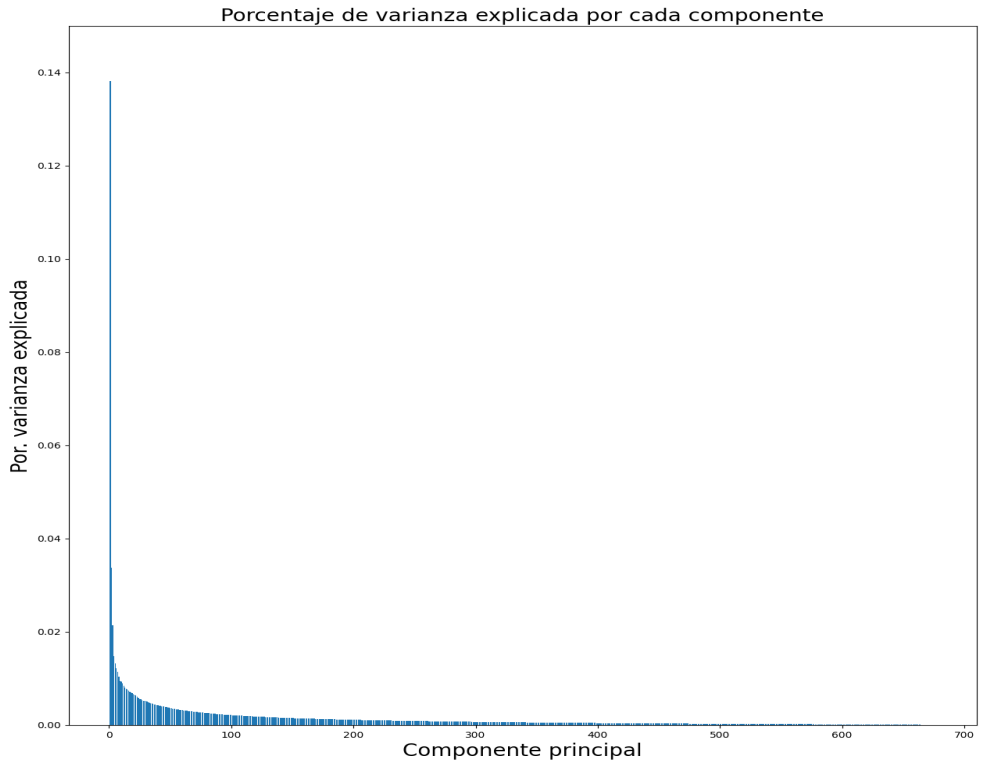

Figura 4.6: Porcentaje de Varianza por componente. El máximo de componentes a calcular es el mínimo entre número de observaciones y el número de variables, por eso es que se calculan solamente 676 componentes principales.

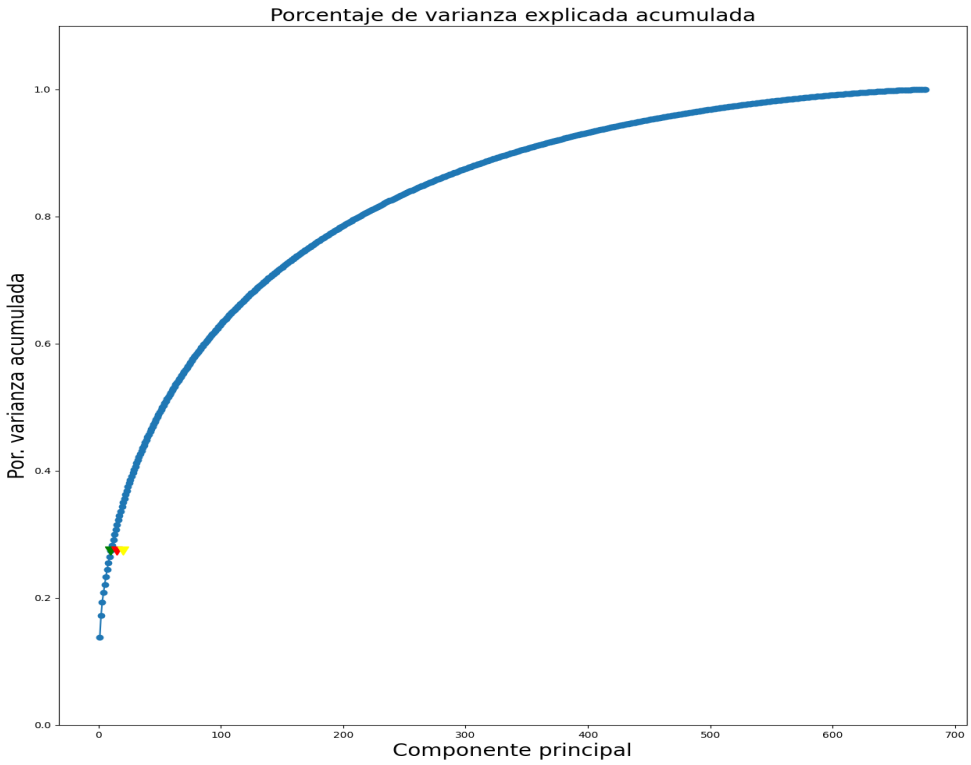

Figura 4.7: Porcentaje de varianza explicada acumulada, se marcan en verde, rojo y amarillo las primeras 10, 15 y 20 componentes.

cer una búsqueda de distintos valores para encontrar aquellos que nos dieran un mejor desempeño. En la tabla 4.7 se muestran los valores que se utilizaron para entrenar el modelo y buscar la mejor combinación de parámetros.

Distancia	*minPts*	*eps*
Euclidiana	5, 10, 15, 20, 23, 24, 25, 26, 27, 28	1.0, 1.5, 2.0, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2, 3.3, 3.4, 3.5
Coseno	5, 10, 15, 20	0.12, 0.15, 0.21, 0.25, 0.35, 0.45, 0.5

Tabla 4.7: Valores utilizados para el entrenamiento de DBSCAN

4.4.2. Validación

Una vez obtenidos los parámetros, (*eps* y *minPts*), para la combinación que nos dió el conjunto de entrenamiento, procedimos a utilizar esta clasificación para etiquetar los textos del conjunto de validación.

Para el conjunto de validación procedimos al preprocesamiento, realizar la limpieza de los textos con la función *clean_corpus*, calculamos la bolsas de palabras utilizando el mismo vocabulario que el de la fase de entrenamiento, redujimos el vocabulario quitando las palabras de paro, lematizando y buscando los sinónimos, que al final resulta en los mismos cambios en cuanto a lemas y sinónimos, por lo que se actualizan las mismas columnas que en la fase de entrenamiento.

Una vez obtenida la ultima matriz procedimos a reducir la dimensión de esta utilizando PCA, obtuvimos las matrices para los primeros 10, 15 y 20 componentes de esta matriz.

Con las noticias etiquetadas del conjunto de entrenamiento con la combinación de parámetros que nos dió la mejor exactitud procedimos a clasificar las noticias del conjunto de validación.
La clasificación de las noticias se realizó calculando la matriz de distancias para cada uno de los nuevos puntos, es decir, para cada noticia sin clasificar, calculamos la distancia a los puntos previamente clasificados y su categoría será la que tenga el punto clasificado más cercano.

Capítulo 5

Resultados

En la siguiente sección presentamos los resultados obtenidos al aplicar la metodología.

5.1. Entrenamiento

5.1.1. Preprocesamiento

Comenzaremos mostrando la reducción del vocabulario en las diferentes etapas del preprocesamiento de las noticias. La tabla 5.1 muestra la evolución del tamaño del vocabulario para el conjunto de entrenamiento después de aplicar la lematización y la búsqueda de sinónimos.

Bolsa de Palabras	Número de palabras
Original	24,792
No stopwords y limpio	24,567
Lematizada	17,152
Con sinónimos	12,981

Tabla 5.1: Evolución del tamaño del vocabulario

5.1.2. Reducción de dimensión

Una vez obtenida la última reducción de vocabulario se realizó la reducción de dimensión utilizando PCA a esta última matriz de $[676 \times 12,981]$ para obtener las primeras 10, 15 y 20 componentes que serán utilizadas en la siguiente etapa de experimentación.

5.1.3. Clasificación

Con las matrices obtenidas para las componentes principales procedimos a experimentar en cuanto a la clasificación con DBSCAN, recordemos que los parámetros que recibe el algoritmo son, el mínimo número de puntos ($minPts$), el radio del vecindario (eps) y la métrica de distancia. Para este punto al hacer la comparación entre los resultados obtenidos utilizando como métrica la distancia coseno y la distancia euclidiana, observamos que los resultados utilizando la distancia coseno no fueron satisfactorios y no superaban a los obtenidos con la distancia euclidiana por lo que no seguimos realizando experimentos haciendo uso de la distancia coseno.

En cuanto a la distancia euclidiana continuamos experimentando para diversos valores de los parámetros eps y $minPts$. Así mismo hicimos las mismas pruebas para las matrices de las primeras 10, 15 y 20 componentes principales. Al final nos quedamos únicamente con la matriz para las primeras 10 componentes principales ya que las matrices para las 15 y 20 componentes principales no superaban los resultados obtenidos por la matriz de 10 componentes principales.

Las tablas que muestran los resultados para las primeras 10, 15 y 20 componentes principales se pueden revisar en la parte del apéndice.

De acuerdo a los resultados obtenidos decidimos quedarnos con la clasificación obtenida para las primeras 10 componentes principales con los valores de $minPts$ igual a 15 y el $epsilon$ igual a 3.

5.1.4. Análisis del error

La figura 5.2 muestra la matriz de confusión de nuestros resultado y a partir de esta podemos analizar el como nos estamos equivocando[1].

De acuerdo a esto condujimos un análisis sobre los valores de esta matriz. Comparamos el vocabulario de los Verdaderos Positivos (TP)

[1]Cabe señalar que la etiqueta de Verdadero es que la noticia es Falsa y Falso que la noticia es Verdadera

		Valor Predicho		
		Falso	**Verdadero**	**total**
Valor real	**Falso**	TN 230	FP 108	338
	Verdadero	FN 109	TP 229	338
	total	339	337	

Tabla 5.2: Matriz de Confusión

con los Falsos Positivos (FP) para ver que posiblemente estaba trayendo confusión para no clasificar correctamente las noticias. En la tabla 5.3 se muestra el como se compone el vocabulario de cada una de las clases.

Clase	Número de palabras
FP	3,551
TP	5,535
FP y TP	7,079

Tabla 5.3: Tamaño del vocabulario Falsos positivos y Verdaderos Positivos

Como observamos las suma del número de palabras de las clases no es igual al de la combinación de dichas clases por lo que deben compartir palabras.

Clase	Número de palabras en la clase
FP	1,544
TP	3,528
FP y TP	2,007

Tabla 5.4: Tamaño del vocabulario Falsos positivos y Verdaderos Positivos por clase

La tabla 5.4 muestra que las noticias que etiquetamos como falsas y son verdaderas (FP) comparten 2,007 palabras con las noticias que etiquetamos como falsas y son falsas (TP). Es decir, que las FP, noticias verdaderas, comparten aproximadamente un 56.53 % de palabras con las

noticias falsas y al contrario las noticias falsas (TP) comparten con las noticias verdaderas el 36.26 %. Ahora que tan representativas son las palabras que comparten, tomamos las 10 palabras más frecuentes en ambas clases y en la figura 5.1 se muestra la distribución de éstas.

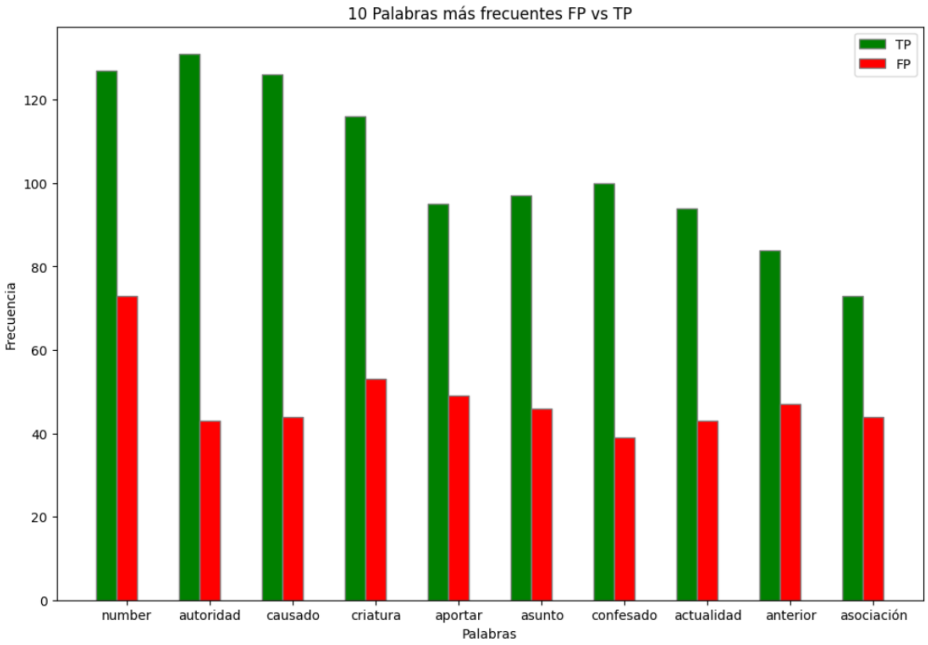

Figura 5.1: 10 palabras más frecuentes FP vs TP

De la misma manera trabajamos sobre el vocabulario de los Falsos Negativos (FN) comparándolos con los Verdaderos Negativos (TN). La tabla 5.5 muestra el como se compone el vocabulario de cada una de las clases.

Clase	Número de palabras
FN	4,366
TN	8,190
FN y TN	9,849

Tabla 5.5: Tamaño del vocabulario Falsos Negativos y Verdaderos Negativos

Como observamos las suma del número de palabras de las clases no es igual al de la combinación de dichas clases por lo que deben compartir palabras.

La tabla 5.6 muestra que las noticias que etiquetamos como verdaderas y son falsas (FN) comparten 2,707 palabras con las noticias que

Clase	Número de palabras en la clase
FN	1,659
TN	5,483
FN y TN	2,707

Tabla 5.6: Tamaño del vocabulario Falsos positivos y Verdaderos Negativos por clase

etiquetamos como verdaderas y son verdaderas (TN). Es decir los FN, noticias que son falsas, comparten aproximadamente un 62 % de vocabulario con las noticias que son verdaderas. Y al contrario las noticias que son verdaderas (TN) comparten con las noticias que son falsas solo es el 33.05 %. Observamos en la figura 5.2 la distribución de las 10 palabras con mayor frecuencia en ambas clases.

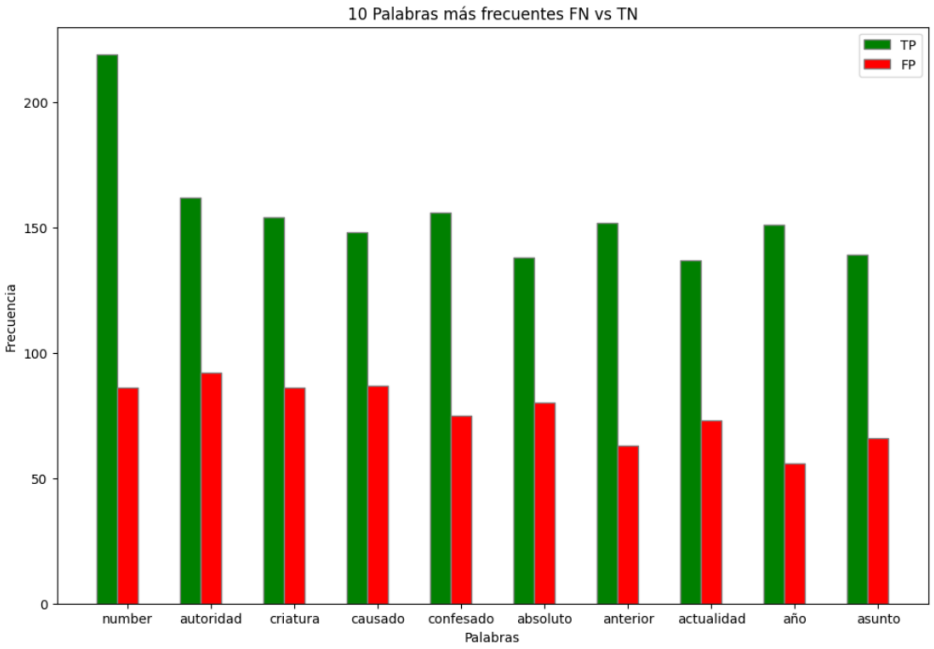

Figura 5.2: 10 palabras más frecuentes FN vs TN

5.2. Validación

Una vez realizado nuestro análisis sobre los errores que encontramos procedimos a realizar nuestra evaluación sobre el conjunto de validación. Para mantener la consistencia del tamaño de los vectores se realizó el mismo procedimiento de preprocesamiento al quitar palabras de paro y la limpieza pero al realizar la vectorización se utilizaron las matrices

de los vocabularios de la fase de entrenamiento para preservar dicha dimensión. Se realizó la reducción de dimensión con PCA y solamente nos quedamos con los primeros 10 componentes de acuerdo a lo obtenido en la fase de entrenamiento.

En cuanto a la clasificación con el conjunto de entrenamiento ya clasificado, para cada una de las nuevas instancias del conjunto de validación se calculó la distancia más cercana a un punto previamente etiquetado para obtener la etiqueta de la nueva instancia. Los resultados sobre el conjunto de validación se muestran en la tabla 5.7.

Precision	0.567
Recall	0.739
F1 Score	0.642
Accuracy	0.603

Tabla 5.7: Tamaño del vocabulario Falsos positivos y Verdaderos Negativos por clase

Capítulo 6

Conclusiones

Partiremos de las preguntas de investigación presentadas anteriormente. ¿Es posible clasificar mediante el uso de aprendizaje no supervisado noticias en falsas y verdaderas? y ¿Cómo se compara el rendimiento de este método de aprendizaje no supervisado con métodos al aprendizaje supervisado?.

Para la primera pregunta basta con irnos a la sección de resultados donde presentamos todos los experimentos realizados utilizando el algoritmo de DBSCAN para mostrar si es posible mediante el agrupamiento espacial, clasificar noticias falsas y verdaderas. Una de las ventajas encontradas es que a partir de un corpus calculando su bolsa de palabras es posible caracterizar otras noticias a partir de éstas ya clasificadas sin la necesidad de volver a aplicar el algoritmo de clusterización sino que se calculan las distancias a los puntos previamente clasificados y a partir de qué punto esté más cerca se clasifica esta nueva entrada. Dentro de las áreas a mejorar en el modelo es el tiempo de cómputo necesario para la reducción del vocabulario, sería posible reducir el tiempo de cómputo afinando los métodos para la búsqueda de los lemas, los sinónimos y la actualización de la matriz de ocurrencias, pero esto va más allá el trabajo aquí presentado. Además de esto, en estos experimentos no aprovechamos ninguna información extra sobre las noticias, no se extrajeron características semánticas o sintácticas, únicamente se utilizó la bolsa de palabras, podría ser que utilizando vectores de características se obtengan mejores resultados usando el algoritmo de DBSCAN sobre estos vectores.

Para la segunda pregunta observamos una diferencia entre el método propuesto y lo presentado en [7], los resultados obtenidos con nuestro

método no alcanzan a igualar los resultados alcanzados por las máquinas de soporte vectorial, la regresión logística, los bosques aleatorios y el *boosting* con la combinación de características propuesta, sin embargo, se tiene una buena exactitud en fase de validación considerando que no se tiene conocimiento a priori de la clasificación de la noticia para el entrenamiento de la metodología.

Como trabajo futuro se podría mejorar el algoritmo para la actualización de las bolsas de palabras. Para continuar con la detección de noticias falsas me gustaría probar métodos de deep learning que es el estado de arte actual con modelos grandes de lenguaje natural (LLMs) con un ajuste fino sobre noticias falsas en español y hacer que este modelo sea más rápido para hacerlo productivo.

Apéndice A

Apéndice

A.1. Tablas de resultados

Las siguientes tablas muestran los resultados obtenidos al aplicar el algoritmo programado de DBSCAN para las matrices de las primeras 10, 15 y 20 componentes principales

clusters	Fake	NoFake	eps	minPts	Precision	Recall	F1	Accuracy
-1	0	676	0.5	5		0		0.5
-1	0	676	1.5	25		0		0.5
-1	0	676	1	25		0		0.5
-1	0	676	0.5	25		0		0.5
-1	0	676	0.5	10		0		0.5
-1	0	676	1	10		0		0.5
-1	0	676	1.5	20		0		0.5
-1	0	676	1	20		0		0.5
-1	0	676	0.5	20		0		0.5
-1	0	676	0.5	15		0		0.5
-1	0	676	1	15		0		0.5
-1	0	676	1.5	15		0		0.5
"1, -1"	5	671	1	5	0.8	0.012	0.023	0.504
"1, -1"	77	599	1.5	5	0.805	0.183	0.299	0.57
"1, -1"	140	536	2	25	0.786	0.325	0.46	0.618
"1, -1"	145	531	2	20	0.772	0.331	0.464	0.617
"1, -1"	151	525	2	15	0.755	0.337	0.466	0.614
"1, -1"	183	493	2	10	0.732	0.396	0.514	0.626
"1, -1"	216	460	2	5	0.713	0.456	0.556	0.636
"1, -1"	243	433	2.5	25	0.72	0.518	0.602	0.658
"1, -1"	249	427	2.5	20	0.723	0.533	0.613	0.664
"1, -1"	267	409	2.5	15	0.719	0.568	0.635	0.673
"1, -1"	278	398	2.5	10	0.712	0.586	0.643	0.675
"1, -1"	295	381	2.5	5	0.702	0.612	0.654	0.676
"1, -1"	323	353	3	25	0.693	0.663	0.678	0.685
"1, -1"	332	344	3	20	0.684	0.672	0.678	0.68
"1, -1"	337	339	3	15	0.68	0.678	0.679	0.679
"1, -1"	347	329	3	10	0.674	0.692	0.683	0.679
"1, -1"	379	297	3	5	0.662	0.743	0.7	0.682
"1, -1"	386	290	3.5	25	0.655	0.749	0.699	0.678
"1, -1"	401	275	3.5	20	0.641	0.76	0.696	0.667
"1, -1"	406	270	3.5	15	0.638	0.766	0.696	0.666
"1, -1"	432	244	3.5	10	0.609	0.778	0.683	0.639
"1, -1"	451	225	3.5	5	0.599	0.799	0.684	0.632

Tabla A.1: Resultados distancia Euclidiana 10 componentes

clusters	Fake	NoFake	eps	minPts	Precision	Recall	F1	Accuracy
-1	0	676	0.5	5		0		0.5
-1	0	676	0.5	15		0		0.5
-1	0	676	0.5	20		0		0.5
-1	0	676	1	20		0		0.5
-1	0	676	1.5	20		0		0.5
-1	0	676	2	20		0		0.5
-1	0	676	1.5	10		0		0.5
-1	0	676	1	15		0		0.5
-1	0	676	1	10		0		0.5
-1	0	676	0.5	25		0		0.5
-1	0	676	1	25		0		0.5
-1	0	676	1.5	25		0		0.5
-1	0	676	2	25		0		0.5
-1	0	676	1	5		0		0.5
-1	0	676	0.5	10		0		0.5
-1	0	676	1.5	15		0		0.5
"1, -1"	8	668	1.5	5	0.875	0.021	0.04	0.509
"1, -1"	40	636	2	15	0.775	0.092	0.164	0.533
"1, -1"	45	631	2	10	0.8	0.107	0.188	0.54
"1, -1"	73	603	2	5	0.822	0.178	0.292	0.57
"1, -1"	133	543	2.5	25	0.744	0.293	0.42	0.596
"1, -1"	141	535	2.5	20	0.73	0.305	0.43	0.596
"1, -1"	144	532	2.5	15	0.729	0.311	0.436	0.598
"1, -1"	165	511	2.5	10	0.733	0.358	0.481	0.614
"1, -1"	187	489	2.5	5	0.717	0.396	0.51	0.62
"1, -1"	221	455	3	25	0.719	0.47	0.569	0.643
"1, -1"	227	449	3	20	0.714	0.479	0.573	0.643
"1, -1"	235	441	3	15	0.719	0.5	0.59	0.652
"1, -1"	243	433	3	10	0.724	0.521	0.606	0.661
"1, -1"	263	413	3	5	0.719	0.559	0.629	0.67
"1, -1"	286	390	3.5	25	0.71	0.601	0.651	0.678
"1, -1"	293	383	3.5	20	0.71	0.615	0.659	0.682
"1, -1"	299	377	3.5	15	0.706	0.624	0.662	0.682
"1, -1"	309	367	3.5	10	0.689	0.63	0.658	0.673
"1, -1"	326	350	3.5	5	0.684	0.66	0.672	0.678

Tabla A.2: Resultados distancia Euclidiana 15 componentes

clusters	Fake	NoFake	eps	minPts	Precision	Recall	F1	Accuracy
-1	0	676	0.5	5		0		0.5
-1	0	676	2	25		0		0.5
-1	0	676	1.5	25		0		0.5
-1	0	676	1	25		0		0.5
-1	0	676	0.5	25		0		0.5
-1	0	676	2	20		0		0.5
-1	0	676	1.5	20		0		0.5
-1	0	676	1	20		0		0.5
-1	0	676	0.5	20		0		0.5
-1	0	676	1.5	15		0		0.5
-1	0	676	1	15		0		0.5
-1	0	676	0.5	15		0		0.5
-1	0	676	2	15		0		0.5
-1	0	676	1.5	10		0		0.5
-1	0	676	1	10		0		0.5
-1	0	676	0.5	10		0		0.5
-1	0	676	1	5		0		0.5
"1, -1"	7	669	1.5	5	0.857	0.018	0.035	0.507
"1, -1"	20	656	2	10	0.85	0.05	0.095	0.521
"1, -1"	28	648	2	5	0.893	0.074	0.137	0.533
"1, -1"	51	625	2.5	25	0.804	0.121	0.211	0.546
"1, -1"	57	619	2.5	20	0.789	0.133	0.228	0.549
"1, -1"	68	608	2.5	15	0.779	0.157	0.261	0.556
"1, -1"	82	594	2.5	10	0.805	0.195	0.314	0.574
"1, -1"	102	574	2.5	5	0.765	0.231	0.355	0.58
"1, -1"	149	527	3	25	0.745	0.328	0.456	0.608
"1, -1"	157	519	3	20	0.752	0.349	0.477	0.617
"1, -1"	160	516	3	15	0.75	0.355	0.482	0.618
"1, -1"	170	506	3	10	0.729	0.367	0.488	0.615
"1, -1"	224	452	3.5	25	0.705	0.467	0.562	0.636
"1, -1"	225	451	3.5	20	0.707	0.47	0.565	0.638
"1, -1"	230	446	3.5	15	0.713	0.485	0.577	0.645
"1, -1"	242	434	3.5	10	0.702	0.503	0.586	0.645
"1, -1"	254	422	3.5	5	0.713	0.536	0.611	0.66

Tabla A.3: Resultados distancia Euclidiana 20 componentes

Bibliografía

[1] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[2] Helena Gómez-Adorno, Yuridiana Alemán, Darnes Vilariño Ayala, Miguel A Sanchez-Perez, David Pinto, and Grigori Sidorov. Author clustering using hierarchical clustering analysis. In *CLEF (Working notes)*, 2017.

[3] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36, 2017.

[4] Carlos Rodríguez Pérez. No diga fake news, di desinformación: una revisión sobre el fenómeno de las noticias falsas y sus implicaciones. *Comunicación*, (40):65–74, 2019.

[5] Seyedmehdi Hosseinimotlagh and Evangelos E Papalexakis. Unsupervised content-based identification of fake news articles with tensor decomposition ensembles. In *Proceedings of the Workshop on Misinformation and Misbehavior Mining on the Web (MIS2)*, 2018.

[6] Kasra Majbouri Yazdi, Adel Majbouri Yazdi, Saeid Khodayi, Jingyu Hou, Wanlei Zhou, and Saeed Saedy. Improving fake news detection using k-means and support vector machine approaches. *International Journal of Electronics and Communication Engineering*, 14(2):38 – 42, 2020.

[7] Juan-Pablo Posadas-Durán, Helena Gómez-Adorno, Grigori Sidorov, and Jesús Jaime Moreno Escobar. Detection of fake news in a

new corpus for the spanish language. *Journal of Intelligent & Fuzzy Systems*, 36(5):4869–4876, 2019.

[8] Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2. 02 2008.

[9] Víctor Germán Mijangos de la Cruz, 2017. Capítulo 1 Introducción, Capítulo 2 Introducción a la Lingüistica.

[10] D. Sarkar. *Text Analytics with Python: A Practitioner's Guide to Natural Language Processing*. Apress, 2019.

[11] Ronen Feldman, Ronen, Sanger, and James. *The text mining handbook: Advanced approaches in analyzing unstructured data*. 01 2007.

[12] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval: Scoring, term weighting, and the vector space model. 2008.

[13] Nltk · nltk documentation.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] Lemmatizer · spacy api documentation.

[16] Charu C. Aggarwal and ChengXiang Zhai. *A Survey of Text Classification Algorithms*, pages 163–222. Springer US, Boston, MA, 2012.

[17] Pavel Pudil and Jana Novovičová. *Novel Methods for Feature Subset Selection with Respect to Problem Knowledge*, pages 101–116. Springer US, Boston, MA, 1998.

[18] Laurens van der Maaten, Eric O. Postma, and Jaap van den Herik. Dimensionality reduction: A comparative review. 2008.