Universidad del Bío-Bío, Chile

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Selección de métodos de Machine Learning para Generar una Base de Datos de Complejos Proteína-Proteína con Estructura Tridimensional desde Textos Biológicos

Tesis presentada por Jonathan Gatica
para obtener el grado de Magíster en Ciencias de la Computación
Dirigida por Tatiana Gutiérrez
Mónica Caniupán

2023

Agradecimientos

Quiero darle las gracias a mis docentes guía por su constante apoyo, y a mis cercanos por su contención.

Abstract

Las proteínas participan en una gran cantidad de procesos biológicos permitiendo la activación de funciones en los seres vivos. Debido a distintas circunstancias químico-físicas se generan interacciones entre proteínas denominadas *complejos proteína-proteína*. Es posible consultar coordenadas de complejos proteína-proteína desde la base de datos de estructuras tridimensionales de proteínas denominada *Protein Data Bank*. Sin embargo, esta base de datos no contiene todos los *complejos proteína-proteína*. En esta tesis se propone el uso de un lexicón de intensidad formulado específicamente para identificar interacciones entre proteínas en el texto, como una nueva representación de documentos. Además, se utilizan algoritmos de clasificación para crear diversos modelos mediante la combinación de diferentes representaciones de documentos, con el fin de obtener una mayor cantidad de información y resultados más precisos. Esto permite generar un listado de interacciones de proteínas a partir de los modelos de clasificación desarrollados.

Keywords — Interacción proteína-proteína, estructura de proteínas, complejos proteína-proteína, Machine Learning, minería de texto, lenguaje natural, Lexicón de intensidad

Índice general

Índice de Figuras

Índice de Tablas

Listado de acrónimos

SVM Support Vector Machine.

IPP Interacción Proteína Proteína.

TF IDF Term frequency – Inverse document frequency.

KNN K-Nearest Neighbors.

NB Naive Bayes.

RF Random Forest.

ML Machine Learning.

DL Deep Learning.

CNN Convolutional Neural Network.

RNN Recurrent Neural Network.

LSTM Long Short-Term Memory.

GRU Gated Recurrent Unit.

WEMB Word Embedding.

RMSD Root-mean-square deviation.

Log LR Log-likelihood ratio.

Acc Accuracy.

Corpus Conjunto de textos biológicos ya clasificado.

Capítulo 1

Introducción

Las interacciones entre proteínas (IPP) juegan un papel fundamental en una gran cantidad de procesos biológicos como la replicación del ADN y las funciones estructurales, además de prácticamente todos los procesos de una célula. Entre algunas aplicaciones del estudio de las IPP se encuentran el diseño de tratamiento de enfermedades y la comprensión de procesos celulares. El estudio de interacciones entre proteínas es de gran relevancia, siendo su análisis experimental un proceso costoso y examinar los textos que contienen información acerca de IPP consume una gran cantidad de tiempo. Por esta razón, recurrir al análisis masivo de datos, mediante herramientas como la minería de datos y Machine Learning para encontrar estas interacciones en los textos biológicos es una alternativa mucho más eficiente respecto a costos. (Braga et al., 2009; Luscombe et al., 2001).

La minería de texto se basa en la recuperación de información, aprendizaje automático, estadísticas y lingüística computacional, por lo que puede aplicar en múltiples disciplinas. La minería de texto utiliza diversas técnicas tales como, árboles de decisión, redes neuronales, algoritmo Naive Bayes, regresión logística, Máquinas de Vectores Soporte, entre otras (Srivastava y Sahami, 2009). Por otro lado, Machine Learning o aprendizaje automático es un área de la inteligencia artificial, que se centra en la implementación de algoritmos basados en modelos matemáticos y/o estadísticos para identificar tendencias o patrones de interés en conjuntos de datos. Los algoritmos de Machine Learning pueden ser utilizados para clasificar documentos biológicos que contengan interacciones entre proteínas.

En este trabajo se describen algoritmos de clasificación que se basan en técnicas de minería de texto, Machine Learning y Procesamiento de Lenguaje Natural, sobre textos biológicos para la obtención de interacciones de proteínas.

Para el funcionamiento de los algoritmos se necesita datos. Esta información puede obtenerse al integrar Corpus, que son conjuntos de datos no necesariamente previamente clasificados. Lo que se busca es usar los corpus para el proceso de aprendizaje del algoritmo, método que se ha observado en la revisión de diferentes algoritmos que realizan procesos similares. La utilización de los corpus en el entrenamiento de los algoritmos trabajados se detalla en la sección 4. Por otro lado, también se crea un nuevo corpus basado en los existentes y en los resultados obtenidos de los algoritmos que se implementaron.

A partir de los corpus se obtienen textos científicos etiquetados que se utilizan co-

mo base para generar un Lexicón de intensidad orientado a la búsqueda de interacciones proteína-proteína (IPP). El lexicón de intensidad es una herramienta emergente, con un gran potencial recientemente creado como se detalla en (Segura-Navarrete et al., 2021). En esta ocasión, se modificó para adaptarlo a los objetivos de esta tesis. Una vez creado, se utiliza como una nueva representación de documentos, junto con otras herramientas como Word embedding y TF-IDF. Estas herramientas permiten filtrar y organizar la información de los corpus para que pueda ser procesada y aplicar los algoritmos de clasificación recientemente mencionados. Luego de esto, se lleva a cabo un proceso de Docking para obtener el listado de pares de proteínas que se busca.

El acoplamiento molecular, también conocido como Docking, es una técnica computacional utilizada para predecir cómo se unirán dos o más moléculas para formar un complejo estable, ya que explora las posibles orientaciones y posiciones de los ligandos de las proteínas para descubrir la manera óptima de unirse. (Ver Figura 2.2), Este enfoque se usa con frecuencia en biología molecular e investigación de fármacos para pronosticar la estructura y la energía de las interacciones entre proteínas y ligandos o entre diferentes moléculas. (Trott y Olson, 2010)

Según lo mencionado en (de Vries et al., 2006) Este método utiliza funciones de puntuación, las que se encargan de evaluar la forma de la estructura tridimensional contando la cantidad de interacciones positivas, las que pueden ser según los parámetros interacciones hidrofóbicas o de hidrógeno.

1.1. Planteamiento del problema

Las interacciones entre proteínas son esenciales para numerosos procesos biológicos, como: señalización celular, replicación del ADN, plegamiento y ensamblaje de proteínas, contracción de los músculos, catalizar reacciones bioquímicas para el metabolismo, entre otros procesos (Pawson y Nash, 2003). Dado esto, es fundamental para los científicos encontrar información que sugiera la posible interacción entre dos proteínas determinadas, ya que puede conducir a importantes descubrimientos.

En sintonía con facilitar la investigación de IPP en ciencias biológicas, esta tesis propone la implementación de algoritmos de clasificación basados en técnicas de minería de texto, procesamiento de lenguaje y algoritmos de Machine Learning, para la búsqueda de interacciones entre proteínas, sobre grandes conjuntos de textos biológicos. Este enfoque puede permitir analizar una cantidad significativa de información y extraer conocimientos útiles para el campo de la biología molecular. De acuerdo a que el objetivo de esta tesis es incrementar el actual conjunto de *complejos proteína-proteína* con su estructura tridimensional, las contribuciones de la tesis son:

- Hacer un análisis actual de los algoritmos de minería de texto.
- Proponer mejoras y aportes para la búsqueda de interacciones entre proteínas en el texto.

- Proponer un diccionario de palabras claves para identificar interacciones de proteínas, a partir del lexicón de intensidad generado.
- Proponer modelos eficientes basados en minería de texto, procesamiento de lenguaje y técnicas de Machine Learning, que permitan la identificación de interacciones de proteínas sobre grandes conjuntos de textos biológicos.
- Generar un listado de *complejos proteína-proteína* con estructuras tridimensionales conocidas, que incorporen las estructuras del Protein Data Bank.

1.2. Hipótesis

Es posible identificar *complejos proteína-proteína* con una estructura tridimensional conocida, mediante la implementación de algoritmos de clasificación basados en técnicas de minería de texto, Procesamiento de Lenguaje Natural, y Machine Learning.

1.3. Objetivos

Generar una base de datos ampliada de *complejos proteína-proteína* validando la estructura tridimensional de complejos y proteínas de forma independiente, mediante la implementación de algoritmos de clasificación basados en técnicas de minería de texto, Machine Learning y Procesamiento de Lenguaje Natural sobre textos biológicos.

La investigación propuesta tiene como objetivos específicos los siguientes:

1. Analizar algoritmos para la búsqueda de *complejos proteína-proteína* en la literatura.
2. Implementar algoritmos de clasificación para la detección de interacciones entre proteínas.
3. Evaluar la eficiencia resultante de experimentar en la clasificación de textos con una interacción entre proteínas presentes en ellos.
4. Generar listado de *complejos proteína-proteína* con estructura tridimensional.
5. Validar *complejos proteína-proteína* con métodos de docking.

1.4. Alcance de la investigación

Esta investigación se enfoca en la aplicación de algoritmos de clasificación para la identificación de interacciones entre proteínas sobre textos biológicos y la creación de un listado de complejos proteína-proteína.

Para el análisis de texto se consideran las base de datos de proteínas UniProt (Consortium, 2015) y el Protein Data Bank (PDB) (Berman et al., 2000). Se consideraron 10,000 textos obtenidos de Pubmed Central (PMC) y este proyecto tiene la capacidad de utilizar

tanto los textos completos del PMC, como los abstract de Pubmed. Para el proceso de aprendizaje de los algoritmos se utilizan los siguientes corpus de interacciones entre proteínas; AIMed (Bunescu et al., 2005), Biocreative3 (Arighi et al., 2011) , HPRD50 (Fundel et al., 2007b), IEPA (Ding et al., 2002), LLL (Nédellec, 2005).

A partir de las proteínas obtenidas de la interacción y posterior a su validación en Protein Data Bank, se valida el uso de aplicaciones de docking para la creación del listado de *complejos proteína-proteína*.

1.5. Metodología de trabajo

Se describen a continuación las tareas que contempla la metodología de trabajo desarrollada en esta tesis.

- Revisión exhaustiva de la literatura respecto a:
  - Métodos de identificación de interacciones de proteína.
  - Técnicas de clasificación utilizadas en minería de texto y Machine Learning para la búsqueda de patrones en textos.
  - Modelos de interacción (docking) para seleccionar el complejo adecuado a ser usado con el conjunto de complejos proteína-proteína identificados.
  - Corpus existentes de textos con interacciones entre proteínas y sus limitaciones.
- Búsqueda de repositorios de artículos del área biológica enfocados en proteínas.
- Generación de un lexicón de intensidad a partir de la información contenida en los corpus revisados, el cual se utiliza para la búsqueda de Interacciones Proteína-Proteína (IPP).
- Generar representación de documentos utilizando el lexicón de intensidad y otras formas de representación. Esto permite filtrar y organizar la información de los corpus para que pueda ser procesada de manera más efectiva.
- Aplicación de algoritmos de clasificación para identificar los datos.
- Desarrollar modelos que combinen el uso de algoritmos y representaciones de documentos para mejorar la precisión y eficacia del proceso de Identificación de IPP en textos biológicos.
- Aplicar los modelos desarrollados para clasificar el texto y detectar los pares de proteínas, utilizando las herramientas generadas en los pasos anteriores.
- Realización de Docking para predecir la formación de posibles complejos proteína-proteína, mediante el acoplamiento de los pares de proteínas detectados.
- Evaluación de resultados e interpretación de los experimentos realizados para validar hipótesis y elaborar conclusiones.

1.6. Organización de tesis

El presente documento de tesis de magíster se organiza en seis capítulos, en los que se describen el trabajo realizado en diferentes secciones dentro de cada uno de ellos.

Capítulo 1. Se presenta una introducción de la problemática y su definición, además de los objetivos de esta tesis.

Capítulo 2. Se dan a conocer los conceptos básicos biológicos necesarios para comprender la problemática de la búsqueda de interacciones entre proteínas, además de conceptos de minería de texto explicando su proceso y técnicas utilizadas.

Capítulo 3. Se expone el estado del arte y se describen los algoritmos de clasificación que se utilizan como referencia para la creación del algoritmo propuesto.

Capítulo 4. Se describe el diseño de modelos, el conjunto de datos destinados para el entrenamiento y pruebas de este, y las representaciones de texto utilizadas por el mismo.

Capítulo 5. Se analiza el proceso de experimentación con los modelos propuestos y se describen los resultados obtenidos por la investigación.

Capítulo 6. Se analizan y describen los resultados obtenidos por las aplicaciones de docking en la validación del listado de complejos proteína-proteína.

Conclusión Se destacan las conclusiones obtenidas en base al análisis de los resultados descritos en el capítulo 6 y a la validación de los objetivos e hipótesis de esta tesis.

Capítulo 2

Marco Teórico

Este capítulo reúne los conceptos teóricos necesarios como base para dar a entender lo abordado y utilizado en esta investigación. Se contemplan primeramente los conceptos de Ciencias Biológicas relacionados con este estudio y conceptos de Procesamiento de Lenguaje Natural y Machine Learning.

2.1. Conceptos Biológicos

Las interacciones entre proteínas cumplen un rol fundamental en diversos procesos fisiológicos, por lo que son ampliamente estudiadas. El presente capítulo es una síntesis del conocimiento necesario para el emprendimiento de esta investigación, definiendo qué son las proteínas, sus interacciones, bases de datos donde se registra esta información, especificación del acoplamiento molecular (Docking) y por último herramientas informáticas para realizar este proceso de acoplamiento.

2.1.1. Proteínas

Las proteínas tienen una gran cantidad de funciones en los seres vivos, son consideradas como herramientas biológicas debido a que determinan la forma y la estructura de las células, además de dirigir la mayoría de los procesos biológicos (Karp, 2009) . Las proteínas están formadas por la unión de aminoácidos por medio de enlaces covalentes llamados enlaces peptídicos. Por lo general llamamos proteína a la unión de más de 50 aminoácidos. Por ejemplo, la proteína insulina es una molécula proteica con 51 aminoácidos, mientras que la apolipoproteína B (una proteína que transporta colesterol) contiene 4.536 aminoácidos (Bastarrachea et al., 2005). Entre las variadas funciones que pueden realizar las proteínas se encuentran: dar resistencias a las estructuras tales como el pelo, las uñas y la piel; transportar moléculas o iones a través de membranas o células, entre otras.

Las proteínas se estructuran en diferentes niveles de plegamiento, catalogándose como: estructura primaria, estructura secundaria, estructura terciaria y estructura cuaternaria (Seguí, 2011). Estas dos últimas poseen una estructura tridimensional. A continuación se describen estos niveles en detalle:

- **Estructura primaria**: La secuencia de aminoácidos de una proteína es su estructura primaria, la cual determina su estructura tridimensional, como en el caso de la insulina.

- **Estructura secundaria**: La estructura secundaria de una proteína es la conformación tridimensional de segmentos cortos, como hélices alfa o láminas beta, estabilizada por enlaces de hidrógeno y cargas parciales de los aminoácidos.

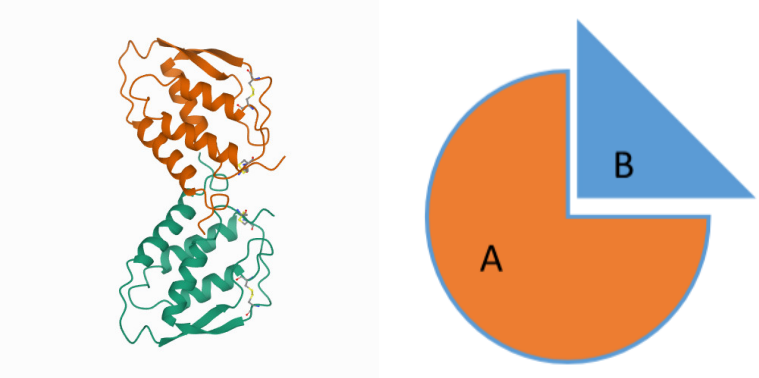
- **Estructura terciaria**: La estructura terciaria de una proteína es su disposición tridimensional completa, determinada por las interacciones entre residuos de aminoácidos distantes. Ejemplos son la mioglobina, el colágeno y la hemoglobina.

- **Estructura cuaternaria**: La estructura cuaternaria es la organización tridimensional de proteínas formadas por dos o más subunidades. La hemoglobina es un ejemplo con cuatro subunidades alfa y beta, y las interacciones entre ellas son importantes para su estabilidad y función.

2.1.2. Interacciones entre proteínas - IPP

Las IPP pueden ser entre dos o más proteínas, y estas interactúan debido a las características de cada una y al entorno en que se encuentran. Por ejemplo, en Figura 2.1.a), se puede ver que proteína A, interactúa con proteína B. Las interacciones que se producen entre dos o más proteínas se denominan *complejos proteína-proteína*, los cuales cuando existe su estructura tridimensional, pueden ser almacenados y registrados en la base de datos Protein Data Bank (Berman et al., 2000), la cual almacena una gran variedad de ácidos nucleicos y proteínas. Estas IPP pueden ser de diversos tipos, que difieren según su composición, la afinidad y del tiempo que estas interactúan (Mintseris y Weng, 2005; Nooren y Thornton, 2003). En la Figura 2.1.b) muestra la representación de un complejo en su estructura tridimensinal.

Los estudios se enfocan en la zona donde las proteínas interaccionan, debido a que uno de los principales problemas para entender y clasificar las interacciones proteína-proteína es la caracterización de sus interfaces y superficie de interacción (Caffrey et al., 2004).



(a) Estructura tridimensional complejo proteico1CSG[1]

(b) Complejos proteína-proteína

Figura 2.1: Proteína en su estructura tercearia y participación en un complejo.)

2.1.3. Bases de datos de proteínas

Existe un gran número de bases de datos especializadas en el registro de organismos y moléculas biológicas, entre estas bases de datos hay un grupo encargado en el registro de proteínas y su información relevante como su estructura tridimensional o su secuencia de aminoácidos, además de referencias de dónde son mencionadas. En la Tabla 2.1 se señalan algunas de las bases de datos más relevantes, y cabe destacar que se trabaja con las tres primeras.

2.1.4. Bases de datos de IPP

Las bases de datos de interacciones entre proteínas (IPP) se especializan en su registro mediante el análisis por especialistas, utilizando diferentes fuentes de datos para la obtención de nuevos resultados. En la Tabla 2.2 se pueden ver las principales bases de datos de IPP.

2.1.5. Acoplamiento molecular - Docking

Según lo mencionado en (de Vries et al., 2006), para resolver la problemática de predecir la precisión de los *complejos proteína-proteína* a partir de estructuras tridimensionales, se utiliza el método conocido como acoplamiento molecular o Docking (Ver Figura 2.2), su funcionamiento consiste en determinar la afinidad de enlace entre dos proteínas, tales como moléculas de proteínas, ácidos nucleicos, carbohidratos o lípidos. Este método utiliza funciones de puntuación, las que se encargan de evaluar la forma de la estructura tridimensional contando la cantidad de interacciones positivas, las que pueden ser interacciones hidrofóbicas o de hidrógeno, según los parámetros.

Base de datos	**Descripción**	**Tamaño (Marzo 2023)**
Uniprot	Base de datos enfocada al coleccionismo de información sobre cómo funcionan las proteínas. Teniendo en cuenta el orden de citas, entre otras cosas, y aminoácidos. El Swiss Bioinformatics Institute, el European Bioinformatics Institute y Protein Information Resource, estos trabajan juntos para formar el Consorcio UniProt, el cual es responsable de mantención de esta base de datos. https://www.uniprot.org/	292 000
PDB	Protein Data Bank, es una base de datos que realiza un seguimiento de la estructura tridimensional de proteínas y aminoácidos. Esta base de datos es mantenida por la organización Worldwide Protein Data Bank (wwPDB). http://www.rcsb.org/	202.292
MMDB	Base de datos de modelamiento molecular (Molecular Modeling Database) es una base de datos experimental de estructura tridimensional de moléculas biológicas. MMDB es mantenida por el Centro Nacional de Información Biotecnológica (NCBI). https://www.ncbi.nlm.nih.gov/Structure/MMDB/mmdb.shtml	200.000
SCOP	Base de datos de clasificación estructural de proteínas (Structural classification of Proteins) es una base de datos utilizada para la clasificación que utiliza similitud estructural y secuencias de aminoácidos. La base datos SCOP es mantenida por el equipo SCOP de la Universidad de Cambridge (Reino Unido). http://scop.mrc-lmb.cam.ac.uk/	861.631

Tabla 2.1: Bases de datos de proteína

Base de datos	**Descripciones**	**Tamaño (Marzo 2023)**
HPID	Base de datos de interacciones entre proteínas humanas la que está formada por la bases de datos HPRD, BIND y DIP. La mantención de esta base de datos es realizada por el Molecular Interaction Lab de la Universidad de California en Los Ángeles (UCLA). http://wilab.inha.ac.kr/hpid/	11.487
Int Act	Base de datos de código abierto, además de una herramienta para el análisis de interacciones entre proteínas. Esta base de datos es mantenida por el Laboratorio Europeo de Biología Molecular - Instituto Europeo de Bioinformática (EMBL-EBI) en el Reino Unido. https://www.ebi.ac.uk/intact/home/	5.565.271
MINT	Base de datos de interacciones entre moléculas biológicas. Se centra en experimentar y verificar las interacciones entre proteínas, a partir de textos biológicos curados por expertos. La base de datos MINT ya no se mantiene y sus datos se fusionaron con los de IntAct. https://mint.bio.uniroma2.it/	133.087
STRING	Base de datos de interacciones proteína-proteína que integra sus datos de interacciones de diversas fuentes, una gran cantidad de organismos y asociaciones funcionales, además de físicas. La base de datos STRING es mantenida por una agrupación de socios académicos y de la industria. La Universidad de Zúrich y la Universidad Técnica de Dresden son las principales instituciones académicas que trabajan en el mantenimiento de esta base de datos. https://string-db.org/	20.000 millones

Tabla 2.2: Bases de datos IPP

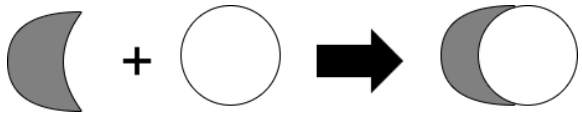


Figura 2.2: Diagrama de ejemplo del acoplamiento molecular entre dos proteínas

2.1.6. Herramientas de docking

A lo largo de los años grupos de investigación han desarrollado diversas herramientas informáticas para la implementación de simuladores de acoplamiento molecular y para validar las interacciones entre proteínas. En la Tabla 2.3 se puede ver algunos de estos softwares para docking con su respectiva url.

2.1.7. Métricas de docking

Existen diversas métricas de evaluación para valorar o evaluar el acoplamiento. Las que se utilizaron en esta investigación son:

Herramienta	Descripción	Referencia
pyDockWeb	Es una herramienta de docking basada en la web que permite calcular la energía de unión entre dos moléculas y pronosticar cómo se unirán e interactuarán a nivel molecular. https://life.bsc.es/pid/pydockweb/	Jiménez-García et al., 2013
PatchDock	Es una herramienta que predice con rapidez y precisión las estructuras tridimensionales de los complejos proteína-proteína y proteína-ligando a partir de sus estructuras. https://bioinfo3d.cs.tau.ac.il/PatchDock/	Schneidman-Duhovny et al., 2005
GRAMM	Con esta aplicación de docking se puede analizar la energía libre de las interacciones entre ligandos y proteínas. Produce modelos tridimensionales precisos de estructuras moleculares utilizando técnicas de búsqueda y optimización. https://gramm.compbio.ku.edu/	Tovchigrechko y Vakser, 2006
Hex	Es una aplicación de docking útil para examinar las interacciones entre moléculas pequeñas. Emplea un método de búsqueda de emparejamiento y se puede utilizar para analizar cómo interactúan los ligandos y las proteínas. http://hexserver.loria.fr/	Macindoe et al. , 2010
PEPPI	Es una aplicación de docking que utiliza documentos o proteínas en formato FASTA y se caracteriza por retornar un resultado binario, sobre si existeo no una IPP en el acoplamiento. https://zhanggroup.org/PEPPI/	Bell et al., 2022

Tabla 2.3: Herramientas de docking

- Desviación cuadrática media (RMSD): Como lo indica su nombre, indica un valor correspondiente a la media aritmética de la desviación cuadrática entre la posición de unión determinada experimentalmente y la posición de unión predicha para el ligando. Esta puede calcularse con la Ecuación 2.1. Es posible calcular el RMSD sin contar con la estructura nativa de referencia comparando entre sí las demás estructuras (Torres et al., 2019).

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[(x_i - y_i)^2 + (y_i - y_i^{'})^2 + (z_i - z_i^{'})^2\right]} \tag{2.1}$$

  Siendo:

  N el número de átomos de la alineación.

  x_i, y_i, y z_i son las coordenadas x, y, y z del del átomo i en la estructura de referencia.

  $y_i^{'}$ y $z_i^{'}$ son las coordenadas y y z del átomo i en la estructura de prueba, que es alineado con la estructura de referencia.

- Log-likelihood ratio (Log LR): El logaritmo de razón de probabilidad o log(LR) es una medida que indica la probabilidad de que un complejo formado sea un complejo proteína-proteína genuino. El valor de log(LR) se aplica para evaluar la calidad de las predicciones de acoplamiento proteína-proteína y puede calcularse según lo expuesto en la ecuación 2.2 Cuando se calcula el log(LR) sin un complejo nativo como

referencia, se compara la energía de enlace del complejo predicho con la energía de enlace esperada de una estructura generada aleatoriamente (Read et al., 2023).

$$log(LR) = -log(1 + (RMSD \times 2/2)) \tag{2.2}$$

- Probabilidad de unión: En docking se hace referencia a la probabilidad de unión, como la probabilidad de que 2 moléculas se unan de forma estable como un complejo. Y se puede calcular utilizando el valor de Log(LR) como se detalla en la Ecuación 2.3.

$$\text{Probabilidad de unión} = 10^{log(LR)} \tag{2.3}$$

2.2. Minería de Datos - Data mining

La minería de datos es una técnica utilizada en la investigación científica para analizar de manera rápida y efectiva grandes conjuntos de datos y revelar patrones y tendencias ocultas que de otro modo podrían pasar desapercibidas. La minería de datos se usa en muchos campos, incluida la inteligencia artificial, la estadística, la ingeniería, entre otros. La minería de datos es el proceso de identificación de patrones, tendencias y relaciones en conjuntos de datos considerables, utilizando para esto técnicas y algoritmos estadísticos y de aprendizaje automático. (Witten y Frank, 2002).

Algunas áreas en las que se aplica comúnmente la minería de datos son conocidas y desarrolladas como se detalla a continuación:

- **Genómica:** La minería de datos se puede utilizar para analizar grandes conjuntos de datos sobre genoma, incluidas secuencias de ADN y expresión génica, para identificar patrones y relaciones que pueden ayudar a comprender mejor la función genética y la biología celular.
- **Proteómica:** La minería de datos se puede utilizar para analizar grandes conjuntos de datos de proteínas y péptidos, incluida la información de espectrometría de masas, para identificar patrones y relaciones que pueden ayudar a comprender mejor la función y la estructura de las proteínas.
- **Pronóstico y diagnóstico de enfermedades:** Mediante la extracción de datos, se pueden analizar grandes conjuntos de datos clínicos, incluida información sobre pacientes, resultados de pruebas y tratamientos, para encontrar patrones y relaciones que puedan mejorar el diagnóstico. y el pronóstico de una enfermedad.

2.2.1. Minería de texto

La minería de datos se concentra en datos estructurados y analiza los datos utilizando métodos y algoritmos estadísticos, mientras que la minería de texto se concentra en datos no estructurados y utiliza métodos de procesamiento de lenguaje natural para extraer información pertinente de los datos de texto. Los tipos de datos analizados son la principal

diferencia entre la minería de texto y la minería de datos. Las bases de datos, hojas de cálculo y tablas que contienen datos estructurados son el foco principal de la minería de datos.

Como método de análisis de datos, la minería de textos tiene como objetivo obtener conocimiento e información de colecciones de texto. Las aplicaciones de procesamiento de lenguaje natural, como el análisis de sentimientos, la clasificación de documentos y la extracción de información, utilizan con frecuencia la minería de texto (Gémar y Jiménez-Quintero, 2015).

La Minería de texto es un método que extrae información de manera automática, con el objetivo de aplicarse en grandes conjuntos de texto. Una gran problemática de estos algoritmos al utilizarse en artículos biológicos es que estos poseen diferentes estilos de información, ejemplo de esto sería la diferencia entre artículos científicos e informes clínicos. Entre los conceptos fundamentales a conocer al estudiar la minería de texto se encuentran: (Manning, 2009)

- **Tokenización:** Es el proceso de dividir un texto en distintas unidades, como palabras o frases.
- **Lematización:** Implica condensar una palabra a su forma más simple (el lema) para facilitar el análisis y la comparación de palabras.
- **Stemming:** Es el proceso de despojar a una palabra de sus sufijos y conjugaciones, como ***s*** o ***ing***, para devolverla a su forma original. Por ejemplo, ***amistades*** se cambiaría por ***amistad***.
- **Part-of-speech tagging:** Es el proceso de identificar cada palabra en un texto según su función gramatical, como un adjetivo, verbo o sustantivo.

2.2.2. Algoritmos de clustering

El Clustering es una técnica de minería de datos utilizada para examinar la estructura de los datos y encontrar patrones. Los algoritmos de clustering o de agrupamiento, son algoritmos que tienen como objetivo agrupar información no etiquetada según la similitud entre los datos utilizados. Estos algoritmos son no supervisados debido a que no se utilizan en datos clasificados, además que al generar grupos se tiende a perder información relevante. Algunas de las aplicaciones de los algoritmos de agrupamiento son para generar categorías de palabras que tengan una gran similitud o para su uso en diagnósticos médicos. A continuación se menciona K-means, uno de los algoritmos de este tipo el cual va a ser utilizado, y una métrica de utilidad.

2.2.2.1. K-means

El algoritmo K-means, también conocido como agrupamiento de k medias, es un método de aprendizaje no supervisado que se utiliza para clasificar datos no etiquetados en K

grupos. El algoritmo se basa en la creación aleatoria de K puntos, que se ajustan sucesivamente hasta que se logra la mejor agrupación posible. La Figura 2.3 ilustra este proceso, donde las subfiguras 2.3.a) y 2.3.b) muestran combinaciones aleatorias no óptimas de los K puntos, mientras que la subfigura 2.3.c) representa la agrupación óptima de los datos. Cabe destacar que el algoritmo se ajusta en cada iteración en función del valor de K.

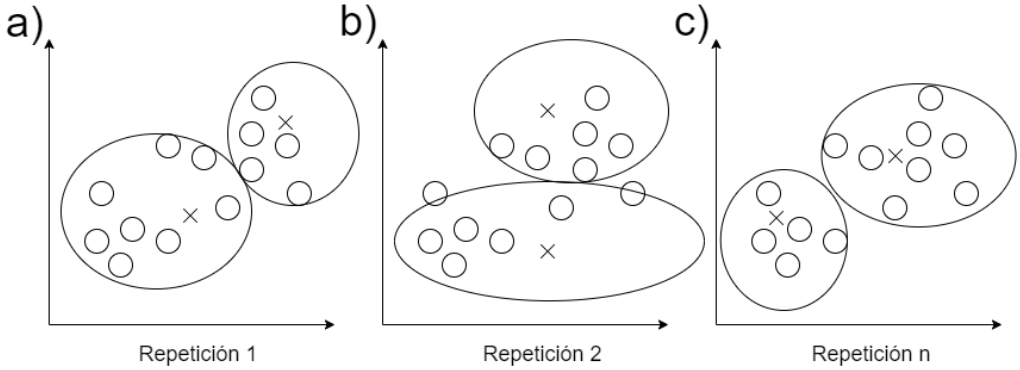


Figura 2.3: Ejemplo de creación de grupos con el uso de K-means

2.2.2.2. Métrica de inercia

La medida de inercia es una métrica que se utiliza en algoritmos de clustering para evaluar la calidad de un agrupamiento. La inercia es esencialmente la suma de las distancias al cuadrado entre cada punto en el conjunto de datos y el centroide del grupo al que pertenece. La inercia mide cuánto se están alejando los puntos del centroide de su grupo. (Hastie et al., 2009) Al utilizarla con algoritmos de Clustering, el objetivo es minimizar la inercia, es decir, agrupar los puntos para que la suma de las distancias al cuadrado de los puntos al centroide sea lo más pequeña posible. Es una métrica comúnmente utilizada en el algoritmo k-means con el objetivo es minimizar la inercia en cada iteración hasta obtener una solución ideal. (Hastie et al., 2009)

Otra forma de verlo es que la métrica de inercia indica el número ideal de grupos en que debería distribuir el algoritmo de agrupamiento.

Para una comprensión más clara de esta métrica, a continuación se presenta su formulación matemática junto con un ejemplo que ilustra cómo funciona la métrica de inercia.

Suponiendo que X es una matriz de frecuencias de términos por documentos, donde cada fila corresponde a un término y cada columna corresponde a un documento específico. Podemos denotar la frecuencia del término i en el documento j como f_{ij}. Por lo tanto, la matriz X se puede representar como $X = [f_{ij}]_{i,j}$. Entonces la métrica de inercia se define de la siguiente manera:

$$I(x_k, x_l) = \sum_{i=1}^{n} \frac{(f_{ik} - f_{il})^2}{f_{i\cdot}},$$

- x_k y x_l son dos términos de la matriz X.
- $f_{i\cdot}$ es la frecuencia total del término i en todos los documentos.

Para ejemplificar un caso de agrupamiento de palabras con alta relación entre sí, supongamos que contamos con las siguientes palabras y sus respectivas frecuencias en tres documentos:

Palabra	Documento 1	Documento 2	Documento 3
hogar	3	1	2
erizo	2	3	1
hamster	1	2	2

Por lo tanto, la matriz de frecuencias X quedaría conformada de la siguiente manera:

$$X = \begin{bmatrix} 3 & 1 & 2 \\ 2 & 3 & 1 \\ 1 & 2 & 2 \end{bmatrix}$$

El cálculo de la métrica de inercia entre las palabras "hogar" y "erizo" se llevaría a cabo de la siguiente manera:

$$I(\text{hogar}, \text{erizo}) = \frac{(3-2)^2}{6} + \frac{(1-3)^2}{6} + \frac{(2-1)^2}{5} \approx 0{,}93$$

Mientras que el cálculo de la métrica de inercia entre las palabras "hogar" y "hamster" se llevaría a cabo de la siguiente manera:

$$I(\text{hogar}, \text{hamster}) = \frac{(3-1)^2}{6} + \frac{(1-2)^2}{6} + \frac{(2-2)^2}{5} \approx 1{,}2$$

Podemos notar que la métrica de inercia entre "hogar" y "hamster" es mayor que la que existe entre "hogar" y "erizo". Esto sugiere que "hogar" y "hamster" están más estrechamente relacionados que "hogar" y "erizo".

2.3. Conceptos de machine learning

A continuación se describen algunas métricas, algoritmos de clasificación y de agrupamiento que se desprenden del Machine Learning y que son de utilidad durante el desarrollo de esta investigación.

2.3.1. Representación de los documentos

Los algoritmos de inteligencia artificial no entienden el lenguaje de forma natural en el texto, por lo que debemos entregarles representaciones o formatos que puedan procesar. Los formatos más usados para los algoritmos de Machine Learning son TF-IDF o Bag of words, mientras que de Deep Learning utilizan Word Embedding. A continuación se describen estos formatos.

- **Bag of words**: El método Bag of words es utilizado para el Procesamiento de Lenguaje Natural para representar los documentos. Se basa en un diccionario que contiene una bolsa de palabras la cual va a representar el texto (Tsai, 2012). En la Tabla 2.4 se puede observar como es representado el texto de las siguientes 3 oraciones utilizando el enfoque de Bag of words. La tabla contiene todas las palabras que aparecen en las 3 oraciones y luego se procede al llenado, donde es 1 cuando la palabra aparece y 0 cuando esta no aparece.

  1. Mi casa es bonita.
  2. Mi casa es grande y tiene muchos animales.
  3. Mi casa tiene un jardín muy grande.

	animales	**bonita**	**casa**	**es**	**grande**	**jardin**	**mi**	**muchos**	**muy**	**tiene**	**un**	**y**
1	0	1	1	1	0	0	1	0	0	0	0	0
2	1	0	1	1	1	0	1	1	0	1	0	1
3	0	0	1	0	1	1	1	0	1	1	1	0

Tabla 2.4: Ejemplo bag of words

- **Lexicón de intensidad**:

  Un lexicón de intensidad es un conjunto de palabras claves con un determinado puntaje que al ser sumado establece si el texto tiene una connotación positiva o negativa (Segura-Navarrete et al., 2021). En la Tabla 2.5 se puede observar un conjunto de palabras con un puntaje asociado a la emoción de sorpresa.

Palabras	Intensidad
idea	10
urgencia	28
abatimiento	34
absorto	28

Tabla 2.5: Ejemplo lexicón de intensidad de sorpresa (Segura-Navarrete et al., 2021).

Para ejemplificar el uso del enfoque de lexicón de intensidad en la creación de un vector de características, es necesario comprender que el análisis realizado por (Segura-Navarrete et al., 2021) involucra la evaluación de distintos sentimientos, cada uno con una puntuación correspondiente. Estos sentimientos incluyen Enojo, Anticipación, Disgusto, Miedo, Alegría, Tristeza, Sorpresa y Confianza, así como también una lista de palabras negativas.

A modo de ejemplo, se utilizará la oración Çuanta gente sin trabajo y el ctm de Kramer rindiéndole culto a los delincuentes", cuya evaluación se muestra en la Tabla 2.6. Las primeras ocho columnas de la tabla corresponden a la suma de las puntuaciones de las palabras de la oración según cada uno de los sentimientos. La columna 9 indica la proporción de palabras negativas encontradas dentro de la oración en relación al total de palabras. Por último, la columna 10 muestra el número total de palabras en la oración.

Enojo	Anticipación	Disgusto	Miedo	Alegría	Tristeza	Sorpresa	Confianza	MP/CP	CP
62	0	0	46	64	0	0	46	0.16	6

Tabla 2.6: Ejemplo de un vector de características utilizando lexicón de intensidad

- **TF-IDF**:

  Es un método del Procesamiento de Lenguaje Natural, al igual que el Bag of Words, que representa los documentos a partir de la frecuencia de ocurrencia de cada término, en este caso palabra. Este método permite representar lo importante de cada palabra dentro del documento, y así facilita que se clasifiquen estos de mejor manera. En las siguientes Ecuaciones se puede observar como se obtiene la puntuación de cada palabra. En la Ecuación 2.4 se indican los valores necesarios para dar una puntuación de importancia de cada una de las palabras. t es la palabra, d es el documento/oración.

  1. *TF-IDF(t,d)*: La multiplicación de la frecuencia de una palabra dentro de una oración ($TF(t,d)$), por la frecuencia de la palabra dentro de un conjunto de oraciones ($IDF(t)$).
  2. $IDF(t)$: El logaritmo del número total de documentos $|D|$, dividido por el número de oraciones que contienen t ($|d \in D : t \in d|$).

$$TF - IDF(t,d) = TF(t,d) \times IDF(t) \tag{2.4}$$

$$IDF(t) = \log\left(\frac{|D|}{|d \in D : t \in d|}\right) \tag{2.5}$$

Se puede realizar un ejemplo de TF-IDF con el siguiente conjunto de oraciones:

1. ”Mi casa es bonita.”

2. ”Mi casa es grande y tiene muchos animales.”
3. ”Mi casa tiene un jardín muy grande.”

El funcionamiento de TF-IDF se puede explicar en los siguientes pasos:

1. Crear una matriz de términos-documentos: se construye una tabla que contiene las palabras únicas de los documentos en las filas y los documentos en las columnas.
2. Calcular la frecuencia de términos (TF): se cuenta cuántas veces aparece cada palabra en cada documento. Por ejemplo, en la primera oración ”Mi casa es bonita”, la palabra ”casa” aparece una vez, mientras que la palabra ”bonita” aparece una vez.
3. Calcular la frecuencia inversa de documentos (IDF): se calcula el número total de documentos y se divide por el número de documentos que contienen cada palabra. Para el ejemplo, la palabra ”casa” aparece en todos los documentos, por lo que su IDF es $\log(3/3) = 0$. La palabra ”jardín” aparece en un solo documento, por lo que su IDF es $\log(3/1) = 1.1$.
4. Calcular TF-IDF: se multiplica la frecuencia de términos (TF) por la frecuencia inversa de documentos (IDF) para obtener un valor que indica la importancia de cada palabra en cada documento. Por ejemplo, en la tercera oración ”Mi casa tiene un jardín muy grande”, el valor de TF-IDF para la palabra ”jardín” es 1 (TF) x 1.1 (IDF) = 1.1, mientras que el valor de TF-IDF para la palabra ”casa” es 1 (TF) x 0 (IDF) = 0, ya que aparece en todos los documentos.

Los valores de TF-IDF pueden ser útiles para identificar las palabras más relevantes en un documento o conjunto de documentos. En el ejemplo anterior, la palabra ”jardín” es considerada como la palabra más relevante en el tercer documento debido a su valor alto de TF-IDF.

- **Word embedding**: Word Embedding es un método de Procesamiento de Lenguaje Natural que representa las palabras en un espacio vectorial, por lo que cada documento es representado por un conjunto de vectores que representan cada una de las palabras. Estos vectores representa la relación entre el diccionario de palabras utilizado en el entrenamiento (Mikolov et al., 2013).

  En la Figura 2.4 se grafica un conjunto de palabras dentro de un espacio vectorial. Algunas tienen mayor relación entre sí que otras. Por ejemplo “femenino” se ve más relacionado con “mujer” y “masculino” más relacionado con “hombre”. Similar a esto es como se registran internamente las palabras y se relacionan entre sí dentro de Word embedding, cada una se ubica en el espacio en una coordenada específica cercana a otra en el espacio, con la cual estaría relacionada.

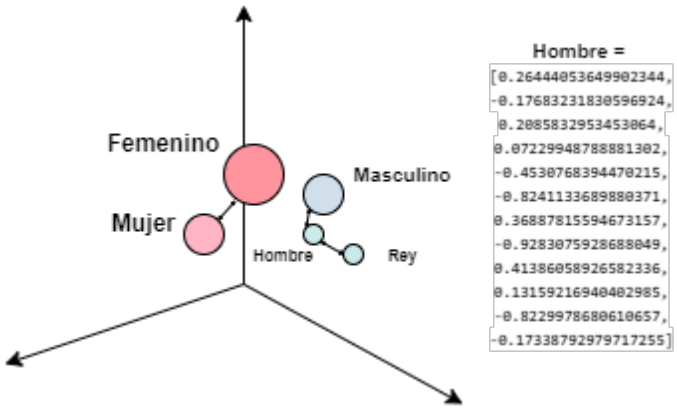


Figura 2.4: Ejemplo word embedding

Cada documento es representado en un espacio vectorial de características, compuesto por palabras y datos relevantes. La forma en que se hace la representación del documento se muestra en la Ecuación 2.6. Por otra parte el conjunto de datos utilizado por los métodos de minería de texto es representado en la matriz de la Ecuación 2.7 en la cual se puede ver las posibles clases en las que se puede clasificar un documento y sus características.

$$x = [x_1, x_2, ..., x_n]^t \tag{2.6}$$

$$matriz = \begin{bmatrix} \omega_1 & x_{1,1} & x_{1,2} & ... & x_{1,n} \\ \omega_2 & x_{2,1} & x_{2,2} & ... & x_{2,n} \\ ... & ... & ... & ... & ... \\ \omega_c & x_{m,1} & x_{m,2} & ... & x_{m,n} \end{bmatrix} \tag{2.7}$$

- x: Es un vector que representa cada documento de texto.
- n: Representa las diferentes características de cada documento (palabras y datos relevantes).
- t: Es el vector transpuesto que representa el listado de documentos.
- ω_c : Representa las clases en las que se puede clasificar el documento, las categorías en este caso son si el documento contiene o no una interacción entre proteínas.
- m: Representa un documento en específico.

2.3.2. Algoritmos de clasificación

Los algoritmos de clasificación se pueden explicar como la búsqueda de un patrón entre las diferentes valores de las características presentes en un vector que haga referencia a una clase, en este caso, si en el texto se menciona una interacción entre proteínas. Estos algoritmos de clasificación en Machine Learning y minería de datos funcionan en dos etapas. La primera consiste en el entrenamiento del modelo, donde se ingresa un conjunto de datos de entrada al modelo previamente clasificadas para la creación de un modelo. La segunda etapa consiste en realizar las pruebas del modelo entrenado, utilizando un conjunto de datos ya clasificados, mediante la verificación de la respuesta (Kowsari et al., 2019).

Existen dos tipos de aprendizaje: supervisado y no supervisado. En el aprendizaje supervisado, se le proporciona al algoritmo preguntas o características y sus correspondientes respuestas etiquetadas, las cuales son utilizadas para entrenar el modelo. Luego, el algoritmo combina esta información y realiza predicciones. En el aprendizaje no supervisado, se entregan únicamente las características sin proporcionar etiquetas. En el caso de la agrupación, la función del aprendizaje no supervisado es clasificar por similitud y crear grupos, sin especificar previamente los integrantes del grupo.

A continuación, se presenta un ejemplo de aprendizaje supervisado en el análisis de sentimientos para la clasificación de sentimiento en comentarios de redes sociales.

1. Para construir el modelo se requiere un conjunto de datos etiquetados que contenga comentarios y sus correspondientes etiquetas de sentimiento (positivo o negativo).

2. Se aplican técnicas de procesamiento de lenguaje natural para extraer características relevantes de los comentarios, tales como la presencia de palabras o frases comunes asociadas con sentimientos positivos o negativos.

3. El modelo de aprendizaje supervisado se entrena utilizando un algoritmo clasificador y el conjunto de datos etiquetados, junto con las características extraídas.

4. El modelo aprende a clasificar nuevos comentarios en función de su similitud con las reseñas del conjunto de datos de entrenamiento y sus correspondientes etiquetas de sentimiento.

5. Una vez entrenado, el modelo se utiliza para clasificar nuevos comentarios y predecir su sentimiento, como clasificar automáticamente un comentario como positiva o negativa en función de su contenido.

Algunos de los algoritmos más utilizados para la clasificación de textos son los siguientes: K-vecinos más cercanos, árboles de decisión, redes neuronales, máquinas de soporte vectorial (SVM) y Naive Bayes. A continuación se presenta una breve descripción de estos algoritmos.

Por otra parte, la técnica de clustering de documentos es un ejemplo de aprendizaje no supervisado en clasificación de textos. En esta técnica, se utiliza un algoritmo de agrupamiento para dividir un conjunto de documentos en grupos basados en su similitud, lo que se describe a continuación:

1. Se comienza con el preprocesamiento de los documentos mediante técnicas de procesamiento de lenguaje natural, como la eliminación de stopwords, la lematización y la eliminación de signos de puntuación. Posteriormente, se utiliza un modelo de vectorización de palabras para representar cada documento en forma de vector numérico.

2. Después, se emplea un algoritmo de agrupamiento, como por ejemplo K-means, para agrupar los documentos en distintos grupos según su similitud. El objetivo del algoritmo es maximizar la similitud entre los documentos dentro de cada grupo y minimizar la similitud entre los grupos.

3. Después de agrupar los documentos, se pueden etiquetar los grupos según su contenido. Si los documentos son artículos, se pueden asignar etiquetas como "ecología", "genética", "biodiversidad", etc.

Se puede concluir que los distintos tipos de aprendizaje son útiles para abordar diversas problemáticas según la necesidad.

2.3.2.1. K-vecinos más cercanos

De acuerdo a (Kubat, 2017) el algoritmo K-vecinos más cercanos (K-Nearest Neighbour) puede clasificar un elemento en una categoría x o y dependiendo del número de características similares con otros elementos pertenecientes a las categorías. Cada elemento es representado con un punto en el espacio y el vecino más cercano, de acuerdo a la distancia euclidiana calculable con el teorema de pitágoras, define una mayor similitud a un elemento. Para un análisis más robusto se puede identificar no sólo un vecino, sino varios. A esto se refiere el K que es el número de vecinos a considerar.

Para abordar la problemática de minería de texto en la búsqueda de interacciones entre proteínas y en la clasificación de textos, es posible emplear el algoritmo KNN de la siguiente manera: Primero se proporciona un conjunto de entrenamiento S, que consiste en n textos etiquetados, cada uno representado por un vector de características x_i y su correspondiente etiqueta y_i. Luego, el modelo KNN (k-nearest neighbors) se utiliza para predecir la etiqueta de un nuevo texto x a partir de los vecinos más cercanos en el conjunto de entrenamiento.

El algoritmo KNN para la clasificación de texto se basa en encontrar los k vecinos más cercanos al nuevo texto en el conjunto de entrenamiento, utilizando una métrica de distancia, como la distancia euclidiana. La etiqueta del nuevo texto se predice a partir de la mayoría de las etiquetas de sus vecinos más cercanos.

El algoritmo KNN para la clasificación de texto se fundamenta en encontrar los k vecinos más cercanos al nuevo texto en el conjunto de entrenamiento. Para medir la cercanía entre los vectores de características, se utiliza una métrica de distancia, como la distancia euclidiana. Posteriormente, la etiqueta del nuevo texto se predice basándose en la mayoría de las etiquetas de sus vecinos más cercanos, utilizando la misma métrica de distancia para calcular su similitud. De esta forma, el algoritmo KNN clasifica el nuevo texto en función de la mayoría de las etiquetas de sus k vecinos más cercanos en el conjunto de

entrenamiento, empleando la métrica de distancia para calcular la cercanía de los vectores de características. A continuación se puede observar la ecuación de la distancia euclidiana:

$$d(x_i, x_j) = \sqrt{\sum_{m=1}^{M} (x_{im} - x_{jm})^2}$$

Para clasificar un nuevo texto x, se calcula su distancia a cada uno de los textos de entrenamiento en S utilizando una métrica de distancia, como la distancia euclidiana, donde M es el número de características del vector. Se seleccionan entonces los k vecinos más cercanos, y la etiqueta de x se determina por mayoría de votos de las etiquetas de los vecinos más cercanos, es decir, se asigna la etiqueta más frecuente entre los vecinos más cercanos. Esto se puede expresar de la siguiente manera:

$$y = \arg\max_{c} \sum_{i=1}^{k} [y_i = c]$$

1. y es la etiqueta asignada a x
2. c es una posible etiqueta
3. $[y_i = c]$ es una función indicadora que toma el valor de 1 si la etiqueta de y_i es igual a c y 0 en caso contrario.

Un ejemplo del uso de KNN se puede observar a continuación:

1. Se tiene un conjunto de entrenamiento S con tres textos etiquetados:

$$S = \{(\begin{bmatrix}1\\0\end{bmatrix}, 0), (\begin{bmatrix}0\\1\end{bmatrix}, 1), (\begin{bmatrix}2\\2\end{bmatrix}, 1)\}$$

2. Se quiere clasificar un nuevo texto $x = \begin{bmatrix}1\\1\end{bmatrix}$ utilizando el algoritmo KNN con $k = 2$.

3. Se calcula la distancia euclidiana entre x y cada uno de los textos en S como:

$$d(x, \begin{bmatrix}1\\0\end{bmatrix}) = \sqrt{(1-1)^2 + (1-0)^2} = 1$$

$$d(x, \begin{bmatrix}0\\1\end{bmatrix}) = \sqrt{(1-0)^2 + (1-1)^2} = 1$$

$$d(x, \begin{bmatrix}2\\2\end{bmatrix}) = \sqrt{(1-2)^2 + (1-2)^2} = \sqrt{2} \approx 1{,}41$$

4. Los dos vecinos más cercanos del nuevo texto $x = \begin{bmatrix}1\\1\end{bmatrix}$ son $(\begin{bmatrix}1\\0\end{bmatrix}, 0)$ y $(\begin{bmatrix}0\\1\end{bmatrix}, 1)$, ambos a una distancia de 1 unidad.

5. Como ambos tienen etiquetas diferentes, la etiqueta del nuevo texto x se determina por mayoría de votos de las etiquetas de los vecinos más cercanos, lo que en este caso resulta en una clasificación indeterminada.

2.3.2.2. Árboles de decisión

A diferencia de otros algoritmos de clasificación, en los árboles de decisión no se reciben los valores de los atributos al mismo tiempo. El clasificador puede elegir mejor los atributos de uno a la vez, inducido por un conjunto de reglas. Los elementos ingresados al algoritmo poseen una cantidad de atributos que permiten realizar la clasificación. Estos algoritmos pueden trabajar con datos tabulados como el ejemplo en la Tabla 2.7 (Kubat, 2017), la cual incluye tres atributos, que son tamaño del borde del elemento geométrico, la forma del elemento geométrico y el tamaño del relleno, los cuales son relevantes para determinar la clase etiquetada con *positivo* y *negativo*. Los nodos intermedios del árbol representan un control (pregunta) sobre el valor de un atributo, los arcos indican como proceder de acuerdo al resultado del test sobre el atributo, las hojas contienen las etiquetas de la clase. El resultado del test sobre el valor de un atributo permite descender en el árbol, hasta alcanzar un nodo hoja.

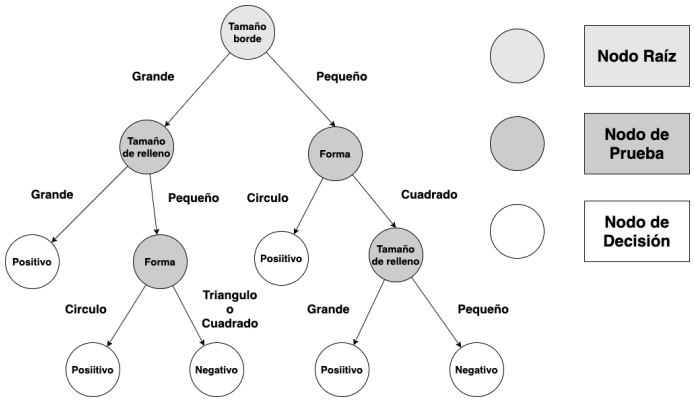


Figura 2.5: Ejemplo árbol de decisión

En la Figura 2.5 se visualiza cómo se lleva a cabo la clasificación de los elementos en la Tabla 2.7. Se utiliza como nodo raíz el tamaño del borde, los nodos intermedios que

permiten descender a las hojas del árbol son los atributos de forma y tamaño del relleno. A medida que se baja en el árbol, se observan los resultados de los tests que permiten describir el patrón para determinar la clase del elemento (positivo o negativo). Inicialmente si el tamaño del borde y relleno de la Figura es grande la clase del nodo hoja es positivo, pero si el relleno es pequeño el valor podría ser positivo o negativo, dependiendo del atributo de forma. Si el tamaño del borde es pequeño, se verifica la forma, si es circular la clase es positiva, de lo contrario, si es cuadrado, se considera el atributo del tamaño del relleno para determinar su clase.

Elemento	**Tamaño borde**	**Forma**	**Tamaño del relleno**	**Clase**
e_1	grande	circulo	pequeño	positivo
e_2	pequeño	círculo	pequeño	positivo
e_3	grande	cuadrado	pequeño	negativo
e_4	grande	triángulo	pequeño	negativo
e_5	grande	cuadrado	grande	positivo
e_6	pequeño	cuadrado	pequeño	negativo
e_7	pequeño	cuadrado	grande	positivo
e_8	grande	círculo	grande	positivo

Tabla 2.7: Clasificación de elementos geométricos (Kubat, 2017)

A continuación se explicará cómo se puede utilizar el algoritmo de Random Forest (RF) para clasificar textos. Para ello, se requerirá de los siguientes elementos:

- Un conjunto de entrenamiento D, el cual estará compuesto por n textos y p características.
- Un vector de etiquetas Y, que asigna una clase a cada una de los textos en D. Cada etiqueta Y_i indicará la clase a la que pertenece la observación i.
- El número de árboles T que se incluirán en el modelo de RF.

El proceso del algoritmo Random Forest (RF) consta de los siguientes pasos:

1. Para construir e l modelo de RF, se inicia seleccionando una muestra aleatoria con reemplazo de tamaño m de todo el conjunto de datos de entrenamiento D. A este conjunto de datos se le llama conjunto de entrenamiento bootstrap, el cual se utilizará para construir cada árbol de decisión en el modelo.
2. Para cada árbol t en el modelo de RF, se construye un árbol de decisión utilizando una muestra aleatoria de $p' < p$ características, seleccionadas de manera aleatoria y sin reemplazo.

3. Para cada nodo en el árbol, se elige la característica y punto de corte que maximizan la ganancia de información o reducen la impureza en el conjunto de datos de entrenamiento.

4. Se realiza la repetición de los pasos 1 a 3 para construir T árboles de decisión independientes. Cada árbol se construye de manera independiente, utilizando una muestra de entrenamiento de bootstrap y seleccionando un subconjunto aleatorio de características en cada nodo del árbol. La aleatoriedad en la selección de los datos y las características ayuda a reducir el sobreajuste y mejorar la generalización del modelo.

5. La clasificación de una nueva observación x en el modelo de RF se realiza aplicando cada uno de los T árboles construidos en el paso anterior y asignando la clase más frecuente entre los T resultados obtenidos. Es decir, se hace una votación entre los árboles y se elige la clase con mayor frecuencia como la clase predicha para x.

La predicción de una observación x con el modelo de RF se puede expresar matemáticamente como sigue:

$$\hat{Y}(x) = \text{mode}\{Y_t(x)\}_{t=1}^{T}$$

1. $\hat{Y}(x)$ es la etiqueta predicha para la observación x.

2. mode es la función moda que devuelve la clase más frecuente entre los T árboles de decisión en el modelo de RF.

3. $Y_t(x)$ es la etiqueta predicha para el texto x utilizando el árbol t.

2.3.2.3. Máquinas de Vectores de Soporte - SVM

El algoritmo SVM (Support Vector Machine) es utilizado para clasificación y se destaca por el uso de métodos de kernel. Este algoritmo se basa en un conjunto de entrenamiento para construir un modelo que pueda predecir la clase o categoría de una nueva muestra.

En una SVM, las clases son separadas en dos espacios, y se busca maximizar la distancia entre ellas (conocida como margen) mediante la creación de un hiperplano que funciona como la línea de separación entre estas clases. Los puntos que conforman las dos líneas paralelas al hiperplano son llamados vectores de soporte.

El modelo construido por SVM es utilizado para clasificar nuevas entradas, tomando en cuenta los espacios a los que pertenecen. Un ejemplo de una SVM con clases A y B se muestra en la Figura 2.6, donde el hiperplano se genera considerando el punto de la clase A que está más cercano a un punto de la clase B (Ng, 2017).

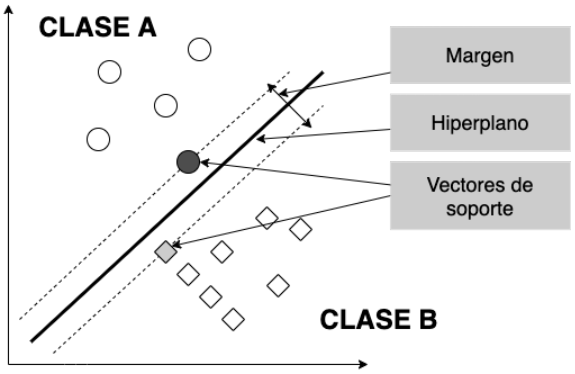


Figura 2.6: Ejemplo máquina de soporte vectorial

Para clasificar textos que puedan contener interacciones entre proteínas, se utiliza la formulación matemática del algoritmo SVM. En esta metodología, se dispone de un conjunto de datos de entrenamiento $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, donde $\mathbf{x}_i$ es un vector de características de dimensión d correspondiente al i-ésimo texto, y $y_i \in \{0, 1\}$ es su etiqueta de clase correspondiente. El objetivo es encontrar un hiperplano que permita la separación de los datos en dos clases. Este modelo busca el hiperplano que maximiza la distancia entre los puntos más cercanos de las dos clases, también conocidos como vectores de soporte. La ecuación que define el hiperplano es la siguiente:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

1. $\mathbf{w} \in \mathbb{R}^d$ es el vector de pesos.

2. b es el sesgo o término de intersección con el eje $y = 0$.

El problema de encontrar el hiperplano óptimo en el modelo SVM se puede expresar como un problema de optimización:, que se puede expresa de la siguiente forma:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{sujeto a } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i$$

1. $\|\mathbf{w}\|^2$ representa la norma euclidiana del vector de pesos.

La función de decisión que se utiliza para clasificar un nuevo texto $\mathbf{x}$ se define como:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

1. La función sign se define como la función signo.

La función signo es una función matemática que devuelve un valor positivo (+1) si su argumento es positivo, un valor negativo (-1) si su argumento es negativo y cero si su argumento es cero. En la función de decisión utilizada para clasificar un nuevo texto, se emplea la función signo para determinar la clase a la que se asigna el texto en función del valor del argumento $\mathbf{w}^T\mathbf{x} + b$, el cual se evalúa como positivo o negativo.

2.3.2.4. Naive bayes - Clasificador bayesiano ingenuo

Naive Bayes es un algoritmo estadístico simple pero poderoso, es útil para ser usado sobre pequeños y grandes conjuntos de datos, utilizado en problemáticas de clasificación. Una ventaja de este con respecto a otros modelos, es que requiere una pequeña cantidad de datos de entrenamiento. Este algoritmo se basa en el teorema de Bayes (Bayes, 1763) ilustrado en la Ecuación 2.8, donde h y D son eventos.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{2.8}$$

$P(h)$**:** Probabilidad de que la hipótesis h sea cierta (independiente de los datos).

$P(D)$**:** Probabilidad de los datos D (independiente de la hipótesis).

$P(h|D)$**:** Probabilidad de la hipótesis dada los datos D.

$P(D|h)$**:** Probabilidad de los datos D dado que la hipótesis h es cierta.

Este método es considerado un clasificador ingenuo debido a que si las características de la información entregada al modelo está interconectada, éste las considera características independientes.

2.3.2.5. Métricas de evaluación de algoritmos de clasificación

En esta tesis se trabaja con modelos supervisados, como lo son los algoritmos de clasificación. Por lo tanto, se presentan las principales métricas para este tipo de aprendizaje.

Las métricas de evaluación nos permiten saber si los modelos son correctos. Estas métricas utilizan los valores verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN) y falsos negativos (FN) de una matriz de confusión para sus evaluaciones, obteniendo un porcentaje resultante. Una matriz de confusión es una herramienta estadística en la que se utilizan sus parámetros en métricas de evaluación de algoritmos de clasificación en Machine Learning. Según la definición de (Kubat, 2017) los parámetros de la matriz de confusión son:

- Verdaderos Positivos (TP): El clasificador reconoce correctamente un elemento positivo.
- Verdadero Negativo (TN): El clasificador reconoce correctamente un elemento negativo.
- Falso Negativo (FN): El clasificador etiqueta al elemento como negativo cuando es positivo.
- Falso Positivo (FP): El clasificador etiqueta al elemento como positivo cuando es negativo.

La Figura 2.7 muestra la tabulación de los valores de una matriz de confusión, elaborada en base a (Kubat, 2017) tal como lo anteriormente descrito.

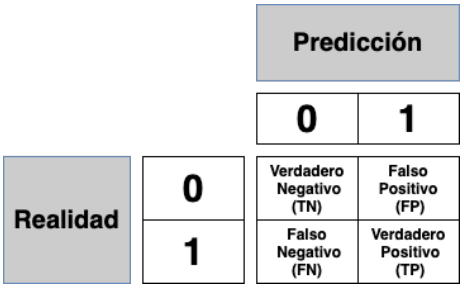


Figura 2.7: Matriz de confusión

Algunas de las métricas a utilizar son: accuracy, precision, recall y $F1$-score, las que se definen a continuación (Kubat, 2017), donde TP, TN, FP y FN se definen de acuerdo a la matriz de confusión en Figura 2.7.

- Accuracy: indica la exactitud del modelo, es decir, el número de elementos clasificados correctamente. Entre más cercano esté el resultado a 1 indica una mejor clasificación. Esta medida se representa en la Ecuación 2.9.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.9}$$

- Precision: valida la capacidad del modelo de no clasificar un elemento en una categoría equivocada. Esta medida se representa en la Ecuación 2.10.

$$precision = \frac{TP}{TP + FP} \tag{2.10}$$

- Recall: evalúa la capacidad del clasificador de detectar los elementos de una categoría. Esta medida se representa en la Ecuación 2.11.

$$recall = \frac{TP}{TP + FN} \tag{2.11}$$

- $F1$-score: corresponde al promedio ponderado entre los valores de las métricas de precisión y recall. La idea es tener una métrica unificada, de tal manera de calibrar su importancia. Esta medida se representa en la Ecuación 2.12.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.12}$$

2.3.3. Deep learning

Una red neuronal es un modelo computacional que intenta emular el funcionamiento de una red neuronal biológica. Estas redes utilizan un gran número de unidades simples interconectadas para el desarrollo de problemáticas complejas (Kubat, 2017). Estas unidades son conocidas como neuronas artificiales. Según lo mencionado por Rosenblatt (Rosenblatt, 1958) las neuronas artificiales están compuestas por las señales de entrada, los pesos de cada entrada, una unión sumadora, una función de activación y la salida. La función de activación utilizada puede ser de distintos tipos, un ejemplo de esto es la función *sigmoide* la que realiza una suma ponderada de las entradas. La Figura 2.8 muestra la estructura básica de una red neuronal, la cual esta constituida por un numero n de capas con una función en específico para realizar el procesamiento de forma organizada, por ejemplo las capas ocultas, que se especializan en la obtención de información de las capas de entrada. Para la creación de esta imagen se tomó como referencia lo descrito en (Kubat, 2017).

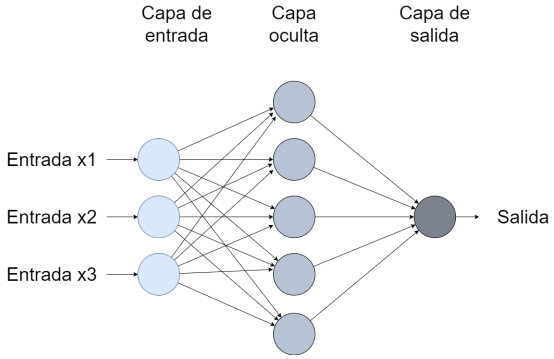


Figura 2.8: Estructura de una red neuronal artificial

Ya entendiendo lo que son las redes neuronales se puede dar paso a los algoritmos de Deep Learning, los cuales son un grupo de algoritmos de Machine Learning que busca realizar un análisis de los datos en un alto nivel de abstracción. Estas redes neuronales necesitan utilizar una representación de los datos, de esta manera se crean modelos que asimilen la representación de los datos entregados para dar un resultado. Existen diferentes arquitecturas de redes neuronales algunas de estas son las redes neuronales convolucionales (CNN), redes neuronales recurrentes (RNN) y Transformers. Entre las aplicaciones para las redes neuronales existen:

- Procesamiento de Lenguaje Natural/Natural language processing (NLP): Entendimiento de frases, Clasificación de frases y Traducción.
- Análisis de datos multimedia: Reconocimiento de voz/Speech recognition y Detección de objetos en imágenes o videos.
- Aplicaciones médicas.

Como anteriormente se mencionaba existen diferentes arquitecturas de redes neuronales, cada una de ellas especializada en el manejo de un tipo de estructura, entre estas hay algunas que se utilizan en el manejo de imágenes, manejo de texto o entre otras. A continuación se describen algunas de las arquitecturas de las redes neuronales de utilizadas para la minería de texto en la extracción de proteínas.

2.3.3.1. Redes neuronales convolucionales - CNN

Las redes CNN (Convolutional Neural Network) es un tipo de red neuronal basada en una red de un perceptrón multicapa, esta red está enfocada en el análisis (clasificación y segmentación) de imágenes, siendo el área con mejores resultados en el uso de esta red, pero también siendo utilizada para otras aplicaciones como el Procesamiento de Lenguaje Natural (NLP) o reconocimiento de voz. Esta red se caracteriza por el uso de capas que reducen el peso de entrada de los datos para un mayor entendimiento (extracción de características) y luego proceder a su reconstrucción (Peng y Lu, 2017). En la Figura 2.9 se puede observar un ejemplo del funcionamiento de este tipo de red, donde la Figura 2.9.a) representa una imagen completa la que es reducida en la Figura 2.9.b).

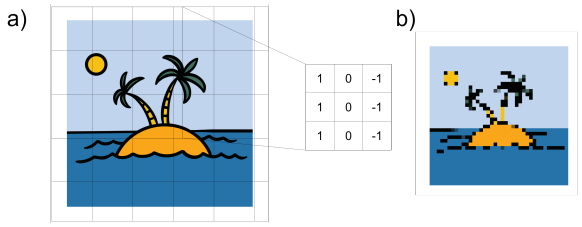


Figura 2.9: Ejemplo de una red convolucionales

La formulación matemática del modelo CNN (Convolutional Neural Network) para la clasificación de textos consiste en considerar un conjunto de entrenamiento $\{x_1, x_2, ..., x_n\}$ con etiquetas $\{y_1, y_2, ..., y_n\}$, donde x_i es un vector que representa el texto i y y_i es su correspondiente etiqueta (0 o 1). El objetivo del modelo CNN es aprender una función de clasificación $f(x)$ que minimice la pérdida en la clasificación de los datos de prueba.

En la mayoría de los casos, una CNN consta de una o varias capas convolucionales, capas de pooling, capas de activación y una capa completamente conectada. A continuación, se describen brevemente estas capas:

1. **Capa convolucional**: La capa convolucional es una de las capas principales de una CNN, y se encarga de realizar una operación de convolución entre una ventana de tamaño k y el vector de entrada x. Esta operación aplica filtros que se desplazan a lo largo de los datos de entrada y extraen características relevantes. La convolución se define matemáticamente como una operación entre una matriz de pesos (el filtro) y una sección de la entrada, y produce una salida que se utiliza como entrada a la siguiente capa. La que se define a continuación:

$$c_i = f(\sum_{j=1}^{k} w_j x_{i+j-1} + b)$$

a) w: Vector de pesos correspondiente a la ventana utilizada en la operación de convolución.

b) b: Sesgo utilizado en la misma operación.

c) f: Función de activación no lineal.

d) c_i: Resultado de la operación de convolución en la posición i.

2. **Capa de pooling**: La capa de pooling reduce la complejidad de la salida de la capa convolucional al elegir el máximo o el promedio dentro de una ventana de tamaño k.

3. **Capa de activación**: La capa de activación aplica una función no lineal a la salida de la capa de pooling, lo que permite al modelo aprender relaciones no lineales. En este contexto, una relación no lineal se refiere a una función matemática que no sigue una relación proporcional constante entre la entrada y la salida, es decir, que no se puede representar por una línea recta. La capa de activación es esencial en las redes neuronales convolucionales, ya que permite la extracción de características no lineales de los datos de entrada.

4. **Capa completamente conectada (FC)**: La capa completamente conectada utiliza las características extraídas por las capas anteriores para producir una salida final. Esta capa tiene conexiones entre todas las neuronas de la capa anterior y todas las neuronas de la capa siguiente. La salida final se utiliza como entrada en una función de pérdida que permite calcular la pérdida en la clasificación.

Para resumir, la función de clasificación $f(x)$ se obtiene al aplicar en conjunto las capas convolucional, de pooling, de activación y completamente conectada. En otras palabras, $f(x)$ es el resultado de la composición de estas capas y se puede expresar de la siguiente manera:

$$f(x) = softmax(W_2 h_2 + b_2)$$

- W_2 y b_2 son los parámetros de la capa completamente conectada.
- h_2 es la salida de la capa de activación.

2.3.3.2. Redes neuronales recurrentes - RNN

Las redes neuronales recurrentes/recurrent neural network es un tipo de red neuronal enfocada en el Procesamiento de Lenguaje Natural, ya que estas utilizan información secuencial en su funcionamiento, agregando información de la entrada previa a la nueva, permitiendo de esta manera el entendimiento del contexto de la frase. Este tipo de red tiene

una problemática con el tema de mantener la relevancia de una palabra, por ejemplo en la Figura 2.10 podemos ver como las palabras se relacionan entre si, el problema se distingue en la relación de las palabras ”expression” y ”laboratory”, perdiendo la importancia entre la relación de ambas palabras debido a la distancia entre ellas. (Goodfellow et al., 2016)

Existen dos tipos de variantes de RNN conocidas como Long short-term memory (LSTM) y Gated recurrent unit (GRU) que solucionan la problemática de la pérdida de relevancia de palabras/entradas. (Bengio et al., 1994; Cho et al., 2014).

La Figura 2.10 Explica cómo funciona las RNN. Se desarrolla la relación entre palabras de forma consecutiva, ya que para cada una se toma el valor de la palabra previamente analizada. El gran problema que podemos ver en este tipo de redes neuronales es que existe pérdida de información al pasarla de una palabra a otra, es decir, que se pierde la relación entre las primeras y últimas palabras de la red.

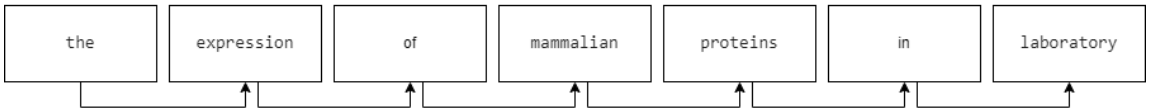


Figura 2.10: Ejemplo relación de las palabras en una RNN

- La arquitectura de una red LSTM (Long Short-Term Memory) está compuesta por una celda de memoria y puertas de entrada, salida y olvido que regulan el flujo de información dentro de la red. La formulación matemática del modelo LSTM se expresa de la siguiente manera:

  Dado un ejemplo de texto de entrada $x = (x_1, x_2, ..., x_T)$, donde T es la longitud del texto, y una salida binaria $y \in 0, 1$, la red LSTM recibe como entrada la secuencia x y produce una salida $p(y|x)$ que representa la probabilidad de que el texto pertenezca a una de las dos clases.

  Se definen los vectores de estado de la celda de memoria en el tiempo t como primer paso:

  $$h_t = \text{estado oculto en el tiempo } t$$
  $$c_t = \text{estado de la celda de memoria en el tiempo } t$$

  Se procede a definir las puertas de la red neuronal.

  $$\begin{aligned} i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ \tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \end{aligned}$$

  Las matrices $W_i, U_i, b_i, W_f, U_f, b_f, W_o, U_o, b_o, W_c, U_c, b_c$ y los vectores de sesgo son parámetros que se aprenden durante el entrenamiento. La función σ representa la función sigmoide.

Las puertas controlan el flujo de información a través de la celda de memoria de la siguiente manera:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \tanh(c_t)$$

La operación $\odot$ representa el producto elemento a elemento. Para obtener la salida de la red, se aplica una capa densa.

$$y' = \text{softmax}(W_y h_T + b_y)$$

La capa densa utiliza la matriz de pesos W_y y el vector de sesgo b_y, mientras que la función de activación softmax se aplica para obtener la salida de la red.

La función de pérdida de entropía cruzada binaria se minimiza durante el entrenamiento del modelo entre la salida y' y la etiqueta verdadera y:

$$L(y', y) = -y \log(y') - (1 - y) \log(1 - y')$$

La red LSTM, en resumen, utiliza una celda de memoria y puertas de entrada, salida y olvido para controlar el flujo de información y generar una salida binaria para la clasificación de textos.

- La ecuación del modelo GRU (Gated Recurrent Unit) para la clasificación de textos se define de la siguiente manera:

  Considerando un conjunto de entrenamiento $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, donde x_i es el i-ésimo documento de texto y y_i es su respectiva clase binaria ($y_i \in \{0, 1\}$), el objetivo es crear un modelo que pueda clasificar nuevos documentos de texto con precisión.

  Para procesar secuencialmente el texto de entrada x_i a través de una capa oculta h_t, se utiliza la siguiente ecuación:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$\tilde{h_t} = \tanh(W x_t + U(r_t \odot h_{t-1}) + b_h)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t}$$

  A continuación, se presentan las definiciones de los símbolos utilizados en la formulación matemática del modelo GRU:

  1. Durante el entrenamiento, se ajustan las matrices de pesos W_z, U_z, W_r, U_r, W, y U en el modelo GRU.

2. b_z, b_r y b_h son vectores de sesgo.

3. σ es la función sigmoide.

4. $\odot$ es el producto componente a componente entre dos vectores.

La última capa oculta h_n se utiliza para calcular la salida del modelo, que se expresa como:

$$\hat{y} = \sigma(Vh_n + c)$$

La matriz de pesos V y el vector de sesgo c son utilizados para calcular la salida del modelo. Durante el entrenamiento, se utiliza la función de pérdida de entropía cruzada binaria:

$$L(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

La optimización se realiza mediante el método de descenso de gradiente estocástico (SGD) para actualizar los parámetros del modelo, siguiendo la siguiente regla:

$$\theta \leftarrow \theta - \alpha \nabla_\theta L$$

Los parámetros del modelo están representados por θ, mientras que la tasa de aprendizaje se denota como α. El descenso de gradiente estocástico (SGD) se utiliza para actualizar los parámetros del modelo mediante la siguiente regla, donde $\nabla_\theta L$ es el gradiente de la función de pérdida L con respecto a los parámetros del modelo.

2.3.3.3. Transformers

La arquitectura de redes neuronales Transformer se caracteriza por el uso un de mecanismo de atención, es comúnmente utilizada en el procesamiento natural debido a su uso de datos secuenciales al igual que las RNN. A diferencia de este tipo de redes no se manejan los datos en orden como sería normalmente en lenguaje natural, permitiendo una mejor extracción del contexto de la información ya que reduce los tiempos de entrenamiento utilizados en variantes de las RNN como LSTM y GRU. Esto permite la creación de modelos o inteligencias artificiales pre entrenadas para el análisis de texto como BERT y MEGATRON utilizados para la transferencia de aprendizaje (TL) en algunos algoritmos actuales por la falta de datos de entrenamiento (Devlin et al., 2018).

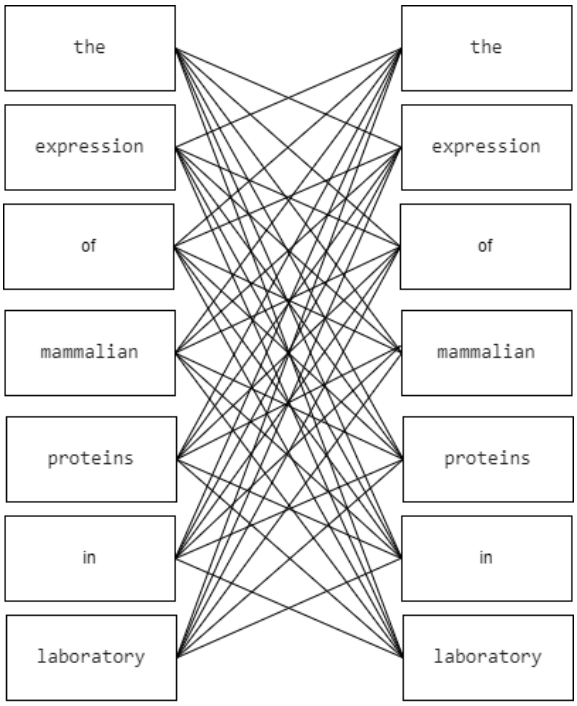


Figura 2.11: Ejemplo del funcionamiento de transformer

El ejemplo de la Figura 2.11 muestra de forma simplificada cómo se lleva a cabo la relación entre palabras con la arquitectura Transformer. A diferencia de las redes recurrentes, se busca la relación de una palabra con cada una de las palabras de la oración, entonces no existe una pérdida de relación entre la primera y última palabra de la oración.

La transferencia de aprendizaje (TL) es una técnica de Machine Learning que soluciona la falta de datos mediante la reutilización de datos previamente aprendidos en problemas similares. Al aplicar un modelo preentrenado con datos de artículos biológicos, se mejora la relación entre los conceptos. Los modelos más populares son BERT y MEGATRON, los cuales poseen sus versiones entrenadas con PubMed y PMC (Peng et al., 2019).

- BERT: Bidirectional Encoder Representations from Transformers (BERT) es modelo desarrollado por Google (Devlin et al., 2018) para su uso en el procesamiento del lenguaje natural. BERT posee distintas variantes entrenadas con diferentes conjuntos de datos para su mejor uso en las áreas del análisis de sentimientos o el análisis de textos biológicos.

- MEGATRON: Megatron es un modelo desarrollado por NVIDIA para la tarea de Procesamiento de Lenguaje Natural (NLP), debido a la gran masa de información utilizada en sus entrenamientos Megatron ha obtenido mejores resultados que BERT, Megatron al igual que BERT posee su versión orientada al análisis de textos Biológicos conocido como Bio Megatron.

En este capítulo se introdujeron conceptos biológicos y de bioinformática relevantes para el estudio de interacciones proteína-proteína, así como la minería de datos. Esta revisión bibliográfica es esencial para comprender los próximos segmentos de esta tesis. A continuación, se presenta el estado del arte actual.

Capítulo 3

Estado del arte

En este capítulo se presenta el proceso de búsqueda de artículos relacionados con algoritmos de extracción de IPP, donde se recopila el avance de las investigaciones y el contenido actualizado en el área. Para ello, se utilizó un conjunto de filtros que se detallan en las secciones siguientes para realizar una exhaustiva búsqueda en la bibliografía de las investigaciones.

3.1. Proceso de revisión

Para llevar acabo la revisión, se llevó acabo una búsqueda de los artículos más recientes sobre Minería de Texto para la extracción de proteínas en Google Scholar. Se utilizó la palabra clave ”text mining protein complex” y se aplicó un filtro de fecha desde 2017 hasta 2022, seleccionando los primeros 1000 artículos. La Figura 3.1 muestra el proceso llevado a cabo para filtrar los algoritmos considerados en la implementación de la tesis. Durante la búsqueda bibliográfica, se encontraron diversos algoritmos enfocados en la extracción de interacciones entre proteínas, los cuales utilizan diferentes clasificadores para su desarrollo. En la Tabla 3.1 se detalla la cantidad de algoritmos que utilizan los clasificadores más comúnmente empleados. Es notable que la Máquina de Vectorial Soporte (SVM) es ampliamente utilizada para este propósito.

Una vez identificados los textos con mayor cantidad de referencias, se realizó un análisis detallado de cada uno de ellos. Se resumen los resultados de este análisis en las Tablas F.1, F.2, F.3, F.4 y F.5 del anexo. En total, se encontraron 29 artículos relacionados con los temas de minería de texto e interacciones entre proteínas. Se seleccionaron los 10 artículos con mayor relevancia para la investigación, es decir, aquellos que mencionan el uso de algoritmos de clasificación para la detección de textos con posibles complejos de proteínas. Luego, se priorizó la selección de 4 artículos que utilizan redes neuronales y máquinas de soporte vectorial como algoritmos de clasificación.

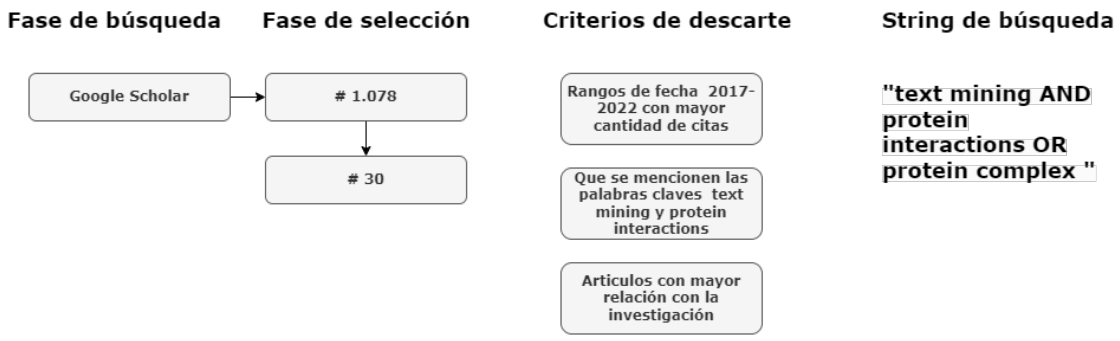


Figura 3.1: Proceso de revisión bibliográfica

Algoritmos de Clasificación	Cantidad
Support Vector Machine (SVM)	19
Naive Bayes (NB)	7
Decision Tree (DT)	5
Logistic Regression (LR)	1
Random Forest (RF)	1
K-Nearest-Neighbors (KNN)	3
Convolutional Neural Network (CNN)	2
Alineamiento de secuencias (Smith –Waterman)	3
Deep Recursive Neural Network (DRNN)	1
Propuestos	2

Tabla 3.1: Número de artículos encontrados según el algoritmo clasificador utilizado

3.2. Artículos principales

En la sección anterior se seleccionaron 30 artículos principales como referencia para el desarrollo del algoritmo propuesto en esta tesis. A continuación se describen en detalle.

3.2.1. Extracción de IPP con CNN - 2017

El algoritmo descrito en (Peng y Lu, 2017) utiliza información relevante como entrada de datos, tales como la distancia entre proteínas y palabras, y las dependencias entre pala-

bras. Aunque no se especifica el lenguaje o la biblioteca utilizada para su implementación, el pseudocódigo se describe detalladamente. Para la validación del algoritmo, se utilizaron los conjuntos de datos Aimed (Bunescu et al., 2005) y Bioinfer (Pyysalo et al., 2007), que son corpus estándar para la detección de interacciones entre proteínas. El texto de entrada se proporciona en formato de word embedding (vectores de palabras).

- **Input**:Se utilizan los conjuntos de datos AImed y BioInfer como valores de entrada en formato de word embedding de modelos pre-entrenados, como por ejemplo, word2vec. Estos modelos son útiles para capturar las relaciones semánticas entre las palabras presentes en el texto, lo que puede resultar muy beneficioso para la tarea de extracción de interacciones proteína-proteína.

- **Output**: El modelo produce una clasificación binaria para cada frase de entrada, donde una predicción de 1 indica la presencia de una interacción proteína-proteína (PPI) en la frase, mientras que una predicción de 0 indica su ausencia. Para evaluar el rendimiento del modelo, se utilizaron métricas de evaluación estándar como la precisión, la recuperación y la puntuación F1, que comparan las PPI predichas con las anotaciones de los corpus utilizados.

3.2.2. Biocppi extraction - 2018

El algoritmo Biocppi (Tran y Kavuluru, 2018) se basa en una red neuronal convolucional que utiliza textos preprocesados a través de Word embedding como datos de entrada. La implementación del algoritmo se realizó en Python utilizando la biblioteca de aprendizaje profundo TensorFlow para construir la red neuronal. Este método se destaca por crear su propio conjunto de datos procesados internamente, seleccionando artículos clasificados por Biocreative y abstract de la base de datos de artículos biológicos Pubmed.

- **Input**: Se utilizó el dataset BioCreative V CDR, el cual consta de 1,500 resúmenes de PubMed y 500 artículos completos. Estos documentos fueron anotados manualmente con información sobre mutaciones genéticas y sus efectos en las interacciones proteína-proteína (PPIs). Una vez obtenidos los datos, se procedió a su preprocesamiento y tokenización, y se alimentaron a la arquitectura de DL diseñada para la extracción de PPIs. La arquitectura empleada es una red bidireccional de memoria a corto plazo de larga duración (BiLSTM) encargada de codificar los datos de texto.

- **Output**: La salida del algoritmo consiste en pares de proteínas y sus respectivos tipos de interacción, tales como unión o inhibición, junto con información sobre mutaciones genéticas específicas que puedan afectar dichas interacciones.

3.2.3. Graph kernels for RE - 2018

El algoritmo Graph kernels for RE (Panyam et al., 2018) utiliza una bolsa de palabras relacionadas con conceptos biológicos para el preprocesamiento de los textos biológicos.

Fue desarrollado en Java y para validar sus resultados utiliza conjuntos de textos previamente clasificados con posibles interacciones, como AImed (Bunescu et al., 2005), BioInfer (Pyysalo et al., 2007), LLL (Nédellec, 2005), IEPA (Ding et al., 2001) y HPRD50 (Fundel et al., 2007a). Este algoritmo se distingue por utilizar un clasificador Support Vector Machine (SVM) con un kernel modificado.

- **Input**: Los datos de entrada para este método consisten en textos del ámbito biomédico, como artículos de revistas médicas o informes clínicos, que pueden ser obtenidos a través de corpus especializados como BioInfer o textos de acceso público en Pubmed. En estos textos, las entidades biomédicas, como genes, proteínas, enfermedades o fármacos, pueden ser identificadas y extraídas mediante técnicas de reconocimiento de entidades con nombre.

  Además, para extraer las relaciones entre estas entidades, es necesario llevar a cabo un análisis sintáctico de dependencias que permita identificar las relaciones sintácticas entre ellas. Por lo tanto, el método requiere de un preprocesamiento de los datos que incluye tanto el reconocimiento de entidades con nombre como el análisis sintáctico de dependencias.

- **Output**: La salida del método incluye pares de entidades relacionadas y los tipos de relación asociados, tales como ”el gen A se asocia con la enfermedad B” o ”el medicamento C trata la enfermedad D”. Este método puede extraer varios tipos de relaciones, dependiendo de las entidades involucradas, tales como asociaciones gene-disease, tratamientos drug-disease, o interacciones proteína-proteína.

3.2.4. Extracción de IPP con DRNN - 2021

El algoritmo desarrollado por Badal et al. (Badal et al., 2021) tiene la capacidad de extraer información sobre las interacciones entre proteínas a partir de artículos y resúmenes disponibles en bases de datos como Pubmed y PMC. La implementación del algoritmo se realizó en Perl, un lenguaje de programación de bajo nivel que ofrece mayor velocidad de ejecución de la red neuronal, pero que también presenta ciertas limitaciones en cuanto a la complejidad de la codificación y a la replicabilidad de las versiones de las librerías utilizadas.

Este programa se basa en la búsqueda de texto mediante el uso de una lista de proteínas, en conjunto con los servicios web de las bases de datos de proteínas Uniprot y PDB, y emplea un modelo pre-entrenado para la clasificación de textos que puedan contener posibles interacciones entre proteínas.

- **Input**: Para el procesamiento de los datos, se emplea este algoritmo que acepta tanto textos completos como resúmenes de artículos biológicos provenientes de bases de datos como PMC y PubMed. Además, se utiliza un diccionario de frecuencia de palabras como entrada para el algoritmo clasificador en conjunto con los textos de interés.

- **Output**: El resultado del método es un modelo predictivo de la estructura del complejo proteína-proteína, el cual puede ser visualizado y analizado para comprender la naturaleza de las interacciones entre las proteínas que lo componen. La precisión de las predicciones se puede evaluar mediante diferentes métricas de rendimiento, tales como la desviación cuadrática media (RMSD) o las curvas de precision y recall.

La Tabla 3.2 presenta un resumen de las características más relevantes de los artículos seleccionados, mientras que la Tabla 3.3 muestra los resultados obtenidos por estos.

Artículo =>	(Peng y Lu, 2017)	(Tran y Kavuluru, 2018)	(Badal et al., 2021)	(Panyam et al., 2018)
Año	2017	2018	2021	2018
Número de citas	68	8	no	25
Revista/ Conferencia	Biomedical Natural Language Processing Workshop (2017)	Database	Bioinformatics	Journal of biomedical semantics
Lenguaje	no	python	perl	java/python
Datos de entrenamiento	AImed y BioInfer	BioCreative y Pubmed	Pubmed y PMC	AImed, BioInfer, HPRD50, IEPA, LLL
Estructura de datos	Word embedding	Word embedding	Word embedding	Bag of word
Repositorio	no	si	si	si
Algoritmo de clasificación	CNN	CNN	RNN/SVM	SVM
Validación cruzada	si	si	no	si
Nombre del algoritmo	no	biocppi_extraction	no	Graph kernels for RE

Tabla 3.2: Resumen de las características de los algoritmos principales por artículo

Artículo	**Conjunto de datos**	**Acc**	**Prec**	**Recall**	**F1**
1	Biocreative		38.22	37.34	37.78
2	AImed	72.9	29.3	78.0	39.3
3	AImed		67.3	60.1	63.5
4	Abstracts	69.3			

Tabla 3.3: Resumen de las características de los algoritmos principales por artículo

En conclusión, este capítulo ha revelado que los algoritmos analizados comparten diversas características, como el uso de corpus similares, algoritmos clasificadores y lenguajes

de programación comunes. Para el desarrollo de un nuevo algoritmo, es importante tener en cuenta estas características a fin de facilitar su implementación. Por ejemplo, el uso de Python podría ser beneficioso ya que es compatible con una amplia variedad de librerías en el campo del Procesamiento del Lenguaje Natural.

Capítulo 4

Desarrollo e implementación

En este Capítulo se describe los modelos desarrollados junto a los corpus utilizados. Aquí se exponen los diferentes modelos a partir de las distintas formas de representación del texto.

4.1. Proceso de búsqueda de interacciones entre proteínas

La Figura 4.1 resume los procesos conforman el trabajo realizado en el desarrollo de esta tesis. El primer paso consiste en generar el lexicón de intensidad orientado a la búsqueda de IPP. Seguidamente, utilizando como entrada textos científicos etiquetados previamente con una connotación positiva o negativa, se procede a utilizar el Lexicón para la representación de estos documentos, y luego se comparara esto con el uso de otros tipos de representaciones, entre las que se incluyen TF-IDF y Word embedding. Es importante destacar que además se llevaron a cabo distintas combinaciones entre estas para explorar si se obtienen mejores resultados. El tercer paso implica aplicar algoritmos de clasificación, y se desarrollaron diferentes modelos para este fin. En cuarto lugar, se lleva a cabo la aplicación de los modelos desarrollados, utilizando textos completos de PMC sin clasificar y separados por párrafos, el lexicón positivo y el diccionario de proteínas para buscar las interacciones proteína-proteína (IPP) en el texto. Para generar el diccionario de proteínas se utilizó Biopython y el Protein data bank (PDB). Los textos que contienen una posible interacción se seleccionan y se utilizan los códigos PDB para el docking. Por último, se realiza el docking, y luego un análisis detallado de los resultados obtenidos por este proceso.

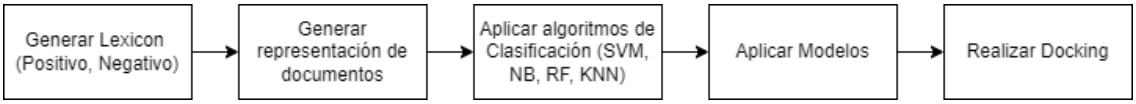


Figura 4.1: Proceso de búsqueda

La Figura 4.2 detalla las entradas y salidas en cada proceso. Es importante destacar que tanto el proceso 1 como el proceso 2, la generación del lexicón y de representación de do-

cumentos respectivamente, son las principales innovaciones, ya que no existía previamente un lexicón de intensidad orientado específicamente a la búsqueda de IPP.

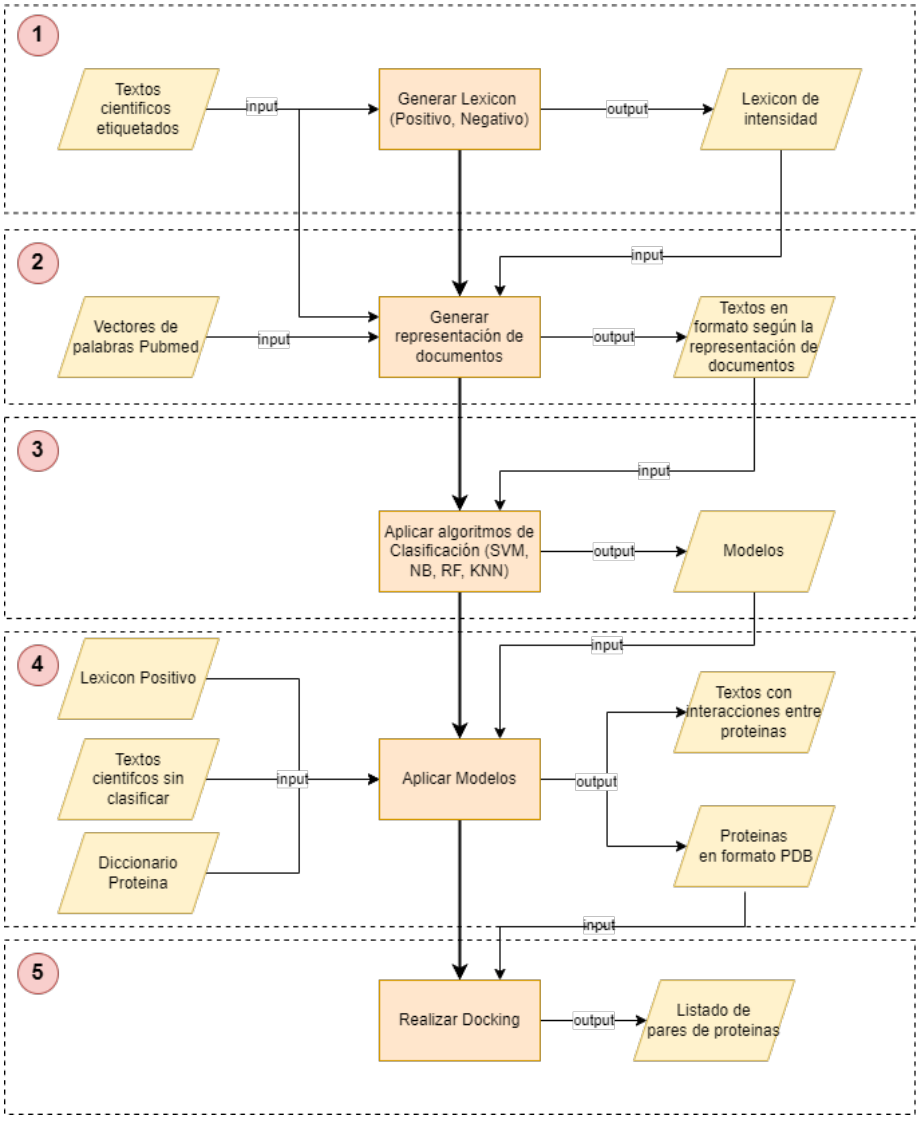


Figura 4.2: Proceso de búsqueda de interacciones entre proteínas

4.2. Corpus utilizados

Los corpus utilizados para el proceso de entrenamiento y pruebas de los algoritmos de clasificación son un conjunto de textos biológicos etiquetados con una connotación positiva o negativa de acuerdo a si contienen una interacción entre proteínas o no. Entre estos se encuentran Biocreative 3 (Arighi et al., 2011), AImed (Bunescu et al., 2005) y una

combinación entre partes de los Corpus AImed (Bunescu et al., 2005), BioInfer (Pyysalo et al., 2007), LLL (Nédellec, 2005), IEPA (Ding et al., 2001) y HPRD50 (Fundel et al., 2007a). La cantidad de textos que componen cada corpus se expone en la Tabla 4.1.

En otras palabras, todos estos corpus son conjuntos de datos estándar para la evaluación de sistemas de búsqueda de interacciones entre proteínas. Se asemejan en que sus textos están etiquetados con entidades biológicas como proteínas o genes, lo que facilita su búsqueda y análisis, pero HPRD50 varía en que cuenta con anotaciones que hacen referencia a si existe o no una IPP, lo cual sigue siendo útil para este trabajo y en la práctica sigue siendo similar a los demás.

También se realizaron pruebas de todos en conjunto utilizando solo una parte de cada uno de los corpus mencionados con un total de 3763 textos, a lo que en esta oportunidad se le llama Corpus completo. En la Figura 4.3 a se puede observar el porcentaje de textos etiquetados positivamente y negativamente según cada corpus utilizado.

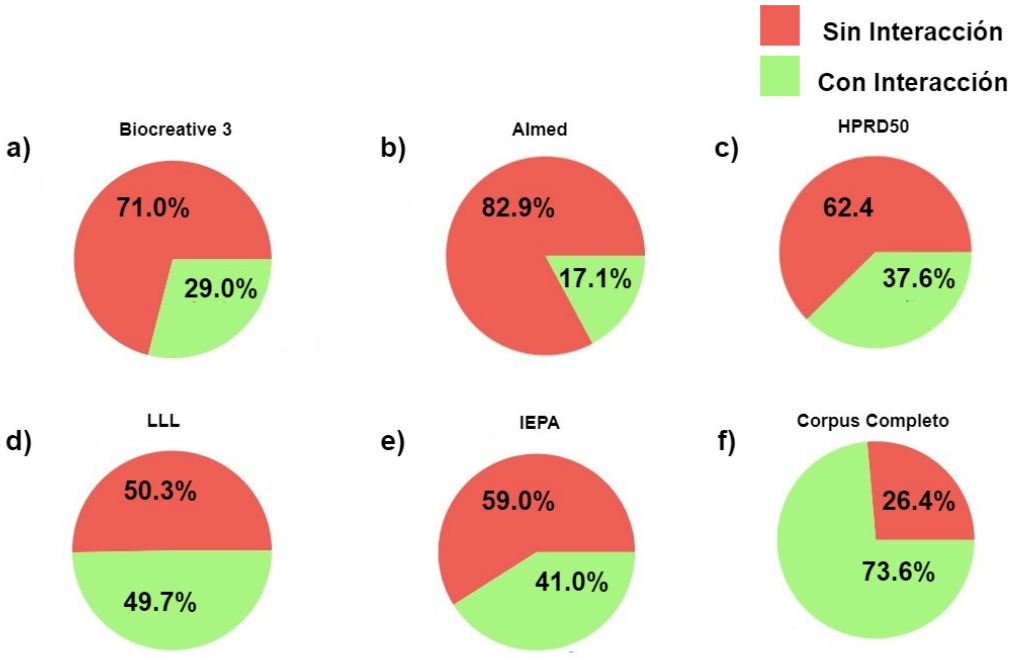


Figura 4.3: Detalle de porcentajes de interacción en los corpus utilizados

Los corpus se filtraron mediante la librerías Spacy o mediante expresiones regulares, además del uso de palabras claves que hacen mención a una IPP en el texto.

La librería Spacy[1][2] es una biblioteca de procesamiento de lenguaje natural (NLP) de código abierto desarrollada en Python. Spacy analiza el texto y asigna etiquetas usando modelos estadísticos. En este caso se usa para el reconocimiento de entidades, como las

[1] https://spacy.io/
[2] https://allenai.github.io/scispacy/

Corpus	Total	Train	Test
AImed	5.834	3.908	1.926
Biocreative 3	6.280	4.207	2.073
HPRD50	433	290	143
IEPA	817	547	270
LLL	330	221	109
Combinado	1.349	903	446

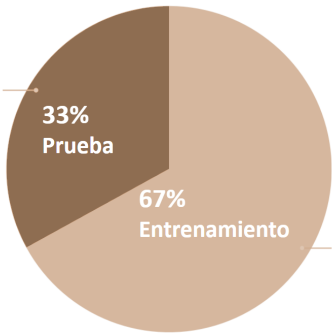


Figura 4.4: Detalle de porcentajes de datos utilizados de los corpus para el entrenamiento y pruebas de los modelos

Corpus	Total de textos del corpus		Total filtrados con spacy		Total filtrados con expresiones regulares	
	Cantidad	Porcentaje con interacciones	Cantidad	Porcentaje con interacciones	Cantidad	Porcentaje con interacciones
Biocreative 3	6280	29 %	3316	45,1 %	6167	29,5 %
AImed	5775	17,1 %	414	13,8 %	2781	14,6 %
Corpus Completo	3763	73,6 %	2995	54,9 %	3516	29,5 %
HPRD50	433	38 %	138	44,2 %	307	37,5 %
IEPA	817	41 %	158	32,3 %	520	40 %
LLL	330	41 %	80	22,5 %	185	43,2 %

Tabla 4.1: Corpus de textos etiquetados con interacciones entre proteínas

proteínas.

Respecto a expresiones regulares [3], a partir de la base de datos swiss prot se crea un patrón, usando la primera palabra de los nombres de proteínas registradas en esta (base de datos). Con el uso de expresiones regulares (Regex) en Python, como herramienta para la búsqueda de patrones en el texto, se detectan posibles nombres de proteínas.

De acuerdo con lo anteriormente mencionado en el Pseudocódigo 1, se puede observar el proceso de filtrado y extracción de datos de los Corpus.

La Tabla 4.1 resume la cantidad de textos filtrados por cada corpus, y los textos señalados con interacciones según la librería utilizada.

En el proceso de filtrado del Corpus, como se puede observar en el pseudocódigo 1, lo primero que se hace es ingresar la Ruta del Corpus utilizado (*ruta_corpus*) y el filtro a utilizar (*buscador_p*). En este caso, el filtro puede ser spacy o patrones, refiriéndose a la búsqueda de proteínas con expresiones regulares o con la librería spacy.

A continuación, se recorre el conjunto de resúmenes del Corpus seleccionado y se valida que haya alguna mención de algún nombre de proteína mediante la función ExtraerProteinas(), o alguna palabra clave que haga mención a una interacción mediante la función

[3] https://docs.python.org/3/library/re.html

ValidarPalabrasClaves().

Los valores obtenidos en el filtro se registran y se crea un nuevo documento del Corpus en formato CSV.

Algoritmo 1: ExtracionDatosCorpus

Require: *ruta_corpus*: ruta al archivo del corpus y buscador utilizado
Ensure: *corpus_procesado*: corpus procesado

```
Importar spacy
Importar pandas como pd
jnlpba ← spacy.load("en_ner_jnlpba_md")
corpus ← pd.read_csv(ruta_corpus).values.tolist()
corpus_filtrado ← []
for i in corpus:
    abstract ← i[0]
    doc ← jnlpba(abstract)
    buscar_proteinas ← ExtraerProteinas(abstract, buscador_p)
    validacion_proteinas ← buscar_proteinas[0]
    listado_proteinas ← buscar_proteinas[1]
    if buscar_proteinas[0] == False:
        continue
    palabras_claves ← ValidarPalabrasClaves(doc)
    if buscar_palabras_claves[0] == False:
        continue
    RegistrarProteinas(listado_proteinas)
    RegistrarPalabrasClaves(palabras_claves)
    corpus_filtrado.append({"Abstract" : abstract,"Clasificacion" : i[1]})
corpus_filtrado ← pd.DataFrame(corpus_filtrado)
ruta_corpus_filtrado ← "Documentos/{buscador_p}/{nombre_corpus}.csv"
corpus_filtrado.to_csv(ruta_corpus_filtrado)
```

4.3. Generar lexicón de intensidad de IPP

Cómo se menciona en el Capítulo 2 los lexicones de palabras contienen un conjunto de palabras claves que nos permiten indicar una mayor predominancia de un sentimiento dentro de un texto. Utilizando el mismo concepto se desarrollaron dos lexicones de palabras que indican si hay una posible interacción entre proteínas o si no, dándole un puntaje positivo o negativo al texto, permitiendo de esta manera tener otro criterio de descarte para la clasificación, como se muestra de forma resumida en la Figura 4.5 y a lo largo de la sección se explicará más a profundidad.

Para crear el lexicón de palabras claves sobre interacciones entre proteína, se utilizan 1000 abstract del corpus AImed clasificados positivamente, como se puede observar en

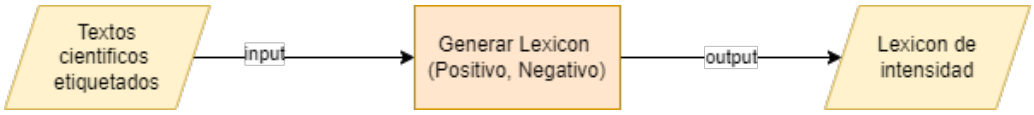


Figura 4.5: Resumen proceso de creación de lexicón de intensidad

el pseudocódigo 2, el proceso comienza con la creación del diccionario de palabras. Para ello, se utilizan los valores de los textos clasificados positivamente (*listado_positivo*) y los textos clasificados negativamente (*listado_negativo*). Luego, se analizan los textos y se extrae la frecuencia inversa de cada palabra dentro de ellos, generando el diccionario de palabras positivas (*diccionario_positivo*) y el diccionario de palabras negativas (*diccionario_negativo*).

Finalmente, se crea un documento en formato JSON para almacenar el diccionario de palabras positivas y otro documento para el diccionario de palabras negativas.

Algoritmo 2: Diccionario

Input: Corpus pre etiquetado con una clasificación positiva o negativa de acuerdo a si se menciona una interacción dentro del texto
Output: Exportación diccionario de palabras en formato JSON

1: *listado_postivo* ← Listado de abstracts clasificados positivamente con una interacción
2: *listado_negativo* ← Listado de abstracts clasificados negativamente sin una interacción
3: *diccionario_postivo* ← Listado de palabras positivas con su frecuencia inversa presente dentro del total de textos
4: *diccionario_negativo* ← Listado de palabras negativas con su frecuencia inversa presente dentro del total de textos
return Creación documentos en formato JSON de los diccionarios positivo y negativo

Para la selección de las palabras, se utiliza como criterio de selección aquellas palabras que explícitamente mencionan una interacción (por ejemplo “protein-protein”, “complex”, “interaction”) y las palabras que representan la función que cumple la interacción entre proteínas, como resultado se obtienen un número de 126 palabras claves que representan una posible interacción.

Se utiliza como ejemplo la medida de intensidad obtenida a partir de la similitud de las palabras en (Segura-Navarrete et al., 2021). En este caso, se emplean los Word Embedding de Pubmed BERT para calcular la similitud coseno de cada una de las palabras, lo que proporciona la medida de similitud para asignar una intención a cada una de ellas. Para llevar a cabo esta tarea, se decide utilizar el algoritmo K-means, el cual se implementa en el pseudocódigo 3.

Continuando con la descripción del pseudocódigo 3, la función Cluster recibe como parámetro de entrada el diccionario de palabras creado en el pseudocódigo 2. Su resultado es el grupo de palabras que presenta una mayor relación, y se devuelve en formato JSON. Para lograr esto, se utilizan las funciones EmbeddingLemmaPalabras descrita en el pseudocódigo 4 y CrearCluster explicada en el pseudocódigo 5.

En primer lugar, se define el número de grupos en los que se va a dividir el listado de palabras. Se utilizan los Word Embedding de Pubmed BERT para visualizar la relación entre ellas. Además, se emplea el diccionario de palabras creado anteriormente. Estos elementos se pasan como parámetros de entrada a la función EmbeddingLemmaPalabras 4.

La función EmbeddingLemmaPalabras 4 recibe el listado de palabras, una instancia de la librería de Spacy para obtener el lemma de cada palabra y el listado de vectores de palabras.

En el pseudocódigo 4, se observa que se crean dos listados: uno donde se eliminan todas las palabras repetidas del listado original, y otro donde se busca el lema de cada palabra y se registra. A continuación, se codifican estos lemmas en formato de vector de palabras (Word Embedding) para obtener la relación entre ellos. Por último, se retorna el listado de palabras en formato de vectores y el listado de lemmas correspondientes a cada palabra.

Continuando con el pseudocódigo 3, los valores obtenidos por la función 4 se guardan en las variables *corpus_embedding* y *palabras_lemma*. Estas variables se utilizan como parámetros de entrada, junto con el número de grupos de palabras a utilizar, en la función 5.

En el pseudocódigo 5, se crea una instancia de la clase del algoritmo K-means, a la cual se le entregan como parámetros de entrada el número de grupos de palabras a utilizar y el conjunto de vectores de palabras obtenidos anteriormente. Luego, se entrena el algoritmo y se retornan las etiquetas, que representan las palabras. De esta manera, se puede determinar a qué grupo pertenece cada palabra a partir de su índice. Estas etiquetas se retornan para ser registradas en formato JSON.

Algoritmo 3: Cluster

Input: Ruta diccionario de palabras
Output: Exportación grupos de palabras con mayor relación en formato JSON

```
numero_cluster ← Cantidad de grupos en los que se quiere separar el diccionario
embedder ← SentenceTransformer()
jnlpba ← spacy.load()
Librería de o
palabras_lexicon ← Diccionario de palabras filtrado por expertos
datos_emb_lemma ← EmbeddingLemmaPalabras( palabras_lexicon, jnlpba, embedder)
corpus_embedding ← datos_emb_lemma[0]
palabras_lemma ← datos_emb_lemma[1]
clustered_senteces ← CrearCluster( numero_cluster, corpus_embedding, palabras_lemma )
```

Algoritmo 4: EmbeddingLemmaPalabras

Input: Diccionario de palabras
Output: Lemma de la palabras con su representación en word embedding

```
palabras ← []
palabras_lemma ← []
for i in palabras do
    if i not in palabras then
        palabras.append(i)
for j in palabras do
    lemma ← Utilizando una librería de procesamiento de lenguaje natural para la obtención del lemma de la palabra j
    if lemma not in palabras_lemma then
        palabras_lemma.append(lemma)
corpus_emb ← Utilizando un modelo pre entrenado de word embedding se codifica las palabras del listado palabras_lemma
return corpus_emb, palabras_lemma
```

Posterior a esto, se organizan manualmente grupos de acuerdo a si existe mayor o menor relación con IPP. Se comienza formando el grupo con mayor relación y luego los siguientes grupos en orden decreciente. Estando ya clasificados y ordenados los grupos. Una vez creados los grupos de palabras, como se describe en el pseudocódigo 3, se procede a realizar la creación del lexicón, como se indica en el pseudocódigo 6. Este proceso requiere

Algoritmo 5: CrearCluster

Input: Numero de grupos, embedding de palabras y lemma de cada palabra
Output: Listados de grupos de palabras con mayor relación entre si

```
model ← KMeans( num_clusters )
model.fit(corpus_embeddings)
cluster_assignment ← model.labels_
cluster_sentences ← Listado de grupos de palabras para la creación del lexicón
for sentence_id in cluster_assignment do
    cluster_sentences.append(palabras_lemma[sentence_id])
return clustered_sentences
```

como parámetros de entrada el número de grupos de palabras (*cluster_num*) y la lista de grupos de palabras (*clustered_sentences*).

A partir del número de grupos, se crea una variable llamada "puntajes", que es una lista donde se registran los valores de puntaje asignados a cada grupo de palabras, dependiendo del número de grupos existentes. Estos puntajes se asignan en un rango de 10 a 90, en función de la cantidad de grupos.

Finalmente, utilizando la lista de puntajes generada junto con la lista de palabras, se utilizan como parámetros de entrada en la función ÇrearLexicon"descrita en el pseudocódigo 6. En esta función, a cada palabra se le asigna un puntaje equivalente a su intensidad, según el listado de puntajes realizado previamente. El resultado se retorna en formato JSON.

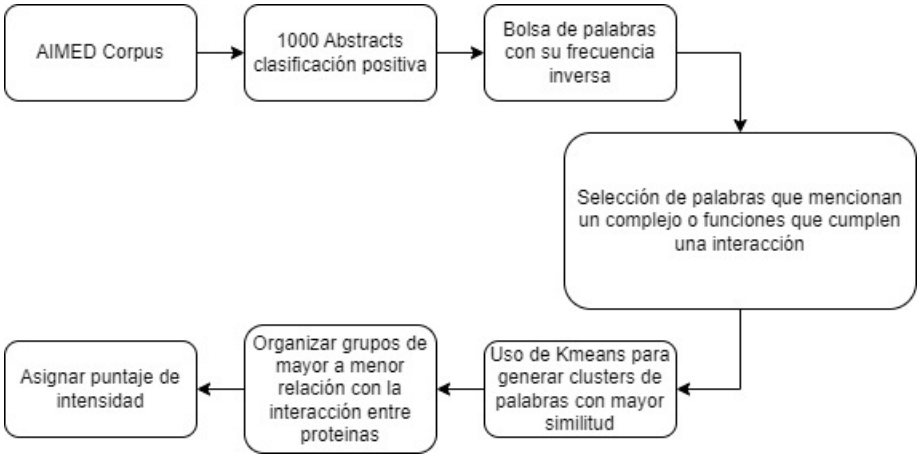


Figura 4.6: Creación lexicón de intensidad de palabras de indiquen una interacción entre proteínas.

En la Figura 4.6 se ilustra el proceso de creación del lexicón de intensidad descrito recientemente. En las Tablas 4.2a) y 4.2b) se presentan ejemplos del lexicón de intensidad.

Tanto la Tabla 4.2a) como la Tabla 4.2b) muestran listados de palabras obtenidas y los puntajes asignados correspondientes. En la Tabla 4.2a), se indican las palabras que tienen una interacción entre proteínas junto con sus puntajes asignados obtenidos durante el proceso de creación del lexicón. Por otro lado, en la Tabla 4.2b), se incluyen palabras que no hacen referencia a una interacción entre proteínas, junto con sus puntajes asignados

obtenidos durante el mismo proceso de creación del lexicón.

a)

Palabras	**Intensidad**
protein-protein	90
interact	64
affinity	37
heteroprotein	10

b)

Palabras	**Intensidad**
mutant	90
extracellular	64
oncoprotein	37
protein-rna	10

Tabla 4.2: Ejemplo lexicón de intensidad. a) palabras que indiquen una IPP. b) palabras que no indiquen una IPP

Para determinar el número de grupos correcto se utiliza la medida de inercia (en la Figura 4.7 a se puede observar los resultados obtenidos para los diferentes números de grupos), dando como resultado un número de 5 grupos que se les asignan un puntaje de 10 a 90, siendo 90 hipotéticamente el grupo con una mayor intensidad de interacción y 10 una menor intensidad (Ver Pseudocódigo 6 y 7).

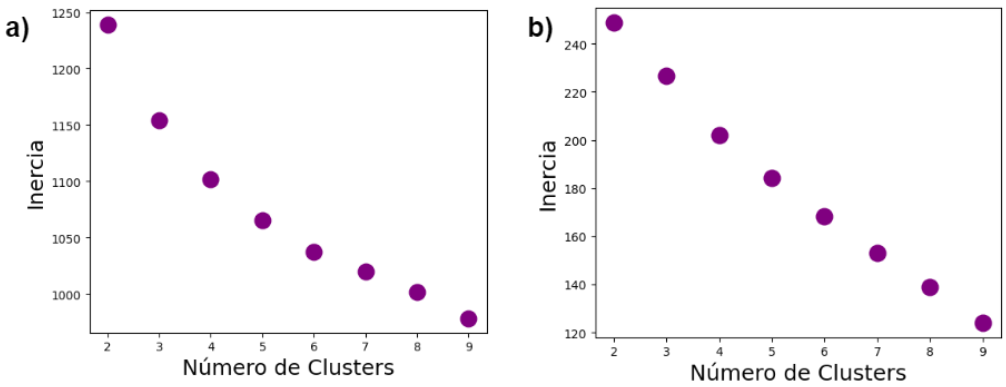


Figura 4.7: a) Selección de número de clusters para la creación del lexicón de intensidad que indique una interacción entre proteínas. b) Selección de número de clusters para la creación del lexicón de intensidad que no indique una interacción entre proteínas.

La creación del lexicón de intensidad negativa o que no indique una interacción entre proteínas, consiste en realizar el mismo proceso anteriormente descrito, con la diferencia que el conjunto de datos utilizado están etiquetados de forma negativa. En la Figura 4.7.b) se puede observar la medida de inercia según la cantidad de grupos de palabras y en la Tabla 4.2.b) se señala un ejemplo del lexicón de intensidad negativo.

Después de finalizar la creación del Lexicón, se obtuvo un diccionario con 182 palabras positivas que se refieren a la presencia de interacciones entre proteínas, con puntajes que van desde 10 hasta 90, dependiendo de la fuerza de la mención de la interacción en el texto. También se generó un diccionario negativo con 27 palabras que indican la ausencia

Algoritmo 6: Lexicon

Input: Ruta de los grupos de palabras con mayor relación
Output: Exportación lexicón de palabras con intensidad en formato JSON

```
clustered_sentences ← Listado de grupos de palabras
cluster_num ← Numero de grupos dentro de clustered_sentences
puntajes ← Es un listado con valores 0 con su tamaño dependiendo del numero de grupos en la variable cluster_num
pts ← 10
pts_sum ← cluster_num − 1
pts_sum ← 80/pts_sum
pts_sum ← round(pts_sum, 0)
for index, value in enumerate( puntajes ) do
    if index + 1 == cluster_num then
        pts ← 90
    puntajes[index] ← pts
    pts ← pts + pts_sum
puntajes ← sorted(puntajes)
CrearLexicon(clustered_sentences, puntajes)
```

Algoritmo 7: CrearLexicon

Input: Grupos de palabras, listado de puntajes y ruta registro del lexicón
Output: Exportación lexicón de intensidad en formato JSON

```
lexicon ← []
for index, value in enumerate( clusters ) do
    porcentaje ← puntajes[index]
    for i in value do
        datos ← {"Palabras": i,"Intensidad": porcentaje }
    lexicon.append(datos)
Registro del lexicón de intensidad en la ruta de entrada en formato JSON
```

de interacciones entre proteínas.

4.4. Generar Representación de documentos

El segundo proceso en el desarrollo de la búsqueda de interacciones entre proteínas en el texto consiste en generar una representación de documentos. Para ello, se utilizará el Lexicón de intensidad creado en el proceso anterior para generar una representación de texto comprensible para el algoritmo clasificador. Además, se utilizarán otras representaciones de documentos, como TF-IDF y Word Embedding, para obtener más información.

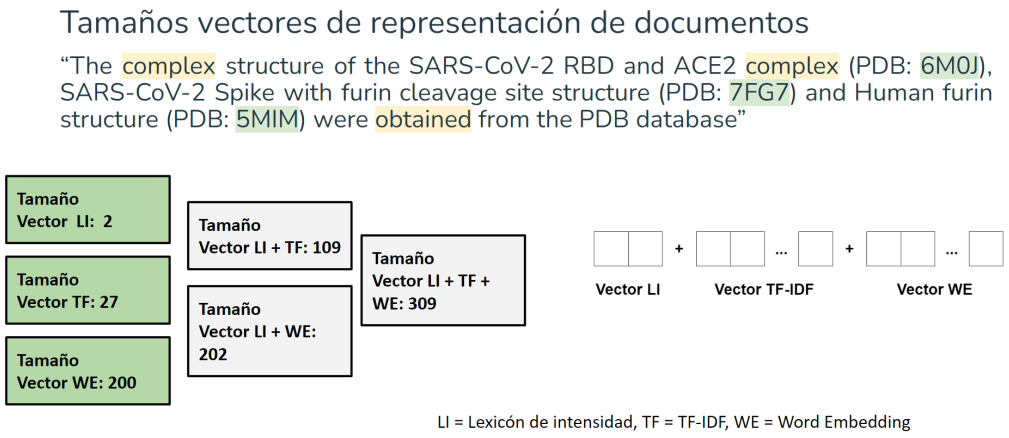


Figura 4.8: Resumen proceso de representación de documentos

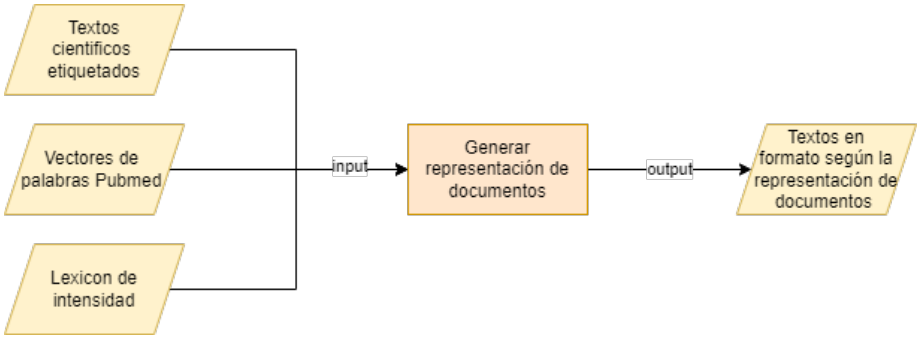


Figura 4.9: Resumen proceso de representación de documentos

Se considera de entrada a este proceso tanto el lexicón de intensidad como TF-IDF y Word Embedding, integrándose al diagrama general de la metodología de trabajo como se muestra en la Figura 4.9.

En la Tabla 4.3 se detalla la combinación de las representaciones mencionadas anteriormente, que incluye TF-IDF + Lexicón, la combinación de las tres (TF-IDF, Lexicón y Word Embedding), así como el uso de TF-IDF con Lexicón y la combinación de ambos. De esta manera, se combinan los resultados obtenidos de forma individual por el clasificador utilizado, y se comparan los resultados de TF-IDF, Lexicón y la clasificación resultante de su combinación.

La misma técnica se utiliza para combinar Word Embedding y el Lexicón de intensidad. En el ejemplo de la Figura 4.8, se ilustra el tamaño de los vectores obtenidos a partir de cada representación y de la combinación de las mismas. Se muestra un texto y se presenta el tamaño correspondiente a cada una de las representaciones utilizadas.

Clasificador	**Representación de documentos**	**Corpus**
NB	Lexicón de Intensidad (LEXI)	AImed
SVM	TF-IDF	Biocreative 3
RF	Word Embedding	Corpus Completo
RF	TF-IDF/LEXI	HPRD50
KNN/NB/RF	Word Embedding/LEXI	IEPA
SVM/KNN/RF	Word Embedding/ TF-IDF/LEXI	LLL
SVM/NB/KNN	TF-IDF/LEXI/ (TF-IDF+LEXI)	
SVM/NB/RF	Word Embedding/LEXI/ (Word Embedding+LEXI)	

Tabla 4.3: Algoritmos clasificadores, representaciones de documentos y corpus utilizados, incluyendo combinaciones entre estos.

4.5. Proceso de aplicación de algoritmos de clasificación

Para aplicar los algoritmos clasificadores se toma como base el análisis del texto para la extracción de proteínas (esto se observa en la Figura 4.10), donde:

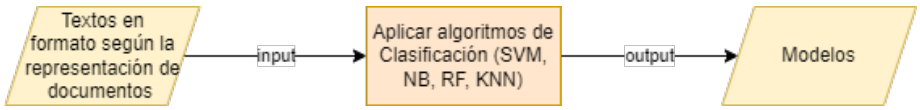


Figura 4.10: Resumen proceso de representación de documentos

1. Primero se utilizan criterios de descarte en el conjunto de datos 0, estos serían los corpus previamente mencionados sin ser filtrados. Estos criterios son que la clasificación de textos no haga mención de que no existe una interacción en el texto, que el texto tenga un mínimo de dos nombres de proteínas y por último que contenga al menos una palabra del lexicón de intensidad positivo que se describe en la sección anterior. Como resultado del filtro realizado se obtiene un primer conjunto que son todos aquellos textos que cumplen con el criterio de descarte, además de identificar la posible mención de proteínas. Esta información será necesaria en la creación del conjunto de datos final, descrito a continuación.

2. Luego, para la creación del conjunto de datos final se realizan combinaciones de las proteínas en pares, destacándolas en el texto, ampliando la cantidad de datos del primer conjunto.

3. Finalmente, la información es preprocesada de acuerdo a la representación de documentos seleccionada y luego clasificada por un algoritmo de clasificación también seleccionado. Como resultado se obtiene un porcentaje que valida si es que existe una interacción entre proteínas dentro del texto y un listado de proteínas, que se les asignan sus respectivos códigos del PDB.

En la Figura 4.11 se puede observar paso a paso el proceso del algoritmo mencionado con sus diferentes etapas.

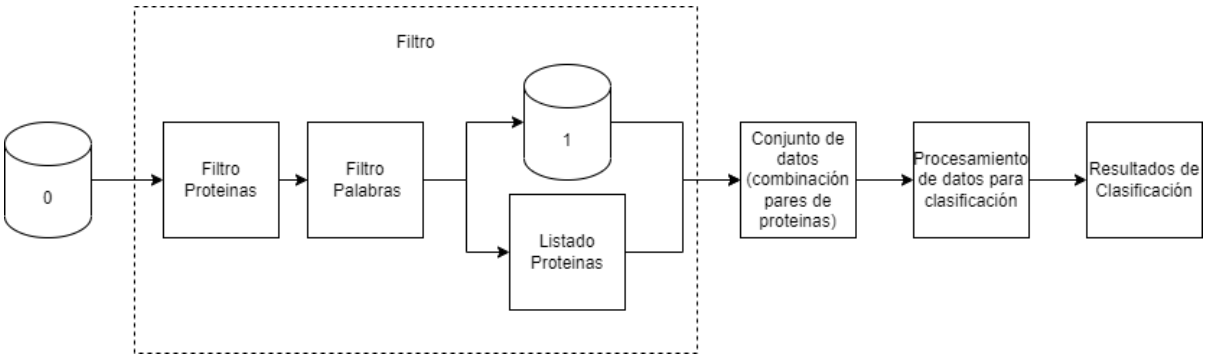


Figura 4.11: Esquema diseño de modelos

4.6. Modelos

Para el desarrollo de esta tesis se formulan diferentes modelos a trabajar e implementar, para lo cual se hace uso de algoritmos de clasificación de Machine Learning y Deep Learning con diferentes tipos de representaciones de documentos para la detección de interacciones entre proteínas en textos biológicos.

Para empezar, un modelo es el resultado de la combinación entre un algoritmo clasificador, una representación de documentos y un corpus que lo entrena. Se describen diferentes opciones para combinar entre algoritmos, representaciones de documentos y corpus, cuyos elementos a combinar se detallan en la Tabla 4.3, pudiendo utilizarse también combinaciones de algoritmos entre sí, o entre representaciones, previo a la combinación propiamente tal que formará un modelo, como puede observarse también en la tabla descrita. A continuación se describen los modelos con los que se trabajó y qué elementos se combinaron para diseñarlos. Los resultados de los modelos desarrollados se pueden observar en el Capítulo 5.

4.6.1. Modelos con uso de representaciones de documentos

Como anteriormente se menciona existen diferentes tipos de representación de documentos para la obtención de información dentro del texto, a continuación se mencionan los modelos desarrollados utilizando como representación de documentos lexicón de intensidad, TF-IDF y Word Embedding. En la Tabla 4.3 se puede observar los diferentes modelos con los clasificadores y los corpus utilizados.

4.6.1.1. Modelos con uso de lexicón de intensidad

En la Sección 2 se explica que una forma de representar los textos biológicos es a través del uso de un Lexicón de intensidad de interacciones entre proteínas, especialmente diseñado para este propósito. Este lexicon asigna un puntaje positivo o negativo al texto, lo que proporciona otro criterio para la clasificación. La suma de estos puntajes permite determinar si el texto tiene una connotación positiva o negativa, lo que ayuda a validar la existencia de una interacción entre proteínas. Para la tokenización del texto, se utiliza la librería de procesamiento de lenguaje natural Spacy. Es importante destacar que el Lexicón de intensidad solo utiliza palabras consideradas positivas o negativas.

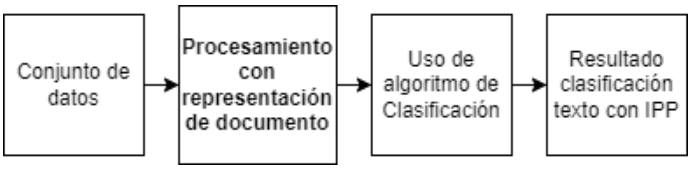


Figura 4.12: Procesamiento del modelo con el uso de representación de documentos

En la Figura 4.12 se puede ver el proceso del preprocesamiento del modelo, donde como dato de entrada se ingresa el conjunto de datos creado a partir de la combinación de los diferentes pares de proteínas encontrados en el texto. Esta información es preprocesada a un formato de lexicón de intensidad y luego clasificado por un algoritmo de Machine Learning de clasificación.

4.6.1.2. Modelos con uso de TF-IDF

La segunda forma de representar los textos biológicos que se propone en la formulación de modelos es a partir de la técnica TF-IDF. Como se menciona en el Sección 2 el vector para cada texto está compuesto por una bolsa de palabras con un valor de frecuencia para cada una de ellas, para determinar la importancia de cada palabra de acuerdo a su frecuencia de aparición en el corpus. Para la creación de la matriz TF-IDF se utiliza la librería Scikit-learn, y como tokenizador la librería de procesamiento de lenguaje natural Spacy.

En la Figura 4.12 se puede observar el proceso del preprocesamiento del modelo, que con datos de entrada similares al modelo anterior, esta información es preprocesada a una matriz TF-IDF y luego clasificado por un algoritmo de Machine Learning de clasificación.

4.6.1.3. Modelos con uso de word embedding

Otro tipo de modelos propuestos son con el uso de Word Embedding para la representación de los documentos. Esta técnica descrita en la Sección 2 consiste en representar las palabras en un espacio vectorial de tal manera que estas pueden ser agrupadas por su similitud semántica, entendiendo esto se puede determinar el contexto de una oración a partir de la similitudes del conjunto de palabras. Para realizar la tokenización se utiliza la librería

Spacy y para la creación de los vectores de palabras se utiliza un modelo pre entrenado de Word Embedding a partir de textos biológicos de Pubmed (Moen y Ananiadou, 2013).

En la Figura 4.12 se puede ver el proceso del preprocesamiento del modelo, donde la diferencia con los anteriores está en que la información es preprocesada a un formato Word Embedding y luego clasificada por un algoritmo de Machine Learning de clasificación.

4.6.2. Modelos con uso combinado de representación de documentos

Se plantea crear también diferentes combinaciones entre las representaciones de datos. Cada tipo de representación contempla un vector de características, y estos van a ser concatenadas para obtener mayor información. Por consiguiente, se describen las combinaciones realizadas con sus respectivas Figuras y Tablas que explican los modelos desarrollados. En todos los casos, luego de que la información del modelo es preprocesada según cada representación de documentos, es clasificada por un algoritmo de clasificación de Machine Learning.

- TF-IDF - Lexicón de Intensidad
- Word Embedding - Lexicón de Intensidad
- Word Embedding - TF-IDF - Lexicón de Intensidad
- TF-IDF - Lexicón de Intensidad - (combinación TF-IDF con Lexicón de Intensidad)
- Word Embedding - Lexicon de Intensidad - (combinación Word Embedding con Lexicon de Intensidad

4.6.2.1. TF-IDF con lexicón de intensidad

La primera combinación se esquematiza en la Figura 4.13 y corresponde a la concatenación del vector TF-IDF (que en la figura se representa con ”X”) con el de Lexicón de intensidad para el preprocesamiento de la información, en otras palabras, el conjunto de datos es preprocesado por ambos enfoques y luego es concatenado en un solo vector, para ser clasificado por un algoritmo de clasificación de Machine Learning.

4.6.2.2. Word embedding con lexicón de intensidad

La segunda combinación también se esquematiza en la Figura 4.13, y es la concatenación del vector Word Embedding (que en la figura se representa con ”X”) con el de Lexicón de intensidad para el preprocesamiento, y luego clasificado por el algoritmo.

4.6.2.3. Word embedding, lexicón de intensidad con TF-IDF

Como se puede observar en la Figura 4.14 la tercera combinación se basa en la concatenación de las tres representaciones de datos de documentos previamente mencionadas. Consiste en dar como datos de ingreso la combinación de pares de proteínas que va a

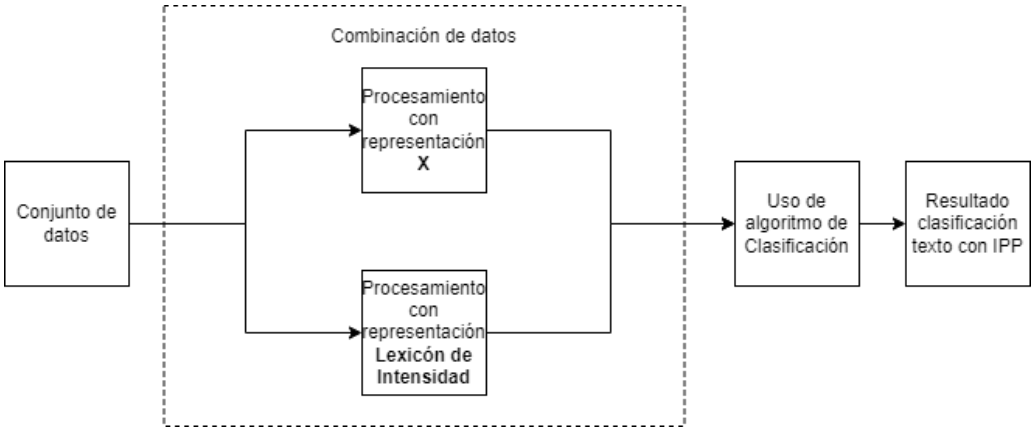


Figura 4.13: Esquema modelo con uso de TF-IDF o Word embedding con lexicón de intensidad

ser preprocesada por cada una de las representaciones, creando tres vectores que serán concatenados, y finalmente se clasifica por el algoritmo.

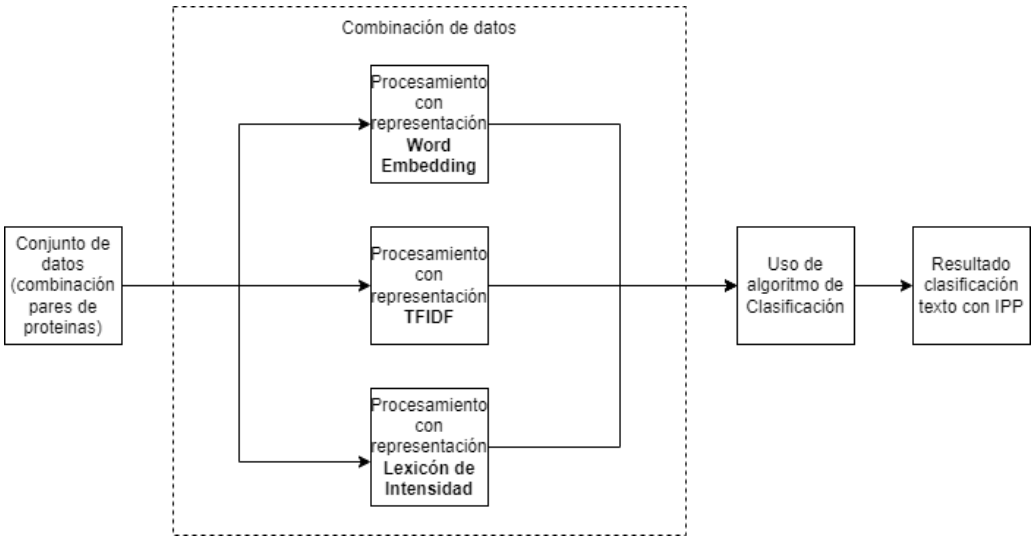


Figura 4.14: Esquema modelo con uso de word embedding, TF-IDF y lexicón de intensidad

4.6.2.4. TF-IDF, lexicón de intensidad y la combinación de TF-IDF con lexicón de intensidad

En este caso se combinan tres modelos con diferentes tipos de representaciones de documentos, pero utilizando el mismo clasificador. Las representaciones utilizadas son TF-IDF, Lexicón de Intensidad y la combinación de ambos, como una tercer tipo de representación. Lo anterior se presenta en la Figura 4.15 donde "X" representa en este caso a TF-IDF.

4.6.2.5. Word embedding, lexicon de intensidad y la combinación de word embedding con lexicon de intensidad

Se desarrolla el modelo con combinación de tres maneras de representar documentos, estas son Word Embedding, Lexicón de Intensidad y la tercera es la combinación de ambos, como se detalla en la Figura 4.15, donde "X" representa en este caso a Word embeadding. Se usa el mismo clasificador, pero se aplica por separado para cada forma de representación.

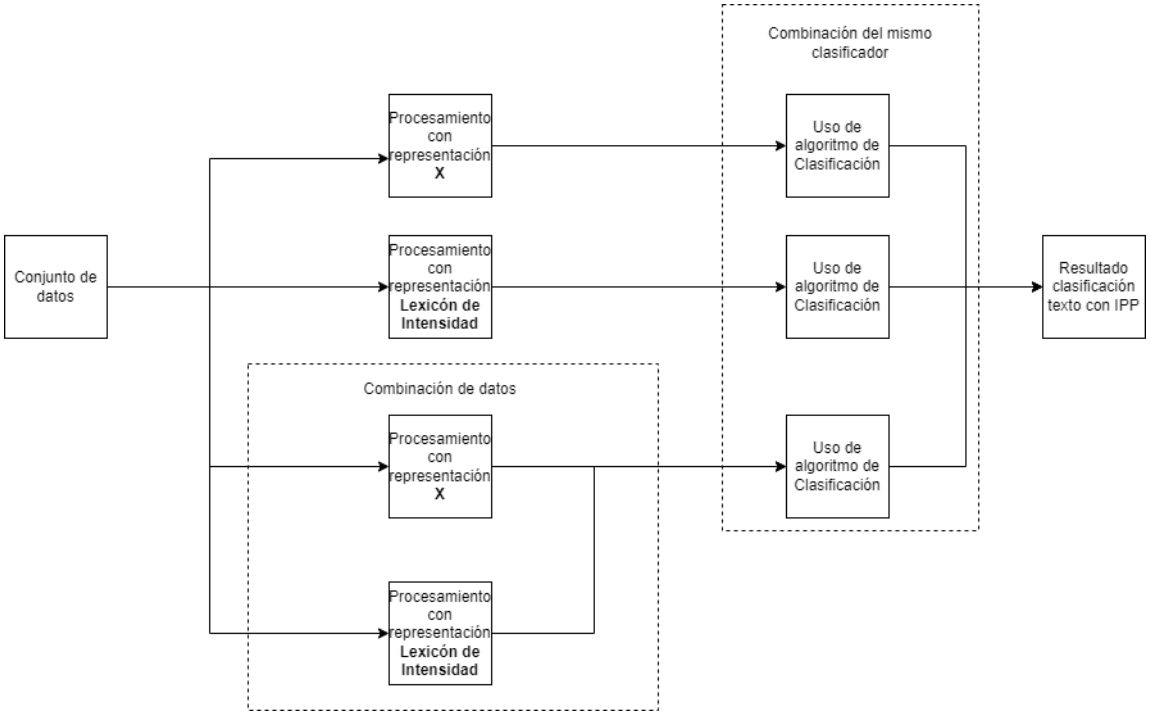


Figura 4.15: Esquema modelo con uso de TF-IDF o Word embeadding, con lexicón de intensidad y la combinación de ambos

4.6.3. Modelos con uso de combinación de clasificadores

Estos modelos se diferencian en combinar los algoritmo de clasificación para variar los resultados. Para esta experimentación se utiliza la representación de texto combinada de TF-IDF con lexicón de intensidad, además de los grupos de clasificadores que se mencionan a continuación:

- KNN-NB-RF
- SVM-KNN-RF
- SVM-NB-KNN
- SVM-NB-RF

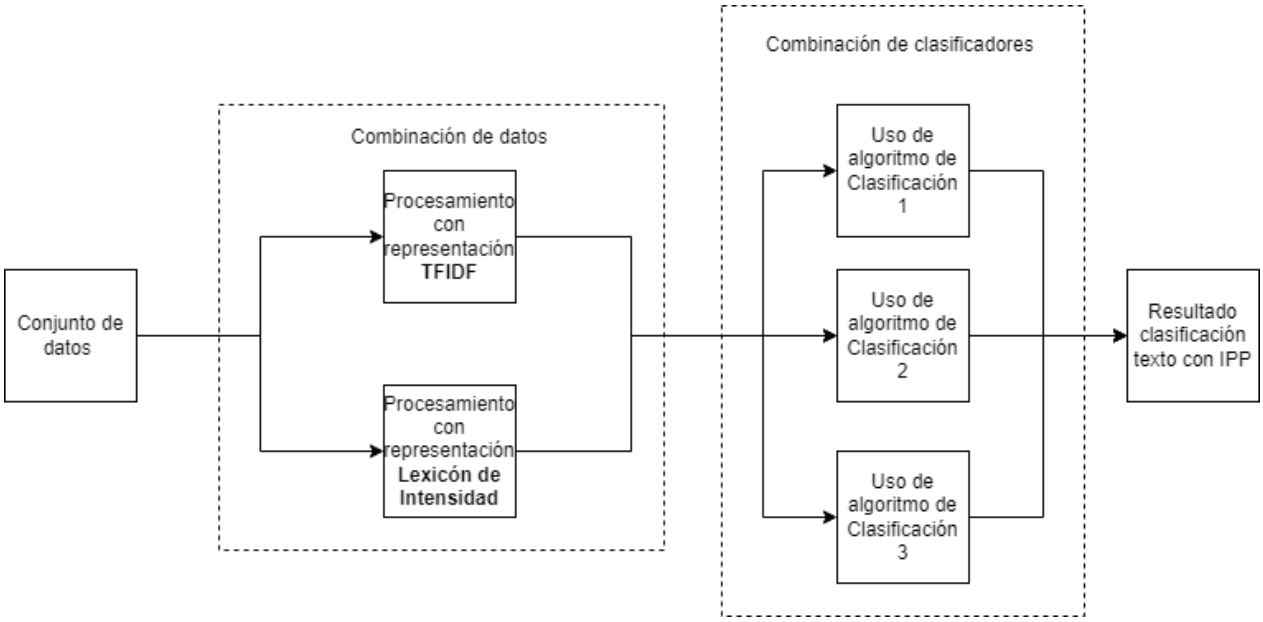


Figura 4.16: Esquema modelo con uso de TF-IDF con lexicón de intensidad y combinación de clasificadores

Acerca de este capítulo se puede concluir que para obtener una mayor cantidad de información, es posible combinar diferentes representaciones de documentos, cada una con un enfoque específico. En este trabajo, se empleó el lexicón de intensidad junto con las diferentes representaciones mencionadas en el marco teórico. Además, se implementó un filtro para optimizar los recursos, descartando los textos que no mencionan posibles nombres de proteínas o palabras relacionadas con interacciones. En el próximo capítulo se detallan con mayor profundidad los resultados obtenidos. Cabe destacar que, teóricamente, al obtener más información, deberían obtenerse mejores resultados en la clasificación de textos con IPP.

Capítulo 5

Experimentación, resultados de modelos y selección

En este Capítulo se describe cómo se experimentó y trabajó con los diferentes modelos mencionados en el Capítulo anterior. Para la experimentación se utiliza un equipo con las características siguientes: Sistema Operativo Ubuntu 20.04.4 LTS, procesador Intel(R) Xeon(R) CPU E3-1220 v3 @ 3.10GHz y 16262728 de RAM.

La información de proteínas fue filtrada previamente, la Tabla 5.1 que sintetiza el tiempo de filtrado (tf) de cada corpus con ambas librerías, y se desprende que varía entre 0,83 (50 minutos) y 29 horas con la librería Spacy, y entre 1.26 y 90 horas con el uso de expresiones regulares. En la experimentación se debe considerar el tiempo que toma el filtrado de los corpus utilizados para cada modelo.

Corpus	**Spacy**	**Exp.Regulares**
Biocreative 3	29	90
AImed	22	90
Corpus Completo	2	4
HPRD50	0.83	1.4
IEPA	1.23	2.12
LLL	0.83	1.26

Tabla 5.1: Tiempos de filtrado (tf) de los corpus utilizados, en horas

5.1. Resultados con cada representación de datos

En esta sección se presentan los resultados obtenidos mediante las representaciones individuales de los documentos: Lexicón, TF-IDF y WEMB. Para cada tipo de representación de datos, se aplicaron los siguientes clasificadores: Naive Bayes (NB), Máquinas de

Vectores de Soporte (SVM), Bosques Aleatorios (RF) y K-Vecinos más Cercanos (KNN). Además, se incluye el uso de WEMB con Deep learning (DL).

Cada tabla muestra los clasificadores utilizados, el corpus empleado, si se realizaron filtrados o no, en caso afirmativo, se indica el tipo de filtro aplicado, el tiempo de entrenamiento y el índice de precisión (Accuracy).

5.1.1. Resultados con representación de datos - Lexicón de intensidad

Se empleó el lexicón propuesto para este trabajo y se presentan los resultados en la Tabla 5.2. La tabla muestra que los modelos que utilizaron la representación de documentos del Lexicón de Intensidad tuvieron diferentes niveles de éxito en la clasificación de textos.

En particular, el modelo que utilizó el conjunto de datos AImed, filtrado por Spacy para buscar proteínas en el texto y el algoritmo clasificador Naive Bayes (NB), logró una exactitud del 90 % en la clasificación de los textos, lo que indica que fue el más exitoso de los modelos evaluados. Por otro lado, el modelo que combinó el corpus HPRD50 no filtrado, que contiene una cantidad menor de textos clasificados, y el clasificador Naive Bayes (NB), tuvo una exactitud del 62 % en la clasificación de los documentos, lo que indica que fue el menos exitoso de los modelos evaluados.

5.1.2. Resultados con representación de datos - TF-IDF

Se utilizó la representación de datos TF-IDF y los resultados se presentan en la Tabla 5.3. Los modelos que utilizaron esta representación otuvieron diferentes niveles de éxito en la clasificación de textos.

El modelo que utilizó el conjunto de datos AImed, filtrado por Spacy, y el algoritmo clasificador SVM, logró una exactitud del 88 % en la clasificación de los textos, lo que indica que fue el modelo más exitoso de los evaluados. Por otro lado, el modelo que utilizó la combinación del corpus IEPA filtrado por Spacy, que contiene una cantidad menor de textos clasificados, y el clasificador Naive Bayes (NB), tuvo una exactitud del 62 % en la clasificación de documentos, lo que indica que fue el menos exitoso de los modelos evaluados. Los resultados se muestran detalladamente en la tabla.

Clasificador	Corpus	Filtrado	Filtro	t entrenamiento	Accuracy
NB	**AImed**	**SI**	**spacy**	**8.19 s**	**0.90**
	Biocreative3	No	ninguno	316.78 s	0.72
	Corpus Completo	Si	spacy	24.41 s	0.78
	HPRD50	**No**	**ninguno**	**19.96 s**	**0.62**
	IEPA	Si	spacy	2.54 s	0.74
	LLL	Si	spacy	731.79 s	0.70
SVM	AImed	Si	spacy	8.91 s	0.89
	Biocreative3	No	ninguno	942.18 s	0.72
	Corpus Completo	Si	spacy	21.20 s	0.78
	HPRD50	Si	patrones	38.01 s	0.65
	IEPA	Si	spacy	4.24 s	0.74
	LLL	Si	spacy	903.19 s	0.70
RF	AImed	Si	spacy	61.28 s	0.86
	Biocreative3	No	ninguno	316.60 s	0.66
	Corpus Completo	Si	spacy	25.84 s	0.75
	HPRD50	No	ninguno	5.75 s	0.66
	IEPA	Si	spacy	2.44 s	0.72
	LLL	Si	patrones	3.02 s	0.71
KNN	AImed	Si	spacy	8.12 s	0.88
	Biocreative3	No	ninguno	292.12 s	0.67
	Corpus Completo	Si	spacy	30.37	0.78
	HPRD50	No	ninguno	5.05 s	0.64
	IEPA	Si	spacy	2.32 s	0.72
	LLL	Si	spacy	13.05 s	0.70

Tabla 5.2: Resultados con representación de datos - Lexicón de intensidad

Clasificador	Corpus	Filtrado	Filtro	t entrenamiento	Accuracy
NB	AImed	Si	spacy	6.53 s	0.67
	Biocreative3	No	ninguno	314.34 s	0.77
	Corpus Completo	Si	spacy	27.75 s	0.76
	HPRD50	No	ninguno	8.10 s	0.66
	IEPA	**Si**	**spacy**	**3.70 s**	**0.62**
	LLL	Si	spacy	2.16 s	0.63
SVM	**AImed**	**Si**	**spacy**	**6.61 s**	**0.88**
	Biocreative3	No	ninguno	327.43 s	0.86
	Corpus Completo	Si	spacy	27.83 s	0.84
	HPRD50	Si	patrones	8.82 s	0.76
	IEPA	Si	spacy	4.22 s	0.75
	LLL	Si	patrones	6.24 s	0.71
RF	AImed	Si	spacy	6.91 s	0.87
	Biocreative3	No	ninguno	309.99 s	0.82
	Corpus Completo	Si	spacy	28.20	0.81
	HPRD50	Si	patrones	7.63 s	0.75
	IEPA	Si	spacy	4.27 s	0.77
	LLL	Si	patrones	6.60 s	0.73
KNN	AImed	Si	spacy	6.63 s	0.87
	Biocreative3	No	ninguno	312.15 s	0.77
	Corpus Completo	Si	spacy	27.44 s	0.82
	HPRD50	Si	patrones	5.84 s	0.68
	IEPA	Si	patrones	11.12 s	0.65
	LLL	Si	spacy	2.64 s	0.70

Tabla 5.3: Resultados con representación de datos - TF-IDF

5.1.3. Resultados con representaciones de datos - WEMB

Se utilizó la representación de datos final de Word Embedding, para la cual se aplicaron dos conjuntos diferentes de clasificadores: los mismos ya utilizados en las subsecciones anteriores y los de Deep learning (DL), con el fin de comparar los resultados obtenidos y determinar si los algoritmos de DL son superiores a los de ML.

Los resultados obtenidos por los algoritmos de clasificación de ML se muestran en

la Tabla 5.4. Se observó que la mejor combinación se obtuvo con el clasificador SVM y el corpus AImed filtrado por la librería Spacy, con un tiempo de entrenamiento de 7.10 segundos y un accuracy de 0.89 %. Por otro lado, los peores resultados se obtuvieron con el clasificador NB y el corpus HPRD50 sin ningún filtro, con un tiempo de entrenamiento de 8.22 segundos y un accuracy de 0.65.

Por su parte, los resultados obtenidos por los algoritmos de DL se presentan en la Tabla 5.5. La mejor precisión se logró con el clasificador LSTM, el cual es una variante de la RNN especializada en el análisis de texto y utilizó el corpus LLL filtrado por la librería Spacy, con un tiempo de entrenamiento de 1.04 segundos y un accuracy de 0.88 %. En contraposición, los peores resultados se obtuvieron con la red CNN y el corpus completo, que es una combinación de parte de todos los corpus, filtrado por la librería Spacy, con un tiempo de entrenamiento de 27.17 segundos y un accuracy de 0.26.

En conclusión, se puede observar que el algoritmo de ML SVM obtuvo mejores resultados que el algoritmo de DL LSTM.

Clasificador	Corpus	Filtrado	Filtro	t entrenamiento	Accuracy
NB	AImed	Si	spacy	6.62 s	0.71
	Biocreative3	Si	patrones	397.30 s	0.80
	Corpus Completo	Si	spacy	28.46 s	0.81
	HPRD50	**No**	**ninguno**	**8.22 s**	**0.65**
	IEPA	Si	patrones	11.09 s	0.70
	LLL	Si	spacy	2.82 s	0.70
SVM	**AImed**	**Si**	**spacy**	**7.10 s**	**0.89**
	Biocreative3	No	ninguno	336.90 s	0.86
	Corpus Completo	Si	patrones	38.61 s	0.84
	HPRD50	Si	patrones	8.71 s	0.70
	IEPA	Si	spacy	3.59 s	0.79
	LLL	Si	spacy	2.21 s	0.70
RF	AImed	Si	spacy	7.04 s	0.87
	Biocreative3	No	ninguno	329.82 s	0.84
	Corpus Completo	Si	patrones	39.50	0.84
	HPRD50	Si	patrones	9.84 s	0.68
	IEPA	Si	spacy	4.58 s	0.77
	LLL	Si	patrones	5.18 s	0.71
KNN	AImed	Si	patrones	60.22 s	0.83
	Biocreative3	No	ninguno	329.03 s	0.71
	Corpus Completo	Si	spacy	30.76 s	0.79
	HPRD50	Si	patrones	8.77 s	0.66
	IEPA	Si	spacy	3.84 s	0.68
	LLL	Si	spacy	2.60 s	0.70

Tabla 5.4: Resultados con representaciones de datos - WEMB

5.1.3.1. Clasificadores DL

Clasificador	Corpus	Filtrado	Filtro	t entrenamiento	Accuracy
LSTM	AImed	Si	patrones	343.89 s	0.85
	Biocreative3	No	ninguno	24.26 s	0.71
	Corpus Completo	No	ninguno	13.82 s	0.27
	HPRD50	Si	patrones	393.31 s	0.63
	IEPA	No	ninguno	5.34 s	0.62
	LLL	**Si**	**spacy**	**1.04 s**	**0.88**
GRU	AImed	Si	patrones	658.32 s	0.87
	Biocreative3	No	ninguno	17.69 s	0.72
	CorpusCompleto	No	ninguno	9.72 s	0.26
	HPRD50	Si	spacy	22.53 s	0.71
	IEPA	Si	patrones	341.48 s	0.72
	LLL	Si	spacy	61.15 s	0.81
CNN	AImed	Si	patrones	2.72 s	0.87
	Biocreative3	No	ninguno	5.75 s	0.71
	Corpus Completo	**Si**	**spacy**	**27.17 s**	**0.26**
	HPRD50	No	ninguno	0.53 s	0.68
	IEPA	Si	spacy	0.45 s	0.72
	LLL	Si	spacy	0.26 s	0.75
BERT	AImed	Si	patrones	1,226.23 s	0.86
	Biocreative3	Si	patrones	2,712.67 s	0.86
	CorpusCompleto	Si	spacy	407.12 s	0.78
	HPRD50	Si	patrones	145.87 s	0.68
	IEPA	Si	spacy	85.46 s	0.75
	LLL	Si	spacy	57.63 s	0.81

Tabla 5.5: Resultados con representaciones de datos - WEMB y DL

Resultados con representaciones de datos - WEMB y DL

5.2. Resultados generales de pruebas de combinaciones de lexicón, clasificadores, corpus

Se ejecutaron pruebas combinando representaciones de datos, clasificadores, para verificar si se puede mejorar los resultados obtenidos de manera individual. Los resultados fueron los siguientes:

5.2.1. Combinación de representaciones de datos

Respecto a los modelos desarrollados utilizando diferentes combinaciones de representación de textos biológicos, con el fin de abarcar más características y complementar así los resultados, se presenta lo obtenido con cada combinación.

TF-IDF - lexicón de intensidad

En cuanto a los modelos que utilizan la combinación de TF-IDF y Lexicón de Intensidad en la representación de textos, sus resultados se presentan a continuación.

El modelo con mejor resultado en esta representación de textos fue el que utiliza el conjunto de datos AImed filtrado por Spacy en la búsqueda de proteínas en el texto, además de emplear el algoritmo clasificador Máquina de Soporte Vectorial (SVM), obteniendo un 89 % de textos clasificados correctamente. Por otro lado, el modelo con peores resultados fue el que utiliza la combinación del corpus HPRD50 filtrado con patrones para la búsqueda de proteínas, el cual contiene una cantidad menor de textos clasificados y se emplea el clasificador K-vecinos más cercanos (KNN), que otorga un Accuracy de 58 % en la clasificación de los documentos. En la Tabla B.1 se encuentran todos los resultados obtenidos por los modelos con las diferentes combinaciones de corpus y algoritmos de clasificación. Además, para un mejor análisis de estos resultados, se pueden observar los gráficos de la Figura D.4.

Word embedding - lexicón de intensidad

Se presentan los resultados de los modelos hechos con una combinación de Word Embedding y Lexicón de Intensidad. El modelo con mejores resultados fue el que utilizó el conjunto de datos AImed filtrado por Spacy, junto con el algoritmo clasificador Máquina de Soporte Vectorial (SVM), ya que obtuvo un 89 % de textos clasificados correctamente. Por el contrario, el modelo con peores resultados fue el que utilizó el corpus HPRD50 filtrado con patrones para la búsqueda de proteínas, ya que contiene una cantidad menor de textos clasificados, y empleando el clasificador K-vecinos más cercanos (KNN), que resultó en un Accuracy de 64 % en la clasificación de los documentos. Todos los resultados obtenidos por los modelos con diferentes combinaciones de corpus y algoritmos de clasificación se presentan en la Tabla B.2. Además, para un mejor análisis de estos resultados, se pueden observar los gráficos de la Figura D.5.

Word embedding - lexicón de intensidad - TF-IDF

Se describen los resultados obtenidos por la combinación de Word Embedding, Lexicón de Intensidad y TF-IDF. El modelo con mejor resultado fue el que emplea el conjunto de datos AImed filtrado por Spacy y el algoritmo clasificador Máquina de Soporte Vectorial (SVM), otorgando como resultado un 89 % de textos clasificados correctamente. El caso opuesto fue el modelo que utiliza el Corpus Completo no filtrado y se emplea el clasificador Naive Bayes (NB), siendo el peor resultado con un Accuracy de solo 60 % en la clasificación. Todos los resultados obtenidos por modelos que combinan distintos corpus y algoritmos de clasificación se pueden observar en la Tabla B.3. Además, para un mejor análisis de estos resultados, se pueden observar los gráficos de la Figura D.6. Además, para un mejor análisis de estos resultados, se pueden observar los gráficos de la Figura D.7.

5.2.2. Combinación de clasificadores

A continuación se describen los mejores resultados obtenidos por los modelos con combinación de clasificadores. Puede entenderse cierta continuidad entre el conjunto de comparaciones anterior y este, ya que también se agrupan los valores considerando los corpus o el tipo de representación de documentos.

TF-IDF - lexicón de intensidad - TF-IDF/lexicón de intensidad

Se presentan los resultados de los modelos que utilizan esta combinación de representaciones (TF-IDF, Lexicón de Intensidad y la combinación de ambos) pero empleando el mismo clasificador. El modelo con mejor rendimiento fue aquel que empleó el conjunto de datos AImed filtrado por Spacy y el algoritmo clasificador K-vecinos más cercanos (KNN), logrando una tasa de acierto del 88 % en la clasificación de los textos. En contraste, el modelo que empleó el corpus HPRD50 filtrado por patrones, que contiene menos textos clasificados, y el clasificador K-vecinos más cercanos (KNN) obtuvo la tasa de acierto más baja, con un 58 % de textos clasificados correctamente. Se pueden encontrar todos los resultados obtenidos por los modelos con diferentes combinaciones de corpus y algoritmos de clasificación en la Tabla B.4.

Word embedding - lexicón de intensidad - word embedding/lexicón de intensidad

Se presentan los resultados de los modelos que utilizan Word Embedding, Lexicón de Intensidad y la combinación de ambas, y en cada modelo comparten el mismo clasificador, pero al formular nuevos modelos se realiza el mismo procedimiento con los distintos clasificadores y corpus. El modelo más efectivo en la clasificación de textos fue aquel que empleó el conjunto de datos AImed filtrado por Spacy para buscar proteínas en el texto, y utilizó el algoritmo clasificador Máquina de Soporte Vectorial (SVM), con una tasa de aciertos (accuracy) del 89 %. El modelo menos efectivo utilizó el corpus HPRD50 filtrado por patrones, el cual contiene menos textos clasificados, y se empleó el clasificador K-vecinos más

cercanos (KNN), obteniendo un 62 % de precisión en la clasificación de documentos. La Tabla B.5 muestra todos los resultados obtenidos por los modelos que utilizan diferentes combinaciones de corpus y algoritmos de clasificación. Además, para un mejor análisis de estos resultados, se pueden observar los gráficos de la Figura D.8.

Resultados de modelos con uso de combinación de clasificadores

Otra estrategia para explorar diferentes resultados consiste en utilizar modelos con la misma representación de documentos pero con diferentes algoritmos de clasificación, combinando en diferente orden el uso de K-vecinos más cercanos (KNN), Random Forest (RF) y Naive Bayes (NB). En este caso, se utiliza la combinación de TF-IDF y lexicón de intensidad como representación de texto, y se experimenta con los grupos de clasificadores **KNN-NB-RF**, **SVM-KNN-RF**, **SVM-NB-KNN** y **SVM-NB-RF**. Los valores de accuracy obtenidos oscilan entre el 86 % y el 88 %, y se muestran en detalle en las Tablas C.1, C.2, C.3 y C.4.

5.3. Comparación de resultados

A continuación se describen los mejores resultados obtenidos por los modelos mencionados anteriormente, agrupados según el algoritmo de clasificación, corpus o el tipo de representación de documentos utilizado.

Mejores resultados según el clasificador utilizado

En la primera comparación de resultados, se han separado los modelos según los algoritmos de clasificación utilizados. El modelo que obtuvo el mejor resultado alcanzó un 90 % de accuracy, empleando el clasificador Naive Bayes (NB) con lexicón de intensidad como representación de documentos y utilizando el corpus AImed filtrado con Spacy. En la Tabla A.1, se presentan los resultados obtenidos por los otros modelos desarrollados.

Mejores resultados según la representación del documento

En la Tabla A.3 se muestran los resultados obtenidos por los modelos desarrollados. En esta segunda comparación de resultados, se analiza la representación del documento utilizada. El modelo que obtuvo el mejor resultado, con una precisión del 90 %, utilizó la representación de lexicón de intensidad y el clasificador Naive Bayes (NB), junto con el corpus AImed filtrado con Spacy. Este modelo es el mismo que se obtuvo en la comparación anterior, que fue según el algoritmo de clasificación utilizado. Es importante mencionar que, en todos los modelos con mejores resultados en cuanto a la representación de los textos, se utilizó el corpus AImed filtrado con Spacy para la búsqueda de proteínas dentro del texto.

Mejores resultados según el corpus utilizado

En la Tabla A.4 se presentan los resultados obtenidos por los diferentes modelos desarrollados. Esta tercera comparación de resultados se centra en los corpus utilizados, y el modelo con el mejor resultado de 90 % de precisión utiliza el corpus AImed filtrado con Spacy junto con el clasificador Naive Bayes (NB) y lexicón de intensidad como representación de documentos. Este modelo también fue el mejor en las comparaciones anteriores, donde se evaluaron los algoritmos de clasificación y las representaciones de documentos. Cabe destacar que, en todos los modelos con los mejores resultados según la representación de los textos, se utilizó el corpus AImed filtrado con Spacy para la identificación de proteínas en el texto.

Mejores resultados de la combinación de clasificadores

A continuación se describen los mejores resultados obtenidos por los modelos con combinación de clasificadores. Puede entenderse cierta continuidad entre el conjunto de comparaciones anterior y este, ya que también se agrupan los valores considerando los corpus o el tipo de representación de documentos.

Mejores resultados de según el corpus utilizado

Esta cuarta comparación de resultados tiene como separador los corpus utilizados por los modelos con combinación de clasificadores. El mejor resultado obtenido es de 88 % de accuracy, el modelo utiliza el corpus AImed filtrado con Spacy al igual que los otros modelos con mejores resultados, usando como clasificador SVM-NB-KNN y la combinación de TF-IDF con lexicón de intensidad como representación de documentos. En la Tabla A.6 se pueden observar los resultados obtenidos por los otros modelos desarrollados.

Mejores resultados según la combinación de clasificadores utilizada

La quinta comparación de resultados tiene como separador los mejores resultados según la combinación de algoritmos de clasificación. El mejor resultado obtenido es de 88 % de accuracy, el modelo utiliza el clasificador SVM-NB-KNN con la combinación de TF-IDF con lexicón de intensidad como representación de documentos y el corpus utilizado es el de AImed filtrado con Spacy, siendo el mismo modelo que en la cuarta comparación. Algo importante a mencionar es que en todos los modelos con combinación de clasificadores los mejores resultados fueron utilizando el corpus AImed filtrado con Spacy para la búsqueda de proteínas dentro del texto. En la Tabla A.7 se pueden observar los resultados obtenidos por los otros modelos desarrollados.

Entre algunos aspectos importantes que se pueden destacar en este capítulo:

Podemos observar que la mayoría de los modelos con mejores resultados obtuvieron valores similares entre 86-88 % de accuracy siendo el mejor valor de un 90 %, se diferencian en el algoritmo clasificador utilizado. También se pudo observar en la experimentación que algoritmos de Machine Learning tradicionales o combinados, no podían soportar una

cantidad exorbitante de datos a diferencia de los algoritmos de Deep Learning que podían soportar una cantidad mayor de datos.

Al generar diferentes variantes de algoritmos de clasificación, algoritmos de clasificación combinados y algoritmos en los cuales se utilizan diferentes representaciones de documentos combinadas, combinación de representaciones de documento o diferentes clasificadores, no siempre van a dar buenos resultados, en ocasiones los resultados mejores son usando un clasificador y una representación de documentos, ya que a pesar de suponer una posible sinergia entre representaciones de documentos, esto no ocurría. En este caso los mejores resultados se obtuvieron desde el uso del clasificador Naive Bayes junto con el lexicon de intensidad como representación de documentos, con el corpus AImed.

5.4. Conclusión de capítulo

Se intentaron diferentes enfoques para lograr mejores resultados, entre ellos, la combinación de representaciones de documentos, la utilización de algoritmos de clasificación de ML y el empleo de clasificadores de DL para su comparación. No obstante, no se encontró una variación significativa en relación con el uso individual de las representaciones de documentos. En este contexto, la representación de Lexicón de Intensidad propuesta para la búsqueda de IPP fue la que arrojó los mejores resultados.

Capítulo 6

Aplicación de modelos, validación por docking y listado final

En este capítulo se describe el proceso de implementación del docking o acoplamiento molecular dentro de la búsqueda de IPP en los textos, para esto se describe la búsqueda de las proteínas mediante sus códigos del Protein data bank (PDB), las aplicaciones utilizadas para realizar el docking y se comparan de resultados obtenidos.

6.1. Aplicaciones de Modelos

Previo al proceso de docking se aplican los modelos diseñados en el Capítulo 4, cuyos resultados se revisan en el Capítulo 5. En aquella etapa, se obtiene la información de textos completos extraídos del PMC sin clasificar. Junto a esto, se hace uso del Lexicón positivo y un diccionario de proteínas generado a partir de Biopython y PDB, y estos contribuyen a facilitar la búsqueda de IPP en el texto. Entonces, al aplicar los modelos se obtienen de salida textos que probablemente contendrán información de IPP y, por el motivo que se detalla más adelante dentro de este Capítulo, se espera extraer las proteínas del texto en formato PDB. Dentro del proceso general este paso es el que se resume en la Figura 6.1.

Al trabajar con textos completos descargados desde PMC (Pubmed central), al algoritmo clasificador le demandaba más trabajo procesar los textos, por lo que luego de descargarlos se separaron por párrafos para facilitar su procesamiento. La Figura 6.2 muestra un ejemplo de un párrafo extraído de un artículo.

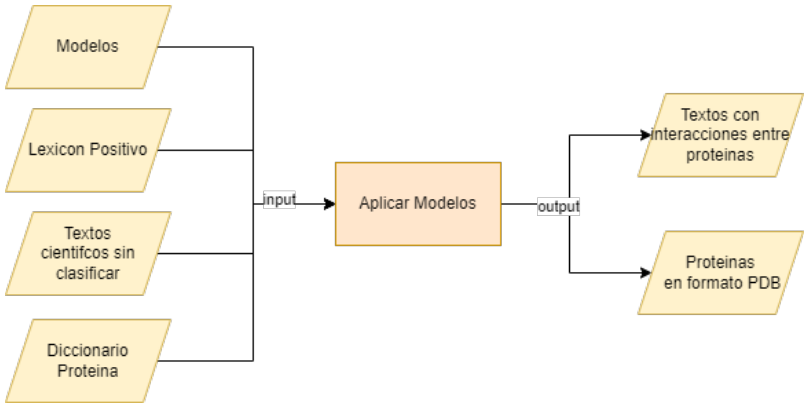


Figura 6.1: Resumen aplicar modelos de clasificación para la búsqueda de ipp

The complex structure of the SARS-CoV-2 RBD and ACE2 complex (PDB: 6M0J), SARS-CoV-2 Spike with furin cleavage site structure (PDB: 7FG7) and Human furin structure (PDB: 5MIM) were obtained from the PDB database.

Figura 6.2: Ejemplo párrafo extraído del articulo 10008190 de la base de datos de artículos biológicos PMC

La solución presentada trajo consigo un riesgo, ya que en algunos casos se mencionaba una proteína en un párrafo del texto y la otra en otro, mientras que la interacción se mencionaba en otra sección del texto. En consecuencia, si se utiliza un modelo clasificador que se basa únicamente en la clasificación por párrafos, puede resultar eficiente en ciertos casos, pero en otros puede ser contraproducente ya que la relación entre las proteínas sólo se podría determinar al analizar el texto completo y no se detectaría al analizar los párrafos por separado.

El uso de los nombres de las proteínas para llevar a cabo la búsqueda generó complicaciones, ya que en muchos casos la búsqueda por nombre no arrojaba el resultado preciso. En algunas ocasiones se generaban docking con dos proteínas que no interactuaban entre sí, lo que resultaba en un resultado negativo en el docking. Por esta razón, se decidió realizar la búsqueda principalmente a través del código del PDB de las proteínas, ya que esto permitía encontrar el código exacto correspondiente a la proteína mencionada en el texto. Al hacer esto, se previenen errores y se logra detectar correctamente las interacciones proteína-proteína (IPP), lo que mejora la precisión del proceso de docking.

Siguiendo esta idea, se usó el API de PMC para descargar 10,000 papers que hicieran mención de Protein-protein interaction. En la Figura 6.3 se detalla el proceso de filtrado y los criterios de descarte. En primer lugar, se aplicó el filtro A, el cual consideró únicamente

los párrafos que contenían al menos 2 códigos del PDB y palabras presentes en el diccionario del lexicón positivo, es decir, que su mención se asocie a una mayor probabilidad de referirse a una interacción entre proteínas. Paralelamente se aplicaron los modelos de clasificación que en el capítulo 5 otorgaron mejores resultados y se consideraron los textos seleccionados por ambos filtros. Este primer criterio de descarte permitió obtener 220 párrafos, que contenían un total de 1078 pares de proteínas

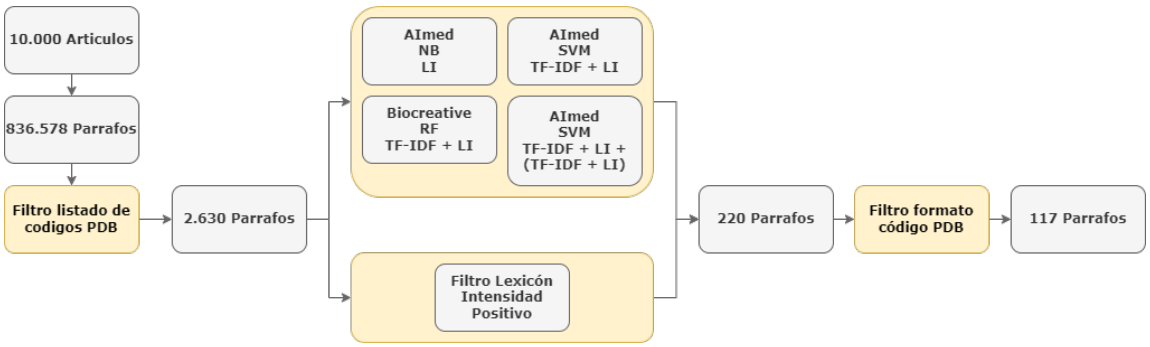


Figura 6.3: Proceso de revisión de párrafos

Posteriormente, se aplicó el criterio de descarte B, el cual validó que los PBD ID respectivos a las proteínas mencionadas en los párrafos tuvieran el formato correcto, que corresponde a un código alfanumérico de 4 caracteres, por lo que se descartan códigos que contengan más caracteres o estén combinados con palabras. De este paso se obtuvo 117 párrafos, con 772 pares de proteínas.

Como tercer criterio o C, se realizó una revisión de los pares obtenidos, descartando aquellos que no mencionaban realmente una interacción de proteínas, lo que resultó en 555 pares de proteínas. Finalmente, se aplicó el criterio D, que eliminó los pares repetidos, obteniendo como resultado final un total de 490 pares de proteínas.

6.2. Realización de Docking

Teniendo los artículos filtrados, se descargan los códigos correspondientes a las proteínas encontradas en formato PDB, se realiza el Docking propiamente tal con las aplicaciones de Docking anteriormente descritas, se registra el complejo generado en formato PDB, por lo tanto con su estructura tridimensional, a partir de cada par de proteínas que interactúan. Dado esto, se obtiene el listado de pares de proteínas y el listado de complejos generados con estos pares. Dentro del proceso general de esta Tesis, lo detallado aquí corresponde al último paso, rescatado en la Figura 6.4.

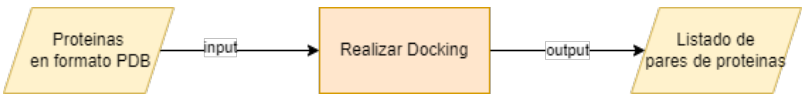


Figura 6.4: Resumen proceso de realizar docking con los pares de proteínas encontrados

Para el acoplamiento de las proteínas encontradas en los textos se hace uso de diferentes aplicaciones de docking disponibles de forma web o de escritorio. Las aplicaciones son utilizadas a modo de caja negra y como parámetro de entrada se entregan los pares de proteínas. Durante el proceso de validación que se detallará más adelante en este capítulo, se agregó otra aplicación de Docking, ClusPRO. Por lo tanto, el listado de aplicaciones utilizadas se detalla a continuación:

- pyDockWeb[1] (Jiménez-García et al., 2013)
- PatchDock[2] (Schneidman-Duhovny et al., 2005)
- GRAMM[3] (Tovchigrechko y Vakser, 2006)
- Hex[4] (Macindoe et al., 2010)
- PEPPI[5] (Bell et al., 2022)
- ClusPro[6] (Kozakov et al., 2013)

Previo a la implementación se hicieron pruebas del proceso de docking con las aplicaciones mencionadas, que arrojaron resultados deficientes al hacer la búsqueda de nombres de proteínas, dificultad que se detallará más adelante, pero en general en estas pruebas se observó que las aplicaciones funcionaban correctamente. Posteriormente, al momento de hacer nuevamente el proceso de docking con nueva información para la búsqueda, se observó que los servidores de Patchdock y Hex no estaban disponibles, el servidor de GRAMM cambió de URL, lo cual se resolvió al encontrar la nueva dirección, y PEPPI contemplaba un tiempo excesivo en comparación con los demás. Finalmente se trabajó con pyDockWeb, ClusPro y principalmente GRAMM.

Un aspecto a comparar es que pyDockWeb requiere que se proporcione la cadena de la proteína con la que interactúan las demás, mientras que ClusPro lo considera opcional. Por otro lado, GRAMM no solicita este parámetro, ya que lo determina automáticamente. Cabe señalar que todas las aplicaciones requieren que se ingrese una proteína en formato PDB como parámetro de entrada.

Durante el proceso, las tres aplicaciones funcionaron correctamente, pero GRAMM destacó por su capacidad para realizar el docking sin mayores complicaciones en términos de tamaño de las proteínas ingresadas o tiempos de uso.

Luego de esto, es necesario evaluar los resultados obtenidos en esta tarea, y para hacerlo existen las métricas de docking mencionadas en el Capítulo 2, que permiten hacer una comparación entre el acoplamiento desarrollado por alguna aplicación y el complejo nativo. Al no conocer el complejo nativo, como ocurre en este caso, se hace una comparación entre

[1] https://life.bsc.es/pid/pydockweb/
[2] https://bioinfo3d.cs.tau.ac.il/PatchDock/
[3] https://gramm.compbio.ku.edu/
[4] http://hexserver.loria.fr/
[5] https://zhanggroup.org/PEPPI/
[6] https://cluspro.org/login.php

los mejores complejos predichos y así se obtiene un aproximado. Con este resultado se puede calcular el valor de las métricas RMSD, y posteriormente el Log(LR), teniendo así la manera de evaluar la calidad del Docking. En la Tabla 6.1 se exponen un parte de los resultados obtenidos de evaluar con métricas los complejos agrupados por las aplicaciones de docking, valores que se explicarán a continuación.

Los resultados arrojados por las aplicaciones de Docking y sus métricas evaluadoras se detallan en la Tabla 6.1.

PMC Id	**Proteína 1**	**Proteína 2**	**GRAMM RMSD**	**GRAMM Log(LR)**	**pyDockWeb RMSD**	**pyDockWeb Log(LR)**	**ClusPro RMSD**	**ClusPro Log(LR)**
9645286	4GCJ	3QTW	1.45	-0.72	24.74	-5.73	0.51	-0.12
9722428	1NFI	6VXX	41.39	-6.75	19.57	-5.26	35.5	-6.45
9746828	4CI2	1BG1	45.02	-6.92	33.75	-6.35	23.39	-5.62
10008190	7FG7	5MIM	20.39	-5.34	21.92	-5.49	0.28	-0.04
9719429	3UT3	7DV6	23.28	-5.61	0	0	40.24	-6.7
9750146	3MXF	4CI3	14.14	-4.61	0	0	9.09	-3.75

Tabla 6.1: Resultados obtenidos de los complejos predichos por las aplicaciones de docking

En la Tabla 6.1 se muestran 6 artículos como ejemplo, en los que en cada uno se seleccionó un par de proteínas que posiblemente interactúan entre ellas, en las que se ejecutaron las aplicaciones de docking GRAMM, pyDockWeb y ClusPro. Para poder evaluar los resultados obtenidos se implementaron las métricas RMSD y Log(LR). En relación a los resultados obtenidos en Log(LR) los mejores resultados son los que dan valores de Log(LR) entre 0 y -1, ya que se correlacionan con una mayor probabilidad de unión, mientras que los valores menores a -1 se asocian a una muy baja probabilidad de unión de los pares de proteínas implicadas en esos casos. Con este criterio, mejor resultado lo podemos observar en el acoplamiento de las proteínas contenidas en el artículo de código 10008190 presente en la misma tabla, con un Log(LR) de -0,04.

Como se mencionó anteriormente, se decidió utilizar principalmente la herramienta GRAMM para analizar los 490 pares de proteínas.

Sin embargo, durante el proceso de docking, se detectaron algunos problemas en la aplicación GRAMM, lo que permitió analizar de forma correcta solo un total de 351 pares de proteínas. Para seleccionar los pares que presentaran una mayor precisión en el docking, se utilizó la métrica RMSD como criterio de descarte, y se descartaron los pares con un valor menor a 10. Finalmente, se obtuvieron un total de 180 pares de proteínas considerados como los posibles pares encontrados en el estudio.

6.3. Comparación de nuevo listado con Base de datos IPP

Después de realizar el docking en 351 pares de proteínas, se obtuvieron resultados positivos en 180 de ellos. Para validar la existencia de estos pares de proteínas, se utilizó la base de datos de interacciones entre proteínas Biogrid.

Durante la búsqueda en Biogrid, se descubrió un problema de integración con la base de datos de proteínas Protein Data Bank. Al analizar la base de datos de Biogrid, se encontró que esta registraba los códigos y Uniprot de los pares de proteínas.

Para integrar los datos, se empleó Uniprot como intermediario, debido a que esta base de datos también almacena la relación entre la proteína de Uniprot y la proteína de Protein Data Bank. Se utilizaron los servicios de Protein Data Bank para buscar los códigos Uniprot y los servicios de Biogrid para buscar los pares correspondientes a dichos códigos Uniprot.

De los 180 pares validados, se encontraron 51 registrados en Biogrid, lo que significa que los otros 129 no se encontraron registrados en esta base de datos, es decir, se trata de nuevos pares de proteínas.

El proceso llevado a cabo se resume en la Figura 6.5, en la cual se muestra cómo los 180 pares de proteínas se utilizaron como parámetro de entrada para buscar información en la base de datos de Uniprot y posteriormente validarlos dentro de la base de datos de Biogrid.

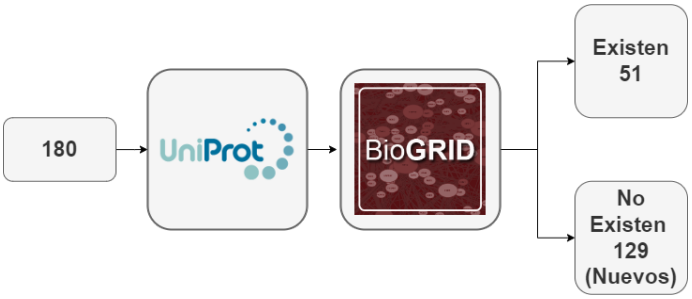


Figura 6.5: Comparación con BD de IPP

Capítulo 7

Conclusión y trabajos futuros

El motivo de esta tesis es encontrar un método de minería de texto que permita la clasificación de textos con la posible interacción entre proteínas dentro del mismo, por lo cual se utilizó un conjunto de corpus pre entrenado para la evaluación de los modelos desarrollados, además del uso de abstracts con una posible interacción de Pubmed para la búsqueda de complejos proteína-proteína y la creación de sus modelos tridimensionales con el uso de aplicaciones de docking.

Para abordar esta tarea, se descompuso el problema en varios procesos: generación de un lexicón, generación de representaciones de documentos, aplicación de algoritmos de clasificación, aplicación de modelos seleccionados y validación mediante docking. Como resultado de estos procesos, se obtuvieron los siguientes resultados:

Se plantea innovar en los algoritmos de clasificación utilizando un lexicón de intensidad específico para buscar interacciones proteína-proteína (IPP). Utilizando BERT y K-means, se creó un lexicón de intensidad con 182 palabras positivas y 28 negativas.

Se representaron documentos usando Word Embedding, TF-IDF, lexicón de intensidad y su combinación, tras preprocesar el texto original al convertirlo a minúsculas, eliminar stopwords y lematizar palabras.

Se aplicaron diferentes algoritmos de clasificación en corpus etiquetados, sin variaciones notables al combinar diferentes representaciones y algoritmos. El modelo más exitoso utilizó el algoritmo clasificador Naive Bayes junto con el léxico de intensidad, alcanzando una precisión del 90 %.

Se analizaron 10000 artículos de PMC en busca de interacciones entre proteínas, identificando 117 párrafos, con un total de 490 pares de posibles IPP.

Se llevó a cabo el docking en los 490 pares obtenidos utilizando la aplicación GRAM con sus parámetros por defecto. Como resultado de la validación utilizando las métricas RMSD y Log(LR), se logró clasificar correctamente un total de 180 pares. Se validaron 180 pares mediante docking y se compararon con la base de datos Biogrid. Se encontraron 51 interacciones existentes y 129 nuevas.

Con estos resultados se puede decir con lo anterior que se cumplió con el objetivo general y la hipótesis.

7.1. Trabajos futuros

Para trabajos futuros se desea mejorar los resultados obtenidos en el desarrollo de los objetivos de esta tesis, por lo que algunas ideas a proponer se relacionan con lo siguiente.

El uso del Lexicón de Intensidad como herramienta emergente puede mejorar significativamente la representación de documentos. En la investigación realizada, se logró obtener buenos resultados al orientarla por primera vez hacia la búsqueda de IPP. Por lo tanto, se recomienda ampliar el tamaño del Lexicón para maximizar su efectividad en la representación de documentos. Además de aplicar algoritmos de deep learning sobre el nuevo Lexicón de intensidad y comparar los nuevos modelos.

Se propone explorar el uso de redes neuronales para mejorar la búsqueda y detección precisa de proteínas en textos biológicos, dado que actualmente la identificación de los nombres de proteínas representa una dificultad y genera un considerable esfuerzo para la generación de una lista precisa de complejos proteína-proteína. Una solución a este problema podría ser el uso de redes neuronales que, a partir del contexto del texto, podrían realizar predicciones precisas, considerando que en la mayoría de los casos no se menciona el nombre completo de la proteína, sino que se hace referencia a ella mediante diferentes palabras, lo que permitiría inferir el nombre completo a partir de la información disponible en el texto.

Dado el problema experimentado en el proceso de validación del Docking, en el que el algoritmo clasificador no detectaba las mismas interacciones que el proceso manual, se debe plantear como trabajo futuro la mejora del modelo clasificador para que sus resultados sean más cercanos a la realidad, ya que al mejorarlo podrían utilizarse de forma conjunta con otros métodos sin discrepancias.

Se puede mejorar la precisión del proceso de docking al utilizar diferentes parámetros. Esto permitiría obtener mejores resultados al realizar el acoplamiento entre pares de proteínas.

Por ultimo, lograr una integración de los resultados obtenidos con la base de datos Protein data bank (PDB).

Referencias

Cecilia N Arighi, Phoebe M Roberts, Shashank Agarwal, Sanmitra Bhattacharya, Gianni Cesareni, Andrew Chatr-Aryamontri, Simon Clematide, Pascale Gaudet, Michelle Gwinn Giglio, Ian Harrow, et al. Biocreative iii interactive task: an overview. *BMC bioinformatics*, 12(8):1–21, 2011.

Varsha D Badal, Petras J Kundrotas, y Ilya A Vakser. Text mining for modeling of protein complexes enhanced by machine learning. *Bioinformatics*, 37(4):497–505, 2021.

R. A. Bastarrachea, H. Laviada-Molina, I. Machado-Domínguez, Kent J., J. C. López-Alvarenga, y A. G. Comuzzie. El receptor de insulina como objetivo farmacogenómico: potenciando su señalización intracelular. *Revista de endocrinología y nutrición*, 13(4):180–189, 2005.

Thomas Bayes. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418, 1763.

Eric W Bell, Jacob H Schwartz, Peter L Freddolino, y Yang Zhang. Peppi: Whole-proteome protein-protein interaction prediction through structure and sequence similarity, functional association, and machine learning. *Journal of Molecular Biology*, pág. 167530, 2022.

Yoshua Bengio, Patrice Simard, y Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Helen Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady Bhat, Helge Weissig, Ilya Shindyalov, y Philip Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

Luis Paulo Vieira Braga, Luis Iván Ortiz Valencia, y Santiago Segundo Ramirez Carvajal. *Introducción a la Minería de Datos*. Editora E-papers, 2009.

Razvan Bunescu, Ruifang Ge, Rohit J Kate, Edward M Marcotte, Raymond J Mooney, Arun K Ramani, y Yuk Wah Wong. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial intelligence in medicine*, 33(2):139–155, 2005.

Daniel R Caffrey, Shyamal Somaroo, Jason D Hughes, Julian Mintseris, y Enoch S Huang. Are protein–protein interfaces more conserved in sequence than the rest of the protein surface? *Protein Science*, 13(1):190–202, 2004.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, y Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

UniProt Consortium. Uniprot: a hub for protein information. *Nucleic acids research*, 43(D1):204–212, 2015.

Sjoerd J de Vries, Aalt DJ van Dijk, y Alexandre MJJ Bonvin. Whiscy: what information does surface conservation yield? application to data-driven docking. *Proteins: Structure, Function, and Bioinformatics*, 63(3):479–489, 2006.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, y Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jing Ding, Daniel Berleant, Dan Nettleton, y Eve Wurtele. Mining medline: abstracts, sentences, or phrases? En *Biocomputing 2002*, págs. 326–337. World Scientific, 2001.

Jing Ding, Daniel Berleant, Dan Nettleton, y Eve Syrkin Wurtele. Mining medline: Abstracts, sentences, or phrases? En *Pacific Symposium on Biocomputing*, págs. 326–337. 2002.

Katrin Fundel, Robert Küffner, y Ralf Zimmer. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007a.

Katrin Fundel, Küffner Robert, y Zimmer Ralf. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007b.

Germán Gémar y José Antonio Jiménez-Quintero. Text mining social media for competitive analysis. *Tourism & Management Studies*, 11(1):84–90, 2015.

Ian Goodfellow, Yoshua Bengio, y Aaron Courville. *Deep learning*. MIT press, 2016.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, y Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, tomo 2. Springer, 2009.

Brian Jiménez-García, Carles Pons, y Juan Fernández-Recio. pydockweb: a web server for rigid-body protein–protein docking using electrostatics and desolvation scoring. *Bioinformatics*, 29(13):1698–1699, 2013.

Gerald Karp. *Cell and molecular biology: concepts and experiments*. John Wiley & Sons, 2009.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, y Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

Dima Kozakov, Dmitri Beglov, Tanggis Bohnuud, Scott E Mottarella, Bing Xia, David R Hall, y Sandor Vajda. How good is automated protein docking? *Proteins: Structure, Function, and Bioinformatics*, 81(12):2159–2166, 2013.

Miroslav Kubat. *An introduction to machine learning*. Springer, 2017.

Nicholas M Luscombe, Dov Greenbaum, Mark Gerstein, et al. What is bioinformatics? an introduction and overview. *Yearbook of medical informatics*, 1(83-100):2, 2001.

Gary Macindoe, Lazaros Mavridis, Vishwesh Venkatraman, Marie-Dominique Devignes, y David W Ritchie. Hexserver: an fft-based protein docking server powered by graphics processors. *Nucleic acids research*, 38(suppl_2):W445–W449, 2010.

Christopher D Manning. *An introduction to information retrieval*. Cambridge university press, 2009.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, y Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Julian Mintseris y Zhiping Weng. Structure, function, and evolution of transient and obligate protein–protein interactions. *Proceedings of the National Academy of Sciences*, 102(31):10930–10935, 2005.

SPFGH Moen y Tapio Salakoski2 Sophia Ananiadou. Distributional semantics resources for biomedical text processing. *Proceedings of LBM*, págs. 39–44, 2013.

Claire Nédellec. Learning language in logic-genic interaction extraction challenge. En *4. Learning language in logic workshop (LLL05)*. ACM-Association for Computing Machinery, 2005.

Andrew Ng. Supervised learning, discriminative algorithms. *ML CS229 Lecture Notes*, págs. 1–30, 2017.

Irene MA Nooren y Janet M Thornton. Diversity of protein–protein interactions. *The EMBO journal*, 22(14):3486–3492, 2003.

C Nédellec. Learning language in logic - genic interaction extraction challenge. En *Proceedings of the Learning Language in Logic 2005 Workshop at the International Conference on Machine Learning*. 2005.

Nagesh C Panyam, Karin Verspoor, Trevor Cohn, y Kotagiri Ramamohanarao. Exploiting graph kernels for high performance biomedical relation extraction. *Journal of biomedical semantics*, 9(1):1–11, 2018.

Tony Pawson y Piers Nash. Assembly of cell regulatory systems through protein interaction domains. *science*, 300(5618):445–452, 2003.

Yifan Peng y Zhiyong Lu. Deep learning for extracting protein-protein interactions from biomedical literature. *BioNLP 2017*, págs. 29–38, 2017.

Yifan Peng, Shankai Yan, y Zhiyong Lu. Transfer learning in biomedical natural language processing: an evaluation of bert and elmo on ten benchmarking datasets. *arXiv preprint arXiv:1906.05474*, 2019.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, y Tapio Salakoski. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC bioinformatics*, 8(1):1–24, 2007.

Randy J Read, Claudia Millán, Airlie J McCoy, y Thomas C Terwilliger. Likelihood-based signal and noise analysis for docking of models into cryo-em maps. *Acta Crystallographica Section D: Structural Biology*, 79(4), 2023.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Dina Schneidman-Duhovny, Yuval Inbar, Ruth Nussinov, y Haim J Wolfson. Patchdock and symmdock: servers for rigid and symmetric docking. *Nucleic acids research*, 33(suppl_2):W363–W367, 2005.

Alejandra Segura-Navarrete, Claudia Martinez-Araneda, Christian Vidal-Castro, y Clemente Rubio-Manzano. A novel approach to the creation of a labelling lexicon for improving emotion analysis in text. *The Electronic Library*, 2021.

M Seguí. Estructura y propiedades de las proteínas. 2011.

Asho Srivastava y Mehran Sahami. *Text mining : classification, clustering, and applications*. CRC Press, 2009.

Pedro HM Torres, Ana CR Sodero, Paula Jofily, y Floriano P Silva-Jr. Key topics in molecular docking for drug design. *International journal of molecular sciences*, 20(18):4574, 2019.

Andrey Tovchigrechko y Ilya A Vakser. Gramm-x public web server for protein–protein docking. *Nucleic acids research*, 34(suppl_2):W310–W314, 2006.

Tung Tran y Ramakanth Kavuluru. An end-to-end deep learning architecture for extracting protein–protein interactions affected by genetic mutations. *Database*, 2018:1–13, 2018.

Oleg Trott y Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

Chih-Fong Tsai. Bag-of-words representation in image annotation: A review. *International Scholarly Research Notices*, 2012, 2012.

Ian H Witten y Eibe Frank. Data mining: practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, 31(1):76–77, 2002.

Apéndice A

Tablas con los mejores resultados.

A continuación se presenta cada tabla con los mejores resultados según, clasificador, representación de datos y combinaciones de estos. Cada tabla puede contener los siguientes datos, Representación de documento (RP), Algoritmo clasificador, corpus, si es filtrado el corpus, con que filtro, en caso que si, tiempo de entrenamiento (en segundos) Accuracy, Recall, Precision, F1-Score.

RP	Clasificador	Corpus	Filtrado	Filtro	t (seg.)	Accuracy	Recall	Precision	F1 Score
TF-IDF/LEXICON	SVM	AImed	Si	Spacy	17.91	0.89	0.89	0.89	0.89
TF-IDF/LEXICON	RF	Biocreative3	No	ninguno	659.62	0.82	0.82	0.82	0.82
MULTIPLE TF-IDF	KNN	AImed	Si	Spacy	171.84	0.88	0.88	0.88	0.88
LEXICON	**NB**	**AImed**	**Si**	**Spacy**	**8.19**	**0.90**	**0.90**	**0.90**	**0.90**

Tabla A.1: Mejores resultados según el clasificador

Clasificador	Corpus	Filtrado	Filtro	t (seg.)	Accuracy	Recall	Precision	F1-Score
LSTM	**LLL**	**Si**	**Spacy**	**1.04**	**0.88**	**0.88**	**0.88**	**0.88**
GRU	AImed	Si	patrones	658.32	0.87	0.87	0.87	0.87
CNN	AImed	Si	patrones	2.72	0.87	0.87	0.87	0.87
BERT	AImed	Si	patrones	1,226.23	0.86	0.86	0.86	0.86

Tabla A.2: Mejores resultados según el clasificador de DL

RP	Clasificador	t (seg.)	Accuracy	Recall	Precision	F1 Score
TF-IDF	SVM	6.61	0.88	0.88	0.88	0.88
TF-IDF/LEXICON	SVM	17.91	0.89	0.89	0.89	0.89
WEMB	SVM	7.10	0.89	0.89	0.89	0.89
WEMB/LEXICON	SVM	37.21	0.89	0.89	0.89	0.89
WEMB/LEXICON/TF-IDF	SVM	23.34	0.89	0.89	0.89	0.89
MULTIPLE/TF-IDF	KNN	171.84	0.88	0.88	0.88	0.88
MULTIPLE/WEMB	SVM	197.95	0.89	0.89	0.89	0.89
LEXICON	**NB**	**8.19**	**0.90**	0.90	0.90	0.90

Tabla A.3: Mejores resultados según la representación de documento

RP	Clasificador	Corpus	Filtrado	Filtro	t (seg.)	Accuracy	Recall	Precision	F1 Score
LEXICON	**NB**	**AImed**	**Si**	**Spacy**	**8.19 s**	**0.90**	0.90	0.90	0.90
WEMB/LEXICON/TF-IDF	SVM	Biocreative3	No	ninguno	13,792.37 s	0.88	0.88	0.88	0.88
WEMB/LEXICON/TF-IDF	SVM	Corpus Completo	Si	patrones	438.82 s	0.86	0.86	0.86	0.86
TF-IDF	SVM	HPRD50	Si	patrones	8.82 s	0.76	0.76	0.76	0.76
WEMB/LEXICON	SVM	IEPA	Si	Spacy	7.20 s	0.83	0.83	0.83	0.83
TF-IDF	RF	LLL	Si	patrones	6.60 s	0.73	0.73	0.73	0.73

Tabla A.4: Mejores resultados según el corpus utilizado

Clasificador	Corpus	Filtrado	Filtro	t (seg.)	Accuracy	Recall	Precision	F1 Score
GRU	AImed	Si	patrones	658.32 s	0.87	0.87	0.87	0.87
BERT	Biocreative3	Si	patrones	2,712.67 s	0.86	0.86	0.86	0.86
BERT	Corpus Completo	Si	Spacy	407.12 s	0.78	0.78	0.78	0.78
GRU	HPRD50	Si	Spacy	22.53 s	0.71	0.71	0.71	0.71
BERT	IEPA	Si	Spacy	85.46 s	0.75	0.75	0.75	0.75
LSTM	**LLL**	**Si**	**Spacy**	**1.04 s**	**0.88**	**0.88**	**0.88**	**0.88**

Tabla A.5: Mejores resultados según el corpus utilizado con clasificadores de DL

Clasificador	t (seg.)	Accuracy	Recall	Precision	F1 Score
KNN-NB-RF	259.88 s	0.86	0.86	0.86	0.86
SVM-KNN-RF	**256.07 s**	**0.88**	**0.88**	**0.88**	**0.88**
SVM-NB-RF	348.81 s	0.86	0.86	0.86	0.86
SVM-NB-KNN	275.87 s	0.88	0.88	0.88	0.88

Tabla A.7: Mejores resultados según la combinación de clasificadores

Clasificador	Corpus	Filtrado	Filtro	t (seg.)	Accuracy	Recall	Precision	F1 Score
SVM-NB-KNN	**AImed**	**Si**	**Spacy**	**275.87 s**	**0.88**	0.88	0.88	0.88
SVM-NB-RF	Biocreative3	No	ninguno	9,694.70 s	0.81	0.81	0.81	0.81
SVM-NB-RF	Corpus Completo	Si	Spacy	744.56 s	0.85	0.85	0.85	0.85
SVM-KNN-RF	HPRD50	Si	patrones	232.88 s	0.76	0.76	0.76	0.76
KNN-NB-RF	IEPA	Si	Spacy	232.51 s	0.77	0.77	0.77	0.77
SVM-KNN-RF	LLL	Si	patrones	227.81 s	0.71	0.71	0.71	0.71

Tabla A.6: Mejores resultados según el corpus utilizado con la combinación de clasificadores

Apéndice B

Tablas de resultados para modelos con combinación representaciones de documentos

Clasificador	Corpus	Filtrado	Filtro	t entrenamiento	Accuracy
NB	AImed	Filtrado	spacy	13.41 s	0.74
	Biocreative3	Si	spacy	418.35 s	0.66
	Corpus Completo	No	ninguno	88.04 s	0.59
	HPRD50	Si	patrones	14.75 s	0.74
	IEPA	Si	spacy	9.88 s	0.75
	LLL	Si	patrones	7.92 s	0.65
SVM	**AImed**	**Si**	**spacy**	**17.91 s**	**0.89**
	Biocreative3	No	ninguno	7,493.10 s	0.75
	Corpus Completo	Si	spacy	462.36 s	0.85
	HPRD50	Si	patrones	18.49 s	0.76
	IEPA	Si	spacy	9.60 s	0.75
	LLL	Si	patrones	8.15 s	0.71
RF	AImed	Filtrado	spacy	14.53 s	0.87
	Biocreative3	No	ninguno	659.62 s	0.82
	Corpus Completo	Si	spacy	58.47 s	0.83
	HPRD50	Si	patrones	14.14 s	0.66
	IEPA	Si	spacy	11.79 s	0.72
	LLL	Si	patrones	14.39 s	0.73
KNN	AImed	Si	spacy	13.56 s	0.86
	Biocreative3	No	ninguno	640.04 s	0.68
	Corpus Completo	Si	spacy	59.88	0.80
	HPRD50	**Si**	**patrones**	**17.49 s**	**0.58**
	IEPA	Si	spacy	9.26 s	0.77
	LLL	Si	spacy	7.00 s	0.70

Tabla B.1: Resultados TF-IDF + Lexicon

Representación de los documentos	Algoritmo de Clasificación	Corpus	Corpus Filtrado	Filtro	Tiempo de Entrenamiento	Accuracy
WEMB LEXICON	NB	AImed	Filtrado	spacy	14.30 s	0.71
		Biocreative3	Filtrado	patrones	688.43 s	0.79
		Corpus Completo	Filtrado	spacy	55.45 s	0.81
		HPRD50	No Filtrado	ninguno	16.69 s	0.66
		IEPA	Filtrado	patrones	32.28 s	0.68
		LLL	Filtrado	spacy	4.33 s	0.70
	SVM	**AImed**	**Filtrado**	**spacy**	**37.21 s**	**0.89**
		Biocreative3	No Filtrado	ninguno	1,291.12 s	0.86
		Corpus Completo	Filtrado	patrones	122.03 s	0.85
		HPRD50	Filtrado	patrones	17.34 s	0.72
		IEPA	Filtrado	spacy	7.20 s	0.83
		LLL	Filtrado	spacy	4.51 s	0.70
	RF	AImed	Filtrado	spacy	13.82 s	0.85
		Biocreative3	No Filtrado	ninguno	647.61 s	0.84
		Corpus Completo	Filtrado	spacy	57.65 s	0.83
		HPRD50	Filtrado	patrones	12.27 s	0.73
		IEPA	Filtrado	spacy	6.92 s	0.75
		LLL	Filtrado	spacy	5.23 s	0.70
	KNN	AImed	Filtrado	patrones	130.60 s	0.85
		Biocreative3	Filtrado	patrones	642.45 s	0.69
		Corpus Completo	Filtrado	spacy	59.04 s	0.77
		HPRD50	**Filtrado**	**patrones**	**18.56 s**	**0.64**
		IEPA	Filtrado	spacy	6.92 s	0.75
		LLL	Filtrado	spacy	4.61 s	0.70

Tabla B.2: Resultados word embedding + lexicón de intensidad

Representación de los documentos	Algoritmo de Clasificación	Corpus	Corpus Filtrado	Filtro	Tiempo de Entrenamiento	Accuracy
WEMB LEXICON TF-IDF	NB	AImed	Filtrado	spacy	23.14 s	0.74
		Biocreative3	Filtrado	spacy	691.92 s	0.67
		Corpus Completo	**No Filtrado**	**ninguno**	**128.98 s**	**0.60**
		HPRD50	Filtrado	patrones	26.93 s	0.74
		IEPA	Filtrado	spacy	11.37 s	0.75
		LLL	Filtrado	patrones	17.69 s	0.65
	SVM	**AImed**	**Filtrado**	**spacy**	**23.34 s**	**0.89**
		Biocreative3	No Filtrado	ninguno	13,792.37 s	0.88
		Corpus Completo	Filtrado	patrones	438.82 s	0.86
		HPRD50	Filtrado	patrones	28.66 s	0.75
		IEPA	Filtrado	spacy	16.18 s	0.75
		LLL	Filtrado	patrones	19.04 s	0.71
	RF	AImed	Filtrado	spacy	19.92 s	0.86
		Biocreative3	No Filtrado	ninguno	1,009.87 s	0.83
		Corpus Completo	Filtrado	spacy	87.39 s	0.83
		HPRD50	Filtrado	patrones	29.63 s	0.72
		IEPA	Filtrado	spacy	11.00 s	0.77
		LLL	Filtrado	spacy	9.89 s	0.70
	KNN	AImed	Filtrado	spacy	20.32 s	0.86
		Biocreative3	No Filtrado	ninguno	995.88 s	0.68
		Corpus Completo	Filtrado	spacy	83.41 s	0.80
		HPRD50	Filtrado	patrones	21.21 s	0.63
		IEPA	Filtrado	spacy	10.12 s	0.75
		LLL	Filtrado	spacy	6.70 s	0.70

Tabla B.3: Resultados word embedding + lexicón de intensidad + TF-IDF

Representación de los documentos	Algoritmo de Clasificación	Corpus	Corpus Filtrado	Filtro	Tiempo de Entrenamiento	Accuracy
MULTIPLE TF-IDF	NB	AImed	Filtrado	spacy	175.77 s	0.67
		Biocreative3	No Filtrado	ninguno	1,264.86 s	0.77
		Corpus Completo	Filtrado	spacy	228.14 s	0.76
		HPRD50	Filtrado	patrones	159.56 s	0.72
		IEPA	Filtrado	spacy	155.82 s	0.72
		LLL	Filtrado	spacy	151.51 s	0.63
	SVM	AImed	Filtrado	spacy	183.37 s	0.88
		Biocreative3	No Filtrado	ninguno	6,276.71 s	0.85
		Corpus Completo	Filtrado	spacy	417.75 s	0.84
		HPRD50	Filtrado	patrones	166.12 s	0.75
		IEPA	Filtrado	spacy	161.17 s	0.75
		LLL	Filtrado	patrones	157.61 s	0.71
	RF	AImed	Filtrado	spacy	168.51 s	0.87
		Biocreative3	No Filtrado	ninguno	1,280.16 s	0.80
		Corpus Completo	Filtrado	spacy	233.62 s	0.81
		HPRD50	Filtrado	patrones	159.29 s	0.72
		IEPA	Filtrado	spacy	152.50 s	0.74
		LLL	Filtrado	spacy	153.46 s	0.70
	KNN	**AImed**	**Filtrado**	**spacy**	**171.84 s**	**0.88**
		Biocreative3	No Filtrado	ninguno	1,254.80 s	0.68
		Corpus Completo	Filtrado	spacy	227.11 s	0.81
		HPRD50	**Filtrado**	**patrones**	**164.96 s**	**0.60**
		IEPA	Filtrado	spacy	158.39 s	0.75
		LLL	Filtrado	spacy	148.89 s	0.70

Tabla B.4: Resultados TF-IDF + lexicón de intensidad + (TF-IDF + lexicón de intensidad))

Representación de los documentos	Algoritmo de Clasificación	Corpus	Corpus Filtrado	Filtro	Tiempo de Entrenamiento	Accuracy
MULTIPLE WEMB	NB	AImed	Filtrado	spacy	190.36 s	0.71
		Biocreative3	Filtrado	patrones	1,793.51 s	0.80
		Corpus Completo	Filtrado	spacy	249.57 s	0.81
		HPRD50	No Filtrado	ninguno	176.01 s	0.66
		IEPA	Filtrado	patrones	191.95 s	0.69
		LLL	Filtrado	spacy	164.29 s	0.70
	SVM	**AImed**	**Filtrado**	**spacy**	**197.95 s**	**0.89**
		Biocreative3	No Filtrado	ninguno	2,494.56 s	0.86
		Corpus Completo	Filtrado	patrones	315.98 s	0.84
		HPRD50	Filtrado	patrones	179.50 s	0.70
		IEPA	Filtrado	spacy	168.04 s	0.83
		LLL	Filtrado	spacy	165.80 s	0.70
	RF	AImed	Filtrado	spacy	181.92 s	0.87
		Biocreative3	No Filtrado	ninguno	1,325.41 s	0.84
		Corpus Completo	Filtrado	patrones	274.89 s	0.84
		HPRD50	Filtrado	patrones	178.75 s	0.68
		IEPA	Filtrado	spacy	172.42 s	0.77
		LLL	Filtrado	patrones	174.64 s	0.73
	KNN	AImed	Filtrado	spacy	185.33 s	0.87
		Biocreative3	No Filtrado	ninguno	1,274.04 s	0.71
		Corpus Completo	Filtrado	spacy	243.47 s	0.79
		HPRD50	**Filtrado**	**patrones**	**175.62 s**	**0.62**
		IEPA	Filtrado	spacy	165.56 s	0.72
		LLL	Filtrado	spacy	158.82 s	0.70

Tabla B.5: Resultados word embedding + lexicón de intensidad + (word embedding + lexicón de intensidad)

Apéndice C

Tablas de resultados para modelos con combinación de clasificadores

Representación de los documentos	Algoritmo de Clasificación	Corpus	Corpus Filtrado	Filtro	Tiempo de Entrenamiento	Accuracy
TF-IDF LEXICON	KNN-NB-RF	**AImed**	**Filtrado**	**spacy**	**259.88 s**	**0.86**
		Biocreative3	No Filtrado	ninguno	2,074.35 s	0.77
		Corpus Completo	Filtrado	spacy	340.67 s	0.81
		HPRD50	Filtrado	patrones	242.27 s	0.74
		IEPA	Filtrado	spacy	232.51 s	0.77
		LLL	**Filtrado**	**spacy**	**236.40 s**	**0.70**

Tabla C.1: Mejores resultados de la combinación de los clasificadores KNN-NB-RF

Representación de los documentos	Algoritmo de Clasificación	Corpus	Corpus Filtrado	Filtro	Tiempo de Entrenamiento	Accuracy
TF-IDF LEXICON	SVM-KNN-RF	**AImed**	**Filtrado**	**spacy**	**256.07 s**	**0.88**
		Biocreative3	No Filtrado	ninguno	9,045.60 s	0.76
		Corpus Completo	Filtrado	spacy	689.10 s	0.83
		HPRD50	Filtrado	patrones	232.88 s	0.76
		IEPA	Filtrado	spacy	226.42 s	0.72
		LLL	**Filtrado**	**patrones**	**227.81 s**	**0.71**

Tabla C.2: Mejores resultados de la combinación de los clasificadores SVM-KNN-RF

Representación de los documentos	Algoritmo de Clasificación	Corpus	Corpus Filtrado	Filtro	Tiempo de Entrenamiento	Accuracy
TF-IDF LEXICON	SVM-NB-KNN	**AImed**	**Filtrado**	**spacy**	**275.87 s**	**0.88**
		Biocreative3	No Filtrado	ninguno	8,787.37 s	0.73
		Corpus Completo	Filtrado	spacy	785.84 s	0.84
		HPRD50	Filtrado	patrones	241.81 s	0.72
		IEPA	Filtrado	spacy	242.97 s	0.77
		LLL	**Filtrado**	**spacy**	**227.98 s**	**0.70**

Tabla C.3: Mejores resultados de la combinación de los clasificadores SVM-NB-KNN

Representación de los documentos	Algoritmo de Clasificación	Corpus	Corpus Filtrado	Filtro	Tiempo de Entrenamiento	Accuracy
TF-IDF LEXICON	SVM-NB-RF	**AImed**	**Filtrado**	**spacy**	**348.81 s**	**0.86**
		Biocreative3	No Filtrado	ninguno	9,694.70 s	0.81
		Corpus Completo	Filtrado	spacy	744.56 s	0.85
		HPRD50	Filtrado	patrones	310.29 s	0.76
		IEPA	Filtrado	spacy	255.28 s	0.75
		LLL	**Filtrado**	**spacy**	**259.90 s**	**0.70**

Tabla C.4: Mejores resultados de la combinación de los clasificadores SVM-NB-RF

Apéndice D

Figuras de análisis de resultados obtenidos de las diferentes representaciones de documento

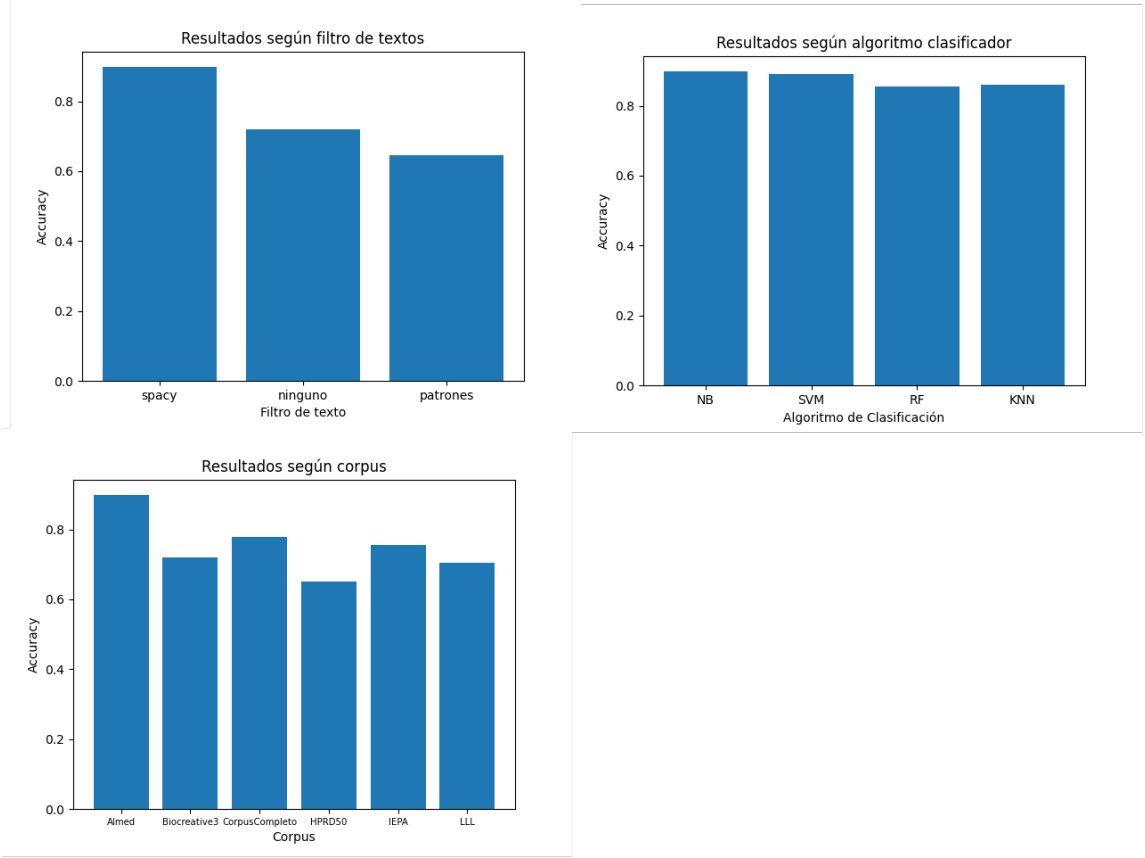


Figura D.1: Análisis resultados modelos con uso de lexicón de intensidad

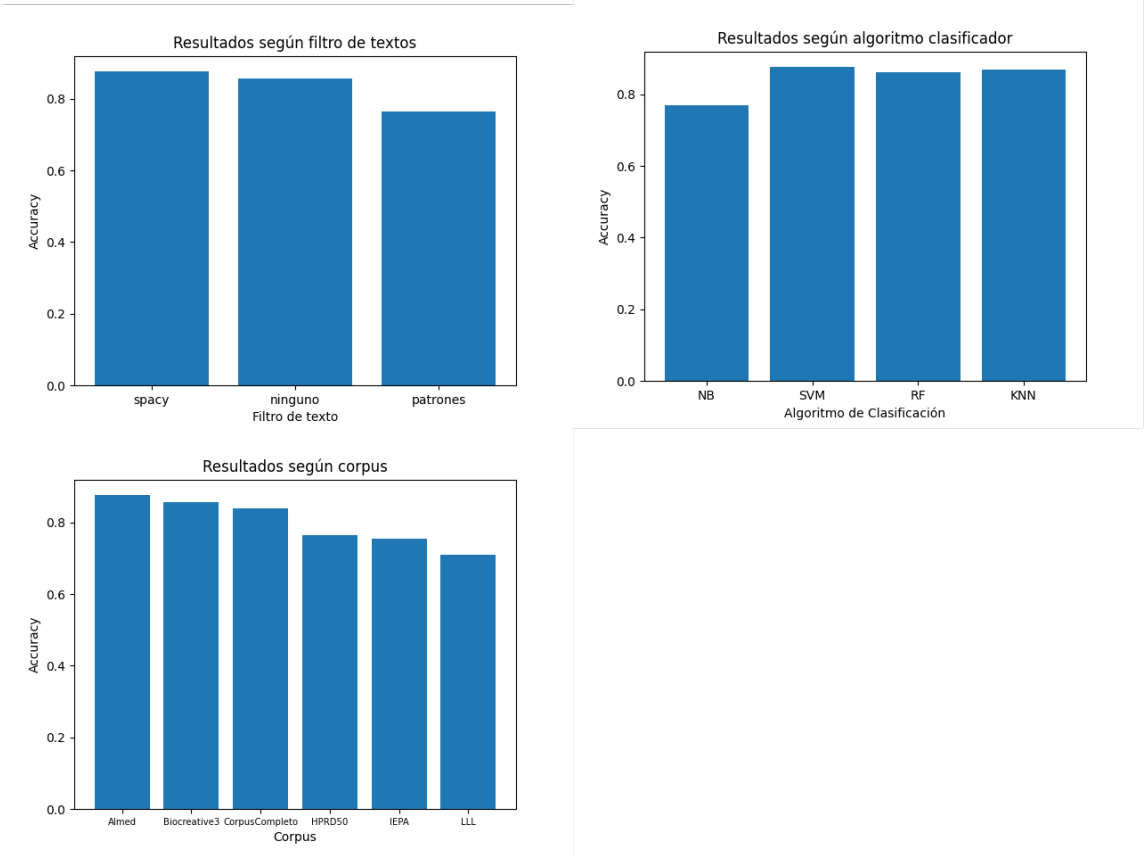


Figura D.2: Análisis resultados modelos con uso de TF-IDF

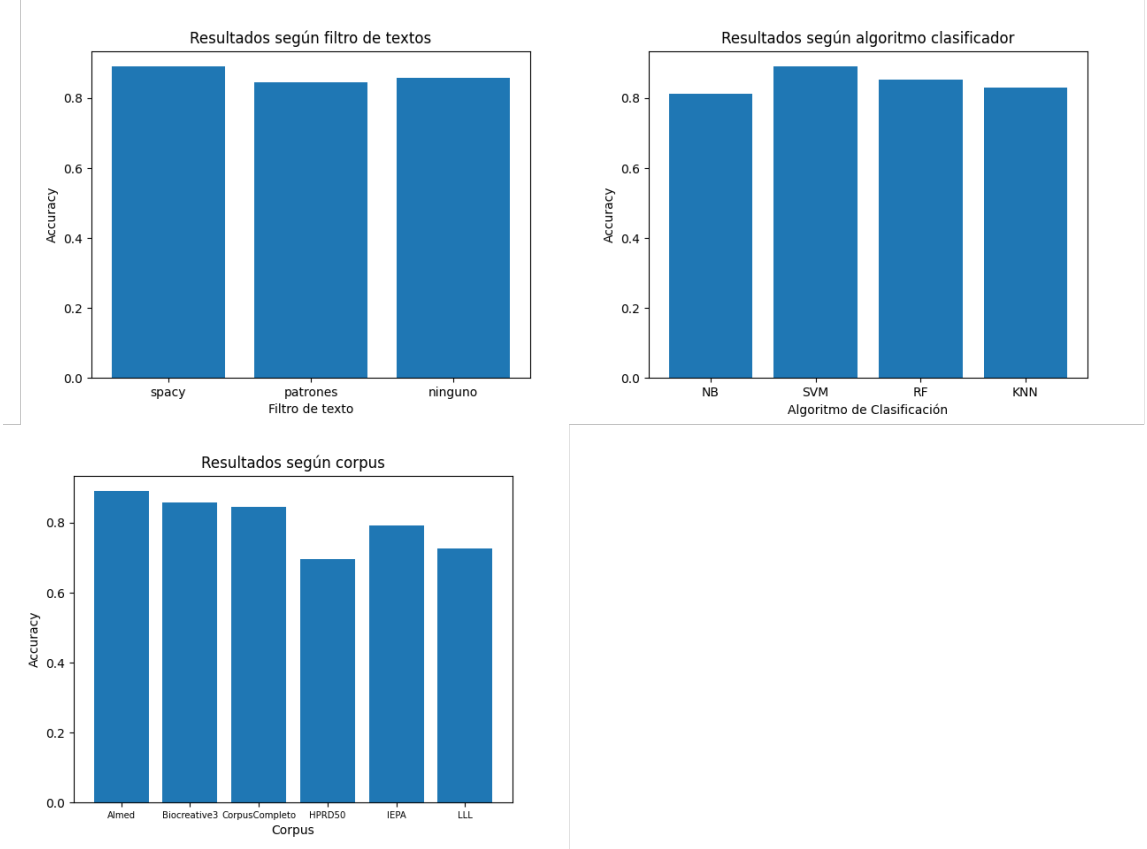


Figura D.3: Análisis resultados modelos con uso de word embedding

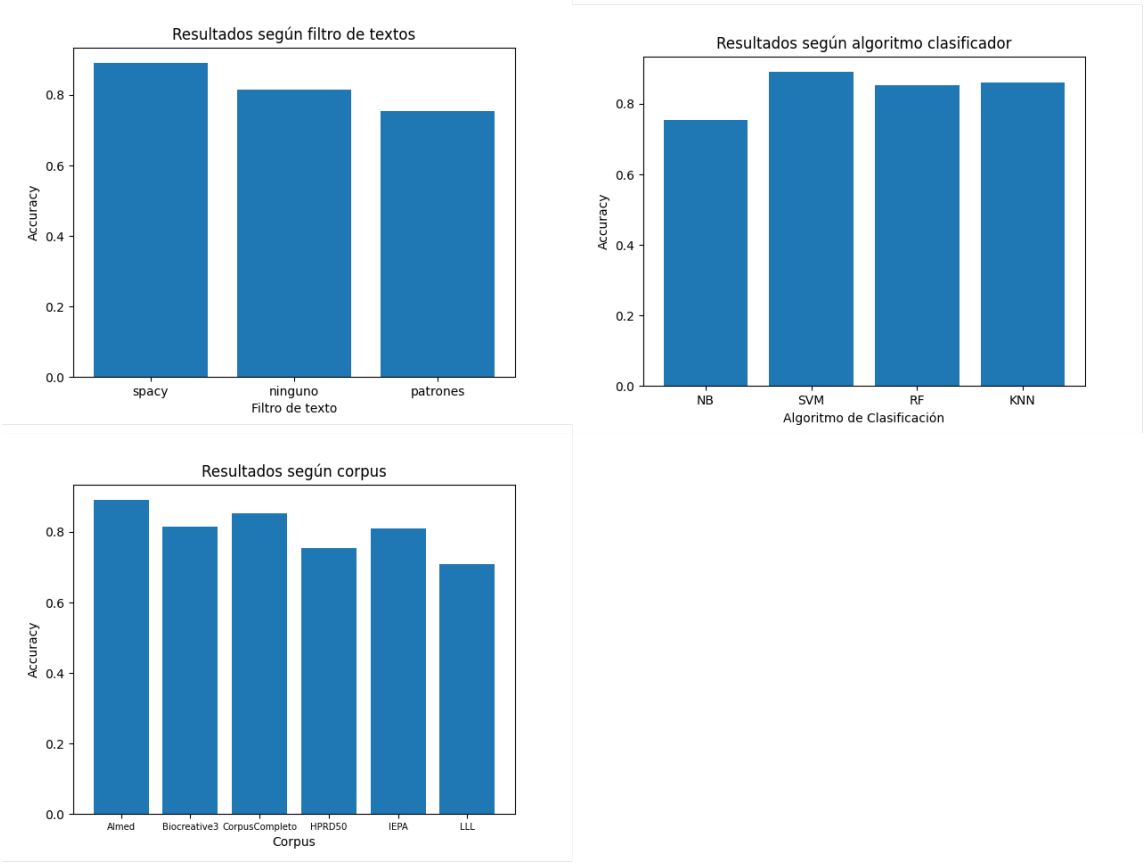


Figura D.4: Análisis resultados modelos con uso de TF-IDF con lexicón de intensidad

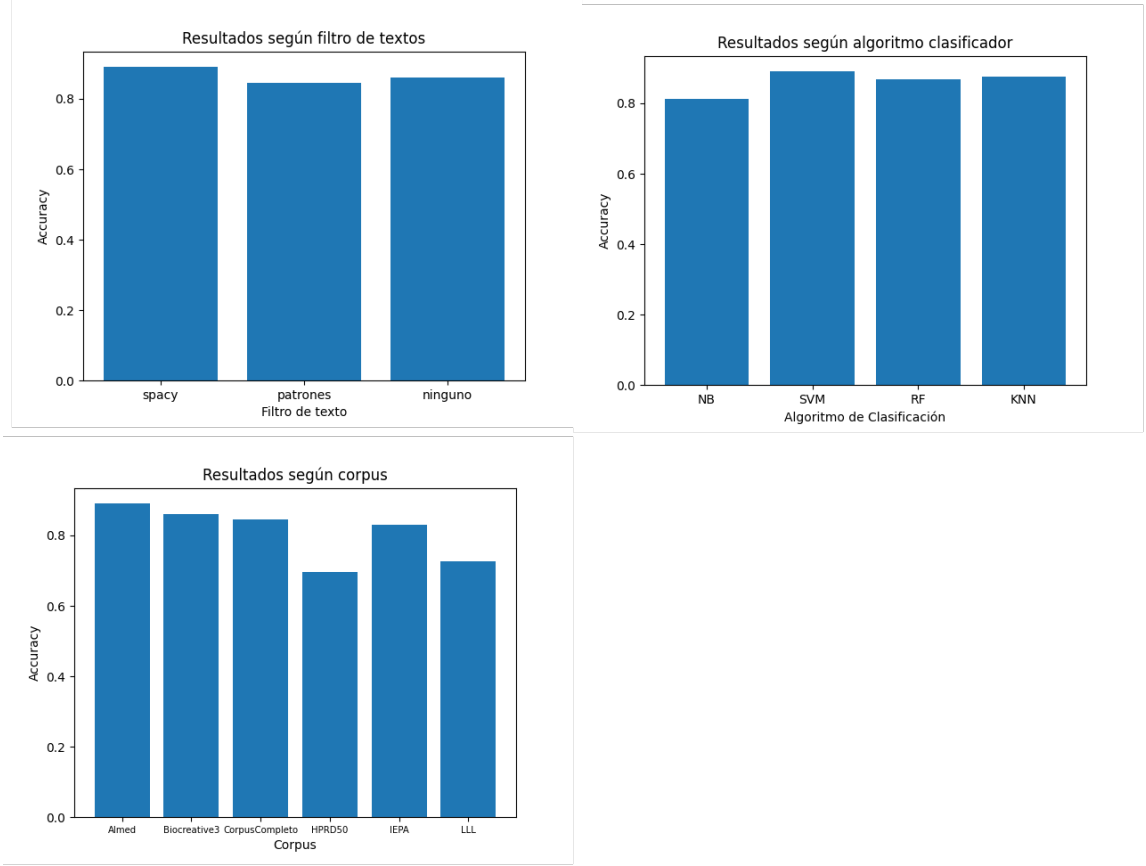


Figura D.5: Análisis resultados modelos con uso de word embedding con lexicón de intensidad

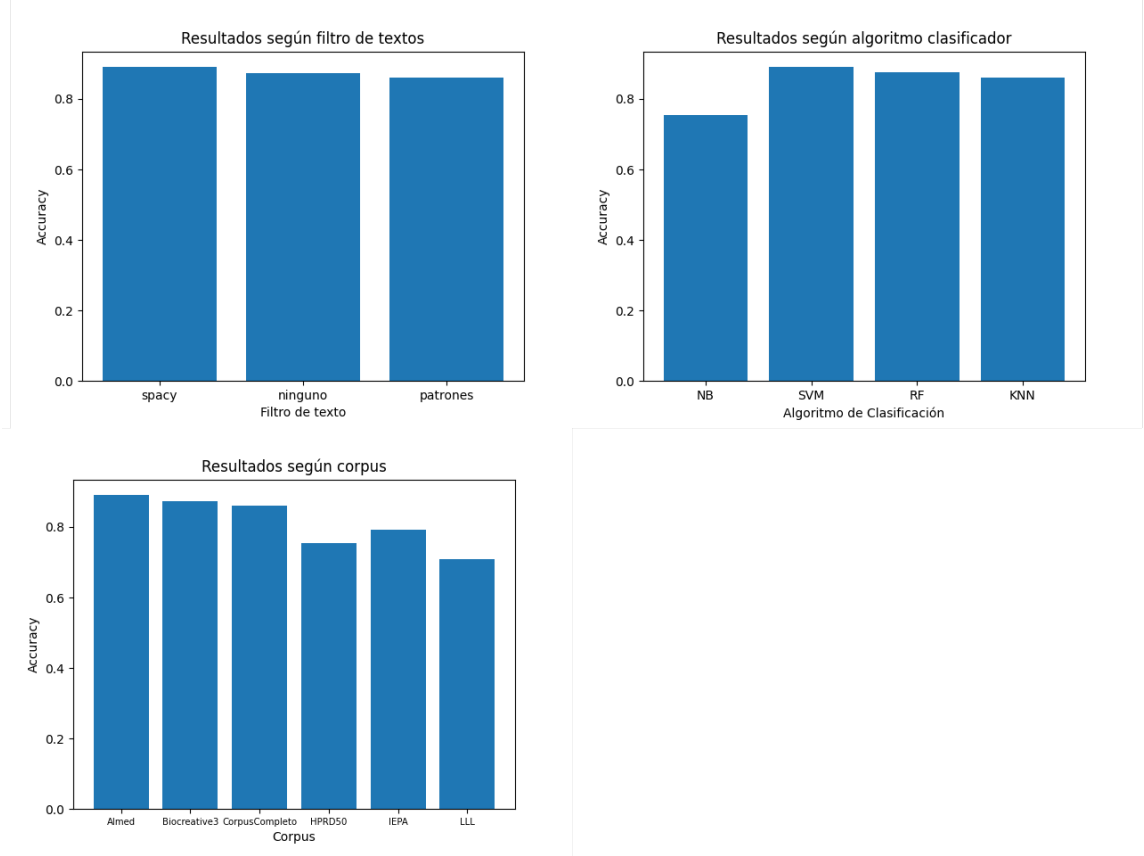


Figura D.6: Análisis resultados modelos con uso de word embedding con TF-IDF y lexicón de intensidad

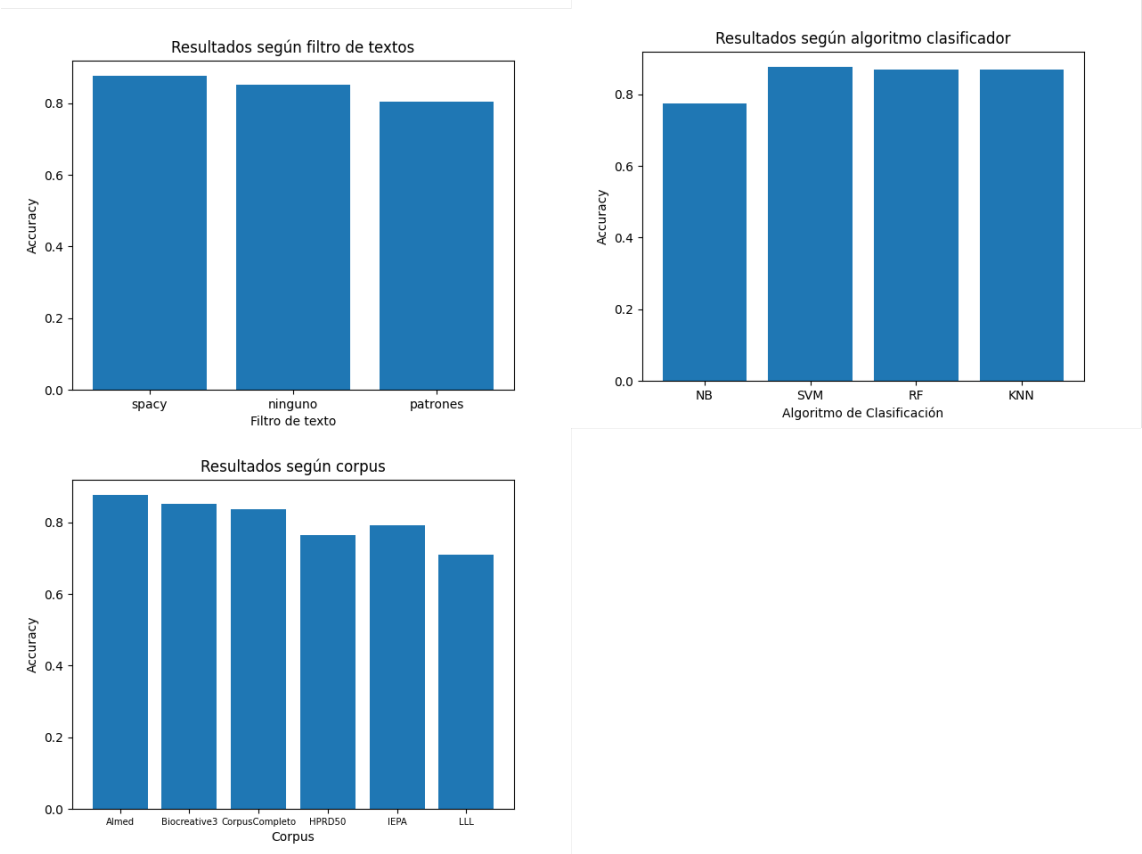


Figura D.7: Análisis resultados modelos con uso de TF-IDF, lexicón de intensidad y la combinación de TF-IDF con lexicón de intensidad

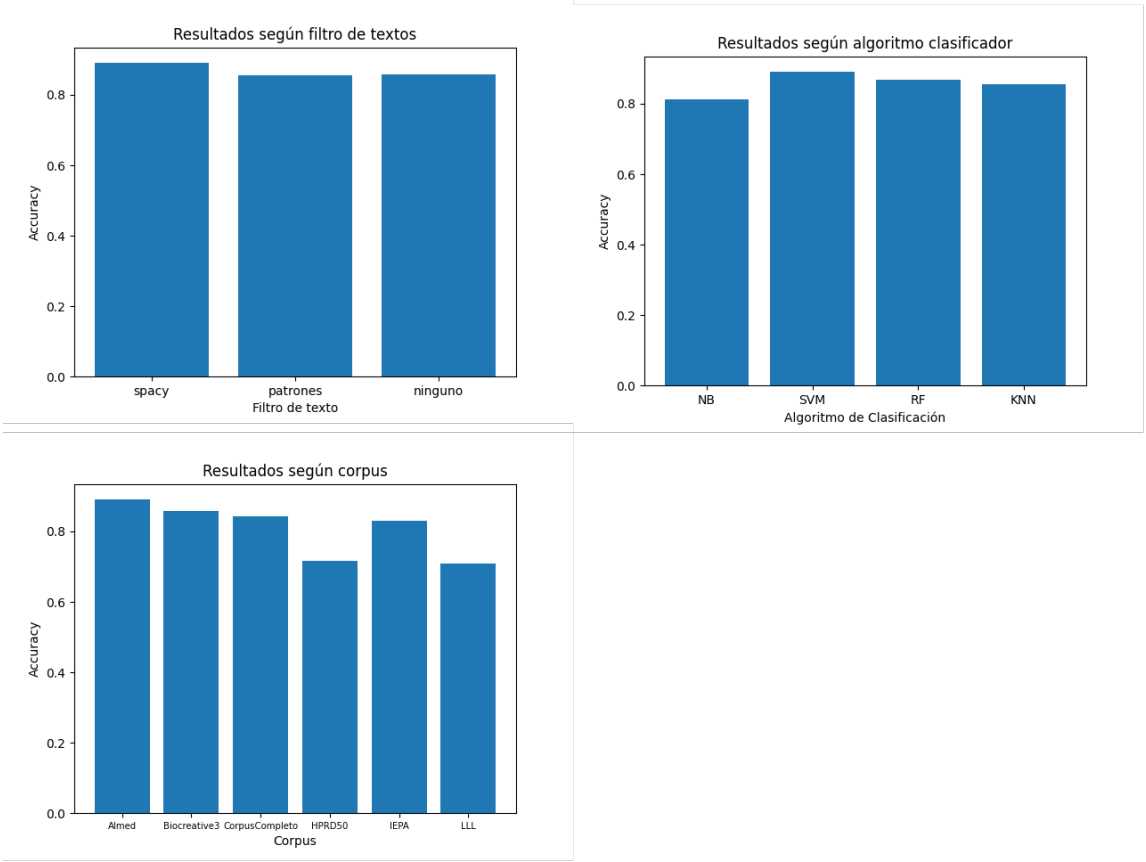


Figura D.8: Análisis resultados modelos con uso de word embedding, lexicón de intensidad y la combinación de word embedding con lexicón de intensidad

Apéndice E

Resultados docking

PDB1	PDB2	Tipo	RMSD	LogLR
6PYS	5HG8	Validado	20.58	-5.36
4WKQ	5WBU	Validado	26.24	-5.84
5FMV	4MXO	Validado	14.43	-4.66
7PQV	6GES	Validado	19.2	-5.22
4KZN	6NJS	Validado	13.16	-4.47
4GCJ	4AUA	Validado	22.44	-5.53
2L7B	6M0J	Validado	23.9	-5.66
1HE8	4TV3	Validado	36.57	-6.51
6W37	6YZ1	Validado	10.9	-4.1
3MXF	4CI3	Validado	14.14	-4.61
6TWQ	3RL7	Validado	28.7	-6.02
7N0I	7LDJ	Validado	52.37	-7.22
6GES	1WZY	Validado	18.45	-5.14
5M6U	1FLT	Validado	35.31	-6.44
3UTU	1LPG	Validado	22.9	-5.57
4ILW	1BUV	Validado	20.88	-5.39
7CEB	3WLW	Validado	52.44	-7.23
4MXO	3WLW	Validado	28.8	-6.03
6M17	6M71	Validado	40.0	-6.69
3QTW	4AUA	Validado	19.23	-5.23
7A25	7N0I	Validado	61.59	-7.55
6NIG	4G8A	Validado	45.99	-6.96
7CEB	4WKQ	Validado	12.04	-4.3
4ZQK	1I8L	Validado	16.4	-4.91
7N0I	7LX5	Validado	36.6	-6.51
1UWH	5FWK	Validado	47.18	-7.02
4KZN	4BQG	Validado	10.11	-3.95
6HKS	6MYF	Validado	26.62	-5.87
7N0I	7CAN	Validado	38.83	-6.63
1LPG	1OYT	Validado	24.09	-5.67

PDB1	PDB2	Tipo	RMSD	LogLR
6W41	7KAG	Validado	22.63	-5.55
2HE4	1I92	Validado	16.51	-4.92
5FWK	1WAO	Validado	44.04	-6.88
6MYF	3RL7	Validado	11.89	-4.27
6WEN	6W41	Validado	11.97	-4.29
7OLZ	7N0I	Validado	21.99	-5.49
1J55	1V7N	Validado	18.03	-5.1
4G6O	5WBU	Validado	36.49	-6.5
2Z3Q	1J55	Validado	10.27	-3.98
4JPS	5HG8	Validado	34.35	-6.38
7N0I	7KN7	Validado	38.22	-6.59
1V7M	1J55	Validado	20.9	-5.39
4F99	1E9H	Validado	31.35	-6.2
6W41	6W9C	Validado	37.4	-6.55
2L7B	6M0J	Validado	23.9	-5.66
4MXO	4G6O	Validado	26.02	-5.83
7N0I	7MEJ	Validado	30.02	-6.11
1UWH	5HPE	Validado	14.49	-4.66
6W37	6W4B	Validado	10.43	-4.01
4WKQ	4MXO	Validado	16.42	-4.91
7N0I	7VNB	Validado	37.58	-6.56
4MME	4JIA	Nuevo	30.61	-6.15
4DJV	4MME	Nuevo	22.05	-5.5
4MXO	5WBU	Nuevo	25.18	-5.76
4WIV	4MME	Nuevo	19.27	-5.23
2Z65	2FSE	Nuevo	16.74	-4.95
6EVQ	6SVC	Nuevo	10.96	-4.11
7B2F	6SVC	Nuevo	11.44	-4.2
4CJ2	5UFE	Nuevo	15.19	-4.76
2Z7X	3L3D	Nuevo	32.7	-6.28

Tabla E.1: Resultados docking parte 1

PDB1	PDB2	Tipo	RMSD	LogLR
1S0Q	1AVX	Nuevo	16.03	-4.86
4WKQ	4G6O	Nuevo	17.28	-5.01
6NR2	6NR3	Nuevo	59.59	-7.48
4MXO	1M4C	Nuevo	22.83	-5.57
7CEB	5WBU	Nuevo	38.19	-6.59
7CEB	4MXO	Nuevo	29.21	-6.06
1J4M	6EVQ	Nuevo	11.52	-4.21
4CJ0	5UFE	Nuevo	23.12	-5.59
1NIC	2IOF	Nuevo	27.29	-5.92
2HE4	6TWQ	Nuevo	16.55	-4.93
1DGB	6FE2	Nuevo	27.46	-5.93
4KZN	2FGF	Nuevo	12.67	-4.4
5FMV	7CEB	Nuevo	41.15	-6.74
7CYN	3A7C	Nuevo	17.78	-5.07
1R4L	3HQ2	Nuevo	38.42	-6.61
4JDV	5TE7	Nuevo	36.67	-6.51
3NGB	3J70	Nuevo	71.83	-7.86
4WKQ	2AZ5	Nuevo	20.04	-5.31
4CJ2	4CJ0	Nuevo	24.05	-5.67
1M4C	3WLW	Nuevo	30.8	-6.16
4WKQ	9ILB	Nuevo	19.94	-5.3
4G6O	2AZ5	Nuevo	27.34	-5.93
3ARP	3GE7	Nuevo	25.74	-5.81
2AGM	1AGM	Nuevo	23.77	-5.65
4JIA	3GE7	Nuevo	19.88	-5.29
3WBP	1KG0	Nuevo	16.29	-4.9
1LPG	3TWP	Nuevo	16.97	-4.98
3WBP	4G8A	Nuevo	25.03	-5.75
2Z3Q	1V7N	Nuevo	22.08	-5.5
3UTU	5C2H	Nuevo	23.08	-5.59

PDB1	PDB2	Tipo	RMSD	LogLR
1PGB	6EVQ	Nuevo	14.0	-4.6
5UFQ	6QBA	Nuevo	15.73	-4.83
3FXI	5HHP	Nuevo	17.11	-4.99
1V7M	2Z3Q	Nuevo	23.86	-5.65
4G8A	1I1Y	Nuevo	47.11	-7.01
6HKS	6QGL	Nuevo	21.6	-5.46
6QGL	6TWQ	Nuevo	40.96	-6.73
4CJ2	6QBA	Nuevo	25.2	-5.76
6QGL	6MYF	Nuevo	18.27	-5.12
4DJV	3GE7	Nuevo	15.44	-4.79
9ILB	2AZ5	Nuevo	26.3	-5.85
4DJV	4WIV	Nuevo	31.36	-6.2
7CYN	4G8A	Nuevo	50.85	-7.17
3WPF	3A7C	Nuevo	15.63	-4.81
1LPG	5C2H	Nuevo	20.68	-5.37
4AH2	3J0A	Nuevo	20.69	-5.37
5FMV	3WLW	Nuevo	50.51	-7.15
9ILB	4MXO	Nuevo	11.08	-4.13
3UTU	1Z9G	Nuevo	25.52	-5.79
7CEB	4G6O	Nuevo	21.86	-5.48
2FGF	1M48	Nuevo	21.89	-5.48
6HG4	4HSA	Nuevo	44.43	-6.9
3SC2	6GHV	Nuevo	19.79	-5.28
4G8A	3A7C	Nuevo	35.63	-6.45
6W37	6W41	Nuevo	10.5	-4.03
2Z65	1H15	Nuevo	32.58	-6.28
5FMV	5WBU	Nuevo	38.99	-6.63
1NSW	3STB	Nuevo	25.41	-5.78
5UFQ	4CJ0	Nuevo	28.1	-5.98
7B2F	7LI2	Nuevo	11.01	-4.12

Tabla E.2: Resultados docking parte 2

PDB1	PDB2	Tipo	RMSD	LogLR
6QBA	4CJ0	Nuevo	22.28	-5.52
4WIV	3GE7	Nuevo	14.31	-4.64
1I92	6MYF	Nuevo	14.92	-4.72
2Z63	3WPB	Nuevo	17.84	-5.08
2A0Z	3FXI	Nuevo	13.42	-4.51
4MME	3GE7	Nuevo	35.84	-6.47
1I85	4ZQK	Nuevo	19.0	-5.2
7CEB	2AZ5	Nuevo	16.4	-4.91
6QGL	3RL7	Nuevo	14.94	-4.72
7CUZ	1P8O	Nuevo	37.31	-6.55
1I92	6QGL	Nuevo	22.67	-5.55
1PGB	6SVC	Nuevo	10.7	-4.06
2A0Z	3L3D	Nuevo	26.37	-5.85
1M48	4BQG	Nuevo	18.79	-5.18
1NFI	6VXX	Nuevo	41.39	-6.75
6LNI	1S3Q	Nuevo	13.93	-4.59
4G6O	1M4C	Nuevo	11.23	-4.16
3TWP	1Z9G	Nuevo	33.05	-6.3
3FXI	3RKI	Nuevo	20.48	-5.35
3ARP	4WIV	Nuevo	22.74	-5.56
3WLW	2AZ5	Nuevo	10.69	-4.06
5FMV	2AZ5	Nuevo	36.21	-6.49
3ARP	4MME	Nuevo	42.35	-6.8
5UFE	5ZAU	Nuevo	19.95	-5.3
4MXO	2AZ5	Nuevo	24.41	-5.7
1S3Q	6NHT	Nuevo	52.32	-7.22
1M4C	5WBU	Nuevo	12.05	-4.3
2IOF	3FMX	Nuevo	25.56	-5.79
1OYT	5C2H	Nuevo	24.93	-5.74
6CYK	4HBT	Nuevo	18.34	-5.13

PDB1	PDB2	Tipo	RMSD	LogLR
1FSD	6EVQ	Nuevo	16.31	-4.9
3UTU	3TWP	Nuevo	15.45	-4.79
4WIV	4JIA	Nuevo	12.63	-4.39
3SGB	1AK4	Nuevo	26.67	-5.88
3WBP	1I1Y	Nuevo	19.3	-5.23
6QUQ	3VX8	Nuevo	33.35	-6.32
2Z7X	5HHP	Nuevo	33.72	-6.34
2HE4	6MYF	Nuevo	16.38	-4.91
4MBS	1HA7	Nuevo	10.69	-4.06
3TWP	5C2H	Nuevo	35.59	-6.45
6R2X	6EVQ	Nuevo	10.32	-3.99
1KG0	1I1Y	Nuevo	34.46	-6.39
5FMV	9ILB	Nuevo	14.74	-4.7
4KZN	1M48	Nuevo	20.07	-5.31
4G8A	3WPF	Nuevo	24.67	-5.72
2A0Z	2Z7X	Nuevo	34.41	-6.39
1NIC	3FMX	Nuevo	18.13	-5.11
1RYJ	1Z9G	Nuevo	10.02	-3.94
5WBU	2AZ5	Nuevo	31.85	-6.23
6QBA	5UFE	Nuevo	23.69	-5.64
9ILB	3WLW	Nuevo	20.36	-5.34
2LAM	2NAO	Nuevo	18.39	-5.14
7CYN	3WPF	Nuevo	28.94	-6.04
7B2F	6EVQ	Nuevo	15.63	-4.81
6LNI	6NHT	Nuevo	54.78	-7.31
6NM3	6EVQ	Nuevo	10.1	-3.95
1LPG	1RYJ	Nuevo	19.27	-5.23
1P8O	1HZV	Nuevo	19.6	-5.26
7X9S	7X9R	Nuevo	29.17	-6.06
4HBT	1YLT	Nuevo	16.46	-4.92

Tabla E.3: Resultados docking parte 3

Apéndice F

Revisión bibliográfica

Título	Año	Resumen
DisProt: intrinsic protein disorder annotation in 2020	2020	DisProt es una base de datos de proteínas intrínsecamente desordenadas que proporciona información completa sobre el desorden de las proteínas, incluyendo las interacciones proteína-proteína (PPIs). Los investigadores utilizaron minería de texto para identificar fragmentos de secuencia proteica relacionados con las PPIs. Sus hallazgos revelaron información clave sobre la naturaleza de las PPIs que podría ayudar en el desarrollo de nuevas aproximaciones terapéuticas. En general, el estudio destaca la utilidad de la minería de texto para facilitar la búsqueda de PPIs dentro del contexto de DisProt.
Text mining approach to explore dimensions of airline customer satisfaction using online customer reviews	2020	Esta publicación no cubre la minería de texto para la búsqueda de interacciones proteína-proteína. Se centra en el uso de la minería de texto para explorar la satisfacción del cliente de aerolíneas a través de reseñas en línea.
BioBERT: a pre-trained biomedical language representation model for biomedical text mining	2020	BioBERT es un nuevo modelo de lenguaje diseñado para ayudar a encontrar interacciones proteína-proteína (PPI) en texto biomédico. Está basado en BERT, un popular modelo para el procesamiento del lenguaje natural, pero específicamente entrenado en literatura biomédica. BioBERT es capaz de identificar palabras y frases relevantes relacionadas con las PPI y logra mejores resultados que modelos anteriores. Tiene el potencial de mejorar en gran medida nuestra comprensión de los sistemas biológicos y ayudar en el descubrimiento de medicamentos.
Collabonet: collaboration of deep neural networks for biomedical named entity recognition	2019	La publicación Collabonet propone un nuevo enfoque para encontrar interacciones proteína-proteína a través de redes neuronales profundas. El método implica coordinar múltiples algoritmos de procesamiento del lenguaje natural para analizar datos textuales e identificar entidades relevantes. Los autores demuestran la efectividad de su enfoque en pruebas de referencia, logrando un mejor rendimiento que los métodos existentes. Este método tiene un gran potencial para avanzar en nuestra comprensión de los datos biomédicos y los mecanismos subyacentes de las interacciones proteína-proteína.
The BioGRID interaction database: 2019 update	2019	La base de datos de interacciones BioGRID ha sido actualizada para el 2019, proporcionando información completa sobre interacciones proteína-proteína que pueden ser analizadas mediante técnicas de minería de texto. La base de datos incluye más de 2.5 millones de interacciones, con anotaciones y datos curados expandidos sobre complejos proteínicos, interacciones genéticas y modificaciones post-traduccionales. La base de datos también incluye nuevos tipos de interacción y herramientas mejoradas de búsqueda y visualización. Estas actualizaciones facilitan la identificación de interacciones proteína-proteína previamente no descubiertas y proporcionan un recurso valioso para avanzar en nuestra comprensión de los sistemas biológicos.
STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets	2019	STRING v11 es un recurso valioso para el análisis de interacciones proteína-proteína (PPI) que proporciona redes PPI completas con mayor cobertura. Soporta el descubrimiento funcional en conjuntos de datos experimentales de todo el genoma, haciendo uso de técnicas de minería de texto para extraer información de PPI de la literatura científica. La plataforma también integra diversas fuentes de información, incluyendo experimentos de alto rendimiento, análisis de coexpresión y bases de datos públicas. En general, STRING v11 es una herramienta esencial para los investigadores que buscan entender las relaciones funcionales entre las proteínas.

Tabla F.1: Resumen revisión bibliográfica parte 1

Título	Año	Resumen
UniProt: a worldwide hub of protein knowledge	2019	UniProt es una plataforma completa para el análisis de datos de proteínas que incluye minería de textos para la identificación de interacciones proteína-proteína. La base de datos integra datos de múltiples fuentes, incluyendo la literatura y datos experimentales, para permitir a los investigadores entender mejor la función y las interacciones de las proteínas. Los algoritmos de minería de textos utilizados por UniProt están en constante evolución para mejorar la precisión y la integridad de los datos.
iPPI-PseAAC (CGR): Identify protein-protein interactions by incorporating chaos game representation into PseAAC	2019	El método iPPI-PseAAC (CGR) combina dos técnicas, la representación de juego caótico (CGR) y la composición de pseudo aminoácidos (PseAAC), para identificar interacciones proteína-proteína. CGR genera una representación visual de la secuencia de proteínas que ayuda en la identificación de los sitios de interacción. PseAAC utiliza información de las secuencias de proteínas para predecir los sitios de interacción. El método iPPI-PseAAC (CGR) funcionó bien en la identificación de interacciones proteína-proteína en conjuntos de datos de referencia.
mCSM-PPI2: predicting the effects of mutations on protein–protein interactions	2019	mCSM-PPI2 es una herramienta que predice los efectos de las mutaciones en las interacciones proteína-proteína utilizando aprendizaje automático y métodos computacionales. Utiliza minería de texto para extraer información clave de la literatura biomédica y las bases de datos de proteínas para aumentar la precisión. La herramienta puede ayudar a los investigadores a comprender cómo ciertas mutaciones pueden afectar sistemas biológicos complejos y ayudar en el desarrollo de fármacos.
Using text mining techniques for extracting information from research articles	2018	Esta publicación discute la aplicación de técnicas de minería de texto para extraer información de interacción proteína-proteína de artículos de investigación. El enfoque implica el uso de herramientas de procesamiento de lenguaje natural que pueden identificar términos relevantes y relaciones entre ellos. Los autores también describen los desafíos y limitaciones de este método, incluyendo la necesidad de datos de entrenamiento de alta calidad y el potencial de falsos positivos/negativos. En general, la minería de texto puede ser una herramienta valiosa para los biólogos que buscan extraer y analizar grandes volúmenes de datos relacionados con proteínas de la literatura.
A comprehensive and quantitative comparison of text-mining in 15 million full-text articles versus their corresponding abstracts	2018	Este estudio comparó la eficacia de la minería de texto en artículos completos frente a sus resúmenes para identificar interacciones proteína-proteína. Los investigadores analizaron 15 millones de artículos y descubrieron que la precisión de la minería de texto se mejoró significativamente al utilizar los artículos completos. Sin embargo, los resúmenes aún proporcionaban información valiosa. El estudio destaca la importancia de considerar el texto completo al realizar la minería de texto para la investigación biomédica.
TRRUST v2: an expanded reference database of human and mouse transcriptional regulatory interactions	2018	TRRUST v2 es una base de datos actualizada para la búsqueda de interacciones proteína-proteína que proporciona información sobre interacciones regulatorias de la transcripción en humanos y ratones. Su objetivo es integrar diversas fuentes de información para mejorar la precisión y la exhaustividad de los datos. Se utilizaron técnicas de minería de texto para extraer interacciones regulatorias de la literatura biomédica. La base de datos actualmente contiene más de 800,000 interacciones entre 8,600 factores de transcripción y 16,200 genes objetivos en humanos y ratones.

Tabla F.2: Resumen revisión bibliográfica parte 2

Título	Año	Resumen
Cytoscape StringApp: network analysis and visualization of proteomics data	2018	La aplicación Cytoscape StringApp es una herramienta utilizada para el análisis y visualización de redes de datos proteómicos. Ayuda en la búsqueda de interacciones proteína-proteína, que son cruciales para entender los procesos celulares y las enfermedades. Con el uso de técnicas de minería de texto, ayuda a identificar interacciones confiables entre proteínas, proporcionando una vista más precisa y completa de los datos de la red. La aplicación también se integra con la base de datos STRING, lo que permite acceder a vastos datos de interacción proteica de fuentes públicas.
... cross-linking/mass spectrometry workflow based on MS-cleavable cross-linkers and the MeroX software for studying protein structures and protein–protein interactions	2018	Esta publicación describe un flujo de trabajo para estudiar estructuras de proteínas e interacciones proteína-proteína utilizando cross-linkers cleavables por MS y el software MeroX. El enfoque se basa en la técnica de cross-linking/espectrometría de masas y puede ayudar en la búsqueda y descubrimiento de PPIs.
SFPEL-LPI: sequence-based feature projection ensemble learning for predicting LncRNA-protein interactions	2018	SFPEL-LPI es un enfoque de aprendizaje automático para predecir interacciones entre ARN no codificante largo (lncRNA) y proteínas. El método utiliza características basadas en secuencia para crear un conjunto de proyecciones, que se entrena en interacciones lncRNA-proteína conocidas. El modelo entrenado luego se usa para predecir interacciones novedosas. Este enfoque podría ser útil para identificar posibles socios proteicos de lncRNA y obtener información sobre su papel en los procesos celulares.
A systematic survey of centrality measures for protein-protein interaction networks	2018	Esta publicación revisa diversas medidas de centralidad utilizadas en la minería de texto para interacciones proteína-proteína. El estudio encontró que la centralidad del grado, la centralidad de intermediación y la centralidad de cercanía son métricas comúnmente utilizadas. Sin embargo, las limitaciones incluyen la incapacidad para capturar la complejidad de las interacciones y el hecho de que las medidas de centralidad están fuertemente influenciadas por la topología de la red. Se necesitan más investigaciones para mejorar la precisión de la minería de texto para interacciones proteína-proteína.
Biomedical text mining for research rigor and integrity: tasks, challenges, directions	2018	Este papel discute la importancia de la minería de texto en la investigación biomédica, especialmente en la identificación de interacciones proteína-proteína. Los autores delimitan los retos comunes en la minería de texto, como la desambiguación y la detección de negación, y proponen nuevas direcciones para futuras investigaciones, incluyendo el uso de técnicas de aprendizaje automático y procesamiento del lenguaje natural. En general, la minería de texto puede ayudar a mejorar la rigurosidad e integridad de la investigación.
Textpresso Central: a customizable platform for searching, text mining, viewing, and curating biomedical literature	2018	Textpresso Central es una plataforma de minería de texto y búsqueda de literatura biomédica. Proporciona herramientas de búsqueda personalizadas para identificar interacciones proteína-proteína, curar y almacenar información relacionada, y visualizar los resultados de la búsqueda. Esta herramienta ofrece capacidades de minería de texto eficientes y precisas para descubrir las vastas cantidades de información disponibles en la literatura publicada.

Tabla F.3: Resumen revisión bibliográfica parte 3

Título	Año	Resumen
Deep learning for extracting protein-protein interactions from biomedical literature	2017	Este estudio utiliza técnicas de aprendizaje profundo para identificar interacciones proteína-proteína a partir de la literatura biomédica. Demuestra que el enfoque es efectivo y supera a los métodos tradicionales de minería de texto. El estudio destaca la importancia de identificar con precisión las interacciones proteína-proteína para comprender los procesos biológicos y desarrollar tratamientos eficaces para las enfermedades.
Information retrieval and text mining technologies for chemistry	2017	Esta publicación explora el uso de tecnologías de minería de texto y recuperación de información para identificar interacciones proteína-proteína en química. Estas técnicas implican analizar grandes volúmenes de datos textuales y extraer información relevante para facilitar la identificación precisa de dichas interacciones. Los principales desafíos en estos enfoques incluyen la complejidad de las propias interacciones, así como problemas relacionados con la calidad de los datos y la ambigüedad semántica. Sin embargo, los avances recientes en procesamiento de lenguaje natural y técnicas de aprendizaje automático están mostrando resultados prometedores para superar estos desafíos y acelerar el descubrimiento de nuevas interacciones proteína-proteína.
Network pharmacology-based study on the mechanism of action for herbal medicines in Alzheimer treatment	2017	Este estudio sobre el tratamiento del Alzheimer utilizó la farmacología de redes para analizar el mecanismo de acción de las hierbas medicinales. Se utilizó minería de texto para buscar interacciones proteína-proteína. Los resultados sugieren que ciertas hierbas medicinales pueden ser efectivas en el tratamiento del Alzheimer a través de su influencia en proteínas clave en el cerebro.
RAIN: RNA–protein association and interaction networks	2017	El método RAIN utiliza minería de textos para identificar interacciones proteína-proteína en la investigación biomédica. Se enfoca en las interacciones de ARN como un medio para predecir las redes de interacción de proteínas. El estudio encontró que RAIN fue efectivo para identificar muchas interacciones proteína-proteína conocidas y tenía potencial para identificar nuevas interacciones. Los autores sugieren que RAIN podría ser una herramienta útil para identificar posibles objetivos terapéuticos y elucidar las vías de enfermedades.
A review of recent advancement in integrating omics data with literature mining towards biomedical discoveries	2017	La publicación trata sobre la integración de datos ómicos y minería de literatura para descubrimientos biomédicos. La minería de texto puede utilizarse para identificar interacciones proteína-proteína y predecir funciones proteínicas. Sin embargo, existen desafíos que incluyen la necesidad de estandarización y mejor precisión. Globalmente, los avances en la integración de datos ómicos con la minería de literatura muestran un gran potencial para avanzar en la investigación biomédica.
The BioGRID interaction database: 2017 update	2017	La base de datos de interacción BioGRID ha sido actualizada en 2017 para incluir nuevas fuentes de datos de interacción proteína-proteína. La minería de texto se utiliza para identificar estas interacciones y asignarles puntuaciones de confianza. La base de datos también incluye información curada y anotaciones para cada interacción. Sirve como un recurso valioso para estudiar las interacciones proteína-proteína en varios sistemas biológicos.

Tabla F.4: Resumen revisión bibliográfica parte 4

Título	Año	Resumen
A brief survey of text mining: Classification, clustering and extraction techniques	2017	Esta publicación proporciona una breve descripción de las técnicas de minería de texto, como la clasificación, agrupación y extracción, y cómo se pueden aplicar en el contexto de la búsqueda de interacciones proteína-proteína. La minería de texto puede ayudar a automatizar la extracción de información relevante de la literatura científica y, por lo tanto, ayudar en el descubrimiento de nuevas relaciones de interacción proteína-proteína.
DeepPPI: boosting prediction of protein–protein interactions with deep neural networks	2017	DeepPPI es un sistema de red neuronal profunda creado para mejorar la predicción de interacciones proteína-proteína a través de la minería de texto. El sistema integra diversas técnicas bioinformáticas para mejorar la precisión y velocidad de sus predicciones. DeepPPI puede extraer eficientemente conocimiento de grandes volúmenes de literatura existente, convirtiéndose en una herramienta esencial para el desarrollo de investigaciones relacionadas con proteínas y descubrimiento de fármacos. Se encontró que su rendimiento es superior a los métodos existentes en términos de precisión, sensibilidad y puntuación F1.
Protein–protein interactions in virus–host systems	2017	La minería de texto puede buscar eficientemente interacciones proteína-proteína (PPIs) identificando patrones de relación entre entidades biológicas. Esta publicación explora cómo las PPIs en sistemas virus-huésped pueden ser analizadas a través de técnicas de minería de texto que utilizan procesamiento del lenguaje natural, aprendizaje automático y análisis de redes. Los autores destacan la relevancia de la minería de texto para identificar nuevas PPIs, predecir posibles interacciones y en última instancia mejorar la comprensión de los mecanismos de interacción virus-huésped.
Prediction of lncRNA-protein interactions using HeteSim scores based on heterogeneous networks	2017	El artículo describe cómo se utilizó la minería de texto y el algoritmo HeteSim para predecir con precisión las interacciones proteína-proteína. Se crearon redes heterogéneas para revelar las relaciones entre las moléculas de ARN no codificante (lncRNA) y las proteínas, lo que permitió la identificación de posibles nuevas interacciones. Este enfoque podría ser útil en el descubrimiento de medicamentos y la medicina personalizada.
Protein interactions during the flavivirus and hepacivirus life cycle	2017	La publicación se centra en el uso de enfoques de minería de texto para identificar las interacciones proteína-proteína durante el ciclo de vida de los flavivirus y los hepacivirus. Los autores discuten los desafíos para extraer información precisa de los datos de texto y proporcionan una revisión detallada de varias técnicas de minería de texto que se han utilizado para identificar las interacciones proteína-proteína. También brindan información sobre la identificación de posibles objetivos farmacológicos a través del análisis de minería de textos.

Tabla F.5: Resumen revisión bibliográfica parte 5