Aplicaciones en procesamiento de lenguaje natural y teoría de grafos

Mireya Tovar Vidal
Guillermo De Ita Luna
Pedro Bello López

Editores

Dirección General de Publicaciones

Aplicaciones en procesamiento de lenguaje natural y teoría de grafos

Aplicaciones en procesamiento de lenguaje natural y teoría de grafos

Mireya Tovar Vidal
Guillermo De Ita Luna
Pedro Bello López
Coordinadores

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
2023

Benemérita Universidad Autónoma de Puebla
Rectora: Ma. Lilia Cedillo Ramírez
Secretario General: José Manuel Alonso Orozco
Vicerrector de Extensión y Difusión de la Cultura: José Carlos Bernal Suárez
Director General de Publicaciones: Luis Antonio Lucio Venegas

Primera edición **2023**
ISBN: 978-607-8957-18-7

Diseño de portada: Mireya Tovar Vidal

Hecho en México
Made in Mexico

Prólogo

En el presente libro titulado *"Aplicaciones en procesamiento de lenguaje natural y teoría de grafos"* se presentan diferentes áreas de investigación de procesamiento del lenguaje natural y propuestas que utilizan teoría de grafos para su solución que son abordadas por distintos grupos de investigación al interior del país.

La obra incluye diez capítulos de investigación divididos en áreas de procesamiento de lenguaje natural, en aplicaciones de teoría de grafos, en aprendizaje automático, en ciencia de datos, entre otras.

Los capítulos que forman parte de esta obra fueron revisados mediante el sistema de doble par ciego y aprobados para su publicación por expertos en el área de conocimiento, lo que permitió asegurar la calidad científica en las áreas de estudio así como la obra en su totalidad. A continuación se menciona la aportación de cada uno de ellos.

En el Capítulo 1 se aborda el problema del reconocimiento de firmas falsas utilizando un algoritmo de aprendizaje profundo; implementan una arquitectura de red neuronal convolucional siamesa y obtienen un mejor comportamiento que una red neuronal convolucional única.

En el Capítulo 2 se presenta un enfoque basado en el algoritmo SVM que utiliza características basadas en coeficientes cepstrales de frecuencia de Mel para reconocer emociones del habla en distintos corporas.

En el Capítulo 3 se describe un método de extracción de entidades utilizando características morfológicas y semánticas en un corpus de textos clínicos en español. En los experimentos las máquinas de soporte vectorial dieron mejores resultados.

En el Capítulo 4 se presenta un enfoque de descubrimiento de tópicos para la detección de la depresión en textos cortos, utilizando una red neuronal convolucional.

En el Capítulo 5 se comparan algoritmos de aprendizaje automático y técnicas de Procesamiento de Lenguaje Natural (PLN) para el análisis de sentimientos de textos en español. En los experimentos, la mejor exactitud en la tarea de clasificación fue usando *random forest*.

En el Capítulo 6 se describe un método que procesa información de un archivo de texto y genera una nube de palabras para determinar que palabras son conectores, preposiciones y cuales aportan un contenido altamente relevante al documento más comunes, se usan librerías de procesamiento de lenguaje natural.

En el Capítulo 7 se presentan modelos de aprendizaje automático para reconocer y categorizar los niveles de depresión de una población estudiantil.

En el Capítulo 8 se hace una comparativa de la eficiencia y precisión de dos heurísticas que se aplican en la solución del coloreo de grafos. Los autores desarrollan la técnica de poda alfa-beta para refinar una propuesta minimax de solución al coloreo de grafos y muestran las bondades de esta técnica por sobre un

método evolutivo desarrollado por los mismos autores. En el Capítulo 9 se aplica la técnica de poda alfa-beta para construir un proceso minimax que construye una solución al juego conecta 4 (Clingo). El problema Clingo se modela como un sistema con restricciones y se aplica programación lógica basada en ASP para la construcción automática de soluciones que satisfacen las restricciones del juego.

Por último, en el Capítulo 10 se presentan diversos resultados del análisis de ancho hiper-arbóreo sobre hipergrafos. En particular, se analizan instancias de hipergrafos provenientes de la clase 2μ-3MON con el fin de determinar si algún ancho hiper-arbóreo garantiza la tratabilidad de la clase de los hipergrafos considerada.

Finalmente, queremos agradecer a cada uno de los autores por su aportación, a nuestros revisores por su valiosa labor y cuidado en el proceso de revisión, a la Facultad de Ciencias de la Computación de la Benemérita Universidad Autónoma de Puebla y a todos aquellos cuya participación contribuyó para la publicación de este libro.

Los editores,
Mireya Tovar Vidal
Guillermo De Ita Luna
Pedro Bello López

Índice general

Capítulo 1

Implementación de un modelo de aprendizaje profundo para el reconocimiento de firmas falsas

Octavio Mendoza Gómez[1], Luis Carlos Altamirano Robles[1]

[1] Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

octaviomen12@gmail.com, luis.altamirano@correo.buap.com

Resumen. El reconocimiento de firmas falsas es un problema cuya solución se puede ver apoyada por la implementación de un algoritmo de aprendizaje profundo. En este trabajo se propone la implementación de una arquitectura de redes neuronales convolucionales siamesas que esté enfocada a la categorización y reconocimiento de dichas firmas. El único obstáculo es el conjunto de imágenes para el entrenamiento del modelo de red neuronal, debido a la dificultad de recolectar y capturar una gran cantidad de imágenes se optó por usar el conjunto de datos BHSig260, en específico las firmas hindús y se seguirá un proceso donde se usará el 70% de las imágenes para el entrenamiento y el 30% restante para la prueba y medidas de precisión del modelo de aprendizaje profundo obtenido del entrenamiento de la red neuronal convolucionales siamesas.

Palabras Clave: Firmas manuscritas, reconocimiento de firmas, redes neuronales convolucionales, redes convolucionales siamesas.

1 Introducción

En toda la historia humana se ha presentado la necesidad de crear y transmitir información de distintos temas de las distintas áreas de estudio, esta razón llevo a la creación de documentos que porten dicha información y para poder darles la credibilidad debida se decidió implementar un sistema donde dicho documento debe estar relacionado con su autor o autores mediante reconocimiento biométrico, que permite una identificación de individuos mediante sus características biológicas, psicológicas y culturales. Las firmas hechas a mano son parte del reconocimiento biométrico y es un método de reconocimiento que se implementado por mucho tiempo hasta la fecha y en la actualidad se usan dos distintos tipos de captura para firmas, digital o no digital. Sin embargo, como cualquier otro, las firmas escritas a mano presentan problemáticas, su reconocimiento y verificación.

En la actualidad se han propuesto un conjunto de herramientas para la aplicación de inteligencia artificial en orden de solucionar este tipo problemas, es aquí donde se presentan los algoritmos de aprendizaje profundo, pues han mostrado niveles de adaptabilidad y precisión altos mientras que su costo computacional es bajo (Swapnil, 2022). Dentro el punto de vista de este tipo de algoritmos la verificación y la clasificación son problemas de clasificación múltiple y clasificación binaria respectivamente.

Uno de los tantos tipos de algoritmos de aprendizaje que se aplican son aquellos clasificados como aprendizaje profundo donde encontramos la familia de las redes neuronales (NNets), el cual es un término para referirnos a un conjunto de algoritmos que nos permiten la creación de un modelo de aprendizaje mediante el procesamiento de datos en cual se hace uso de capas, refiriéndonos a un conjunto de funciones distintas o similares entre sí, de tal manera que la información extraída de un conjunto de datos sea más precisa y no exista necesidad alguna de una interferencia o señalamiento humano para el entrenamiento del algoritmo implementado (Dewangan, 2023).

Entre esta familia de algoritmos podemos encontrar dos tipos de redes neuronales sobre las cuales enfocaremos este trabajo, las redes neuronales convolucionales (Saxena, 2022) a las cuales se les ha dado la gran tarea de procesar imágenes de tal manera que el modelo de aprendizaje proveniente de su entrenamiento sea capaz de clasificar de misma manera imágenes (Lin, 2023), así también podemos encontrar otro tipo de redes neuronales, conocidas cómo redes neuronales siamesas. Las redes neuronales siamesas consisten en entrenar dos redes neuronales cuyas arquitecturas son iguales, pero con la excepción de que su entrenamiento tendrá un comportamiento distinto, este tipo de algoritmos nos permite el entrenamiento de modelos de aprendizaje más profundos con que poseen una mayor precisión en un menor tiempo de cómputo, su única desventaja es el gran procesamiento computacional que conllevan (Hati et al, 2023).

Al unir ambos tipos de redes neuronales obtenemos una arquitectura de redes neuronales convolucionales siamesas (SConvNet) enfocadas a la clasificación cuyo entrenamiento será más preciso con un costo computacional alto en memoria, pero bajo en el tiempo, sin embargo, existe la posibilidad de tener un sobre ajuste, esto se soluciona mediante el guardado de los pesos por época en una carpeta que contenga sus valores de precisión y error (Calik et al, 2019).

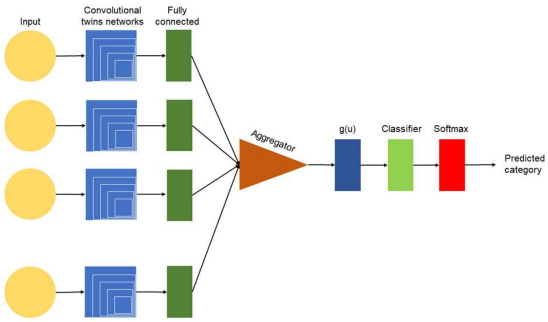


Figura 1. Diagrama de ConvNet Siamesa

La arquitectura de esta red neuronal posee una capa de entrada las cuales se conectan a múltiples capas convolucionales colocadas en paralelo, después pasan a capas de conexión completa y a un agregador que permite unir los valores obtenidos de cada capa convolucional. Para finalizar tenemos las capas de clasificación y de función softmax, el entrenamiento de esta arquitectura resultará en un modelo que dé como resultado una etiqueta de la predicción a la categoría a la cual pertenezca la imagen de la firma.
Las contribuciones principales de este trabajo son:

- Propuesta de red neuronal convolucional siamesa para el reconocimiento de firmas.
- Diseño e implementación de la arquitectura de red neuronal convolucional siamesa para una gran cantidad de imágenes.
- Comparación del comportamiento de la red neuronal convolucional siamesa propuesta contra una red neuronal convolucional simple.

2 Preliminares

Existen distintas aplicaciones de redes neuronales convolucionales para clasificación de imágenes, pero sus fundamentos matemáticos e inspiración bilógica lo podemos observar en el trabajo realizado por Omprakash Dewangan en 2023, donde observamos los pasos necesarios para llevar a cabo el diseño de una red neuronal convolucional, así como su comparación con métodos tradicionales (Dewangan, 2023). Algunas de sus aplicaciones se han visto desde la clasificación de imágenes de marcas (Ruf et al, 2023). Es debido a su gran desempeño que se han diseño herramientas tales como módulos de distintos lenguajes de programación para poder crear programas de redes neuronales (Abu et al, 2019).

Existen trabajos anteriormente dedicados a la aplicación de reconocimiento de firmas con grandes muestras, así como también con pequeñas muestras (Calik et al, 2019). En cuanto a las redes neuronales convolucionales siamesas es posible conocer sus fundamentos

matemáticos e inspiración (Yang et al, 2023) en cuanto a sus aplicaciones algunos trabajos han sido dedicados a la clasificación de imágenes Hyper espectrales (Huang et al, 2020), imágenes SAR (Cheng et al, 2022) así como también se han aplicado al manejo de datos obtenidos en experimentos biotecnológicos (Wang et al, 2022).

3 Conjunto de datos

Al aplicar una red neuronal convolucional tenemos la libertad de utilizar un conjunto de imágenes cómo conjunto de entrenamiento para el modelo de aprendizaje, en este caso se hará uso del conjunto de datos CEDAR el cual está conformado por imágenes de firmas divididas en dos categorías, firmas genuinas y firmas falsificadas. Existen firmas de distintos idiomas y para el modelo de redes neuronales siamesas presentado en este trabajo se llevará a cabo la separación por carpetas de los distintos idiomas, con esto podemos hacer una hipótesis donde obtendremos un algoritmo más eficiente, así como también un modelo de aprendizaje con mejor precisión. El conjunto de datos se utilizará para el entrenamiento del algoritmo de red neuronal, pero antes de ser ingresadas deben ser pre procesadas.

4 Pre procesamiento de imágenes

Para el entrenamiento de la SCovnNet se utilizarán imágenes de nuestro conjunto de datos. Una imagen contiene información numérica, la cual está contenida en una unidad homogénea que se conoce como bit que, para tener una profundidad de color y un nivel de brillo, utilizan un modelo de color. Las imágenes del conjunto de datos, a pesar de ser una firma de tinta negra sobre papel blanco, son leídas por la computadora en el modelo RGB, es decir, una imagen está formada por tres arreglos bidimensionales, uno rojo, uno verde y uno azul, que se representan cómo matrices (Padmavathi, 2016). Es decir, la red neuronal está procesando tres matrices por cada imagen y el entrenamiento de la red neuronal se ve ralentizado, por esta razón se realiza una conversión del modelo RGB al modelo de escala de grises.

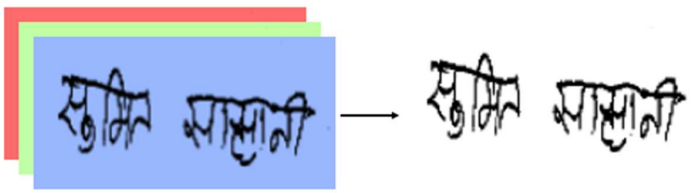


Figura 2. Conversión RBG a escala de grises

El proceso de conversión del modelo RGB al modelo de escala de grises se hará mediante la librería de visión de computadora OpenCV en el lenguaje de programación Python.

5 Entrenamiento del modelo de aprendizaje profundo

Una vez ya establecida la arquitectura del algoritmo de la red neuronal convolucional siamesa se procederá a realizar el entrenamiento, en este proceso se dividirá al conjunto de datos en dos partes, 70% del conjunto para el entrenamiento. Una vez terminado el entrenamiento se obtendrá un modelo de aprendizaje profundo el cuál debe ser probado, para esto se usará el 30% restante del conjunto de datos para realizar dichas pruebas.

De esto obtendremos métricas de la precisión y el error del modelo, para conocer la precisión real del modelo se realizará una matriz de confusión de las pruebas realizadas, donde se compararán las pruebas clasificadas correctamente contra el número total de las pruebas, que se espera tener una precisión de al menos 90%.

Para realizar el entrenamiento las imágenes en escalas de grises se introducirán a la capa de entrada como matrices de valores numéricos de entre 0 a 255, con una dimensión de 955x271 que es el tamaño en pixeles de todas las imágenes en el conjunto de datos.

El proceso de entrenamiento es mediante la generación de un conjunto de filtros (matrices) por cada categoría, firmas genuinas o falsas. Esto se lleva a cabo en las capaz convolucionales de la SConvNet que obtendrán las características que diferencian a cada una de las categorías. Estos filtros se unen en las capas de conexión completa.

Los valores numéricos de los filtros se deberán actualizar para que el modelo tenga una mejor precisión, por esto se aplica un algoritmo de retro propagación que reconoce la capa donde se encuentra el error y busca mejorar los valores de los filtros hasta que se cumpla una condición (Patterson & Gibson, 2017).

6 Resultados experimentales y análisis

Una vez realizado el entrenamiento de la red neuronal convolucional siamesa procederemos a observar cómo es que el aprendizaje fue evolucionando durante el entrenamiento. Primero mostraremos la comparación de los valores de precisión y perdida de la red neuronal convolucional siamesa con una red neuronal convolucional normal. La primera gráfica por presentar es la perdida de la red neuronal convolucional siamesa contra la perdida de la red convolucional normal en un periodo de 100 épocas.

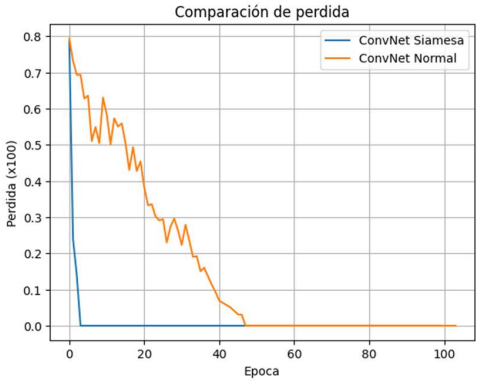


Figura 3. Perdida durante el entrenamiento de SConvNet

En esta gráfica podemos observar que no fue necesario realizar el entrenamiento sobre cien épocas, se presentan casos propios donde el punto mínimo del umbral de perdida está en la tercera época y alrededor de la época 50 para la red neuronal convolucional siamesa y para la red convolucional normal respectivamente. En el caso de la red neuronal convolucional siamesa el hecho de entrenar múltiples redes neuronales convolucionales que poseen la misma arquitectura y pasar por un proceso de agregación ayuda a que el entrenamiento no sea tan alargado y tenga un menor tiempo de ejecución comparado con el entrenamiento de una sola red neuronal convolucional. En la siguiente gráfica se presentará el comportamiento de la precisión en ambos casos de redes neuronales.

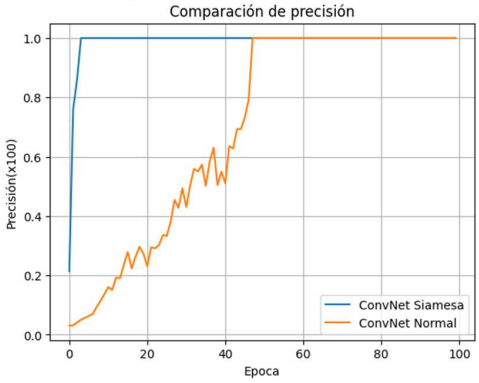


Figura 4. Precisión durante entrenamiento SConvNet

Se pueden observar un comportamiento similar a la perdida, no son necesarias 100 épocas en ninguna de las dos redes neuronales, ambas obtendrán una precisión mayor al 70% antes. Una vez comprobado que la precisión de la red neuronal convolucional siamesa

esta por arriba del 70% es necesario comprobar el funcionamiento del modelo de aprendizaje profundo obtenido del entrenamiento de la red propuesta, para esto ser mostrarán distintas pruebas de la precisión de clasificación de firmas dentro del 30% del conjunto de datos seleccionado para pruebas, estas dirán si una firma es genuina o falsa, el puntaje de diferencia que describe cuan diferentes son ambas firmas y también la clasificación que le da a dicha firma. Primero, es necesario comprobar si es posible para el modelo reconocer una firma falsa de una genuina sobre la misma firma hecha a mano.

Dentro del conjunto de datos seleccionaremos dos firmas y sus respectivas copias, una falsa y otra genuina, la comparación nos deberá dar un puntaje de diferencia, en caso de este puntaje sea mayor o igual a 0.1 entonces la copia será una falsificación, en el caso contrario la firma será genuina. La primera copia de la firma seleccionada pertenece a la carpeta de falsificaciones, entonces el modelo deberá darle una etiqueta de "falsa".

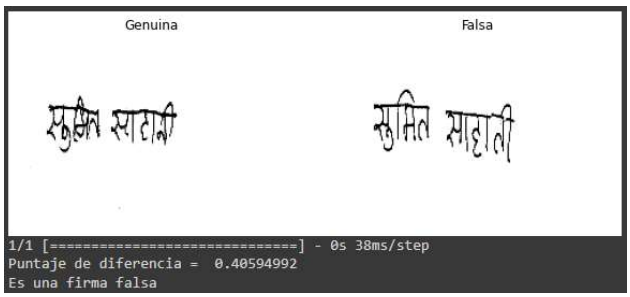


Figura 5. Primera prueba del modelo de SConvNet

En esta prueba la diferencia es mayor a 0.1 entonces la etiqueta de esta copia debería ser "falsa". Mientras que para el ejemplo en la figura 4 tenemos una copia genuina de la firma, por ello su diferencia debe ser menor a 0.1, observemos que este sea el comportamiento.

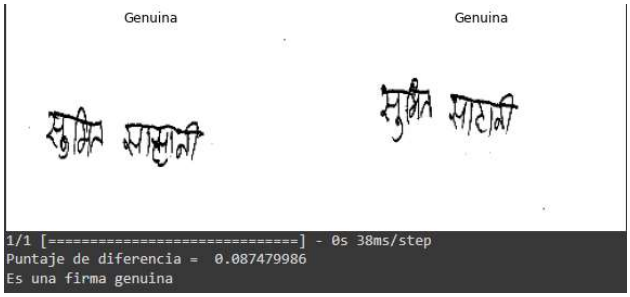


Figura 6. Segunda prueba del modelo de SConvNet

Es posible observar que se cumple la etiqueta "genuina" cuando la diferencia es menor a 0.1, procederemos a realizar pruebas con otra firma, esta firma será seleccionada del

mismo conjunto de datos apartado para realizar las pruebas. De misma manera que la firma anterior compararemos dos copias, una falsa y otra genuina, y el modelo las deberá dar una etiqueta respectiva según la diferencia que tengan con la original.

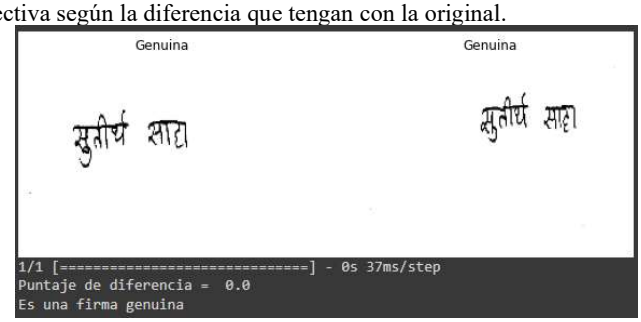


Figura 7. Tercera prueba del modelo de SConvNet

En la primera prueba podemos observar que nuestra copia genuina tiene una diferencia con valor 0, es decir, la copia de la firma y la original son la misma, por ende, debe recibir una etiqueta "genuina". Ahora para una copia de la firma proveniente de la carpeta de copias falsas se obtuvo una diferencia con valor de 0.67 que es estrictamente mayor a 0.1, es decir, la copia recibió una etiqueta "falsa".

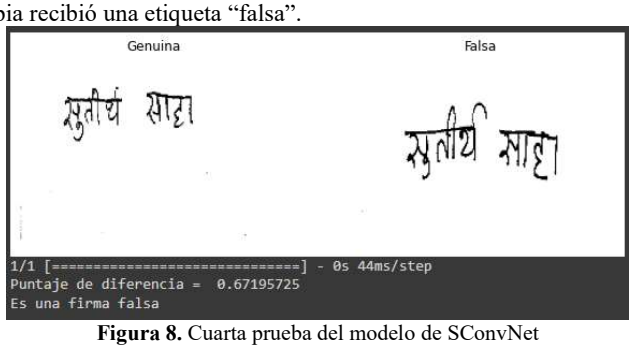


Figura 8. Cuarta prueba del modelo de SConvNet

De todas las pruebas se puede observar un funcionamiento correcto del modelo de aprendizaje obtenido del entrenamiento de la red neuronal convolucional siamesa propuesta en este trabajo. Las pruebas expuestas son para mostrar el funcionamiento adecuado de la, pero para calcular la exactitud del modelo usaremos la matriz de confusión presentada a continuación.

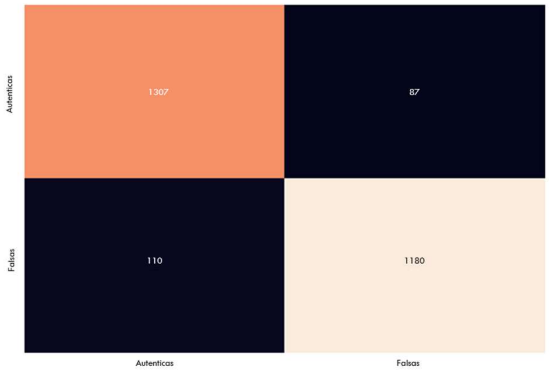


Figura 9. Matriz de confusión de SConvNet

Para las pruebas se usaron un total 2684 imágenes que corresponde al 30% del total de las imágenes de firmas en el conjunto de datos y las clasificaciones correctas (cuadro naranja y cuadro crema) son un total 2487, al dividirlo sobre la cantidad total de imágenes obtenemos una exactitud de 92% del modelo de redes neuronales convolucionales siamesas.

7 Conclusiones

La arquitectura de red neuronal convolucional siamesa propuesta en este trabajo ha demostrado tener un comportamiento similar al esperado, incluso mejor, ya que permitió ahorrar en recurso tales como la memoria del equipo con el que se trabajó, tiempo de ejecución de la red y los recursos del conjunto de datos BHSig260 pues no se necesitaron más datos para poder entrenar.

Otro punto que se puede destacar es la mejora que hay entre ambas redes observadas en este trabajo, la aplicación de una red siamesa con arquitecturas convolucionales como base trabajando juntas tiene un mejor comportamiento que una red neuronal convolucional única. Las pruebas preliminares realizadas nos dieron una idea del comportamiento del modelo de aprendizaje de la arquitectura de red propuesta, sin embargo, no se obtuvo la precisión real del modelo, es decir, no se hizo una medición completa de todas las imágenes de prueba.

La medición de la precisión real del modelo usando una matriz de confusión nos permite conocer cómo se comporta en realidad nuestro modelo, aparte de realizar pruebas de casos en específico, ambos tipos de prueba indican un funcionamiento adecuado del modelo, es decir, un entrenamiento correcto del algoritmo con arquitectura de redes neuronales convolucionales siamesas.

Referencias

Wang, Qian, Duan, Meiyu, Fan, Yusi, Liu, Shuai, Ren, Yanjiao, Huang, Lan y Zhou, Fengfeng. (2022). *Transforming OMIC features for classification using Siamese convolutional networks.* Journal of Bioinformatics and Computational Biology.

Saxena, Aarush. (2022). *An Introduction to Convolutional Neural Networks*. International Journal for Research in Applied Science and Engineering Technology.

Hati, Avik, Velmurugan, Rajbabu, Banerjee, Sayan y Chaudhuri, Subhasis. (2023). *Conditional Siamese Convolutional Network.*

Raj, Swapnil. (2022). *Deep Learning Algorithms.*

Dewangan, Omprakash. (2023*). Study and Innovative Approach of Deep Learning Algorithms and Architecture*.

Lin, Xiyu. (2023). *Research of Convolutional Neural Network on Image Classification*. Highlights in Science, Engineering and Technology.

Ruf, Yesim., Hanne, Thomas y Dornberger, Rolf. (2023). *Classification of Brand Images Using Convolutional Neural Networks*.

Abu, Mohd Azlan, Indra, Nurul Hazirah, Abd Rahman, Abdul, Sapiee, Nor y Ahmad, Izanoordina. (2019). *A study on Image Classification based on Deep Learning and Tensorflow*.

Calik, Nurullah, Kurban, Onur, Yilmaz, Ali, Yildirim, Tülay y Durak Ata, Lutfiye. (2019). *Large-Scale Offline Signature Recognition via Deep Neural Networks and Feature Embedding.* Neurocomputing.

Li, Wanying & Mahpirat, & Xu, Xuebin & Aysa, Alimjan & Ubul, Kurban. (2022). *A Simple Convolutional Neural Network for Small Sample Multi-lingual Offline Handwritten Signature Recognition*.

Yang, Mingting & Huang, Dandan & Yu, Siyu & Liu, Zhi & Duan, Jin. (2023). *Siamese Network Tracking Based on Feature Enhancement*. IEEE Access.

Huang, Lingbo & Chen, Yushi. (2020). *Dual-Path Siamese CNN for Hyperspectral Image Classification With Limited Training Samples*. IEEE Geoscience and Remote Sensing Letters. PP. 1-5.

Cheng, Dongdong & Dong, Zhangyu & Wang, Jun & Yang, Xuezhi. (2022). *A dual-scale siamese densely connected network with MRF for SAR image classification. Remote Sensing Letters*. P. 13.

Patterson, Josh & Gibson, Adam. (2017). *Deep Learning: A practitioner's approah*. O'Reilly.PP. PP. 125-139.

Capítulo 2

Reconocimiento de emociones del habla usando características basadas en Coeficientes Cepstrales de Frecuencia de Mel

Erick Barrios González[1], Mireya Tovar Vidal[1]

[1] Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación, Av. San Claudio y 14 sur, Ciudad Universitaria, 72592 Puebla, Pue., México

erick.barrios@viep.com.mx, mireya.tovar@correo.buap.mx

Resumen. En este artículo son propuestas características basadas en los Coeficientes Cepstrales de Frecuencia de Mel para la tarea de reconocimiento de emociones del habla. Las características propuestas para esta tarea se obtienen aplicando filtros pasa-bajas, pasa-altas y pasa-bandas a la señal de audio antes de obtener los Coeficientes Cepstrales de Frecuencia de Mel. Los resultados son evaluados con las métricas de WAR (*Weighted Average Recall*) y UAR (*Unweighted Average Recall*), también se contempla el uso de los corpus de EMODB, RAVDESS, y SAVEE. La implementación se realizó usando el algoritmo de SVM (*Support Vector Machines*), los resultados muestran que se logró una mejora mínima de 1% en los corpus revisados.

Palabras Clave: MFCCs, SVM, Reconocimiento de emociones.

1 Introducción

El habla es importante principalmente porque nos permite compartir información con otras personas, las emociones forman parte de esas cosas que compartimos con otras personas, las emociones son las reacciones humanas a situaciones particulares y que pueden llegar a ser expresadas con palabras (Wen et al, 2022).

En ciencias de la computación, el habla puede ser analizada usando las señales de audio capturadas. Aunque las emociones pueden ser detectadas automáticamente usando señales de electroencefalografía (EEG) o imágenes faciales (Tuncer et al, 2021), la manera más accesible es usando las señales de audio debido a cuestiones de privacidad o de accesibilidad a herramientas específicas.

En la interacción humano-computadora, el reconocimiento de emociones en el habla consiste en reconocer emociones y estados afectivos de señales de audio (Ye et al, 2016). El reconocimiento de emociones en el habla es igual de importante, debido a que se usa

para reconocer y entender las emociones del ser humano con una gran variedad de propósitos, como lo es la mejora de la experiencia interactiva con las máquinas.

El objetivo principal del reconocimiento de emociones en el habla es el uso efectivo de características de señales de audio para determinar los estados emocionales de una persona (Tuncer et al, 2021). Existen diferentes estados de ánimo, pero algunos de los más conocidos son neutral, felicidad, tristeza y furia.

Un reto esencial en la tarea del reconocimiento de emociones en el habla es extraer atributos comunes de diferentes hablantes (incluyendo diferentes idiomas), especialmente cuando un corpus fuente específico tiene que ser entrenado para reconocer los datos desconocidos provenientes de otro corpus de voz (Ancilin, 2021). Los sistemas del reconocimiento de emociones en el habla se utilizan para determinar la condición mental y física de la persona que utiliza la señal de voz. También se utilizan en diversas aplicaciones, como aplicaciones médicas, sistemas inteligentes, robots de servicio que reconocen las emociones negativas y no negativas del usuario para brindar el servicio correspondiente, sistemas de vigilancia mediante la identificación de emociones de tipo miedo, identificar clientes enojados en centros de llamadas, detección de engaños, etc. (Tuncer et al, 2021) (Wen et al, 2022).

En este trabajo se explorará el uso de filtros (pasa-bajas, pasa-altas y pasa-bandas) para el reconocimiento de emociones del habla. Las próximas secciones están organizadas de la siguiente manera: trabajo relacionado, algoritmo de reconocimiento de emociones del habla, experimentos y conclusiones.

2 Trabajo relacionado

Hay varios enfoques para resolver este problema; los métodos tradicionales se enfocan en la extracción eficiente de características hechas a mano, que se incorporan a los métodos convencionales de aprendizaje automático, como el uso del algoritmo de *Support Vector Machines* (SVM) (Ye et al, 2016)(Tuncer et al, 2021), los métodos recientes se basan en técnicas de aprendizaje profundo que emplean varias arquitecturas como las Redes neuronales convolucionales (CNN) (Mustaqeem, 2021), Redes neuronales recurrentes (RNN) (Tao y Liu, 2018), o la combinación de ambas.

En el reconocimiento de emociones en el habla, los enfoques de modelado temporal se utilizan ampliamente con el objetivo de capturar variaciones temporales dinámicas de las señales de voz. Por ejemplo, en el trabajo de Tao y Liu (2018) se propone un tipo de LSTM (*Advanced LSTM* (A-LSTM), un tipo de RNN), para un mejor modelado del contexto temporal. Este modelo obtuvo un puntaje de F_1 macro de 46.2 con el corpus IEMOCAP (clases neutral, feliz, enojado y triste). O en el trabajo de Ye et al (2016), que propone una arquitectura bidireccional que integra información complementaria del pasado y del futuro para modelar dependencias temporales de largo alcance, usando interdependencias a diferentes escalas temporales que utilizan características de los Coeficientes Cepstrales de Frecuencia de Mel (MFCCs).

Sin embargo, a pesar de que existen enfoques que intentan capturar características temporales, la selección de características también es relevante, como en el trabajo de Mustaqeem (2021) donde se propone una red neuronal convolucional de dos flujos con un análisis iterativo de componentes de vecindad (INCA por sus siglas en inglés), usa características espectrales y selecciona las características óptimas de forma discriminatoria o en el trabajo de Tuncer et al (2021) que empleó un modelo de generación de características multinivel no lineal utilizando una estructura criptográfica, y luego selecciona características (con INCA), utilizando un clasificador SVM para la clasificación.

También se puede observar que las características basadas en los Coeficientes Cepstrales de Frecuencia de Mel (MFCCs) son las más utilizadas para el reconocimiento de emociones en el habla. Los MFCC son la transformada de coseno discreta del logaritmo del espectro de potencia en una frecuencia de escala de Mel, que representa el habla de una mejor manera al aprovechar la percepción auditiva de los humanos y se usa en la mayoría de los estudios de reconocimiento de emociones del habla (Wen et al, 2022).

Por ejemplo, en el trabajo de Ye et al (2021) que usó solo los coeficientes cepstrales de frecuencia de Mel, como entradas y luego los pasa a través del módulo convolucional temporal controlado para generar las características de alto nivel. En el trabajo de Wen et al (2022), se realizan dos modificaciones en la extracción de los coeficientes cepstrales de la frecuencia de Mel (Wen et al, 2022) utiliza el espectro de magnitud en lugar del espectro de energía, la exclusión de la transformada de coseno discreta y la extracción de los coeficientes de magnitud de la frecuencia de Mel.

En este trabajo se hará una comparación de las características utilizadas para el reconocimiento de emociones del habla, utilizando características MFCCs (aplicando previamente al archivo de audio filtros de pasa-bajas, pasa-altas y pasa-bandas) y el algoritmo SVM.

3 Algoritmo de reconocimiento de emociones del habla

En esta sección se describirá el proceso que se llevó a cabo para realizar el reconocimiento de emociones del habla, se describirán los siguientes puntos: el preprocesamiento y la extracción de características.

3.1 Preprocesamiento

A cada archivo de audio se le realizó un preprocesamiento, el preprocesamiento consiste en cambiar su frecuencia de muestreo a 16,000 Hz y reducir los canales de audio a uno solo por archivo, para finalmente normalizar el volumen de audio a 1 dB.

El preprocesamiento reduce las irregularidades al analizar la información en diferentes archivos. Específicamente el reducir la tasa de muestreo y convertir el audio a un solo canal ayuda a reducir la cantidad de información a analizar y reducir el tiempo de ejecución del algoritmo, para esto la tasa de muestreo elegida se determinó a partir de una sugerencia en (Fadheli, 2021).

3.2 Extracción de características

La función más utilizada para el reconocimiento de emociones del habla son los MFCC; sin embargo, se agregaron otras características de uso frecuente, como *Chroma* (chroma), *Mel Spectrogram Frequency* (mel), *Contrast* (contraste) y *Tonnetz* (tonnetz).

Estas características fueron obtenidas a partir del audio preprocesado. Además, como novedad en este trabajo se usaron características basadas en MFCC, estas características se obtuvieron usando 3 diferentes tipos de filtros (filtros pasa-bajas, pasa-altas y pasa-bandas) y usando 3 filtros (Elliptic, Chebyshev tipo 2 y Butterworth) encontrados en la librería Scikit-learn de Python.

- MFCC normal. - Los MFCC se obtienen sobre el audio preprocesado sin aplicar previamente ningún filtro.
- Opción 1.- Se obtienen MFCCs sobre el audio preprocesado, pero previamente se le aplica un filtro de pasa-bajas, a una frecuencia de 2000.
- Opción 2.- Se obtienen MFCCs sobre el audio preprocesado, pero previamente se le aplica un filtro pasa-altas; con una frecuencia de 2000.
- Opción 3.- Se obtienen MFCCs sobre el audio preprocesado, pero previamente se le aplica un filtro pasa-banda, en un rango de frecuencias de 100-800 Hz.

La selección de frecuencias donde se aplican los filtros se basó en la distribución de frecuencias en la voz humana (alrededor de 100 Hz a 4000 Hz (McLoughlin, 2009) y la realización de varias pruebas.

En la Figura 1 se observan los pasos para el preprocesamiento y la extracción de características, se aplica el filtrado de la señal de audio antes de extraer las características de los MFCCs.

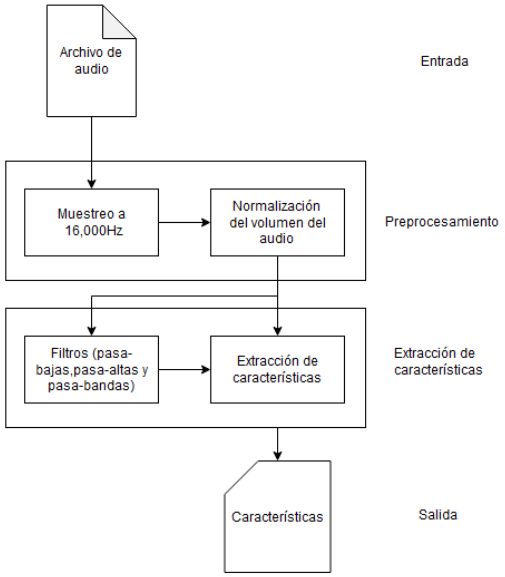


Figura. 1. Secuencia de pasos para preprocesamiento y extracción de características.

4 Experimento y evaluación

En esta sección se mostrarán, los conjuntos de datos utilizados, los experimentos realizados, las métricas utilizadas y los resultados obtenidos.

.

4.1 Conjuntos de datos

Para demostrar las ventajas de este método, se probará con los siguientes corpus:

- EMODB (Burkhardt et al, 2005): corpus alemán que contiene 470 archivos, las emociones del habla incluyen ira (69 instancias), aburrimiento (81 instancias), asco (46 instancias), ansiedad/miedo (71 instancias), tristeza (62 instancias), neutral (79 instancias) y dolor (62 instancias) expresiones (7 clases).
- RAVDESS (Livingstone y Russo, 2018): Corpus en inglés que contiene 1440 archivos, las emociones del habla incluyen calma (192 instancias), felicidad (192 instancias), tristeza (192 instancias), enojo (192 instancias), miedo (192 instancias), sorpresa (192 instancias), neutrales (96 instancias) y disgusto (192 instancias) (8 clases).

- SAVEE (Haq y Jackson, 2010): Corpus en inglés que contiene 960 archivos, las emociones del habla incluyen ira (120 instancias), asco (120 instancias), miedo (120 instancias), felicidad (120 instancias), tristeza (120 instancias), expresiones neutrales (240 instancias) y sorpresa (120 instancias), en total 7 clases.

En la Tabla 1, se observa que el corpus EMODB tiene una distribución de clases diferente, además de que carece de algunos estados de ánimo positivos como felicidad y calma.

Tabla 1. Número de instancias por clase en cada corpus.

Corpus	Miedo	Triste	Sorpresa	Dolor	Neutral
RAVDESS	192	192	192	--	96
EMODB	71	62	--	62	79
SAVEE	120	120	120	--	240

Corpus	Ira	Aburrido	Calma	Disgusto	Felicidad
RAVDESS	192	--	192	192	192
EMODB	69	81	--	46	--
SAVEE	120	--	--	120	120

4.2 Experimento

Las características fueron seleccionadas usando búsqueda de rejilla para todas las posibles combinaciones de los diferentes atributos. Con la finalidad de explorar todas las posibilidades en una búsqueda de rejilla se usó el algoritmo SVM, este algoritmo es más rápido que los clasificadores basados en redes neuronales.

Para los resultados se ha optado por usar validación cruzada usando 2 particiones debido a que la cantidad de instancias por cada clase es menor a 60 instancias en el mejor de los casos si se hicieran 3 particiones.

Para la evaluación del sistema se utilizaron principalmente dos métricas:

- WAR *(Weighted average recall*): El promedio ponderado de recuperación da la media ponderada de recuperación con ponderaciones iguales a la probabilidad de clase.
- UAR (*Unweighted Average Recall*): que se define como el promedio de recuperación obtenido en cada clase.

Las métricas fueron seleccionadas debido al desequilibrio de clases. Además, el trabajo de Ye et al (2016) recopila evaluaciones de esta tarea de diferentes corpus con estas métricas, incluidos los corpus utilizados en este trabajo.

5 Resultados

En esta sección los resultados obtenidos con los diferentes corpus serán mostrados. En la Tabla 2 se observa una comparación usando los filtros Elliptic (Cauer), Chebyshev tipo 2 y Butterworth para el corpus EMODB, se puede observar las diferentes combinaciones de atributos que obtuvieron los mejores resultados. Para cada filtro se contemplan dos resultados uno contemplando las opciones propuestas de MFCCs y el otro solo contempla los atributos normales. Como se observa en la Tabla 2 el mejor resultado se obtiene con el filtro Butterworth usando como atributos algunas de las opciones propuestas en este artículo (Opción 2 y 3), con una diferencia aproximada de 0.01 con el resultado que no contempla las opciones propuestas. Por otro lado, los demás filtros para ese corpus tienen un comportamiento similar.

Tabla 2. Comparación de resultados usando diferentes filtros para corpus EMODB.

Filtro	Combinación	WAR	UAR
Elliptic	Mel, contrast, tonnetz, MFCCs, (Normal, opción 1)	0.7102	0.7055
Elliptic	Chroma, tonnetz, contrast, MFCCs (Normal)	0.7289	0.6951
Cheby II	MFCCs, (Opción 2 y 3)	0.7289	0.7188
Cheby II	MFCCs, (Normal), contrast	0.7289	0.6951
Butterworth	Chroma, mel, contrast, MFCCs (Normal, Opción 2 y 3)	**0.7476**	**0.7293**
Butterworth	Chroma, tonnetz, contrast, MFCCs (Normal)	0.7383	0.7004

Tabla 3. Comparación de resultados usando diferentes filtros para corpus SAVEE.

Filtro	Combinación	WAR	UAR
Elliptic	Mel, contrast, MFCCs, (Normal, opción 1 y 3)	**0.7083**	**0.6702**
Elliptic	Chroma, tonnetz, mel	0.6979	0.6642
Cheby II	Chroma, tonnetz, mel	0.6979	0.6642
Cheby II	Contrast, MFCCs (Opción 3)	0.6666	0.6629
Butterworth	Chroma, tonnetz, mel	0.6979	0.6642
Butterworth	Chroma, tonnetz, contrast, MFCCs (Opción 1)	0.6458	0.6582

En la Tabla 3 se observa una comparación usando los filtros Elliptic (Cauer), Chebyshev tipo 2 y Butterworth para el corpus SAVEE, se puede observar las diferentes combinaciones de atributos que obtuvieron los mejores resultados. Para cada filtro se contemplan dos resultados uno contemplando las opciones propuestas de MFCCs y el otro solo contempla los atributos normales.

Para la Tabla 3 el mejor resultado se obtiene con el filtro Elliptic usando como atributos algunas de las opciones propuestas en este artículo (Opción 1 y 3), con una diferencia aproximada de 0.01 con el resultado que no contempla las opciones propuestas a diferencia de la Tabla 2, los resultados para los demás filtros los mejores resultados no se obtienen con las mejores propuestas.

En la Tabla 4 se observa una comparación usando los filtros Elliptic (Cauer), Chebyshev tipo 2 y Butterworth para el corpus RAVDESS, se puede observar las diferentes combinaciones de atributos que obtuvieron los mejores resultados. Para cada filtro se contemplan dos resultados uno contemplando las opciones propuestas de MFCCs y el otro solo contempla los atributos normales. Para la Tabla 4 el mejor resultado se obtiene con el filtro Chebyshev usando como atributos algunas de las opciones propuestas en este artículo (Opción 2 y 3), con una diferencia aproximada de 0.02 con el resultado que no contempla las opciones propuestas. En esta tabla resaltan mucho los atributos chroma y mel, para los atributos que no contemplan las opciones propuestas.

Tabla 4. Comparación de resultados usando diferentes filtros para corpus RAVDESS.

Filtro	Combinación	WAR	UAR
Elliptic	Chroma, mel	0.4930	0.4911
Elliptic	Chroma, tonnetz, mel, MFCCs (Normal, Opción 2)	0.4618	0.4582
Cheby II	Contrast, tonnetz, mel, MFCCs (Normal, Opción 2 y 3)	**0.5173**	**0.5138**
Cheby II	Chroma, mel	0.4911	0.4930
Butterworth	Chroma, contrast, tonnetz, MFCCs (Opción 1, 2 y 3)	0.5104	0.5032
Butterworth	Chroma, mel	0.4930	0.4911

Tabla 5. Comparación entre los mejores resultados de los diferentes corpus.

Corpus	Filtro	Combinación	WAR	UAR
EMODB	Butterworth	Chroma, mel, contrast, MFCCs (Normal, Opción 2 y 3)	**0.7476**	**0.7293**
SAVEE	Elliptic	Mel, contrast, MFCCs (Normal, Opción 1 y 3)	0.7083	0.6702
RAVDESS	Cheby II	Mel, contrast, tonnetz, MFCCs (Normal, Opción 2 y 3)	0.5173	0.5138

Finalmente, en la Tabla 5 se observan los mejores resultados para cada corpus, cada corpus obtuvo un mejor resultado con diferentes filtros, lo cual no da una conclusión muy específica de qué tipo de filtro funcionaria mejor como propuesta, sin embargo, los mejores resultados son obtenidos usando las opciones propuestas. En los tres resultados la opción 3 (filtro pasa-bandas) aparece, seguido de la opción 2 (filtro pasa-altas) que aparece en 2 de los 3 resultados.

6 Conclusiones

En este artículo, la tarea de reconocimiento de emociones del habla es abordada con una solución basada en características MFCCs, las características fueron creadas aplicando filtros pasa-bajas, pasa-altas y pasa-bandas antes de obtener las características MFCCs.

Los resultados fueron evaluados usando las métricas UAR y WAR, los mejores resultados fueron obtenidos con las características implementadas con una mejora mínima de 0.01 para todos los corpus. Debido a que el filtro tipo pasa-bandas (opción 3) fue el que apareció en todos los mejores resultados se podría concluir que es una buena característica para contemplar en esta tarea

En trabajos futuros se contempla la implementación de estos filtros en otros corpus como lo son CASSIA, EMOVO, IEMOCAP mencionados en el trabajo de Ye et al (2016). Finalmente, se propone explorar el uso de estas características con clasificadores basados en redes neuronales (como lo son redes neuronales convolucionales y recurrentes).

Referencias

Ye, J., Wen, X., Wei, Y., Xu, Y., Liu, K., Shan, H. (2023). “Temporal Modeling Matters: A Novel Temporal Emotional Modeling Approach for Speech Emotion Recognition”. *International Conference on Acoustics, Speech and Signal Processing 2023*, pp. 01--01. IEEE.

Burkhardt, F., Paeschke A., Rolfes, M., et al. (2005). “A database of German emotional speech”. *INTERSPEECH 2005*, pp. 1517–-1520.

Costantini, G., Iaderola, I., Paoloni, A., Todisco, M. (2014). “EMOVO Corpus: an Italian Emotional Speech Database”. *Proceedings of the Ninth International Conference on Language Resources and Evaluation ({LREC}'14)*, pp. 3501--3504.

Smith, J., Tsiartas, A., Wagner, V., Shriberg, E., Bassiou, N. (2018). “Crowdsourcing Emotional Speech”. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5139--5143.

Livingstone, S., Russo, F. (2018). “The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English”. *PLoS ONE13(5): e0196391. Joseph Najbauer, University of Pécs Medical School, HUNGARY* . Recuperado de: https://doi.org/10.1371/journal.pone.0196391

Ancilin, Milton, A. (2021). “Improved speech emotion recognition with Mel frequency magnitude coefficient”. *Applied Acoustics*, pp. 108046. Elsevier , Vienna, Austria

Fadheli, A. (2021). “Speech Emotion Recognition”. *International Journal of Intelligent Systems*, pp. 5116-- 5135. Wiley, Seoul, Republic of Korea

GitHub (2023), *Speech Emotion Recognition*. Recuperado de https://github.com/x4nth055/emotion-recognition-using-speech.

Haq, S., Jackson, P.J.B. (2010). “Multimodal Emotion Recognition”. *Machine Audition: Principles, Algorithms and Systems*, pp. 398--423. IGI Global Press, University of Surrey, UK

McLoughlin, I. (2009). “Applied Speech and Audio Processing”. *First edition. Cambridge University Press*, Cambridge, UK

Mustaqeem, Kwon, S. (2021). “Optimal feature selection based speech emotion recognition using two-stream deep convolutional neural network”. *International Journal of Intelligent Systems*, pp. 5116-- 5135. Wiley , Seoul, Republic of Korea

Scikit-learn (2023). *NuSVC*. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html

Tao, F., Liu, G. (2018). “Advanced LSTM: A Study About Better Time Dependency Modeling in Emotion Recognition”. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2906--2910. IEEE Press, Calgary

Tuncer, T., Dogan, S., Acharya U. (2021). “Automated accurate speech emotion recognition system using twine shuffle pattern and iterative neighborhood component analysis techniques”. *Knowledge-Based Systems*, pp. 106547. Elsevier, Sydney, Australia

Ye, J., Wen, X., Wang, X., Xu, Y., Luo, Y., Wu, C., Chen, L., Liu, K. (2021). “GM-TCNet: Gated Multi-scale Temporal Convolutional Network using Emotion Causality for Speech Emotion Recognition”. *Speech Communication*, pp. 21--35. Elsevier, Valencia.

Wen, X., Ye, J., Luo, Y., Xu, Y., Wang, X., Wu, C., Liu, K., (2022). “CTL-MTNet: A novel CapsNet and transfer learning-based mixed task net for single-corpus and cross-corpus speech emotion recognition”. *IJCAI 2022*, pp. 2305–-2311. Elsevier, Vienna.

Capítulo 3

Extracción de entidades a partir de textos en lenguaje médico en español utilizando características morfológicas y semánticas

Nidia K Serafín-Rojas[1], José A. Reyes-Ortiz[1], Maricela Bravo[1], Gabriela A. García-Robledo[1]

[1] Universidad Autónoma Metropolitana, Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería

{al2221800042,jaro,mcbc,gagr}@azc.uam.mx

Resumen. El campo de la medicina es muy vasto y abarca una gran cantidad de disciplinas, que se incrementan con el tiempo. Esto genera una amplia cantidad de información médica disponible como ensayos clínicos, diagnósticos, informes médicos, estos textos no suelen estar estructurados, es decir, están escritos en lenguaje natural, específicamente en lenguaje médico, el cual es un lenguaje científico y técnico. La información del dominio médico es diversa y compleja, y la lectura para identificar nuevo conocimiento puede ser una tarea costosa y nada sencilla. Con una búsqueda rápida y oportuna se puede realizar análisis, toma de decisiones, automatizar tareas, descubrir conocimiento oculto, organizar documentos de manera sencilla, por lo que, resulta necesario el uso de herramientas computacionales para buscar y localizar información médica especializada de manera precisa. En este trabajo se describe un método para la identificación de entidades, a partir de un conjunto de textos conformado por enunciados escritos en lenguaje médico en español. Este método consiste en procesar los textos, extraer características morfológicas y semánticas; y aplicar algoritmos de aprendizaje supervisado: vecinos más cercanos, árboles de decisión, máquinas de soporte vectorial y perceptrón. También, en este trabajo se presenta un proceso de evaluación basado en métricas bien conocidas en el estado del arte, para la comparación de los diferentes enfoques de características y algoritmos, obteniendo los mejores resultados con el algoritmo de máquinas de soporte vectorial y la combinación de características morfológicas y semánticas con una exactitud del 83.21%.

Palabras Clave: Procesamiento de Lenguaje Médico, Aprendizaje automático, Morfológico, Semántico, Extracción de entidades, texto clínico, Python.

1 Introducción

En México los problemas de salud son una preocupación constante para los expertos, tal es el caso de las epidemias en 2023, la Secretaría de Salud muestra las estadísticas sobre las enfermedades como Dengue, Enfermedades Diarreicas Agudas, Hepatitis Grave Aguda, Infecciones Respiratorias Agudas, Influenza (Secretaría de Salud, 2023). La extracción oportuna y relevante de información en textos en lenguaje médico es de gran importancia debido a que se pueden encontrar entidades específicas como: enfermedades y medicamentos. Sin embargo, el análisis de esta información clínica puede resultar ser complicado y llevar mucho tiempo.

La aplicación de un método automático para realizar esta actividad puede ser de gran ayuda para los expertos en esta área. Los conceptos se encuentran descritos en Lenguaje Médico en diversos documentos como artículos científicos y ensayos clínicos, por mencionar algunos. Esta información no tiene una estructura clara, es decir, puede estar descrita con diversas formas y estructuras del lenguaje español. Otro de los usos de la extracción de información es cuando se requiere información específica urgente, tal fue el caso de lo que ocurrió en la pandemia de COVID-19. Esto ocasiona que exista la demanda de algoritmos, métodos o enfoques computacionales para la extracción automática de conceptos y relaciones.

Este trabajo presenta un método para la identificación de entidades médicas, a partir de un conjunto de textos en lenguaje médico en español. El método se desarrolló utilizando un conjunto de enunciados con temas referentes a la salud, específicamente textos obtenidos de la competencia denominada eHealth-KD Challenge 2021 (Piad-Morffis et al, 2019). Estos textos constan de dos archivos, el primero con sentencias y el segundo contiene anotaciones. Al archivo de texto se aplica tokenización para separarlas en palabras, y poder extraer la entidad de acuerdo con el archivo de anotaciones. Además, se identifica el contexto, que consiste en las cinco palabras antes y cinco palabras después de la entidad. A continuación, se realiza un etiquetado POS y NER mediante un proceso para contar la frecuencia de las etiquetas y crear una matriz de características. Estas características se utilizan como entrada a los algoritmos de aprendizaje automático y permiten clasificar las entidades. Finalmente, se evalúa la exactitud del método. La arquitectura general de la propuesta se puede separar en cuatro fases: recopilación de textos, tareas de Procesamiento de Lenguaje Natural, identificación de entidades y evaluación del método. Con este enfoque, se busca facilitar la extracción de información relevante en el ámbito de la salud y contribuir al avance de la investigación en el procesamiento automático del lenguaje médico aplicado a documentos o textos en español.

El resto del artículo se organiza como se describe a continuación. En la Sección 2, se presenta una revisión de trabajos relacionados con la identificación de entidades, uso de algoritmos y sus evaluaciones, diversos enfoques, y diversos trabajos de extracción de características. En la Sección 3 se detalla el método propuesto, inicialmente la descripción del conjunto de datos, el procesamiento de textos, la extracción de características,

aplicación de los algoritmos de aprendizaje automático, evaluación de los resultados. En la Sección 4 se muestran los diversos experimentos realizados y las comparaciones, tal es el caso de la extracción de características morfológicas y semánticas y algunas combinaciones de características que nos permiten comparar los algoritmos de aprendizaje automático. En la sección 5 se presentan las conclusiones y el trabajo a futuro.

2 Estado del Arte

En esta sección se presenta una revisión del estado del arte en relación con la identificación de entidades en textos de diversos idiomas. Se destaca la escasez de trabajos que utilizan corpus en español y se pone énfasis en los enfoques de aprendizaje automático y profundo empleados para abordar esta tarea, así como en los resultados presentados en cada estudio.

En el estudio (Dey et al, 2022) propuso un enfoque para el reconocimiento de entidades nombradas en un conjunto diverso de textos relacionados con Covid-19 en inglés. Para lograr esto, se emplearon algoritmos de aprendizaje automático que incluye técnicas de limpieza de datos, tokenización, etiquetado POS, *chunking* y NER. La clasificación de entidades se realizó utilizando varios algoritmos como Gradiente Estocástico Descendiente, Naive Bayes, Campos Aleatorios Condicionales y Máquinas de Soporte Vectorial. Los resultados mostraron que el algoritmo de Campos aleatorios Condicionales obtuvo el mejor rendimiento, con una medida *F_1* de 0.96.

En estudio realizado por (Zhang et al, 2019) en el dominio clínico, se utilizaron 500 notas psiquiátricas en inglés del centro médico de Taiwán. El objetivo del estudio fue comparar los algoritmos de Campos Aleatorios Condicionales contra la red neuronal BLSTM-CRF (Memoria a Corto Plazo Bidireccional de Longitud Variable - Campo Aleatorio Condicional), y se encontró que el segundo obtuvo mejores resultados en términos de la medida *F_1*.

El trabajo realizado por (Al-Smadi et al, 2019), consiste en un análisis de sentimiento que utiliza diversas técnicas para la extracción de información. En primer lugar, se lleva a cabo la identificación de categorías de aspectos, empleando una Máquina de Soporte Vectorial para el entrenamiento. Además, se aborda la tarea de extracción de la expresión objetivo de la opinión, esto aplicando tareas de procesamiento de lenguaje natural y extracción de características morfológicas, sintácticas y semánticas. En este estudio se aplican algoritmos de aprendizaje automático como Máquinas de Soporte Vectorial, Naive Bayes, Árboles de decisión y Vecino más cercano, con reseñas de hoteles en árabe.

(Dellanzo et al, 2022) aborda la escasez de trabajos que utilizan corpus en español. Para superar esta limitación, se construye un corpus compuesto por 513 artículos anotados. El objetivo es realizar tareas de reconocimiento de entidades nombradas y extracción de relaciones en textos de informes de salud sobre brotes en América Latina.

Para lograrlo, se emplea una combinación de Redes Neuronales Recurrentes y Campos Aleatorios Condicionales.

Los resultados indican que, para el uso de algoritmos de aprendizaje profundo, la disponibilidad de un corpus reducido representa una limitación. La mayoría de los estudios se han realizado en idiomas dominantes como el inglés. Sin embargo, el trabajo de (Kaur y Khattar, 2021) destaca por abordar el idioma Punjabi y el procesamiento de textos no estructurados. En su trabajo, se utilizan algoritmos como el modelo oculto de Márkov, el principio de máxima entropía y Campos Aleatorios Condicionales. Los resultados obtenidos en términos de la medida *F_1* son 77.61%, 83.65% y 93.21% respectivamente.

En otro estudio relacionado con el dominio clínico (Wei et al, 2020) se realiza una comparación entre un enfoque de aprendizaje profundo y un sistema de máquina de vectores. Los resultados concluyen que el método basado en una red neuronal artificial es más eficiente. En este trabajo se extraen medicamentos y la relación con sus efectos secundarios adversos.

En el ámbito de la atención médica, uno de los aspectos importantes para impulsar las aplicaciones es la construcción a gran escala de grafos de conocimiento médico. Con este objetivo en mente el proyecto liderado por (Wu et al., 2021) se propone el sistema BioIE. Se trata de una red neuronal híbrida para extraer relaciones de textos biomédicos e informes médicos no estructurados. El modelo emplea una red convolucional gráfica mejorada(GCN) con atención de múltiples cabezales para capturar relaciones complejas y contextualizar la información. Se evalúa el modelo en dos tareas principales de extracción de relaciones biomédicas, la relación *químico-enfermedad* (CDR) y la relación *químico-proteína* (CPI). Los resultados demuestran que este enfoque supera a las líneas base. El corpus utilizado en este estudio consiste en informes de patología de cáncer recopilados de hospitales.

En otro trabajo relevante (Jouffroy et al, 2021), se destaca la importancia de un modelo para extraer relaciones entre enfermedades y medicamentos, así como las relaciones de dosis, frecuencia, dirección, vía y condición de administración. Los datos de entrada para este trabajo se encuentran en idioma francés, y la medida de evaluación utilizada es la medida *F_1*, con un porcentaje de 89.9%.

La extracción de información desempeña un papel fundamental en diversas áreas y es de gran ayuda para los expertos en cada campo, tal como se muestra en el estudio de (Kumar et al, 2022) en el ámbito de la agricultura. En este trabajo se aplican técnicas de aprendizaje profundo para extraer conceptos y relaciones, como la relación clima-cultivo.

Por último, el trabajo de (Sánchez-Graillet et al, 2022) proporciona una visión general sobre el uso final de la extracción de conceptos clínicos, haciendo énfasis en la utilización de ontologías.

3 Método propuesto

En esta sección, se describe el método propuesto para la identificación de entidades en textos relacionados con la salud en español, basado en técnicas de aprendizaje automático. El método se divide en cuatro fases, que se describen a continuación.

- Fase 1: Recopilación del conjunto de datos clínicos. Esta fase se encarga de extraer y analizar el conjunto de textos en lenguaje médico en idioma español que contengan etiquetas para entidades; en este caso se usó el corpus del desafío *eHealth-KD Challenge 2021*. IberLEF (2021).
- Fase 2: Procesamiento de textos. Se aplican técnicas de Procesamiento de Lenguaje Natural, para este trabajo se ejecutaron las tareas de limpieza, tokenización, etiquetado parte de la oración (POS), etiquetado de entidades nombradas (NER) y como resultado se extraen características morfológicas y semánticas.
- Fase 3: Extracción de Información. En esta fase se realiza la identificación automática de entidades a partir de textos en lenguaje médico en español. Esta identificación se llevará a cabo utilizando una representación vectorial (obtención de características) de los textos de la etapa anterior. Y estas características se ingresarán a los algoritmos de aprendizaje automático y su salida determinará los tipos de entidades existentes.
- Fase 4: Evaluación. Esta etapa se encargará de obtener el valor de exactitud para la tarea de identificación de entidades.

Las fases del método propuesto se ilustran en la Figura 1, brindando una clara visualización.

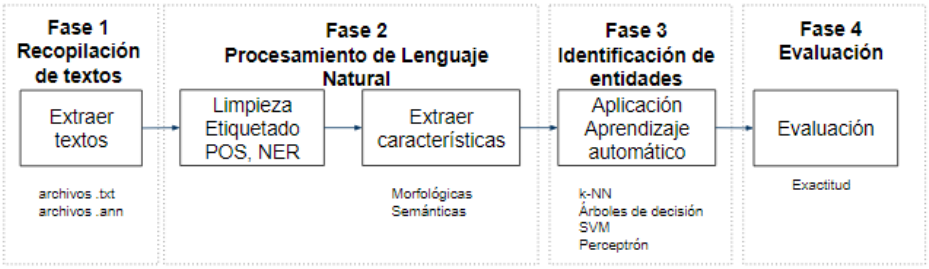


Figura 1. Método propuesto para la identificación de entidades en textos en español.

A continuación, se proporciona el detalle de cada una de las fases que componen el método propuesto. Esto permitirá comprender a fondo el proceso y los pasos involucrados en la identificación de entidades en textos relacionados con la salud en español.

3.1 Recopilación del conjunto de textos en lenguaje médico

El conjunto de textos utilizado en este trabajo se obtuvo del desafío *eHealth-KD Challenge 2021* (Piad-Morffis, 2019). Este conjunto de textos contiene frases extraídas de *MedlinePlus*, *Wikinoticias* y del corpus CORD-19, todas relacionadas con temas de salud. En cuanto al corpus original, contiene sentencias en inglés y español. Sin embargo, en este estudio nos enfocaremos exclusivamente en las sentencias en español. El corpus está formado por pares de archivos de texto que contiene sentencias referentes a la salud y el archivo de anotaciones que consta de tres columnas, la primera columna se refiere al carácter "T" seguido de un consecutivo, la segunda columna muestra el tipo de entidad y las posiciones donde se encuentra y la tercera columna la entidad, como se ve en la Figura 2.

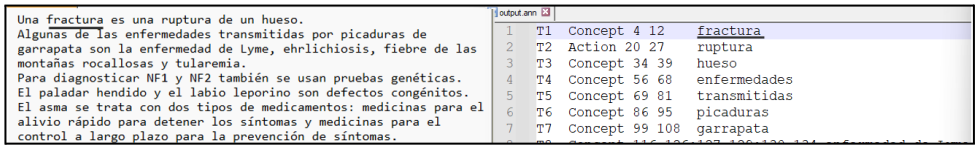


Figura 2. Ejemplo de archivo de texto y anotaciones.

En este texto las entidades se clasifican en cuatro categorías: concepto, acción, predicado y referencia y los contadores se muestran en la Tabla 1.

Tabla 1. Frecuencia de aparición de cada entidad etiquetada en las sentencias.

Tipo entidad	Conteo
Action	512
Concept	1388
Predicate	190
Reference	55
Total	2145

Como salida de la fase de recopilación de textos se construye un *dataframe* con cuatro campos que incluyen un consecutivo, entidad, contexto el cual se forma por cinco palabras antes de la entidad y cinco palabras después de la entidad, este contexto puede ser variable en caso de que no se haya encontrado palabras antes o después y finalmente el campo de la categoría asignada, para esta transformación de los datos, se creó un método en lenguaje *Python*. Esta salida será utilizada en la fase de procesamiento y se muestra en la Figura 3.

	Entidad	Contexto	TipoEntidad
0	sistema vascular	El @ent es la red de vasos	Concept
1	red	El sistema vascular es la @ent de vasos sanguí...	Predicate
2	vasos sanguíneos	vascular es la red de @ent del cuerpo.	Concept

Figura 3. Ejemplo de textos recopilados que muestran la entidad y sus contextos.

3.2 Procesamiento de textos

A partir de los contextos extraídos se aplica una serie de tareas, tales como: etiquetado POS a la entidad, etiquetado POS al contexto y etiquetado NER al contexto.

El etiquetado POS se refiere a asignar la categoría gramatical de las palabras, en este caso podemos ver un ejemplo en la Figura 4, este etiquetado se realiza a la entidad y el contexto segunda y cuarta columna respectivamente, y se aplica etiquetado NER al contexto donde se asigna la categoría del significado de la palabra, estas etiquetas se aplican con la librería de Spacy de Python.

	Entidad	EntidadPOS	Contexto	ContextoPOS	ContextoNER	TipoEntidad
0	sistema vascular	sistema NOUN vascular VERB	El @ent es la red de vasos	El DET es AUX la DET red NOUN de ADP vasos NOUN	El-> es-> la-> red-> de-> vasos->	Concept
1	red	red NOUN	El sistema vascular es la @ent de vasos sanguí...	El DET sistema NOUN vascular VERB es AUX la D...	El-> sistema-> vascular-> es-> la-> de-> vaso...	Predicate

Figura 4. Ejemplo de los datos etiquetados POS y NER.

3.3 Extracción de características

Durante esta fase, se realiza la extracción de características, considerando tanto la información morfológica como la información semántica.

3.3.1 Características morfológicas

A continuación, a manera de ejemplo, se presentan los identificadores más comunes utilizados para el etiquetado POS en español, junto con su respectiva descripción:

ADJ: Adjetivo
ADP: Adposición
ADV: Adverbio
AUX: Verbo auxiliar
CONJ: Conjunción
DET: Determinante
NOUN: Sustantivo
NUM: Número
PART: Partícula
PRON: Pronombre
PROPN:Nombre propio
PUNCT: Puntuación
SCONJ:Conjunción subordinante
VERB: Verbo

Las características morfológicas se derivan del etiquetado POS. En este caso, para obtener la matriz de vectores numéricos, se realiza un conteo de frecuencias de las etiquetas asignadas a cada categoría gramatical de las palabras. En la Figura 5 se muestra un ejemplo ilustrativo de este proceso aplicado en las entidades y en la Figura 6 para el contexto.

	Entidad	EntidadPOS	EPOS
0	sistema vascular	sistema NOUN vascular VERB	NOUN VERB
1	red	red NOUN	NOUN

	adj	adp	adv	aux	cconj	det	noun	num	pron	propn	punct	sconj	verb	Indice	TipoEntidad
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	Concept
1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	Predicate

Figura 5. Ejemplo de extracción de características morfológicas de la entidad.

	Entidad	Contexto	ContextoPOS	POS
0	sistema vascular	El @ent es la red de vasos	El DET es AUX la DET red NOUN de ADP vasos NOUN	DET AUX DET NOUN ADP NOUN
1	red	El sistema vascular es la @ent de vasos sanguí...	El DET sistema NOUN vascular VERB es AUX la D...	DET NOUN VERB AUX DET ADP NOUN ADJ ADP NOUN P...

	adj	adp	adv	aux	cconj	det	intj	noun	num	part	pron	propn	punct	sconj	space	verb	Indice	TipoEntidad
0	0	1	0	1	0	2	0	2	0	0	0	0	0	0	0	0	0	Concept
1	1	2	0	1	0	2	0	3	0	0	0	0	1	0	0	1	1	Predicate

Figura 6. Ejemplo de extracción de características morfológicas del contexto.

3.1.2 Características semánticas.

Para la extracción de características semánticas, se usó el etiquetado NER, esta tarea fue realizada con la librería *Spacy* de *Python* y las etiquetas principales son las que se muestran en la Tabla 2.

Tabla 2. Descripción de etiquetas NER.

Etiqueta NER	Descripción
loc	Ubicación geográfica
misc	Varios
org	Organizaciones y empresas
per	Nombres de personas

Un ejemplo de la extracción de estas características fue aplicado al contexto y se puede visualizar en la Figura 7.

	Entidad	Contexto	ContextoNER	NER	TipoEntidad	Indice
0	sistema vascular	El @ent es la red de vasos	El-> es-> la-> red-> de-> vasos->		Concept	0
1	red	El sistema vascular es la @ent de vasos sanguí...	El-> sistema-> vascular-> es-> la-> de-> vaso...		Predicate	1
2	vasos sanguíneos	vascular es la red de @ent del cuerpo.	vascular-> es-> la-> red-> de-> del-> cuerpo-...		Concept	2
3	cuerpo	red de vasos sanguíneos del @ent .	red->MISC de->MISC vasos->MISC sanguíneos->MI...	MISC MISC MISC MISC	Concept	3

	loc	misc	org	per	Indice	TipoEntidad
0	0	0	0	0	0	Concept
1	0	0	0	0	1	Predicate
2	0	0	0	0	2	Concept
3	0	4	0	0	3	Concept

Figura 7. Ejemplo de extracción de características semánticas.

3.4 Aplicación de algoritmos de Aprendizaje Automático

Las características extraídas se transforman en una matriz de vectores numéricos, la cual es utilizada como entrada para los algoritmos de aprendizaje automático. En este trabajo se emplean algoritmos (Aggarwal, 2018) de naturaleza diversa, aplicados mediante la librería *scikit-learn* de Python descritos a continuación.

3.4.1 Vecino más cercano

El algoritmo del vecino más cercano (*K-Nearest Neighbors*, *k*-NN por sus siglas en inglés) es un algoritmo de aprendizaje automático supervisado que se utiliza para la clasificación y regresión de datos. En la clasificación, se basa en un conjunto de datos de entrenamiento con etiquetas conocidas, para asignar una etiqueta a una nueva instancia. Esto se logra al considerar las etiquetas de las *k* instancias más cercanas en el conjunto de entrenamiento.

En el caso específico del algoritmo *k*-NN, se ha implementado un valor de $k=7$, lo que significa que se consideran las etiquetas de las siete instancias más cercanas para realizar la clasificación. El valor de *k* puede ajustarse según las necesidades del problema y naturaleza de los datos.

3.4.2 Máquinas de soporte vectorial (SVM)

El algoritmo de Máquinas de Soporte Vectorial (*Support Vector Machines*, SVM) es un algoritmo de aprendizaje automático supervisado utilizado para la clasificación y regresión de datos. SVM es capaz de realizar clasificación al encontrar el hiperplano óptimo que mejor separa los puntos de datos en diferentes clases. Los parámetros configurados para este trabajo fue el parámetro de regularización =1.0, con un kernel = 'rbf' (kernel gaussiano) y el grado del *kernel degree =3* del método de la librería *SVC*().

3.4.3 Árboles de decisión

En un árbol de decisión, cada nodo interno representa una característica o atributo, y las ramas que salen de ese nodo representan los posibles valores o categorías de esa característica. Los nodos hoja representan las clases o valores de salida. El proceso de construcción del árbol implica dividir el conjunto de datos en función de los atributos para lograr la mejor separación de las clases o la mayor reducción en la incertidumbre, con los valores default del clasificador llamado *DecisionTreeClassifier*().

3.4.4 Perceptrón

El Perceptrón es un modelo de neurona artificial que toma varias entradas ponderadas y las combina utilizando una función de activación. En el código desarrollado, no se especificaron los parámetros utilizados, lo que implica que se utilizan los valores predeterminados del método *Perceptron*(). Estos valores predeterminados incluyen a*lpha= 0.0001, max_iter=1000, eta0=1.0, tol = 1e-3, suffle= True*.

3.5 Evaluación

La exactitud es la métrica que se utiliza en este trabajo para evaluar los algoritmos y se basa en el porcentaje de valores clasificados como correctos respecto al total de elementos. A partir de la matriz de confusión se obtiene la exactitud con la Fórmula 1.

$$Exactitud(Accuracy) = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (1)$$

donde; TP= Verdaderos Positivos, TN=Verdaderos Negativos, FP=Falsos Positivos, FN=Falsos Negativos

4 Experimentación y resultados

Se llevó a cabo una configuración experimental con el propósito de comparar y evaluar diferentes algoritmos y determinar la mejor evaluación. Las configuraciones utilizadas fueron las siguientes:

1. Variando las características: Extraer características morfológicas a la entidad, y al contexto. Obtener características semánticas al contexto y finalmente uniendo todas las características.
2. Los algoritmos que se aplicaron para en la experimentación fueron: Vecino más cercano *k*-NN, Máquinas de Soporte Vectorial, Árboles de decisión y perceptrón.

En la Tabla 3 se presentan los resultados, obteniendo un promedio a dos decimales de la exactitud por experimento y por algoritmo. Además, esta información se ilustra gráficamente en la Figura 8.

Tabla 3. Resultados experimentales.

Algoritmo	Entidad características morfológicas	Contexto características morfológicas	Contexto características semánticas	Características morfológicas Entidad + características morfológicas contexto + características semánticas Contexto	Total por algoritmo
k-NN	78.08	67.36	63.4	77.62	71.61
SVM	79.72	69.23	63.4	**83.21**	**73.89**
Árboles de Decisión	79.72	55.24	63.4	69.69	67.01
Perceptrón	78.78	42.89	57.1	74.82	63.39
Total por experimento	**79.07**	58.68	61.82	76.33	

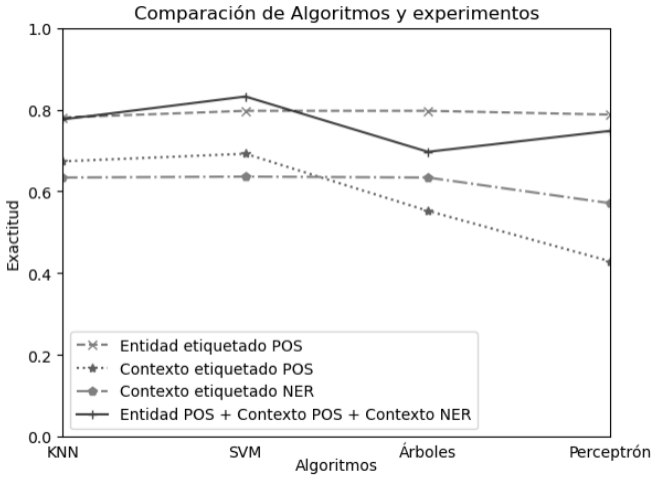


Figura 8. Gráfica de resultados por experimento.

En cuanto al estado actual de la investigación en el campo de la extracción de información en español, se ha identificado la existencia de un amplio espacio para futuros avances. Esto se debe a que la mayoría de los trabajos relevantes se encuentran escritos en inglés, lo que limita la disponibilidad de recursos y modelos en español. Además, se ha observado que en la mayoría de los trabajos en español es necesario construir el conjunto de datos desde cero, lo que implica un esfuerzo adicional en comparación con el uso de conjuntos de datos preexistentes.

En términos generales, los trabajos actuales se centran en la comparación y evaluación de métodos de aprendizaje automático y aprendizaje profundo para la extracción de información. Al revisar la literatura, se encontraron trabajos de la competencia donde se usó el mismo conjunto de datos que se está usando en este trabajo. Por ejemplo, el estudio de Yang (2021) logró obtener una medida *F_1* de 0.56 utilizando un modelo de aprendizaje profundo BiLSTM-CRF, lo que demuestra el potencial de estos enfoques en español.

Otro trabajo relevante es el realizado por (Monteagudo-García et al, 2021). En su investigación, también utilizaron métodos de aprendizaje profundo y el conjunto de datos en español. Lograron alcanzar una medida *F_1* de 0.338, lo que aporta más evidencia sobre la utilidad y los desafíos específicos de la extracción de información en nuestro idioma.

En resumen, existe una necesidad significativa de expandir la investigación en la extracción de información en español, dada la predominancia de trabajos en inglés y la necesidad de construir conjuntos de datos en español. Los resultados de investigaciones anteriores, como las mencionadas, indican un camino prometedor para futuros avances en este campo.

5 Conclusiones y trabajo a futuro

En este trabajo se ha desarrollado un método para la identificación de entidades en textos que contienen sentencias relacionadas a la salud en español. El tratamiento que se le ha dado a estos textos ha sido mediante tareas de Procesamiento de Lenguaje Natural, incluye la tokenización y etiquetado POS y NER. Esto permite realizar un conteo de frecuencias para la extracción de características morfológicas y semánticas relevantes. Como resultado, se obtiene una matriz de vectores que sirve como entrada para los algoritmos de Aprendizaje Automático.

El método propuesto ha contribuido al campo al considerar una comparación y combinación de características morfológicas y semánticas. Además, se ha aplicado una variedad de algoritmos de aprendizaje automático como *k*-NN, SVM, Árboles de Decisión y Perceptrón para obtener resultados más robustos. Esta aportación permite explorar diferentes enfoques y aprovechar las fortalezas de cada algoritmo para mejorar la identificación de entidades en textos relacionados con la salud en español.

Los resultados promedio indican que el algoritmo de Máquinas de Soporte Vectorial obtuvo la mayor exactitud con 73.89%. En cuanto a los experimentos relacionados con las características aplicadas, se observaron resultados alentadores para la entidad con una exactitud promedio de 79.07%. Además, se ha demostrado que la unión de las características morfológicas aplicadas a la entidad, las características morfológicas del contexto y las características semánticas del contexto, utilizando el algoritmo de Máquinas de Soporte Vectorial se obtuvieron los mejores resultados con una exactitud de 83.21%.

Como trabajo a futuro, la explotación de las características sintácticas y la aplicación de algoritmos de aprendizaje profundo para mejorar la extracción de conceptos representa una línea de investigación abierta. Además, es deseable considerar la extracción de relaciones entre los conceptos clínicos, lo cual enriquece aún más el análisis de los textos en lenguaje médico en español.

Referencias

IberLEF (2021). *eHealth Knowledge Discovery Challenge 2021*. Recuperado de https://ehealthkd.github.io/2021/resources

Piad-Morffis, A., Guitérrez, Y., Estevez-Velarde, S., & Munoz, R. (2019). “A general-purpose annotation model for knowledge discovery: Case study in spanish clinical text”, *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pp. 79-88.

Secretaría de Salud (2023). *Informes Semanales para la Vigilancia Epidemiológica 2023*. Recuperado de

https://www.gob.mx/salud/acciones-y-programas/informes-semanales-para-la-vigilancia-epidemiologica-2023

Dey, T., Dey, J., Ghosh, A., Ghosh, S. K., Majumder, M., Mondal, S., & Ghosh, A. (2022). “Name Entity Recognition on Covid-19 Dataset using Machine Learning algorithms”, *2022 International Conference on Machine Learning, Computer Systems and Security (MLCSS)*, pp. 301-306.

Zhang, Y.-C., Lee, C.-H., Liang, T.-Y., Chung, W.-C., Li, K.-H., Huang, C.-C., Dai, H.-J., Wu, C.-S., Kuo, C.-J., Su, C.-H., & Yang, H.-C. (2019). “Depressive Symptoms and Functional Impairments Extraction From Electronic Health Records”, *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, pp. 1-6.

Dellanzo, A., Cotik, V., Lozano Barriga, D. Y., Mollapaza Apaza, J. J., Palomino, D., Schiaffino, F., & Ochoa-Luna, J. (2022). “Digital surveillance in Latin American diseases outbreaks: information extraction from a novel Spanish corpus”, *BMC bioinformatics*, vol. 23, pp. 1-22.

Al-Smadi, M., Al-Ayyoub, M., Jararweh, Y., & Qawasmeh, O. (2019). “Enhancing aspect-based sentiment analysis of Arabic hotels’ reviews using morphological, syntactic and semantic features”, *Information Processing & Management*, vol. 56, pp. 308-319.

Kaur, A., & Khattar, S. (2021). “A systematic exposition of Punjabi Named Entity Recognition using different Machine Learning models”, *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*,pp. 1625-1628.

Wei, Q., Ji, Z., Li, Z., Du, J., Wang, J., Xu, J., Xiang, Y., Tiryaki, F., Wu, S., Zhang, Y., Tao, C., & Xu, H. (2020). “A study of deep learning approaches for medication and adverse drug event extraction from clinical text”, *Journal of the American Medical Informatics Association*, vol. 27, pp. 13-21.

Wu, J., Zhang, R., Gong, T., Liu, Y., Wang, C., & Li, C. (2021).” Bioie: Biomedical information extraction with multi-head attention enhanced graph convolutional network”, *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM),* pp. 2080-2087

Jouffroy, J., Feldman, S. F., Lerner, I., Rance, B., Burgun, A., & Neuraz, A. (2021). “Hybrid Deep Learning for Medication-Related Information Extraction From Clinical Texts in French: MedExt Algorithm Development Study”, *JMIR Medical Informatics*, vol. 9, e17934

Kumar, S., G. Sastry, H., Marriboyina, V., Alshazly, H., Ahmed Idris, S., Verma, M., & Kaur, M. (2022). “Semantic Information Extraction from Multi-Corpora Using Deep Learning”, *Computers, Materials & Continua*, vol. 70, pp. 5021-5038.

Sanchez-Graillet, O., Witte, C., Grimm, F., & Cimiano, P. (2022). “An annotated corpus of clinical trial publications supporting schema-based relational information extraction”, *Journal of Biomedical Semantics*, vol. 13, pp. 1-18.

Aggarwal, C. C. (2018). “Machine Learning for Text”, *Springer International Publishing*, vol. 848

Yang, M. (2021). “Yunnan-1 at eHealth-KD Challenge 2021: Deep-Learning Methods for Entity Recognition in Medical Text”. In IberLEF@ SEPLN (pp. 725-730).

Monteagudo-Garcıa, L., Marrero-Santos, A., Fernández-Arias, M. S., & Canizares-Dıaz, H. (2021). Uh-mmm at ehealth-kd challenge 2021. In Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2021).

Capítulo 4

Descubrimiento de tópicos para la detección de depresión utilizando una red neuronal convolucional

Ana Laura Lezama Sánchez[1], Mireya Tovar Vidal[1], José A. Reyes Ortiz[2], Meliza Contreras González[1]
[1] Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación, Av. San Claudio y 14 sur, Ciudad Universitaria, 72592 Puebla, Pue., México
ana.lezamasanchez@viep.com.mx, mireya.tovar@correo.buap.mx, mel_22281@hotmail.com
[2] Departamento de Sistemas, Universidad Autónoma Metropolitana, Azcapotzalco 02200, México
jaro@azc.uam.mx

Resumen. En los últimos años las redes sociales se han convertido en un medio de comunicación para diferentes personas de cualquier edad. La comunicación en estos medios digitales es posible compartiendo fotos, videos o mensajes. En los cuales es posible compartir diferentes anécdotas, sentimientos, avisos, conocimientos, opiniones o estados de ánimo. Sin embargo, algunos usuarios expresan en redes sociales sentimientos, ideas u opiniones que pueden expresar si se encuentran felices, tristes, en algún tipo de problema o deprimidos. La depresión es uno de los problemas de salud que se manifiesta en estos medios de comunicación. En la literatura existen investigaciones previas que demuestran que el lenguaje que emplean los personas refleja su salud mental. Por tal motivo, existen investigaciones que se centran en aplicar técnicas de procesamiento del lenguaje natural, aprendizaje profundo o aprendizaje automático para predecir si una oración fue escrita por una persona con depresión. Por lo que en este trabajo se propone un modelo que emplea tópicos descubiertos con los algoritmos Análisis de *Dirichlet* Latente (*LDA*), Análisis Semántico Latente Probabilístico (*LSA*), Análisis Semántico Latente (*LSA*), Modelo de Temas Bi-término (*BTM*) y explicación de la correlación (*CorEx*); el propósito es clasificar los tópicos descubiertos y de esta manera predecir si los tópicos pertenecen a tópicos sobre depresión o sin depresión. Los mejores resultados fueron al aplicar como datos de entrada los tópicos descubiertos con el algoritmo *LSA* con una precisión de 0.94, *accuracy* de 0.90, medida F_1 de 0.9 y un *recall* de 0.85.

Palabras Clave: Descubrimiento de Tópicos, Aprendizaje Profundo, Depresión.

1 Introducción

La depresión es un tipo de alteración del ánimo. El cual es consistente en la disminución con un grado variable de perdida de interés o dificultad para experimentar placer en las actividades habituales y acompañado de diversos síntomas como tristeza, alteraciones de la concentración, anorexia-hiperfagia y memoria (Retamal, 2016). La cual puede ser vista desde un enfoque clínico, ya que puede ser definida como un trastorno de la personalidad de carácter afectivo, cognitivo y comportamental (Ahmad et al., 2020). También desde un

punto de vista conductual, donde la depresión es vista como la consecuencia de la falta de refuerzo o de la no contingencia entre conducta y refuerzo. Por otro lado, la teoría cognitiva concibe la depresión como resultado de pensamientos inadecuados, distorsionando el individuo la realidad de forma negativa. Los criterios de diagnóstico varían, aunque generalmente incluyen la presencia de síntomas y una actividad disfuncional en diversas áreas como el trabajo, las relaciones interpersonales y familiares (Woody et al., 2017).

La depresión tiende a ser más frecuente en las personas de mayor edad, en aquellos que padecen enfermedades somáticas crónicas o graves y en las mujeres.

La Organización Mundial de la Salud (OMS) señala que la depresión es la principal causa de deterioro en la salud mental y afecta a 121 millones de personas en el mundo (Woody et al., 2017).

En la década de los 70, las investigaciones sobre la depresión partían de enfoques psicológicos explicativos de la depresión o bien procedían del psicoanálisis o se basaban en las teorías del aprendizaje. La crisis del conductismo y su sustitución por un paradigma cognitivista influyó en los modelos teóricos que trataron de dar cuenta de la aparición de la depresión. Estos iniciaron a finales de 1960, enfatizando en que algunos factores de naturaleza mental o cognitiva juegan un papel importante en el origen de la depresión. En el ámbito de la psicoterapia, un enfoque relevante fue el de Beck que se basó en la hipótesis de que existe una relación causal unidireccional entre el sistema de creencias de la persona, sus afectos y emociones. Para Beck la base de la depresión se encuentra en un triple déficit en el sistema de creencias que lleva a la persona a percibirse a sí misma, al mundo y al futuro en términos negativos (Woody et al., 2017).

En la literatura se describe la complejidad relacionada para identificar la depresión a través de las plataformas de redes sociales, empleando características para el empleo de enfoques de aprendizaje automático (Ahmad, Hussain et al., 2020)

Las características tienen un papel clave para contribuir a lograr resultados positivos. En el estado del arte existen diferentes procedimientos para la extracción de características semánticas para la detección de depresión en textos cortos, sin embargo, no garantizan una correcta detección de la enfermedad. Por lo que, se evalúa la posibilidad de descubrir los tópicos presentes en un corpus compuesto por textos cortos con temas sobre depresión. Los tópicos resultantes serán los datos de entrada a una red neuronal convolucional ya que dichos tópicos aportarán la información central del corpus. Por lo tanto, el resultado esperado es identificar que oraciones o mensajes cortos presentan señales de depresión.

En genera el objetivo principal de este trabajo es identificar la depresión empleando algoritmos de descubrimiento de tópicos y una red neuronal convolucional con un modelo de incrustación de relaciones semánticas.

El resto del artículo está organizado como sigue: en la sección 2 se exponen los conceptos relevantes que dan soporte a esta investigación. En la sección 3 se exponen algunos de los trabajos existentes en la detección de depresión en textos. La sección 4 expone la metodología propuesta para la detección de depresión en textos provenientes de una red social, mientras que en la sección 5 se exponen los resultados obtenidos. Las conclusiones y los trabajos a futuro son expuestos en la sección 6. Finalmente se exponen las referencias consultadas en el desarrollo de este trabajo.

2 Marco teórico

En esta sección se introducen algunos conceptos relevantes que proporcionan soporte a esta investigación. La definición de descubrimiento de tópicos, algunos de los algoritmos empleados y la métrica de evaluación para medir el rendimiento de cada algoritmo se exponen en esta sección. Así como la clasificación de textos, el procesamiento del lenguaje natural, el aprendizaje profundo y las métricas empleadas para evaluar el rendimiento de la detección de depresión. El estudio del significado de las palabras y la forma en cómo se relacionan es una tarea del Procesamiento del Lenguaje Natural (PLN).

El PLN tiene cuatro niveles de estudio del lenguaje humano, uno de ellos es el nivel semántico. El objetivo es descubrir asociaciones entre palabras que permitan definir palabra por palabra el significado implícito de cada frase y que se utilicen en un mismo contexto para dar una idea completa. En la literatura, el descubrimiento de tópicos es definido como área de estudio del PLN y se basa en el descubrimiento de la idea principal de un texto. Algunos de los algoritmos para el descubrimiento de tópicos son el Análisis Latente de *Dirichlet* (*LDA*), el Análisis Semántico Latente (*LSA*), el Análisis Semántico Latente Probabilístico (*PLSA*), Modelo de Temas Bi-término (*BTM*) y explicación de la correlación (*CorEx*).

En los algoritmos *LSA*, *LDA* y *PLSA*, los datos de entrada son transformados en una matriz de tipo documento-término y posteriormente *PLSA* aplica un modelo estadístico, *LDA* y *LSA* la descomposición de valores singulares. Sin embargo, *BTM* modela patrones de co-ocurrencias palabra-palabra. Por otro lado, *CorEx* emplea "palabras de anclaje" que ejemplifican los tópicos potenciales que el modelo podría estar buscando para llevar a cabo el descubrimiento de los tópicos. Para evaluar el rendimiento de un algoritmo de descubrimiento de tópicos se emplea la métrica de coherencia del tópico normalizada expuesta en la ecuación 1

$$CohN(t_i) = \frac{2}{k(k-1)} \Sigma_{j=2}^{k} \Sigma_{\mathrm{i}=1}^{j-1} \frac{\log \frac{P(w_i, w_j)}{P(w_\mathrm{i})P(w_j)}}{-\log P(w_i, w_j)} \qquad (1)$$

Donde:

k= palabras que forman al tópico

$P(w_\mathrm{i})$ es la probabilidad de que la palabra (w_i) aparezcan en una ventana de texto

$P(w_j)$ es la probabilidad de que la palabra (w_j) aparezcan en una ventana de texto

$P(w_i, w_j)$ es la probabilidad de que (w_i) y (w_j) aparezcan juntas

La clasificación de texto surge ante la necesidad de proporcionar al usuario la información que este requiere de forma rápida y precisa existente en grandes volúmenes de información. En la literatura existen algoritmos que realizan la tarea de clasificación de

texto tales como máquinas de soporte vectorial, arboles de decisión, vecinos más cercanos. Estos algoritmos tienen la particularidad que se le deben de proporcionar determinadas características para que el algoritmo proporcione resultados. Si bien son algoritmos que aportan conocimiento e información relevante no son capaces de proporcionar resultados con la misma precisión que los proporcionados por un modelo de aprendizaje profundo; ya que este último no emplea pesados de palabras como *tf-idf* o bolsa de palabras, en su lugar emplea modelos de incrustación de palabras (Viera, 2017). La clasificación de textos se puede llevar a cabo con técnicas de aprendizaje profundo. Los conjuntos de datos deben ser etiquetados manualmente por un experto. Por otro lado, una ventaja importante del aprendizaje profundo es que los resultados obtenidos tienen una mayor precisión que los proporcionados por un clasificador tradicional. Aunado al uso de modelos de incrustación de palabras en los últimos años han surgido los modelos de incrustación de palabras que proporcionan una representación de los textos más precisa. Algunos de los modelos existentes son *glove, word2vec, fastText* (Ucelay et. al., 2021) basados en los modelos *CBOW* y *SKIPGRAM*. Además, existen modelos que se basan en factorización de matrices como en (Saedi et al., 2018) y (Lezama-Sánchez et al., 2022) donde exponen modelos de incrustación de relaciones semánticas.

El aprendizaje profundo es un proceso que se puede llevar a cabo con Redes Neuronales Convolucionales (*CNN*) que han sido adoptadas para tareas de clasificación de textos, generando resultados exitosos (Li et al., 2021). La *CNN* se construye apilando múltiples capas de características. Una capa se compone de *K* filtros lineales y una función de activación (Li et al., 2021). Este tipo de red se distingue por el hecho de que los pesos de la red se comparten entre diferentes neuronas en las capas ocultas (Li et al., 2021). Cada neurona en la red primero calcula una combinación lineal ponderada de sus entradas. La cual se puede visualizar como la evaluación de un filtro lineal en los valores de entrada (Li et al., 2021). Para evaluar el rendimiento de un modelo de clasificación de textos las métricas empleadas son precisión, *accuracy* (exactitud), *recall* (exhaustivdad) y medida-F_1 presentadas en las ecuaciones 2, 3, 4 y 5 respectivamente.

$$Precisión=\frac{Términos\ relevantes\ recuperados}{Términos\ recuperados} \quad (2)$$

$$Exactitud=\frac{Cantidad\ de\ casos\ correctos}{Total\ de\ casos} \quad (3)$$

$$Exhaustividad=\frac{Términos\ relevantes\ recuperados}{Términos\ recuperados} \quad (4)$$

$$Medida - F_1=2*\frac{Precisión*exhaustividad}{Precisión+exhaustivida} \quad (5)$$

3 Trabajos relacionados

En esta sección se exponen los trabajos relacionados en el mismo campo. Algunos autores incorporaron algoritmos para la clasificación de los textos como arboles de decisión, máquinas de soporte vectorial o redes neuronales convolucionales o *BiLSTM*. Sin embargo, para todos los autores el objetivo es descubrir la depresión en los textos empleados en sus experimentos.

En (Ahmad et al., 2020) emplea la técnica de aprendizaje profundo basada en el método *BiLSTM* para detectar si existe depresión en un texto de un conjunto de datos de tipo público. El modelo propuesto lo evaluaron con las métricas de precisión (89%), *recall* (91%), *accuracy* (93%) y una medida F_1 de 90%. No aplicaron un modelo de incrustación de palabras como representación de los datos. Sin embargo, los textos empleados provienen de la red social *Twitter*.

En (Siddiqua et al., 2023) exponen un modelo para detectar depresión en estudiantes de Bangladesh. Los autores aplicaron diez modelos de aprendizaje automático y dos de aprendizaje profundo entre ellos una *CNN* para predecir 3 clases de depresión (normal, moderada y extrema). Los resultados fueron evaluados con la métrica de precisión obteniendo un 92.3% con la *CNN*.

En (Amram et al., 2023) exponen un enfoque para la detección y clasificación de crisis mentales en textos literarios. Los autores emplean técnicas de análisis de texto, para analizar los datos de seis novelas escritas por escritores de la diaspora afgana y pakistaní. El propósito era identificar y clasificar los tópicos y sentimientos relacionados con la depresión en las narrativas. Por lo que emplearon 4 algoritmos para el descubrimiento de tópicos *LDA*, *LSA*, proceso de *Dirichlet* jerárquico (*HDP*) y la factorización de matriz no negativa (*NMF*). Además, aplican una técnica basada en reglas para el análisis de sentimientos utilizando las bibliotecas *VADER* y *TextBlob*. Para la clasificación de la depresión emplearon arboles de decisión, *Naïve Bayes*, *K-Nearest Neighbor* y máquinas de soporte vectorial. Los resultados obtenidos señalaron que *HDP* obtiene una precisión alta en la clasificación con 0.79.

En (Zogan et al., 2021) exponen un modelo para la detección automática de depresión que selecciona contenido existente en tweets. Para la clasificación de los tweets los autores implementaron una red jerárquica de aprendizaje profundo que fusiona varias capas completamente conectadas. Los autores propusieron descartar oraciones con ciertas palabras predefinidas u oraciones dentro de una cierta longitud como resultado el modelo propuesto aprende más atributos discriminativos. El modelo propuesto lo evaluaron con las métricas de precisión, *recall, accuracy* y medida F_1.

Los autores en (Tejaswini et al., 2022) exponen un modelo para la detección de depresión en textos provenientes de redes sociales. El diseño de un modelo híbrido de red neuronal de aprendizaje profundo con mejores representaciones de texto llamado “Red neuronal de convolución de texto rápido con memoria a largo plazo”. El conjunto de datos empleado fue extraído de la red social Reddit. Para la representación de los datos emplearon el modelo de incrustación *fastText*. Las métricas empleadas para la evaluación del diseño fueron *recall*, precisión, *accuracy* y medida F_1.

En (Squires et al., 2023) exponen una visión general de las técnicas y metodologías disponibles para los investigadores para la detección, diagnóstico y tratamiento de la depresión; esta revisión encontró evidencia que en el campo de la psiquiatría las técnicas más usadas son aprendizaje profundo y aprendizaje automático. La mayoría de los autores expuestos en este trabajo aplican un modelo de *deep learning* para la detección de la depresión donde obtienen resultados de medida F_1 desde 0.60 hasta 0.99; con diferentes tipos de conjuntos de datos, pero en general son conjuntos en idioma inglés. Además, exponen un resumen de los trabajos orientados a la predicción de la respuesta al tratamiento farmacológico de la depresión; donde se expone la implementación de algoritmos como *random forest*, técnicas de aprendizaje profundo como *MFFN* entre otros. Los autores encontraron evidencia que existe tendencia a implementar modelos de aprendizaje profundo para la detección de depresión obteniendo resultados altos. Sin embargo, señalan las limitaciones actuales de los sistemas de respuesta al tratamiento para tratar la depresión que incluyen tamaños de muestra pequeños.

En (Zogan, Hamad et al., 2022) exponen un modelo para la detección de depresión de múltiples aspectos aplicando una red de atención jerárquica denominada *Multi-aspect Depression Detection Hierarchical Attention Network* (*MDHAN*). El objetivo es la detección automática de usuarios deprimidos en las redes sociales. El modelo emplea características *multi-aspect*, es decir, características textuales, conductuales, temporales y semánticas. Los conjuntos de datos empleados durante el desarrollo de este trabajo son mensajes que se encuentran en la red social *Twitter* y se encuentran en el rango del año 2009 a 2016.

En (Ucelay et al., 2021) exponen un análisis de la relación que existe entre los modelos computacionales actuales que permiten la detección automática de la depresión. Además de la relación existente con propiedades lingüísticas existentes en un texto escrito por personas que padecen la enfermedad. Los autores utilizaron representaciones textuales ya que encontraron evidencia de que forman parte del estado del arte en clasificación de documentos y que cubren aspectos lingüísticos, sintácticos y semánticos. El conjunto de datos empleado fue extraído de la base de datos *Reddit*. Los datos fueron proporcionados por el laboratorio *eRisk* para la predicción temprana de riesgo en la Web. Los autores emplearon medidas como *tf-idf*, bolsa de palabras, modelo de incrustación *word2vec*, *glove* y *BERT* y además clasificaron con *Logistic Regression* y árboles de decisión. El análisis obtuvo como medida-F_1 resultados que van desde 0.43 hasta 0.52.

En este trabajo se expone una aproximación para la detección de depresión en textos pertenecientes a una red social. Para detectar si en un texto se presentan síntomas de depresión se implementó una red neuronal convolucional. Sin embargo, los datos de entrada en dicha red serán tópicos descubiertos con los algoritmos de descubrimiento de tópicos *LSA*, *LDA*, *PLSA*, *BTM* y *CorEx*; el propósito es proporcionar a la red neuronal convolucional textos con mayor contenido semántico.

4 Aproximación propuesta

En esta sección se presenta la aproximación propuesta para la clasificación binaria de textos cortos en depresivos o no depresivos.

La aproximación propuesta se expone en la Figura 1 e incorpora las siguientes fases: pre-procesamiento del corpus obtenido expuesto en la Tabla 1. La base de datos léxica *WordNet* (Fellbaum, 2010) es utilizada con el objetivo de verificar que cada palabra que forma el corpus cuenta con un significado, ya que se trata de un corpus formado por mensajes extraídos de una red social. Posteriormente se aplica un descubrimiento de tópicos existentes en el corpus con los algoritmos Análisis de *Dirichlet* Latente (*LDA*), Análisis Semántico Latente Probabilístico (*PLSA*), Análisis Semántico Latente (*LSA*), Modelo de Temas Bi-término (*BTM*) y explicación de la correlación (*CorEx*). Posteriormente los tópicos extraídos son los datos de entrada a una red neuronal convolucional con un modelo de incrustación de relaciones semánticas (Lezama-Sánchez et al., 2022). Finalmente, se aplican dos procesos de evaluación; el primero basado en la coherencia del tópico normalizada para la evaluación de los tópicos; el segundo para la evaluación de la clasificación se aplicaron las métricas de precisión, *recall*, *accuracy* y medida F_1.

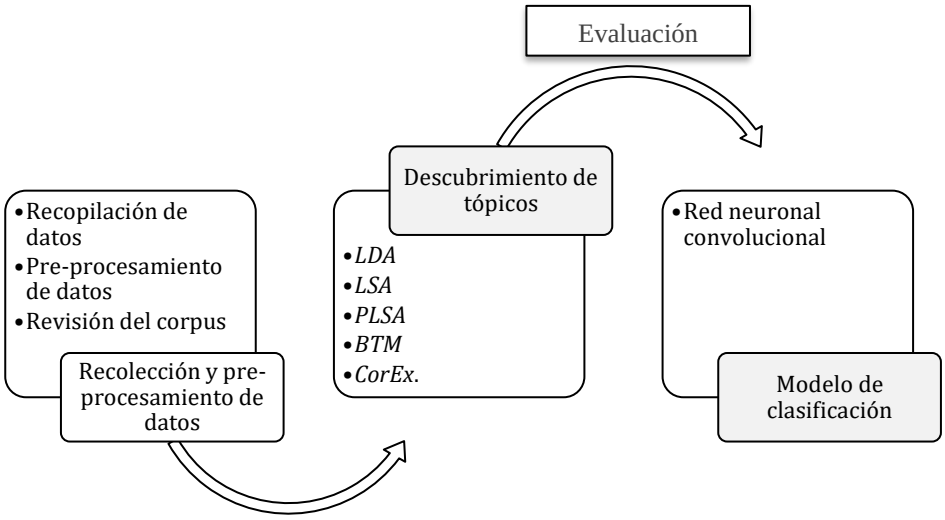


Figura 1. Aproximación propuesta

La aproximación propuesta está compuesta de 4 fases, las cuales están formadas de las siguientes tareas:

- Recolección de datos: Un conjunto de datos donde las opiniones expresadas en una red social etiquetadas como con depresión y sin depresión son el principal criterio para descubrir esta enfermedad en cualquier oración recibida.
- Pre-procesamiento de datos: El conjunto de datos fue sometido a eliminar emoticonos, signos de puntuación, menciones, RT (*Retweet*), números y se convirtieron mayúsculas a minúsculas.
- Revisión del corpus: Se verificó que cada palabra que forma el corpus tuviera contenido semántico. Este proceso se realizó utilizando la base de datos léxica *WordNet*. Es decir, palabras como *zbda*, *awww* y *Ahahahhahahaha*. Sin embargo, en el corpus se mantiene la palabra *covid* a pesar de que en la base de datos léxica *WordNet* esta no figura con significado semántico alguno.
- Descubrimiento de tópicos: Aplicando los algoritmos *LDA, LSA, PLSA, BTM* y *CorEx* se descubren tópicos del corpus utilizado. Por cada algoritmo se extraen 400 tópicos con 10 palabras representativas; 200 tópicos para los textos etiquetados como depresivos o 0 y 200 para los textos etiquetados como no depresivos o 1. Los tópicos son sometidos como datos de entrada a una red neuronal convolucional. El objetivo es proporcionar datos que comparten entre sí similitudes semánticas y por lo tanto en el proceso de clasificación el entrenamiento recibirá datos con fuerte conocimiento que permitirá identificar con más precisión si un texto pertenece a una persona que padece depresión o no. Los tópicos están formados por 10 palabras representativas, pero en este ejemplo se exponen solo 4 palabras representativas en cada uno.
- Evaluación de los tópicos: Los tópicos descubiertos fueron evaluados con la métrica de coherencia del tópico.
- Modelo de clasificación: De forma individual por cada resultado obtenido en el descubrimiento de tópicos los tópicos descubiertos se someten como datos de entrada a una red neuronal convolucional. La red está compuesta de capa de concatenación, capa de convolución, *dropout* de 0.5, *maxpooling1D*, *flatten* y dense. La representación de los datos se realizó aplicando un modelo de incrustación de relaciones semánticas de sinonimia, hiponimia e hiperonimia (Lezama-Sánchez et al., 2022). El número de épocas fue de 70 y un *batchsize* de 128.
- Evaluación de la clasificación: Los resultados obtenidos con la red neuronal convolucional fueron evaluados con la métrica de precisión, *recall, accuracy* y medida F_1.

5 Resultados y discusión

En esta sección se presentan los resultados obtenidos en la clasificación de mensajes con depresión y sin depresión. Además, los resultados obtenidos son comparados con los existentes en la literatura.

5.1 Conjunto de datos

El conjunto de datos fue obtenido de la base de datos *kaggle* (https://www.kaggle.com/datasets/fiorela/data-depresion, descargado el 15 de abril de 2023 expuesta en la Tabla 1) compuesta de 18,721 mensajes. En la Tabla 2 se exponen ejemplos de los mensajes originales que forman parte del corpus sin pre-procesamiento.

Tabla 1. Conjunto de datos

Dominio	Número de mensajes	Número de tokens
Depresión	18,721	175,874

5.2 Resultados y discusión

La aproximación propuesta fue evaluada con 5 métricas. La primera para la evaluación de los tópicos descubiertos fue la coherencia del tópico normalizada para las restantes 4 fueron destinadas a la evaluación de la clasificación realizada con la red neuronal convolucional, es decir precisión, *recall, accuracy* y medida F_1.

Tabla 2. Ejemplo de textos etiquetados con y sin depresión

Mensaje	Clase
RT : I have no life	0 (depresión)
my life is a joke	0
I'm happy and it's where I wanna be	1(no depresión)
RT : omg I'm so in love	1

De los 18,721 documentos se descubrieron 400 tópicos con 10 palabras representativas por cada algoritmo aplicado, es decir en total se descubrieron 2,000 tópicos con 10 palabras representativas cada tópico. La Tabla 3 expone como ejemplo solo 2 tópicos con 4 palabras representativas por algoritmo. Los tópicos están divididos en con depresión (columna con etiqueta 0) y sin depresión (columna con etiqueta 1)

Tabla 3. Ejemplo de los tópicos descubiertos por cada algoritmo

Algoritmo	Tópico 1		Tópico 2	
	0	1	0	1
LDA	Help, need, crazy, roke…	God, happy, love, future…	Unhappy, Covid, Ugly, Boring…	Awesome, dancing, friends, beautiful…
LSA	Life, hate, fuck, boring…	Life, love, good, happy…	Bad, love, worst, feel…	Good, love, life, feel…
PLSA	Sleepy, tormented, hurt, overcome…	Love, life, Atm, lame…	deleted, failure, dirty, pillow…	Done, times, happiness, life…
BTM	Sad, ads, covid, want…	God, blessing, deserve, thanks…	Crying, hating, die, wrong…	True, years, worrying, dreams…
CorEx	Worst, falling, fake, hate…	God, blessing, thank, future…	Died, fair, slow, mirror…	Fabulous, university, glad, forever…

La Tabla 4 expone algunas de las clases recuperadas después de llevar a cabo la clasificación de los tópicos descubiertos por los algoritmos *LDA, LSA, PLSA, BTM y CorEx*. En cada clase se observa que la etiqueta asignada por el modelo de clasificación empleado acierta con la predefinida en los datos originales.

Tabla 4. Ejemplo de la clasificación sobre los tópicos descubiertos.

Mensaje original	Etiqueta original	Etiqueta Descubierta	Algoritmo empleado
unhappy really given brother take championship aaa find falling home	0	0	*LDA*
parents depressed acc sector fight long actually increases guy words	0	0	*LDA*
flop monday info parents follow like sometimes men leave limit	0	0	*LDA*
want know restless feel really like think check time worried	1	1	*LSA*
friend single boyfriend yeah long every afraid truly another	1	1	*LSA*
okay much heart sucks first hard boring take school someone	0	0	*LSA*
using stomach shoulders health photo name wear become example scary	0	0	*PLSA*
beyond wish enough loves imaginations belief fighting nothing simply blessed	1	0	*PLSA*
real started really tiring phase liking effects depressant wear depression	1	1	*PLSA*
anxious calm make future case necessarily slow medicine patient take	1	1	*BTM*
night drinking eating pretzels sprite watching greys friday positive turning	1	0	*BTM*
step mum dad phone smuggles like bottle please love life	1	1	*BTM*
help remember understand need crazy hot turns guard someone person	0	0	*CorEx*
life god feel luck hurry good happy sick loving account	0	0	*CorEx*
god blessing deserve thanks like lazy damn proud called peoples	1	1	*CorEx*

La Tabla 5 expone los resultados obtenidos de la clasificación del corpus. La métrica coherencia del tópico normalizada es utilizada para la evaluación de los tópicos descubiertos. Los resultados se dividen en 1 (sin depresión) y 0 (con depresión). Los resultados obtenidos demuestran que los tópicos descubiertos con los algoritmos *LDA* y *LSA* para mensajes etiquetados con depresión y sin depresión son los que obtuvieron un puntaje mayor. Lo que señala que las palabras que forman los tópicos se encuentran con una relación semántica mayor. Sin embargo, los tópicos descubiertos con el algoritmo *LSA* fueron los que aportaron más información en la tarea de clasificación. Las métricas precisión, *recall, accuracy* y medida F_1 son aplicadas para evaluar el rendimiento de la clasificación realizada. Los resultados obtenidos al aplicar como datos de entrada los tópicos descubiertos con los algoritmos propuestos varían dependiendo del algoritmo utilizado. Se puede observar que los mejores resultados son cuando se utilizan como datos de entrada los tópicos descubiertos con el algoritmo *LSA* obteniendo una precisión y *accuracy* mayor a 0.90 una medida F_1 de 0.9 y un recall de 0.87. Sin embargo, la clasificación realizada con los tópicos descubiertos con el algoritmo *BTM* obtuvo una precisión y *accuracy* y medida F_1 menor a lo obtenido con *LSA*, pero fueron los segundos mejores resultados posicionándose por encima de los resultados obtenidos con los algoritmos *LDA, PLSA* y *CorEx.*

Tabla 5. Resultados obtenidos con los algoritmos de descubrimiento de tópicos y la tarea de clasificación.

Algoritmo para tópicos	Coherencia del tópico		Precisión	*Accuracy*	*Recall*	Medida F_1
	1	0				
LDA	**0.1891**	**0.1777**	0.7187	0.7538	0.7666	0.7419
LSA	**0.1803**	**0.1794**	**0.9473**	**0.9090**	**0.8571**	**0.9**
PLSA	0.1553	0.1367	0.6068	0.6106	0.6008	0.5321
BTM	0.1777	0.1568	**0.9047**	**0.8301**	0.7307	**0.8085**
CorEx	0.1325	0.1475	0.8076	0.7047	0.6666	0.7304

6 Conclusiones y trabajos a futuro

Este artículo presenta una aproximación para la detección de la depresión en textos cortos. La aproximación propuesta inició con el pre-procesamiento del corpus, además de una selección de las palabras con contenido semántico; esto último con ayuda de *WordNet*, además de contemplar la palabra *covid* en el vocabulario aun cuando actualmente *WordNet* no tiene registro sobre el significado de la palabra. Posteriormente se descubrieron 400 tópicos (200 tópicos para textos etiquetados con 0 y 200 para los etiquetados con 1) con 10 palabras representativas presentes en dicho corpus, a través de los algoritmos *LDA, LSA, PLSA, BTM* y *CorEx*. Los tópicos descubiertos fueron el conjunto de datos de entrada a una

red neuronal convolucional que emplea para la representación de los textos un modelo de incrustación de relaciones semánticas de sinonimia, hiponimia e hiperonimia expuesto en (Lezama-Sánchez et al., 2022). Los resultados obtenidos señalaron que clasificar tópicos genera resultados con mayor precisión.

La principal contribución de este artículo es: una aproximación basada en el descubrimiento de tópicos validada en la clasificación de textos; el descubrimiento de tópicos pertenecientes al dominio de depresión; el uso de un modelo de incrustación de relaciones semánticas basada en sinónimos, hipónimos e hiperónimos.

De este modo, los resultados mostraron que seleccionar los tópicos como datos de entrada en lugar del corpus original proporciona más información al clasificador. Se observa que los resultados obtenidos son variables debido a que cada algoritmo para el descubrimiento de tópicos aplica diferentes técnicas estadísticas. En este dominio el *Análisis Semántico Latente* fue el modelo que descubrió tópicos con mayor relación en el dominio estudiado en este trabajo. Además, este enfoque se convierte en un recurso útil en el campo del procesamiento del lenguaje natural ya que cuenta con la capacidad de analizar nuevos datos empleando el modelo previamente obtenido.

Como trabajos a futuro se propone primero aplicar la clasificación de texto y posteriormente descubrir los tópicos, lo que proporcionará tópicos particulares. Por otro lado, se considera que realizar los mismos experimentos con textos en español será una herramienta novedosa ya que en idioma español existe poca investigación en torno a este tipo de enfermedad.

Agradecimientos

Los autores agradecen al Laboratorio Nacional de Supercómputo del Sureste de México (LNS), perteneciente al padrón de laboratorios nacionales CONAHCYT, por los recursos computacionales, el apoyo y la asistencia técnica brindados, a través del proyecto número 202103090C.

Los autores agradecen al Laboratorio de Supercómputo, plataforma de aprendizaje profundo para las tecnologías del lenguaje, INAOE por los recursos computacionales, el apoyo y la asistencia técnica brindados. Al Consejo Nacional de Humanidades de Ciencia y Tecnología (CONAHCYT) con el número de beca 788155 y al proyecto VIEP 2023 en BUAP.

Referencias

Retamal, Pedro., (1998). Depresión. Editorial Universitaria.

Ahmad, Hussain, et al., (2020) "Applying deep learning technique for depression classification in social media text." Journal of Medical Imaging and Health Informatics 10.10 (2020): 2446-2451.

Woody CA, Ferrari AJ, Siskind DJ, Whiteford HA y Harris MG., (2017). A systematic review and meta-regression of the prevalence and incidence of perinatal depression. J Affect Disord. 2017;219:86–92.

Li, Zewen, et al. (2021). "A survey of convolutional neural networks: analysis, applications, and prospects." IEEE transactions on neural networks and learning systems (2021).

Fellbaum, Christiane. "WordNet". (2010). Teoría y aplicaciones de la ontología: aplicaciones informáticas . Dordrecht: Springer Holanda, 2010. 231-243.

Siddiqua, Rokeya, et al. "AIDA: Artificial intelligence based depression assessment applied to Bangladeshi students." Array 18 (2023): 100291.

Amram, Nur Anis Liyana Mohd, Pantea Keikhosrokiani, y Moussa Pourya Asl. (2023). "Artificial intelligence approach for detection and classification of depression among refugees in selected diasporic novels." Social Sciences & Humanities Open 8.1 (2023): 100558.

Zogan, Hamad, et al. (2021). "Depressionnet: A novel summarization boosted deep framework for depression detection on social media." arXiv preprint arXiv:2105.10878 (2021).

Tejaswini, Vankayala, Korra Sathya Babu, and Bibhudatta Sahoo. (2022). "Depression Detection from Social Media Text Analysis using Natural Language Processing Techniques and Hybrid Deep Learning Model." ACM Transactions on Asian and Low-Resource Language Information Processing (2022).

Squires, Matthew, et al. (2023). "Deep learning and machine learning in psychiatry: a survey of current progress in depression detection, diagnosis and treatment." Brain Informatics 10.1 (2023): 1-19.

Zogan, Hamad, et al. (2022). "Explainable depression detection with multi-aspect features using a hybrid deep learning model on social media." World Wide Web 25.1 (2022): 281-304.

Lezama-Sánchez, Ana Laura, Mireya Tovar Vidal, y José A. Reyes-Ortiz. (2022). "An Approach Based on Semantic Relationship Embeddings for Text Classification." Mathematics 10.21 (2022): 4161.

Viera, Ángel Freddy Godoy. (2017). "Técnicas de aprendizaje de máquina utilizadas para la minería de texto." Investigación bibliotecológica 31.71 (2017): 103-126.

Ucelay, María José Garciarena, Leticia Cecilia Cagnina, y Marcelo Luis Errecalde. (2021). "Análisis de rasgos lingüísticos con técnicas de procesamiento del lenguaje natural en la detección temprana de depresión." Anales de Lingüística 7 (2021): 89-116.

Saedi, Chakaveh, et al. (2018). "Wordnet embeddings." Proceedings of the third workshop on representation learning for NLP. 2018.

Capítulo 5

Análisis de sentimientos aplicados a textos en español utilizando técnicas de PLN y características estadísticas

Mario A. Cruz-Miguel[1], José A. Reyes-Ortiz[1], Leonardo D. Sánchez-Martínez[1]

[1] Universidad Autónoma Metropolitana, Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería

{al2221800024,jaro,ldsm}@azc.uam.mx

Resumen. En los últimos años, las redes sociales han adquirido un papel fundamental como fuente de información relevante, desplazando a los medios de comunicación tradicionales. Estas plataformas permiten visualizar y comprender la opinión social, generando diariamente una gran cantidad de datos en diversos temas. El análisis de esta información se encuentra en constante evolución y sin una organización por tema, con lo cual resulta imposible conocer el sentimiento mostrado por el autor de la información de las redes sociales. Por ello, resulta necesario el uso de herramientas computacionales para el análisis automático basado en sentimientos de este tipo de información. Este análisis del sentimiento de la opinión pública en relación con temas relevantes podría ser de gran utilidad en la toma de decisiones en diferentes ámbitos. En este contexto, este proyecto propone realizar un análisis de sentimientos a partir de mensajes de textos en español, organizados en conjunto de datos obtenidos de la literatura. Este enfoque utiliza técnicas de procesamiento de lenguaje natural, características estadísticas y algoritmos de aprendizaje automático. El proceso completo consiste en realizar un procesamiento de los mensajes, una limpieza y llevar el texto a su forma base por medio de lematización y *stemming* para después aplicar algunos algoritmos de aprendizaje automático como máquinas de soporte vectorial y regresión logística. El mejor resultado obtenido fue de 68.28% de exactitud utilizando el algoritmo *random forest*.

Palabras Clave: Red social, Análisis de sentimientos, Procesamiento de Lenguaje Natural, Aprendizaje Automático.

1 Introducción

En la actualidad, las redes sociales en español generan una enorme cantidad de información sobre diversos temas. Conocer la opinión social resulta de gran interés tanto en el sector de industria como científico, ya que se dispone de un amplio volumen de datos para realizar tareas de análisis. Sin embargo, esta gran cantidad de información no estructurada, en forma de textos, requiere ser analizada, particularmente en el sentimiento expresado por los usuarios. Realizar este análisis de manera manual consume mucho tiempo y resulta en una tarea costosa y tediosa. Por esta razón, resulta indispensable emplear enfoques computacionales utilizando técnicas de Procesamiento de Lenguaje Natural (PLN) y algoritmos de aprendizaje automático para identificar automáticamente el sentimiento presente en estos textos. Esta tarea se le conoce como Análisis de Sentimientos y consiste en clasificar textos de forma automática y por medio de la connotación positiva o negativa con la que hayan sido escritos. Para llevar a cabo este proceso, se utilizan técnicas de PLN y algoritmos de aprendizaje automático. Al adoptar estos enfoques computacionales, podemos abordar de manera eficiente el análisis de una gran cantidad de datos provenientes de redes sociales en español. Esto nos permite obtener información valiosa sobre el sentimiento predominante en los textos y utilizarla para diversas aplicaciones. Por lo tanto, la propuesta de este trabajo consiste en realizar una clasificación a partir de mensajes de textos recopilados de la literatura etiquetados en sentimientos y utilizar técnicas de procesamiento del lenguaje natural para extraer características estadísticas a nivel de palabra, de este modo el estudio descarta por ejemplo emoticones ya que se pretende determinar si este enfoque es adecuado para llevar a cabo un análisis de sentimientos. Estas características y sus ponderaciones se utilizarán como entrada para diversos algoritmos de aprendizaje automático con el fin de clasificar los sentimientos. La evaluación de los resultados se realiza utilizando la métrica de exactitud.

El presente artículo abarca la implementación del análisis de sentimiento, incluyendo la descripción de las secciones que se listan a continuación. En la sección 2 se revisa el estado del arte en análisis de sentimiento con algoritmos de aprendizaje automático; en la sección 3 se muestra el enfoque propuesto y se detalla cada una de las etapas implementadas en el análisis de sentimientos; en la sección 4 se presentan los resultados obtenidos y su comparativa; y finalmente en la sección 5 se exponen las conclusiones alcanzadas y se plantean las perspectivas de la investigación a seguir.

2 Estado del Arte

En esta sección, se presenta una revisión del estado del arte en relación con el análisis de sentimientos utilizando textos de redes sociales en diferentes idiomas. Se hace énfasis en los algoritmos de clasificación utilizados y los resultados obtenidos en cada trabajo.

El artículo (Liu, 2010) es uno de los primeros que aborda el problema del análisis de sentimientos, es mencionada la complejidad de obtener un estudio computacional de este

tipo antes de la llegada de la web, debido a que existía muy poco texto opinable disponible y, sin embargo, a pesar de que en la actualidad existe una gran cantidad de reseñas y sitios de opinión, realizar un análisis de sentimiento es todavía una tarea extraordinaria, se hace énfasis en lo multifacético que es realizar un análisis de sentimientos, pues, hay diversos subproblemas a resolver con una reflexión final sobre lo que se ha hecho en este campo y el futuro que vislumbra la solución de problemas muy particulares con investigaciones más refinadas.

Los autores en (Martínez-Cámara et al, 2011) proponen un análisis de opiniones de películas, los datos se procesaron con *stopper* y *stemming* y se extrajeron características con TF-IDF, TF, número de ocurrencias (TO) y ocurrencia binaria (BTO), se usaron máquinas de soporte vectorial cuyo mejor resultado fue con TF-IDF obteniendo una medida *F_1* de 86.37% y con Bayes usando TF-IDF una medida *F_1* 84.4%.

En el artículo (Plaza-del-Arco et al, 2016), se realizó un análisis de sentimientos en opiniones de atención médica con sentimientos positivo y negativo, en el cual se realizó un procesamiento a los textos que consiste en convertirlos a minúsculas, eliminando acentos y caracteres especiales, también se eliminaron palabras vacías y se aplicó TFI-DF para la extracción de características estadísticas, se realizaron dos experimentos donde el corpus no se encontraba balanceado respecto a las clases, obteniendo una exactitud de 87% y 71.35% en la medida *F_1*, con las clases balanceadas la exactitud fue de 88.5% y la medida *F_1* de 89.71%.

Un análisis de sentimientos enfocado en tweets neutrales fue realizado en (Chiruzzo et al, 2020), donde se utilizaron las técnicas de bolsa de palabras, polaridad de palabras y marcadores de categoría, los algoritmos utilizados obtuvieron como resultados en exactitud de 56% para máquinas de soporte vectorial, 52.2% para una red neuronal convolucional y 52.1% en una memoria corto-largo plazo.

En el estudio llevado a cabo en (Samuel et al, 2020), se llevó a cabo un análisis de sentimientos enfocado en el temor hacia el coronavirus, utilizando mensajes de twitter relacionados con la enfermedad COVID-19, estos tweets fueron descargados del sitio web de *kaggle*, el conjunto de datos de los autores consta de más de 170,000 tweets. Se realizó una clasificación teniendo en cuenta tweets de diferentes longitudes. Los resultados mostraron una alta precisión de clasificación del 91% para tweets cortos (77 caracteres) utilizando el método Naïve Bayes, y del 74% para regresión logística. Sin embargo, ambos métodos mostraron un rendimiento relativamente más débil para los tweets más largos (120 caracteres).

En el artículo presentado por (Raheja y Asthana, 2021) se realizó un análisis de sentimientos con el objetivo de obtener la opinión de la población en la India. En este estudio, se clasificó la opinión en positiva, negativa o no partidista. Se utilizaron tres palabras claves principales: COVID, virus Corona y COVID-19. La metodología utilizada se basó en la subjetividad-objetividad y se hizo un conteo de emojis para definir las emociones. No se mencionó el uso de algún algoritmo de aprendizaje automático o técnica de procesamiento del lenguaje natural.

En (Chintalapudi et al, 2021) se analizan tweets de usuarios en la India durante el periodo COVID-19 del 23 de marzo al 15 de julio de 2020, etiquetando el sentimiento

como miedo, tristeza, ira y alegría. El análisis de los datos se llevó a cabo mediante el modelo BERT y se comparó con los modelos regresión logística, máquinas de soporte vectorial (SVM) y memoria de largo y corto plazo (LSTM), la precisión de cada sentimiento se calculó por separado. La exactitud obtenida fue para el modelo BERT de 89% y los otros tres modelos produjeron un 75%, 75% y 65%, respectivamente.

Un análisis de opinión del servicio de taxi por aplicación Uber y Ola es presentado en (Indulkar y Patil, 2021), donde se probaron los algoritmos regresión logística Bayes Multinomial y *Random Forest* en el conjunto de datos de los autores. El número de tweets extraídos fue de 3000. Se evaluó la exactitud de cada conjunto de textos tomando 500 tweets e incrementando el número de tweets. El mejor algoritmo fue *Random Fores*t donde para el conjunto de textos de Uber la exactitud fue de 96.3% y para el conjunto de textos de Ola fue de 83.4%. En las pruebas se observó que el conjunto textos de Ola obtenía exactitudes muy bajas respecto al conjunto de textos de Uber porque el *dataset* no es preciso y contiene muchas palabras que no se encuentran en lenguaje estándar, lo que dificulta la clasificación de los mismos.

En el artículo (Bhargav et al, 2021), se realizó un análisis de opinión para la clasificación de opiniones satisfactorias de un producto, se realizó un preprocesado que consistió en normalizar a minúsculas, reemplazar dos o más puntos con espacios, eliminar espacios y comillas del final del tweet, reemplazar dos o más espacios con un solo espacio, las url sustituirlas por la palabra url, eliminar signos de puntuación, modificar las palabras que tienen letras repetidas, se realiza la extracción de características por unigramas y bigramas, por conteo de palabras, finalmente se utilizaron los algoritmos con resultados de exactitud para *Random Forest* del 76%, árboles de decisión del 75.89 % y el método propuesto de SVM de 81.97%.

En el estudio llevado a cabo en (Hidayat y Sulistiyono, 2022), se realizó un análisis de opinión de la *Aplicación Peduli Protect* que monitorea el aumento de COVID usando una recopilación de 3,588 datos para sentimientos positivos y negativos. El análisis propuesto consiste en un preprocesado de los datos a minúscula, eliminación de caracteres especiales y tokenización, eliminación de palabras vacías, eliminación de signos de puntuación, aplicación de *stemming* y se aplicó el pesado a los términos con TF-IDF, se usaron los algoritmos Bayes obteniendo una exactitud de 80% usando *K-Fold Cross-Validation* y 85% con *K-Fold Cross Validation on the fold* 3, y máquinas de soporte vectorial obteniendo un 86%.

En el estudio descrito en (Parikh et al, 2022), se llevó a cabo un análisis de sentimientos con un conjunto de 1,600,000 textos y dividido equitativamente en positivo y negativo, posterior fueron aplicadas técnicas de tokenización, lematización y fueron eliminadas las palabras vacías, las características se extrajeron con TF-IDF y se aplicó *Word2Vec*, después se utilizaron algoritmos de aprendizaje profundo para la clasificación y se compararon respecto a Bayes y máquinas de soporte vectorial, estos últimos se concluye obtuvieron mejores resultados.

Este trabajo presenta algunas diferencias en comparación con los estudios revisados en el estado del arte. Se enfoca exclusivamente en el análisis de texto, excluyendo el uso de

emoticones y está restringido al idioma español. Además que está centrado en las técnicas de *stemming* y lematización.

3 Enfoque propuesto

En esta sección se presenta el enfoque propuesto para el análisis de sentimientos a partir de textos en español etiquetados en categorías de sentimientos recopilados de la literatura. Este enfoque se compone de cinco fases, tal como se ilustra en la Figura 1 y, primero, se describen de manera general y más adelante, de manera detallada.

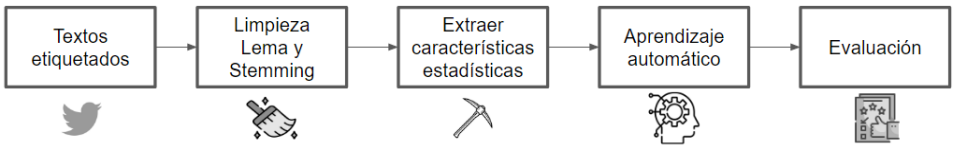


Figura 1. Esquema del enfoque propuesto

Fase 1: Recopilación de textos etiquetados. En esta etapa, se recopilaron diversos conjuntos de textos extraídos de la literatura. A partir de esta selección, se eligieron dos conjuntos de textos llamados *Multilenguaje sentimental* (Qiang, 2023) y *Cardiff* (Barbieri et al, 2022).

Fase 2: Procesamiento de textos. Una vez definidos los conjuntos de textos, se aplicaron técnicas de procesamiento de lenguaje natural. Estas técnicas incluyen la eliminación de palabras vacías, caracteres especiales, url, normalización a minúsculas y separación de palabras en notación *camel*. Además, se realizó el procesamiento de los textos utilizando lematización y *stemming* para llevar las palabras a su forma base.

Fase 3: Extracción de características. En esta fase, los datos ya preprocesados se utilizaron para extraer características. Específicamente, se aplicaron técnicas estadísticas utilizando TF-IDF para generar una matriz de pesos.

Fase 4: Aprendizaje automático. Se emplearon cuatro algoritmos de aprendizaje automático, que incluyen máquinas de soporte vectorial, perceptrón, *random forest* y regresión logística. En esta etapa, los textos procesados de la fase anterior se representaron mediante un modelo de representación vectorial basado en las características extraídas. El objetivo fue clasificar los textos en sentimientos positivos, negativos y neutros.

Fase 5: Evaluación. En la etapa final, se evaluaron los algoritmos utilizados con el fin de medir su desempeño por medio de la métrica exactitud.

A continuación, se presentan las descripciones detalladas de cada fase del enfoque propuesto.

3.1 Recopilación de textos etiquetados

El objetivo de esta fase es obtener dos conjuntos de textos en español extraídos de la literatura con la finalidad de aplicar algoritmos de aprendizaje automático para el análisis de sentimientos.

Se seleccionaron dos conjuntos de textos: *Multilenguaje sentimental y Cardiff.*

Multilenguaje sentimental incluye grupos de mensajes de texto en diferentes idiomas, como inglés, español y árabe, el conjunto de textos completo consta de 270,399 textos del cual solo se enfocó en 1,890 correspondientes al idioma español.

La Figura 2 presenta una muestra de los datos contenidos en el archivo *Multilenguaje sentimental*.

text	label	source	language
estoy hasta el ojete de que me digáis que tengo cara de mala leche	negative	intertass_2017	spanish
@user Por? Tenía pensado verla después de la segunda de Daredevil	neutral	intertass_2017	spanish
Esto de estar feliz mola	positive	intertass_2017	spanish
Ya no es tan divertido	negative	intertass_2017	spanish

Figura 2. Muestra de datos del archivo *Multilenguaje sentimental*

Para el conjunto de textos de *Multilenguaje sentimental* el total de registros por cada etiqueta para el idioma español se muestra en la Tabla 1.

Tabla 1. Textos por clase del idioma español en el archivo *Multilenguaje sentimenta*l

Clasificación	Total de textos
Positivos	613
Negativos	613
Neutros	613

El segundo conjunto de textos es *Cardiff* el cual contiene diferentes idiomas como francés, inglés y español, para este estudio solo se usaron aquellos que corresponden al idioma español. Este archivo cuenta con 3033 textos en español divididos en 3 archivos que contienen el mensaje de los usuarios y 3 archivos que contienen el etiquetado del sentimiento como se muestra en la Tabla 2.

Tabla 2. Textos en cada archivo de *Cardiff*

Archivo	Total de textos
test_text.txt	870
test_labels	870
train_text.txt	1839
train_labels.txt	1839
val_text.tx	324
val_labels.txt	324

Se realizó la unificación de todos los archivos y se aplicó un filtro para seleccionar únicamente los textos en español, la Figura 3 presenta una muestra del conjunto de textos después de este procesamiento donde los valores de la columna "*texto*" continente el mensaje del conjunto de textos y "*etiqueta*" contiene los valores 0 para negativo, 1 para neutral y 2 para positivo.

texto	etiqueta
@user jajajaja dale, hacete la boluda vos jajaja igual a vos nunca se te puede tomar en serio te mando un abrazo desde Perú!	0
cada vez que cito un tweet se va la ubicación sin tampoco poder ponerla en el momento a uds les pasa? TE VEO Y ME PICA VICICONTE	1
@user MAAAAE RAJADO! Pero lo bueno es q uno se va independizandoy logrando metas	2
Bueno hoy fui a almorzar a Nanay con otras 3 dras xq la capacitación mal organizada no nos dió almuerzo y encima nos mandan a comer 2pm	0
Necesito seguir a mas cuentas camren shippers y fans de las armonías. Me recomendais alguna?	1
@user ¡Hola Tomás! ¿Habéis visto los nuevos #dinos de #TierraMagna? Es normal que haya colas antes de que comience el espectáculo	2

Figura 3. Conjunto de Textos *Cardiff* unificando los archivos *test*, *train* y *val*

Para el conjunto de textos del *Cardiff* el total de registros por etiqueta se muestra en la tabla 3.

Tabla 3. Textos por etiquetas en el conjunto de textos de *Cardiff*

Clasificación	Tipo	Total de textos
0	negativo	1011
1	neutro	1011
2	positivo	1011

3.2 Procesamiento de los textos

Esta fase tiene como objetivo preparar los textos para los algoritmos usando el lenguaje *python* y la cual consiste en la limpieza de los datos, aplicación de las técnicas lematización y *stemming* usando las librerías *spacy* y *nltk* respectivamente, y la extracción de características estadísticas usando la técnica TF-IDF por medio de la librería *TfidfVectorizer*, y se describen a continuación.

3.2.1 Limpieza.

Esta tarea se realiza con la finalidad de preparar los textos para etapas posteriores mediante eliminar elementos considerados ruido o que no aportan a la tarea de análisis de sentimientos. A continuación, se detallan las acciones de limpieza realizadas en ambos conjuntos de textos:

- Eliminación de menciones de @usuario.
- Eliminación de caracteres especiales, tales como: ,\.$*¿\?¡!\+\-\[\]\)\(\\\].
- Eliminación de palabras compuestas por un carácter o por más de un carácter repetido.
- Eliminación de emoticones.
- Eliminación de url.

- Eliminación de caracteres especiales, conservando espacios en blanco.
- Eliminación de múltiples espacios en blanco, dejando solo uno.
- Eliminación de números y comas.
- Eliminación de direcciones de correo electrónico.

Se aplicaron también acciones de separación de palabras en notación *camel* para mejorar la legibilidad y comprensión del texto, ya que permite identificar las palabras individuales dentro de una secuencia sin espacios y la normalización a minúsculas para la coherencia y uniformidad de los datos para facilitar.

Finalmente, se emplea la técnica de eliminación de palabras vacías que carecen de relevancia en el texto utilizando la biblioteca *stopwords* de *NLTK*. Esta técnica consiste en identificar y remover palabras comunes como artículos, pronombres y preposiciones que no aportan información significativa al análisis del texto.

Para ambos conjuntos de textos de *Multilenguaje sentimental* y *Cardiff* se realiza esta limpieza, la Figura 4 muestra un ejemplo del texto antes de realizar el procesamiento de limpieza y la Figura 5 muestra el texto después de la limpieza.

estoy hasta el ojete de que me digáis que tengo cara de mala leche	negative
por tenía pensado verla después de la segunda de daredevil	neutral
esto de estar feliz mola	positive
ya no es tan divertido	negative
te recuerdo que soy una persona que tiene criterio equivocado pero lo tengo	neutral

Figura 4. Muestra del conjunto de textos *Multilenguaje sentimental* original

ojete digáis cara mala leche	negative
pensado verla después segunda daredevil	neutral
feliz mola	positive
tan divertido	negative
recuerdo persona criterio equivocado	neutral

Figura 5. Muestra del conjunto de textos *Multilenguaje sentimental* después del proceso de limpieza

3.2.2 Lematización.

La lematización es una técnica lingüística que busca llevar las palabras a su forma base por medio de su forma verbal, esta técnica reduce la variabilidad y facilita el análisis y la comparación de los textos.

Una vez finalizada la etapa de limpieza de los textos, se procede a aplicar un proceso de lematización. Para el conjunto de textos *Multilenguaje sentimental* la Figura 6 muestra el resultado del proceso de lematización aplicado al conjunto de textos donde cada palabra ha sido transformada a su forma base.

ojete,digáis,cara,malo,leche
pensado,ver él,después,segundo,daredevil
feliz,mola
tanto,divertido
recordar,persona,criterio,equivocado

Figura 6. Muestra del conjunto de textos de *Multilenguaje sentimental* después del proceso de lematización

3.2.3 *Stemming*

El proceso de *stemming* es importante ya que permite normalizar las palabras, eliminando afijos, para agrupar variantes de una misma palabra bajo un único término. Esto ayuda a reducir la complejidad del lenguaje y a simplificar el procesamiento de texto. Para el conjunto de textos de *Multilenguaje sentimental* la Figura 7 muestra el resultado del conjunto de textos una vez procesado para llevar las palabras a su forma base usando *stemming*.

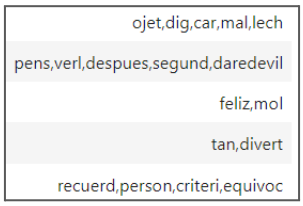


Figura 7. Muestra Conjunto de textos de *Multilenguaje sentimental* después del proceso de *stemming*

3.3 Extracción de características

La fase de extracción de características en este trabajo se basa en el modelo de bolsa de palabras (*bag of words*) utilizando la técnica TF-IDF para la ponderación de las palabras.

TF-IDF (*Term Frequency-Inverse Document Frequency*) es una técnica utilizada para ponderar la importancia de las palabras en un conjunto de documentos. Calcula un valor que combina la frecuencia de una palabra en un documento (TF) con la frecuencia inversa de esa palabra en el conjunto de documentos (IDF). El TF mide qué tan frecuente es una palabra en un documento específico, mientras que el IDF mide qué tan rara es una palabra en el conjunto de documentos. El resultado es un valor que refleja la importancia relativa de una palabra en un documento específico y en todo el conjunto de documentos.

Para el conjunto de textos de *Multilenguaje sentimental* se aplicó la técnica de bolsa de palabras para la construcción de la matriz TF-IDF con la siguiente Fórmula 1.

$$TF\,IDF = TF * IDF \tag{1}$$

donde

$$TF = \frac{Número\ de\ veces\ que\ aparece\ el\ término\ en\ el\ documento}{Número\ total\ de\ términos\ en\ el\ documento} \quad y \quad IDF = \frac{log(Número\ total\ de\ documentos\ en\ la\ colección)}{Número\ de\ documentos\ que\ contienen\ el\ término + 1}$$

La cantidad de términos generados para el conjunto *Multilenguaje sentimen*tal fueron 5038 para la lematización y 4,071 para *stemming*, mientras que para el conjunto de textos *Cardiff* fueron 7,268 para la lematización y 5,646 para *stemming*.

3.4 Aplicación de Aprendizaje Automático.

Para la clasificación de sentimientos basada en las técnicas de PLN, denominadas lematización y *stemming* se utilizaron diversos algoritmos de aprendizaje automático, los algoritmos fueron elegidos debido a su amplio uso en la literatura, implementados con la librería *sklearn* de *Python*, con los siguientes parámetros.

El algoritmo SVM fue implementado con valores para el balance entre la maximización del margen y la minimización de la clasificación incorrecta (C=1.0), el algoritmo del perceptrón simple (una capa, enfoque *one vs rest*) con una tasa de aprendizaje inicial (eta0=1.0) y máximo número de iteraciones (max_iter=1000), el algoritmo de *radom forest* con número de árboles en el bosque (n_estimators=100) y profundidad máxima de cada árbol (max_depth=None), finalmente el algoritmo de regresión logística con el parámetro de regularización (C=1.0).

3.5 Evaluación

Para la evaluación de cada algoritmo es usada la métrica de exactitud, con la siguiente Fórmula 2.

$$Exactitud = \frac{TP + TN}{TP + TN + FP + } \quad (2)$$

donde TP= Verdaderos positivos; TN= Verdaderos negativos, FP=Falsos positivos y FN=Falsos negativos

4 Resultados experimentales

En esta sección se presenta una comparativa con los resultados obtenidos en el experimento para diferentes algoritmos por clase y con ambas técnicas (lematización y *stemming*). La Tabla 4 muestra este comparativo para el conjunto de textos *Multilenguaje sentimental* y la Tabla 5 para el conjunto de textos *Cardiff*.

Tabla 4. Tabla comparativa de exactitud del conjunto de textos *Multilenguaje sentimental*

Método	Clasificación	SVM	Perceptron	Random Forest	Regresión Logística
Stemming	**negative**	**64.49%**	44.86%	53.27%	57.94%
Stemming	**neutral**	34.07%	**52.59%**	48.15%	40.00%
Stemming	**positive**	55.56%	38.89%	53.97%	**57.14%**
Lematización	**negative**	52.00%	41.60%	39.20%	**56.80%**
Lematización	**neutral**	50.00%	44.53%	**57.03%**	44.53%
Lematización	**positive**	52.17%	54.78%	45.22%	**54.78%**

Tabla 5. Tabla comparativa de exactitud del conjunto de textos *Cardiff*

Método	Clasificación	Tipo	SVM	Perceptron	Random Forest	Regresión Logística
Stemming	**0**	**Negativo**	44.66%	**55.34%**	38.83%	50.97%
Stemming	**1**	**Neutral**	**51.23%**	45.81%	54.19%	45.32%
Stemming	**2**	**Positivo**	57.58%	44.95%	56.57%	**61.11%**
Lematización	**0**	**Negativo**	43.52%	**51.85%**	44.44%	45.83%
Lematización	**1**	**Neutral**	63.44%	41.94%	**68.28%**	51.61%
Lematización	**2**	**Positivo**	47.80%	36.10%	49.27%	**52.20%**

Se realizó una prueba de *Mann-Whitney* (Neuhäuser, 2011).cuyo objetivo es comparar la distribución de los rangos de un grupo con los rangos del otro grupo y evaluar si hay una diferencia significativa entre las distribuciones. En el caso de los experimentos presentados en este artículo, no se encontró una diferencia estadísticamente significativa para las exactitudes entre los grupos de resultados obtenidos con la técnica de lematización y *stemming*.

5 Conclusiones y trabajo a futuro

En este artículo se ha presentado un enfoque para el análisis de sentimientos a partir de textos en español, comparando dos técnicas de PLN (lematización y *stemming),* así como cuatro algoritmos de aprendizaje automático.

Las principales aportaciones del trabajo presentado son: a) el enfoque propuesto para el análisis de sentimientos en textos en español considerando tareas de PLN, extracción de características mediante TF-IDF y algoritmos de aprendizaje automático; b) la comparativa de algoritmos de aprendizaje automático para obtener el que muestre los mejores resultados para la tarea; c) y la comparativa de las técnicas de PLN que consistió en, por un lado, realizar la lematización de los textos, y, por otro lado, la tarea de *stemming.*

La mejor exactitud en la tarea de clasificación fue usando *random forest* con 68.28%. Según la prueba de *Mann-Whitney* que nos permite identificar si algunos de los enfoques lematización y *stemming* tiene un desempeño estadísticamente superior al otro en términos de exactitud, se concluye que ningún enfoque es superior.

Los resultados de este trabajo representa una contribución a los analistas de redes sociales al brindarles una herramienta para identificar la polaridad de los mensajes y por otro parte, para la industria, la capacidad de identificar el sentimiento permite tener una

visión de las preferencias y tendencias del mercado que ayuda en la toma de decisiones para estrategias de marketing y satisfacción del cliente.

Como trabajo a futuro, resulta una línea de investigación a explorar, la implementación de formación de *n-gramas(bigramas, trigramas, entre otros)* para encontrar patrones de conjuntos de palabras no solo de una unidad léxica. También la implementación de técnicas de aprendizaje profundo para realizar el análisis de sentimientos es un problema abierto que aportaría en el estado del arte.

Referencias

Liu, B. (2010). "Sentiment analysis: A multi-faceted problem", *IEEE Intelligent Systems*, vol. 25, pp. 76-80.

Martínez-Cámara, E., Martín-Valdivia, M. T., & Urena-López, L. A. (2011). "Opinion classification techniques applied to a spanish corpus", *Natural Language Processing and Information Systems: 16th International Conference on Applications of Natural Language to Information Systems*, vol. 6716, pp. 169-176.

Plaza-del-Arco, F. M., Martın-Valdivia, M. T., Jimenez-Zafra, S. M., Molina-Gonzalez, M. D., & Martınez-Camara, E. (2016). "COPOS: corpus of patient opinions in Spanish. Application of sentiment analysis techniques", *Procesamiento del Lenguaje Natural*, vol. 57, pp. 83-90.

Chiruzzo, L., Etcheverry, M., & Rosá, A. (2020). "Sentiment analysis in Spanish tweets: Some experiments with focus on neutral tweets", *Procesamiento del Lenguaje Natural*, vol. 64, pp. 109-116.

Samuel, J., Ali, G. M. N., Rahman, M. M., Esawi, E., & Samuel, Y. (2020). "Covid-19 public sentiment insights and machine learning for tweets classification", *Information*, vol. 11, pp 314.

Raheja, S., & Asthana, A. (2021). "Sentimental analysis of twitter comments on COVID-19". *2021 IEEE 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 704-708.

Chintalapudi, N., Battineni, G., & Amenta, F. (2021). "Sentimental analysis of COVID-19 tweets using deep learning models", *Infectious Disease Reports*, vol. 13, pp. 329-339.

Indulkar, Y., & Patil, A. (2021). "Comparative Study of Machine Learning Algorithms for Twitter Sentiment Analysis", *2021 IEEE International Conference on Emerging Smart Computing and Informatics (ESCI),* pp. 295-299.

Bhargav, M., Alaria, S. K., & Mukhija, M. K. (2021). "Implementation of Sentiment Analysis and Classification of Tweets Using Machine Learning", Turkish Online Journal of Qualitative Inquiry, vol. 12, pp. 9-21.

Hidayat, M. R., & Sulistiyono, M. (2022). "Comparison of Accuracy and Time Of Naïve Bayes Algorithm with Support Vector Machine Algorithm in Twitter Sentiment Analysis of Peduli Lindungi Application", *2022 IEEE 5th International Conference on Information and Communications Technology (ICOIACT)*, pp. 172-176.

Parikh, A., Pawar, R., Shelke, P., Gadhave, R., & Bagade, J. (2022). "Comparison of Machine Learning Algorithms for Twitter Sentiment Analysis", *2022 IEEE International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, pp. 209-215.

Barbieri, F., Anke, L. E., & Camacho-Collados, J. (2022). "Xlm-t: Multilingual language models in twitter for sentiment analysis and beyond", *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. pp. 258-266.

Qiang, T. Y. (2023). *The AI community building the future: multilingual-sentiments*. Recuperado de https://huggingface.co/datasets/tyqiangz/multilingual-sentiments

Neuhäuser, M. (2011). "Wilcoxon–Mann–Whitney Test" en *Lovric, M. (eds) International Encyclopedia of Statistical Science*. Springer, Berlin, Heidelberg.

Capítulo 6

Nube de palabras para contextualizar vocabulario para el aprendizaje del inglés en pruebas de certificación

Aaron Ramírez Martínez, Meliza Contreras González, Pedro Bello López, Erika Bonfil Barragán

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

maraah.vm99tp@gmail.com, vikax68@gmail.com, pb5pbello@gmail.com
erika.bonfil@gmail.com

Resumen. La certificación TOEFL requiere una gran capacidad de conocimientos y habilidades en el idioma, por lo que es muy importante la preparación para presentar dicha certificación, sí bien existen muchos exámenes en línea que sirven para medir el nivel de conocimientos previo a la certificación, no existen muchas herramientas para extraer el vocabulario de un documento de forma gráfica. En este trabajo se propone un sistema para la interpretación y procesamiento de la información de bancos de información para la certificación TOEFL. Se muestra un método que procesa información de un archivo de texto y como salida muestra una nube de palabras más comunes dentro del mismo, el texto es procesado con librerías de procesamiento de lenguaje natural para determinar que palabras son conectores, preposiciones y cuales aportan un contenido altamente relevante al documento.

Palabras Clave: algoritmo Page Rank, nube de palabras, procesamiento de lenguaje natural, TOEFL.

1 Introducción

La pandemia por SARS-COVID ha traído consecuencias negativas en materia de aprendizaje, puesto que muchas de las instituciones que tenían por bien preparar a los aspirantes a la aplicación de exámenes de certificación han tenido que migrar a plataformas digitales que puedan facilitar el aprendizaje sin comprometer la integridad física de las personas.

Así surge el interés de desarrollar un sistema que permita entrenar a los aspirantes para la aplicación del examen, para lograr una mejora en la calidad de los resultados de las aplicaciones que sean acompañados de esta herramienta.

El sistema parte de experiencias personales de los estudiantes, puesto que el nivel de certificación TOEFL requiere una gran capacidad de conocimiento y experiencia en el manejo del nivel de idioma inglés, se realizó la búsqueda de herramientas que permitan

enriquecer el conocimiento y no existe una gran variedad de las mismas, sí bien existen plataformas de entrenamiento como un simulacro de examen y preguntas en formato escrito como de manera auditiva, de las cuales abundan; no existe una gran variedad de herramientas que en forma didáctica puedan enseñar los tópicos principales del examen.

El presente trabajo tiene como principal objetivo generar una herramienta que haga uso de un banco de información, y a su vez brinde a los usuarios una alternativa de estudio y entrenamiento de la gramática para la aplicación del examen.

El poder del procesamiento de la información del lenguaje Python hoy en día es muy grande, así como el procesamiento de imágenes y de su contenido han crecido de manera exponencial en los últimos años, es por ello que resulta pertinente emplearlo para el procesamiento de la información considerando que cuenta con bibliotecas especializadas para el procesamiento del lenguaje natural en el idioma inglés.

Este trabajo presenta el desarrollo de un sistema en lenguaje Python que procesa la información brindada en un corpus en formato de texto, y extrae la información más relevante, que devuelve en otro documento con el corpus de salida correspondiente, a partir del cual se visualizan de forma gráfica las principales palabras usadas en el corpus.

El contenido del artículo se organiza de la siguiente manera: en la sección 2 se describen los conceptos básicos y los tipos de procesos de recuperación de información y clasificación de estos, en la sección 3 se presenta el algoritmo basado en PageRank para la clasificación para la implementación, mientras que en la sección 4 se mencionan las conclusiones y posible trabajo futuro a desarrollar.

2 Preliminares

El procesamiento del lenguaje natural es una subdisciplina de la inteligencia artificial que se centra en la comprensión y manipulación del lenguaje humano por parte de las computadoras. El campo ha crecido rápidamente en los últimos años debido a la disponibilidad de grandes cantidades de datos y la mayor capacidad de las computadoras para procesar y analizar información. Sin embargo, todavía existen muchos desafíos en la PNL, como la ambigüedad en el lenguaje humano y la comprensión del lenguaje en contexto (Cuba, 2018).

Un corpus es un conjunto de documentos seleccionados y etiquetados que se utiliza en la recuperación de información para evaluar y desarrollar sistemas de recuperación de información. Los corpus proporcionan un entorno controlado para medir la precisión y eficiencia de los sistemas, y permiten a los investigadores desarrollar y mejorar los algoritmos de los sistemas de recuperación de información.

Por otro lado, la nube de palabras es una herramienta visual que se utiliza para representar la frecuencia de palabras en un texto o colección de documentos. Se utiliza comúnmente en la recuperación de información para representar la distribución de términos en un corpus de documentos, y para proporcionar una representación gráfica de los términos más importantes.

Una de las principales ventajas de las nubes de palabras es que permiten identificar rápidamente los términos más relevantes en un corpus de documentos. Otra ventaja de las nubes de palabras es su capacidad para identificar patrones y tendencias en los datos. Por ejemplo, las nubes de palabras pueden utilizarse para comparar la frecuencia de términos entre diferentes documentos o colecciones de documentos, lo que puede ayudar a los investigadores a identificar patrones y tendencias en la información.

Resulta evidente que la selección de los términos que describen la necesidad de información ocurre a partir de un proceso de representación. Esta es una operación compleja en sí misma que involucra los conocimientos (procesos cognitivos) del usuario. Precisamente, Abadal (2005) señala que "el trabajo de representación del conocimiento es el proceso que realiza el documentalista al mediar entre la información producto del conocimiento y el usuario final".

Estos sistemas trabajan con los metadatos son datos que describen o proporcionan información sobre otros datos y proporcionan información sobre los documentos o recursos de información disponibles, lo que permite a los usuarios encontrar y acceder fácilmente a la información que necesitan (Mendez, 2001).

La indexación de la información (Méndez, 2001) es un proceso mediante el cual se asignan términos específicos a los documentos, con el objetivo de permitir su recuperación y búsqueda de manera eficiente. Existen diferentes métodos de indexación, pero los más comunes son la indexación automática y la indexación manual. La indexación automática utiliza algoritmos y programas para analizar el contenido de los documentos y asignar términos específicos a los mismos. La indexación manual, por otro lado, requiere que un humano revise el contenido de los documentos y asigne términos específicos.

También se cuenta con el algoritmo de PageRank (Cuba, 2018), es un algoritmo de búsqueda desarrollado por Larry Page y Sergey Brin, los fundadores de Google, que es utilizado para determinar la importancia de una página web en relación con otras páginas en internet. La idea detrás del algoritmo de PageRank es que una página web es más importante si es enlazada por otras páginas importantes.

El algoritmo de PageRank funciona mediante la asignación de una puntuación inicial a cada página web, que se denomina "PageRank inicial", y luego utiliza esta puntuación para calcular el PageRank de cada página en función de los enlaces que recibe de otras páginas. El algoritmo sigue un proceso iterativo para calcular el PageRank final de cada página web.

Uno de los aspectos clave del algoritmo de PageRank es el concepto de "voto" o enlace. Cada enlace que una página recibe de otra página se considera como un voto de confianza en la importancia de la página enlazada. El algoritmo asigna un peso diferente a cada enlace en función de la importancia de la página que proporciona el enlace. Por ejemplo, un enlace de un sitio web altamente valorado tendrá más peso que un enlace de un sitio web menos valorado.

La representación gráfica de PageRank se conoce como un grafo dirigido, donde cada nodo representa una página web y cada arco representa un enlace entre páginas. En este grafo, la importancia de cada página se representa mediante un valor de rango asignado al nodo correspondiente.

El algoritmo PageRank asigna una importancia inicial a cada página y luego a través de iteraciones va actualizando el rango de importancia en función de los enlaces entrantes y salientes de cada página. El proceso finaliza cuando se alcanza un estado estable de rango de importancia de las páginas.

Por lo general se representa gráficamente los nodos más importantes con un tamaño más grande y los menos importantes con un tamaño más pequeño, además las flechas entrantes tienen un peso mayor que las salientes.

Es importante destacar que el algoritmo de PageRank también tiene en cuenta el factor denominado como "damping factor" o factor de amortiguamiento, que permite evitar la perpetuación de los ciclos de enlaces, esto es que ciertas páginas tienen un cierto porcentaje de posibilidades de ser seleccionadas aleatoriamente, lo que permite tener un mejor rendimiento y precisión en la asignación del rango de importancia a las páginas.

Por otro lado, para la clasificación de los textos las stop words (Abadal, 2005) son palabras comunes y de poco significado que se suelen filtrar de un texto antes de realizar alguna tarea de procesamiento de lenguaje natural. Estas palabras incluyen artículos, preposiciones, conjunciones y pronombres, algunos ejemplos en el idioma inglés incluyen "the", "a", "an", "and", "or", "of", "in", "to", "with", "for", "on", "by", etc. Cada idioma puede tener una lista diferente de stop words, y esta lista puede ser personalizada según las necesidades específicas de una tarea.

Eliminar las stop words antes de realizar un análisis de texto puede mejorar la eficiencia y la precisión del análisis, ya que permite enfocarse en las palabras clave y relevantes en el texto. Muchas bibliotecas de procesamiento de lenguaje natural, como NLTK en Python, tienen listas incorporadas de stop words que se pueden utilizar para filtrar estas palabras en un análisis de texto.

La clasificación de textos es un área de investigación activa en la recuperación de información y el procesamiento del lenguaje natural. Una herramienta fundamental en la clasificación de textos es una lista de palabras 'no válidas' (lista de palabras no válidas) que se utiliza para identificar palabras frecuentes que es poco probable que ayuden en la clasificación y, por lo tanto, se eliminan durante el preprocesamiento. (Hao, 2008)

La tokenización es un proceso en el procesamiento de lenguaje natural que consiste en dividir una cadena de texto en pequeñas piezas llamadas "tokens". Los tokens son unidades lógicas que representan una parte significativa del texto, como palabras o frases.

Hay diferentes formas de tokenizar un texto, como la tokenización por palabras, frases o párrafos. Además, diferentes aplicaciones y contextos pueden requerir diferentes métodos de tokenización.

En Python, existen varias bibliotecas de procesamiento de lenguaje natural, como NLTK y SpaCy, que ofrecen funciones de tokenización incorporadas que se pueden utilizar para tokenizar texto de manera eficiente (Fernández, 2013). También es posible escribir su propia función de tokenización en caso de que sea necesario un enfoque personalizado.

Las bibliotecas de Python permiten a los desarrolladores resolver problemas comunes y complejos más fácil y rápidamente, sin tener que escribir todo el código desde cero (González, 2011).

Algunas de las bibliotecas más populares en Python incluyen:

- NumPy: una biblioteca para el cálculo numérico y la manipulación de arrays multidimensionales (NumPy,2023).
- Pandas: una biblioteca para el análisis de datos y la manipulación de tablas de datos (Pandas, 2023).
- Matplotlib: una biblioteca para la creación de gráficos y visualizaciones de datos.
- Scikit-learn: una biblioteca para el aprendizaje automático y el análisis de datos (Scikit,2023)
- TensorFlow y PyTorch: bibliotecas para el aprendizaje profundo y la inteligencia artificial (TensorFlow, 2023).

3 Algoritmo para la generación de nube de palabras

Bajo la línea de información que se sigue en el presente trabajo, la implementación y conceptualización del algoritmo se presenta a continuación en la Figura 1:

1. **Carga del Documento de Texto:** el algoritmo comienza abriendo un archivo de texto (txt) y guarda su contenido en una variable. Esto permite que el contenido del archivo sea manipulable y procesable en el código.

2. **Importación de Librerías:** se importan las bibliotecas (librerías) necesarias para el procesamiento y análisis de texto. Estas librerías proporcionan funciones y herramientas que facilitarán las diversas tareas en el algoritmo.

3. **Definición de Consultas:** se definen una serie de cadenas de texto que representan las consultas que se realizarán posteriormente en el documento. Estas consultas guiarán el análisis del texto para buscar información específica.

4. **Eliminación de Etiquetas de Respuesta:** se identifican y eliminan las etiquetas de respuesta (A), (B), (C), (D) del texto. Esto es importante para asegurarse de que las etiquetas no interfieran con el análisis y la búsqueda de contenido relevante en el documento.

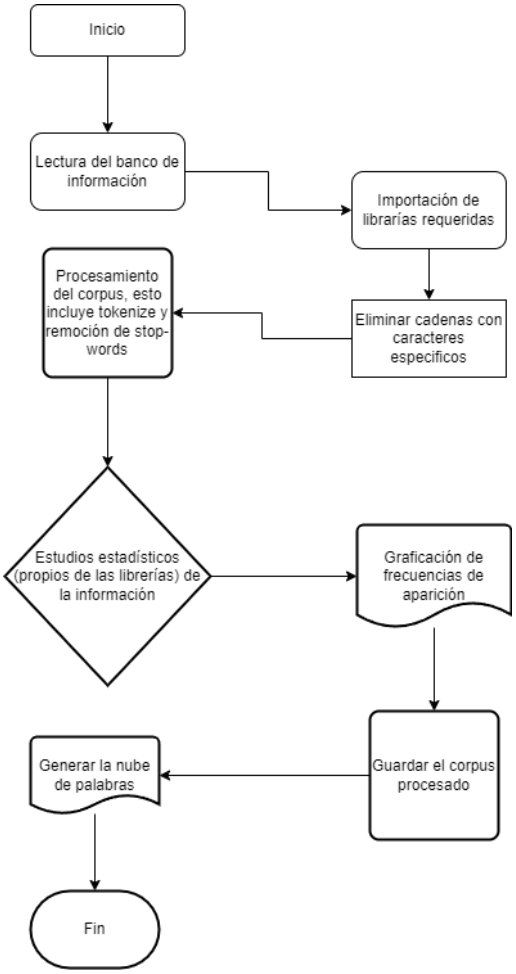


Figura 1. Algoritmo Desarrollado

5. **Extracción de Palabras Vacías (Stopwords):** se obtiene una lista de palabras comunes en el idioma inglés que generalmente no aportan significado sustancial al análisis del texto. Estas palabras, conocidas como "stopwords", se eliminan del texto para enfocarse en las palabras más relevantes.

6. **Tokenización del Texto:** se divide el texto en unidades más pequeñas llamadas "tokens". Estos tokens pueden ser palabras individuales, signos de puntuación u

otros elementos. La tokenización es una etapa fundamental para preparar el texto para su análisis.

7. **Preprocesamiento del Corpus:** se realiza una etapa de preprocesamiento en el corpus (conjunto de texto) eliminando las palabras vacías (stopwords) y los signos de puntuación. Esto ayuda a reducir el ruido y a enfocarse en las palabras más importantes para el análisis.

8. **Consulta y Preprocesamiento de Consulta:** la consulta de búsqueda también se somete a un proceso similar de eliminación de palabras vacías y signos de puntuación. Esto asegura que la consulta esté en un formato compatible con el corpus preprocesado para una búsqueda eficiente.

9. **Obtención de Índices de Palabras de Consulta:** se determinan los índices en el corpus preprocesado donde se encuentran las palabras de la consulta. Esto permitirá identificar dónde aparecen las palabras clave en el texto original.

10. **Creación de Gráfico de Frecuencia:** se genera un gráfico que representa la frecuencia de las palabras en el corpus preprocesado. Esto puede proporcionar información visual sobre las palabras más comunes en el texto y sus distribuciones.

11. **Guardado del Corpus Preprocesado:** se guarda el corpus preprocesado en una nueva variable o archivo. Esto permite reutilizar el corpus limpio en futuros análisis sin tener que repetir el preprocesamiento.

12. **Generación de Nube de Palabras:** se crea una nube de palabras a partir del corpus preprocesado. Una nube de palabras muestra visualmente las palabras más frecuentes en el texto, donde el tamaño de cada palabra está relacionado con su frecuencia de aparición.

En resumen, este algoritmo realiza una serie de pasos desde la carga del texto hasta la generación de visualizaciones y análisis relevantes. Se enfoca en la limpieza y el procesamiento del texto para extraer información valiosa y visualizar patrones importantes en el contenido.

En primera instancia, y como variable central y principal, tenemos el corpus, sin procesar, y sin ningún procesamiento.

```
with open('TOEFL Reading Comprehension 1.txt','r',encoding= "utf8") as miarchivo:
  texto = miarchivo.read()
```

Posterior a ello, importamos las librerías principales que usaremos.

```
from nltk.corpus import stopwords
```

```
from nltk.tokenize import word_tokenize
import string
import nltk
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from collections import Counter
from collections import OrderedDict
```

El código se basa en un programa de recuperación y aún se utiliza la versión de búsqueda de una cadena dentro del corpus.

```
#las cadenas a continuacion son las consultas que realizaremos
#cadena='separation anxiety in infancy (i.e. up to two years of age) and in preschool children, particularly separation of a child from its mother'
#cadena='the toxicity of organic selenium compounds'
cadena='language development in infancy and pre-school age'
#cadena= '!#$%&()*+,-./:;<=>?@[\]^_`{|}~'
#cadena='of not few so on where as no how d before shouldve has weren than will '
#cadena="Perro gato"
```

La siguiente parte de nuestro código, consiste en cambiar las cadenas “(A), (B),(C),(D)”, en primera instancia porque no entran dentro de los signos de puntuación, también es importante saber el contexto, pues al tratarse de un corpus de ensayo para TOEFL, existen preguntas, y por ende opciones de respuestas, es por ello que es importante eliminar estas partes, para que no afecten los resultados.

```
texto = texto.replace('(A)', '')
texto = texto.replace('(B)', '')
texto = texto.replace('(C)', '')
texto = texto.replace('(D)', '')
```

Las stop-words son imprescindibles dentro del procesamiento de texto, es por ello que nuestro siguiente paso es obtenerlas como un punto importante dentro de la ejecución del código.

```
#obtenemos las stop_words en el mismo lenguaje que el corpus
stop_words= set(stopwords.words('english'))
```

Una vez hecho eso, la siguiente parte consiste en aplicar el tokenize de nuestra información, en primera instancia de nuestro corpus principal, como también como con nuestras cadenas de búsqueda.

word_tokens = word_tokenize(texto) #tookenizar significa utilizar toda la palabra y no solo un caracter
word_tokens1 = word_tokenize(cadena)

Posteriormente se realiza el procesamiento del texto, se eliminan stop-words, signos de puntuación, y como anteriormente ya hemos eliminado algunos factores importantes obtenemos un texto de salida puede observarse en la Figura 1, si imprimimos la variable filtro.

Word_tokens = list(filter(lambda token : token not in string.punctuation,word_tokens,)) #Eliminamos caracteres de puntuacón del corpus
word_tokens1= list(filter(lambda token : token not in string.punctuation,word_tokens1)) #Eliminamos caracteres de puntuación de la consulta
filtro=[] #Declaramos una variable de tipo lista que contendrá el corpus una vez finalizado el preprocesamiento
filtro1=[] #Declaramos una variable de tipo lista que contendrá la consulta una vez finalizado el preprocesamiento
aux=[]#Utilizamos una variable auxiliar para realizar la consulta
for palabra in word_tokens: #iniciamos el ciclo para eliminar stop words
if palabra not in stop_words:
filtro.append(palabra)
for I in word_tokens1:
if I not in stop_words:
filtro1.append(i)

Figura 2. Aplicación de los primeros filtros al corpus

Se obtienen los resultados y C es una variable que indica cuántas veces se repite una palabra en el corpus de la Figura 3, y se usa la función de obtención de distribución de frecuencia del corpus:

c=Counter(filtro) # Obtenemos la propiedad del contador de la librería usada
fdist=nltk.FreqDist(filtro) # Usamos una función de la librería para obtener la frecuencia del corpus

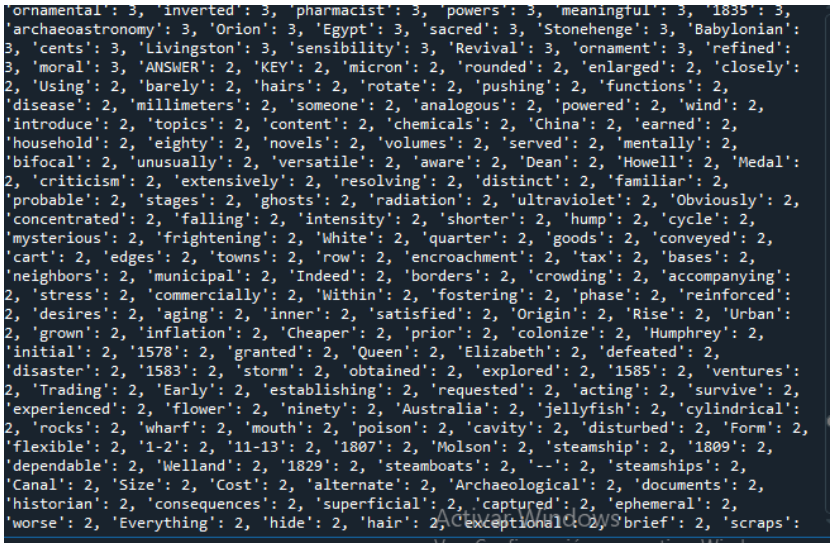


Figura 3. Aplicación de los primeros filtros con conteo de términos

A continuación, se guardan los resultados que se obtienen y se guarda en un archivo de texto, la salida del código será el corpus ya procesado.

```
y=OrderedDict(c.most_common())
with open('salida.txt','w',encoding='utf-8') as file:
    for k,v in y.items():
        file.write(f'{k} ' )
filename = "salida.txt"
```

Una vez realizada la salida del texto, abriremos el corpus, y lo guardamos en una variable que permita el manejo de información.

```
filename = "salida.txt"
with open(filename,encoding='utf-8') as f:
```

```
mytext = f.read()
```

Los resultados obtenidos se muestran de dos maneras, en la Figura 4 observamos la nube de palabras.

```
wordcloud = WordCloud().generate(mytext)
#%pylab inline

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.savefig("resultado.png")
plt.show()
```

Figura 4. Nube de palabras generadas

4 Conclusiones y Trabajo Futuro

El análisis de textos es una tarea importante en diferentes áreas, desde la investigación científica hasta la toma de decisiones en empresas. En este sentido el código presentado utiliza una serie de herramientas para analizar un corpus de texto y responder a consultas específicas.

El trabajo comienza listando conceptos importantes para entender la conceptualización e implementación del código, para así explicar de la mejor manera los resultados, y por ello es importante entender cómo se estructura la información y como se procesa.

Una de las tareas importantes en el preprocesamiento del texto es la eliminación de stop words, es decir, palabras que no aportan información relevante para el análisis. Para ello,

se utilizan herramientas del procesamiento del lenguaje natural, para obtener una lista de stop words en inglés.

Por lo que el presente trabajo ha mostrado cómo se pueden utilizar diferentes herramientas para el análisis de textos, desde la eliminación de stop words hasta la visualización de resultados. La utilización de estas herramientas puede resultar útil en diferentes áreas, como la investigación científica, la gestión empresarial o el análisis de redes sociales, entre varios otros usos que se le pueden dar a esta información.

Como trabajo a futuro se puede realizar una mejora significativa considerando:

- Los dígitos, números y caracteres numéricos en la recuperación de información dependiendo del contexto al que se aplica.
- Uso de expresiones regulares y su importancia en el manejo de información.
- Las contracciones del lenguaje, en distintos lenguajes existen contracciones, que pueden variar el resultado del procesamiento de la información.
- Palabras validas en determinados lenguajes son consideradas correctas y no sólo son un cúmulo de caracteres.

Referencias

Abadal, E., & Codina, L. (2005). "Recuperación de información, Bases de Datos Documentales: Características, funciones y métodos: Madrid, Síntesis", 29-92.

Cuba R., Y., & Olivera B., D. (2018). "Los metadatos, la búsqueda y recuperación de información desde las Ciencias de la Información", *E-Ciencias de la Información*, vol 8, pp. 146-158.

Fernandez, A. (2013). "Python 3 al descubierto", Alfaomega Grupo Editor.

González D., R. (2011). "Python para todos", a Creative Commons Reconocimiento 2.5 España.

Hao, L., & Hao, L. (2008). "Automatic identification of stop words in chinese text classification". *In 2008 International conference on computer science and software engineering* vol. 1, pp. 718-722.

Méndez R., E. M. (2001). "Metadatos y recuperación de información: estándares, problemas y aplicabilidad en bibliotecas digitales", *Tesis doctoral*, Departamento de Biblioteconomía y Documentación, Recuperado de http://hdl.handle.net/10016/26863.

NumPy. (2023). Recuperado en febrero 4, 2023, de Numpy.org website: https://numpy.org/

Pandas - Python Data Analysis Library. (2023). Retrieved February 4, 2023, from Pydata.org website: https://pandas.pydata.org/.

Scikit-learn: machine learning in Python — scikit-learn 1.2.1 documentation. (2023). Retrieved February 4, 2023, de Scikit-learn.org website: https://scikit-learn.org/stable/.

TensorFlow (2023). Recuperado en febrero 4, 2023, de TensorFlow website: https://www.tensorflow.org/?hl=es-419.

Capítulo 7

Algoritmos de aprendizaje automático aplicados al reconocimiento de los factores de depresión en adultos jóvenes

Octavio Mendoza Gómez[1], Mireya Tovar Vidal[1], Fernando Zacarias Flores[1]

[1] Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación, Av. San Claudio y 14 sur, Ciudad Universitaria, 72592 Puebla, Pue., México
octaviomen12@gmail.com, mireya.tovar@correo.buap.mx, fzflores@yahoo.com

Resumen. La depresión es un trastorno mental que afecta a todos los miembros de una población y es complicado establecer los síntomas generales que la generan, esto se debe a que no siempre son los mismos. El estudio de los factores de riesgo de depresión en adultos jóvenes es un campo en rápido crecimiento. Los métodos de aprendizaje automático se están utilizando cada vez más para identificar estos factores, y los resultados de estos estudios son prometedores. En este trabajo se propone aplicar distintos algoritmos de aprendizaje automático para reconocer y categorizar los casos de depresión que existan en una población de adultos jóvenes.

Palabras Clave. aprendizaje automático, agrupamiento, promedios *K*, AGNES, depresión.

1 Introducción

En la psicología clínica se estudian varias enfermedades, entre ellas encontramos la depresión, la cual se caracteriza principalmente por la presencia constante de un estado de ánimo bajo. Entre los pacientes de depresión se suelen presentar los mismos síntomas, sin embargo, los factores que los detonan no son siempre los mismos, cada paciente es único con respecto al camino que los lleva a depresión, así como un tratamiento propio.

La depresión es una de las enfermedades más comunes que se presentan en varios rangos de edad y se han creado sistemas de acompañamiento propio a los individuos con esta enfermedad. Sin embargo, entre los sistemas se ha presentado una problemática, la detección temprana de los casos que pueden llegar a ser severos. Este es un problema de clasificación y para resolverlo se proponen dos tipos de algoritmos, los de clasificación múltiple de tipo supervisado y los algoritmos de agrupación de tipo no supervisado.

El objetivo de los algoritmos enfocados a la clasificación es generar una etiqueta mediante el uso de ecuaciones matemáticas sobre un conjunto de datos. La estructura del conjunto es de filas y columnas donde la primera fila se llama *vector de características* y es la que otorga información a las ecuaciones para generar la etiqueta anteriormente mencionada (Müller & Guido, 2017). Entre los algoritmos de clasificación encontramos a los *K* vecinos cercanos que, dado un punto x_0, se buscan *K* vecinos que tengan características similares a este punto (Gareth et al, 2018).

En el aprendizaje supervisado encontramos un algoritmo llamado máquina de vectores de soporte. Este busca generar un hiperplano sobre una distribución de datos de tal manera que los separe en dos o más categorías dependiendo de la cantidad de vectores de soporte que se generen por cantidad de categorías deseada.

Por el lado de los algoritmos de aprendizaje no supervisados no se le da al *vector de características* un uso cómo en el aprendizaje supervisado, en su lugar, los algoritmos de este tipo de aprendizaje buscan crear una estructura de los datos para asignarlos a un grupo (Patel, 2019).

El objetivo de este trabajo es crear modelos de aprendizaje capaces de reconocer y categorizar casos de depresión en una población de adultos jóvenes.

El resto del artículo tiene la siguiente organización: sección 2 donde se exponen trabajos con temas relacionados al planteado en este trabajo y que sirven como soporte a la investigación. La descripción del conjunto de datos, procedencia, que características se miden con los datos y su tamaño en filas y columnas mientras que la explicación de cómo se llevan a cabo los procesos de análisis de datos e implantación de los algoritmos de aprendizaje automático para su entrenamiento se presenta en la sección 4. En la sección 5 se presentan las propuestas de algoritmos de aprendizaje automático enfocados al agrupamiento, así como algunas ventajas y desventajas. La sección 6 se divide en 6.1 y 6.2 donde se presentan los resultados obtenidos del análisis de datos e implementación de los algoritmos de aprendizaje respectivamente.

2 Preliminares

La depresión en adultos jóvenes y adolescentes es un tema de investigación al cuál se le ha dado mucha atención y varía según la población de estudio. Algunos ejemplos son la ansiedad sobre población de estudiantes de medicina en Sudáfrica (Mhata et al, 2023). También se observan los niveles de depresión de los estudiantes en dos carreras universitarias distintas, ciencias naturales y música (Korte et al,2023).

Los trabajos anteriores tienen un enfoque psicológico, existen trabajos donde se aplica el aprendizaje automático para reconocer otras enfermedades, sean fisiológicas o

psicológicas. En el trabajo de Kasani Hosseinzadeh y colaboradores se estudia la relación entre depresión y alimentación mediante algoritmos de aprendizaje automático (Hosseinzadeh et al, 2023).

Muhammad Istiaq y su equipo realizan una investigación de la depresión usando datos obtenidos de redes sociales, tales como publicaciones, historias o fotos que se suban en un perfil perteneciente a algún miembro de una población (Istiaq et al, 2023).

En los trabajos anteriores se observan casos de depresión más avanzados, Kwang-Sig Lee junto con sus colaboradores realizaron una investigación donde se cree un modelo de aprendizaje que permita utilizar casos donde la depresión empieza, es decir, estudian casos con un diagnóstico de depresión leve (Lee & Ham, 2022).

Se pueden aplicar algoritmos de agrupamiento por promedios K a trabajos con imágenes, por ejemplo, Simon Tongbram y sus colaboradores realizaron un trabajo enfocado a la segmentación de imágenes usando un híbrido de algoritmos de agrupación tales como promedio K y agrupación sustractiva de tal manera que su método obtenga valores de segmentación mayores otros métodos que ellos exponen (Tongbram et al, 2021).

En el caso de las imágenes médicas el trabajo de Farhad Akhbardeh y sus colaboradores publicaron un trabajo donde utilizan un algoritmo de agrupamiento por promedios K para reconocer obstrucciones en las arterias mediante imágenes generada a partir de la angiografía de rayos X y llevando a cabo un proceso propio del análisis de dichas imágenes (Akhbarde & Demirel, 2018).

Los trabajos presentados muestran la capacidad que poseen los algoritmos de agrupamiento para separar un conjunto de datos de acuerdo con las características que posean, sin embargo, también es posible que sean aplicados para la aplicación de la predicción de categorías, es decir, resolver un problema cuyo aprendizaje es de tipo supervisado, pero sin la necesidad de implementar un algoritmo perteneciente a este tipo de aprendizaje.

Emilio Carrizosa y equipo proponen la implementación de un método numérico que asimila a un algoritmo de agrupamiento de tal manera que se conserve e incluso se mejore la precisión del modelo de categorización (Carrizosa et al, 2021).

Existen algoritmos de agrupamiento que aplican un proceso iterativo de tal manera que se respeta una jerarquía de entre las características de los datos. Este tipo de algoritmos son llamados de agrupación jerárquica, entre estos podemos encontrar a los algoritmos de agrupamiento DIANA y AGNES (Ozdemir & Cerman, 2021), siendo el último uno de los cuales se usará como propuesta en este trabajo.

Una aplicación del algoritmo de agrupamiento AGNES lo realizan Xiao Zhou y su equipo donde implementa un algoritmo de agrupamiento AGNES para reconocer las atracciones más populares para los turistas dependiendo de distintos factores psicológicos y cómo pueden mejorar las atracciones (Zhou et al, 2022).

Así como se aplican algoritmos de aprendizaje automático también existen trabajos donde se aplican algoritmos de aprendizaje profundo, por ejemplo, Shikha Tiwari junto con su equipo implementaron ambos tipos de algoritmos para detectar depresión mediante un modelo ya entrenado con el conjunto de datos EEG (Tiwari et al, 2023).

3 Descripción del conjunto de datos

El conjunto de datos es público, para obtenerlo sus autores realizaron una encuesta a una población universitaria delimitada cuyas preguntas miden distintos aspectos de los individuos tales como niveles de depresión, ansiedad, pánico, hábitos de sueño, hábitos de alimentación, promedio, carrera, estado civil, etc.

La encuesta se aplicó en una población de cien estudiantes, es decir, en el conjunto de datos hay cien filas de datos y cada una tiene once columnas que representan las características las preguntas realizadas en la encuesta. Cada pregunta representa una característica medible de los miembros de la población de estudio y los valores obtenidos serán utilizados para entrenar el modelo de aprendizaje (Shariful, 2023).

4 Análisis de datos y aplicación de los algoritmos

Ya una vez expuestas las posibles aplicaciones de los algoritmos de aprendizaje y cómo es el conjunto de datos, procederemos a explicar cómo será el proceso de limpieza y análisis de los datos para aplicar los algoritmos de aprendizaje. Los pasos que se llevaran a cabo para implementar los algoritmos de aprendizaje son los siguientes:

1. Limpieza de datos atípicos y valores no numéricos
2. Análisis exploratorio de datos
3. Implementación de los algoritmos de aprendizaje
4. Comparación de las métricas obtenidas de los algoritmos de aprendizaje
5. Toma de decisión del modelo de aprendizaje
6. Optimización del modelo de aprendizaje

Cada uno de los pasos serán descritos a continuación. En la limpieza de datos se llevan a cabo las siguientes tareas, eliminación de datos atípicos o muy altos con respecto al resto, conversiones de datos, tal como convertir datos categóricos a numéricos, llenado de valores no numéricos, representados con Nan, por valores numéricos, ya sea con el promedio de los datos existentes por columna o su cambio al valor cero.

En el análisis exploratorio de datos se buscan las correlaciones del vector de características con el valor objetivo para así reconocer las características más importantes para el modelo de aprendizaje.

En la implementación de los algoritmos de aprendizaje los algoritmos de aprendizaje propuestos se implementan en el lenguaje de programación Python sobre el conjunto de datos. En la comparación de las métricas se comparan los valores del tiempo de ejecución, la precisión y el error obtenidos de cada modelo obtenido por algoritmo.

De este paso se llega a la toma de decisión del modelo de aprendizaje y se realiza una hipótesis sobre dicha decisión y se buscará cumplirla. El paso de optimización del modelo de aprendizaje es un ajuste sencillo similar al paso de limpieza y análisis de datos, pero lleva un tiempo de planeación menor. Para terminar, se hace la conclusión del proyecto y la discusión de los datos obtenidos para saber si el modelo de aprendizaje obtenido del algoritmo propuesto es correcto o da los resultados planteados en la hipótesis.

5 Algoritmos propuestos

Para realizar una propuesta de los algoritmos que serán utilizados debemos entender el problema, el objetivo es separar los datos en los distintos casos de depresión, es decir, asignarle una etiqueta que indica la pertenencia a un grupo representante del caso de depresión a cada uno de los datos, esto se traduce en un problema de aprendizaje no supervisado enfocado en la agrupación. Los algoritmos de aprendizaje no supervisados enfocados a la agrupación que se proponen son el algoritmo de agrupamiento por promedios *K* y el algoritmo de agrupamiento AGNES.

El algoritmo de agrupamiento por promedios *K,* también conocido como *K*-means, funciona de la siguiente manera, se generan K grupos, cada grupo posee un conjunto de características que lo define, y están formados por distintos datos y tienen un tamaño propio que puede ser distinto para cada grupo. Los miembros del grupo son asignados dependiendo de las condiciones de cada grupo, los miembros, o datos, son evaluados y en caso de que cumplan una condición se agregan o extraen datos a un grupo (Liu, 2020).

El algoritmo de agrupamiento AGNES, o agglomerative nesting por sus siglas en inglés, pertenece a los algoritmos de agrupamiento jerárquico. Su funcionamiento es el siguiente, se crean una *K* cantidad de grupos que sea igual a la cantidad de datos, es decir, si tenemos *n* datos entonces $K=n$. Al inicio de las iteraciones cada grupo tiene como miembro un dato único y empezará uniendo los grupos dependiendo de las similitudes que poseen entre sí, el proceso iterativo se termina hasta llegar a un cierto número de grupos o límite de iteración (Boehmke et al, 2020).

A pesar de que ambos algoritmos pertenecen a los mismos grupos de aprendizaje y tipo de problema objetivo cada uno posee ciertas ventajas y desventajas, por ejemplo, el algoritmo de agrupamiento por promedios *K,* este requiere de pocos pasos para ser realizado por completo, pero posee un tiempo de ejecución alto. Mientras que el algoritmo de agrupamiento AGNES nos permite trabajar con grupos pequeños, pero posee un tiempo de ejecución mayor a promedios *K*.

6 Resultados

De la aplicación de la metodología expuesta en este trabajo se obtuvieron distintos resultados, entre estos datos se muestran los comportamientos de las distintas relaciones entre los datos al momento de realizar el análisis exploratorio de datos, así como también el comportamiento de la implementación de los algoritmos de agrupamiento propuestos en este trabajo. A continuación, presentaremos y explicaremos los resultados obtenidos del análisis exploratorio de datos.

6.1 Análisis exploratorio de datos

En este apartado se presentarán los resultados obtenidos del análisis exploratorio de datos obtenidos de un conjunto de datos público donde se miden distintos aspectos de la depresión sobre una población estudiantil.

Dentro del análisis exploratorio de datos se encontraron distintas relaciones de los datos que nos permitirán reconocer las características del conjunto de datos que servirán para implementar los algoritmos de agrupamiento sobre esas características, esto se traducirá en una precisión alta del modelo de aprendizaje.

Entre estas relaciones podemos destacar las más importantes, en la figura 1 se presenta un histograma donde se muestran las carreras con mayor cantidad de miembros de la población de estudio mientras que la figura 2 muestra en que año de estudio se encuentran.

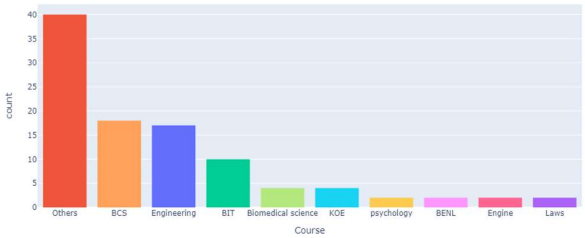


Figura 1. Carreras de los miembros de la población de estudio

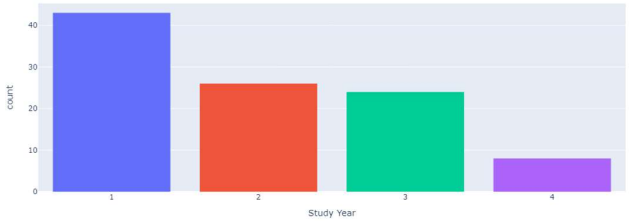


Figura 2. Año de estudio de los miembros de la población de estudio

De la figura 1 podemos destacar tres relaciones, que son la cantidad de miembros de la población con ansiedad, depresión y también si han tenido ataques de pánico, con su carrera o cursos que toman en su carrera. En el análisis encontramos a las carreras que tienen mayor cantidad de miembros de la población de estudio. La primera relación se presenta en la figura 3 donde se compara la depresión con respecto a las carreras más pobladas, es notable que nuevamente el aglomerado de todas las demás carreras llamada "Otras" tiene el mayor número de individuos deprimidos. A pesar de esto no hay tanta diferencia con respecto a las otras carreras.

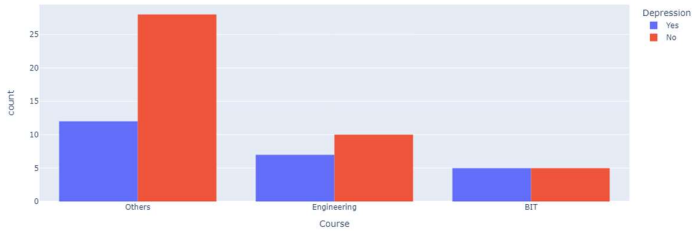


Figura 3. Relación de carreras con depresión

En cuanto a la ansiedad con respecto a la carrera, en la figura 4 notamos que existe un nivel de ansiedad menor en la carrera de ingeniería con respecto al nivel de depresión que se presentó anteriormente, mientras que, para la carrera de negocios, información y tecnología (columna BIT) tiene mayor escala de ansiedad que de depresión.

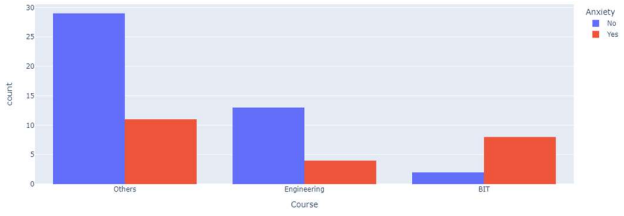


Figura 4. Relación de carreras con ansiedad

En la figura 5 veremos si hay miembros de la población que han sufrido ataques de pánico en algún momento de su vida universitaria y relacionarla con la carrera a la que pertenecen. Esta relación nos arroja niveles altos para las carreras aglomeradas mientras que para las otras dos carreras más pobladas no existe una población tan grande con ataques de pánico. De esta manera podemos hacer una hipótesis donde hacemos una relación entre la depresión de los estudiantes con los ataques de pánico que tienen.

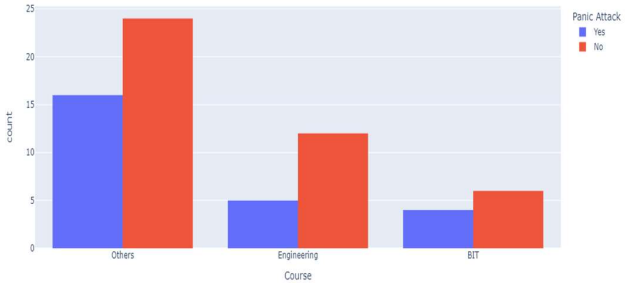


Figura 5. Relación de carrera con ataques de pánico

Otras relaciones que podemos obtener se muestran en las figuras 6 donde se muestra la cantidad de población en cada promedio, donde se concentra mayormente en los promedios de 3.00 a 3.49 y 3.50 a 4.00 GPA. La figura 7 representa la cantidad de miembros en cada una de las edades capturadas, donde domina, las edades de 18, 19, 23 y 24 años.

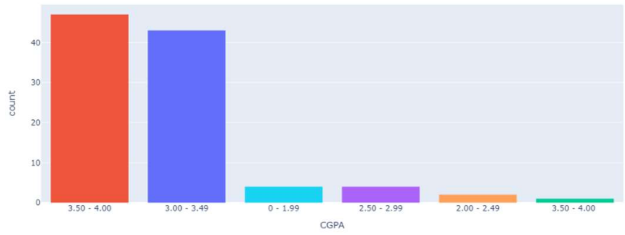


Figura 6. Promedios de la población

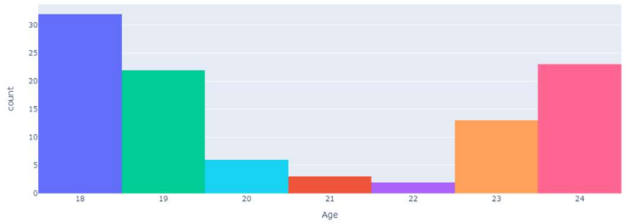


Figura 7. Edades de la población

En la relación de edad-depresión (figura 8) notamos que la edad de 18 años es donde más adultos jóvenes están estresados, el estrés baja en las edades de 20 a 22 años y sube nuevamente en 23 y 24 años. Este patrón se repite en la relación de la figura 9 (edad-ansiedad) donde hay niveles menores de ansiedad con respecto a la depresión. Por último, la figura 10 muestra que no hay ataques de pánico en las edades de 21 y 22 años, caso contrario a las edades de 18 a 20 años y 23 a 24 años.

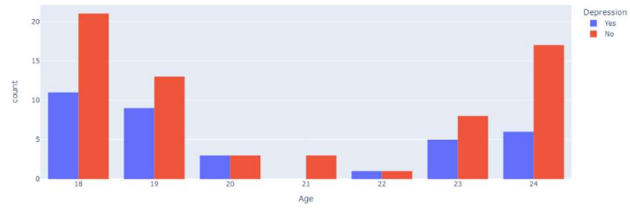


Figura 8. Relación edades y depresión

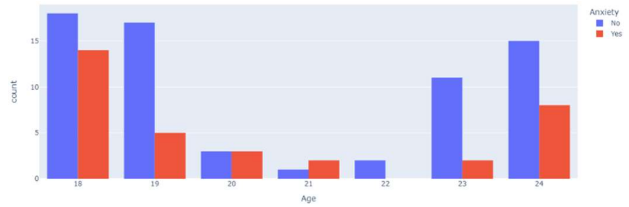


Figura 9. Relación edades y ansiedad

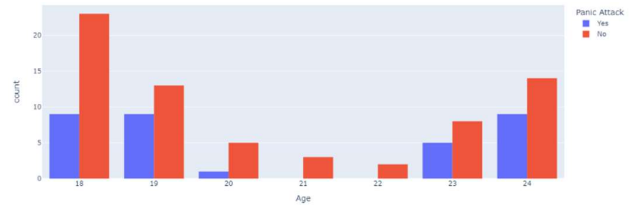


Figura 10. Relación edades y ataques de pánico

Una relación promedio-ansiedad se muestra en la figura 11, donde los mayores promedios presentan más casos de ansiedad que promedios más bajos.

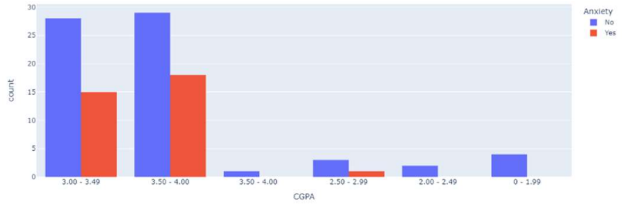


Figura 11. Relación promedio y ansiedad

Otra relación con el promedio es con los casos de ataques de pánico. En la figura 12 se observa que los promedios en el rango de 3.50-4.00 tiene más casos de ataques de pánico que otros rangos de promedio.

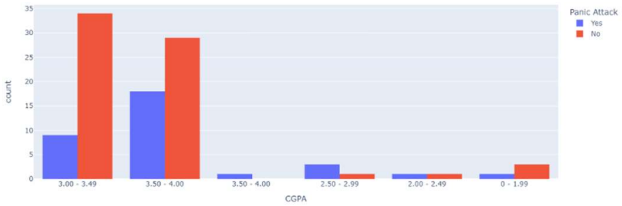


Figura 12. Relación promedio y ataques de pánico

La figura 13 muestra que la mayor cantidad de casos de depresión está más relacionada con promedios altos en el rango de 3.00 a 4.00 GPA.

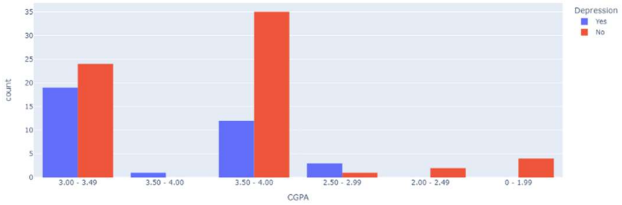


Figura 13. Relación promedio y depresión

Una vez analizadas estas características es posible crear una hipótesis donde estás mismas serán muy importantes para considerar al momento de implementar el algoritmo de agrupamiento y entrenar el modelo de aprendizaje.

6.2 Algoritmos de agrupamiento y modelo de aprendizaje

Se realizó una prueba de codo para ambos tipos de algoritmos de agrupamiento que nos permitió obtener un valor de dos grupos y se presentan en la figura 14. Al ser la misma prueba sobre el mismo conjunto de datos el resultado es el mismo.

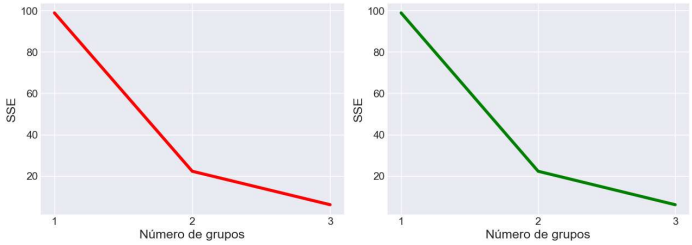


Figura 14.a Promedios K **Figura 14.b** Agnes
Figura 14. Prueba de codo promedio K y AGNES

Ahora presentaremos los resultados obtendidos de la implmentación de los algoritmos de agrupamiento por promedios *K* y AGNES. En la gráfica de la figura 15 se pueden observar dos cruces que marcan el centroide de cada grupo, no deprimidos y deprimidos, que se leen de izquierda a derecha, siendo el primer grupo el de los no deprimidos. Los puntos más cercanos tienen una pertenencia mayor al grupo y aquellos que estén a una distancia similar entre los dos significa que están en un proceso de depresión.

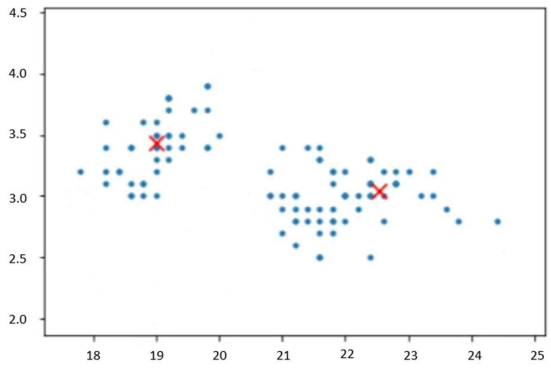


Figura 15. Agrupamiento por promedios k

Este análisis nos sirve para poder reconocer un ritmo con el que evoluciona la depresión en una población. En el caso del algoritmo AGNES se realizaron siete iteraciones donde

el agrupamiento final, presentado en la figura 16, muestra un comportamiento más esparcido de los datos sobre las cruces de los grupos.

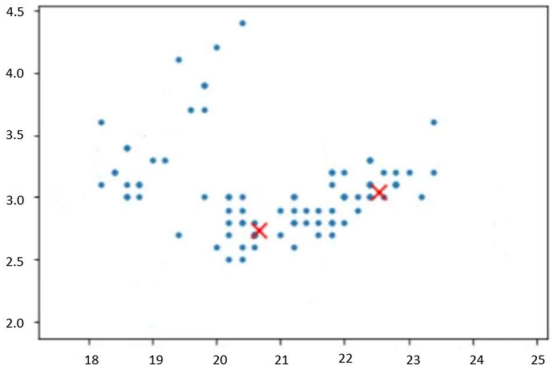


Figura 16. Agrupamiento por AGNES

Es posible notar que el agrupamiento realizado por AGNES se tiene una mayor cercanía entre los dos centroides de los grupos, esto es debido a que por cada iteración del algoritmo se actualizan sus valores.

7 Conclusiones

El reconocimiento de los casos de depresión mediante algoritmos de agrupación depende del rendimiento de los algoritmos de agrupamiento que sean aplicados, en este caso promedios *K* y AGNES. La elección de qué algoritmo usar depende del problema específico que está tratando de resolver. En caso de necesitar en específico la cantidad de grupos por adelantado, promedios *K* es una buena opción y sobre todo nos dará una dispersión más precisa de los datos con respecto a los centriolos.

Si no se está seguro de cuántos grupos hay, o si poder explorar diferentes números de grupos, entonces AGNES es una buena opción, la única desventaja notable es que suelen estar más cercanos los centriolos debido al proceso de actualización iterativa. Sin embargo, ambos algoritmos de agrupamiento mostraron una buena separación de los datos de acuerdo con las características que posee.

Este trabajo fue realizado con un tamaño de población pequeña, pues solo se tenían 100 registros de datos, al observar que los algoritmos propuestos cumplen con la expectativa de su funcionamiento. La implementación de ambos algoritmos con una población mayor es una propuesta para un trabajo a futuro, con los mismos algoritmos de aprendizaje automático.

Referencias

Müller, Andreas C. & Guido Sarah. (2017). *Introduction to Machine Learning with Python: A Guide for Data Scientists* (p. 25). O'Reilly.

Gareth, James et al. (2018*). An Introduction to Statistical learning: with applications in R*. (pp. 39-42). Springer.

Patel, Ankur A. (2019). *Hands-On Unsupervised Learning Using Python How to Build Applied Machine Learning Solutions from Unlabeled Data* (p. 5). O´Reilly.

Liu, Yuxin. (2020). *Python Machine Learning By Example* (pp. 316-318). Packt editorial.

Boehmke, Bradley y Greenwell, Brandon. (2020). *Hands-on Machine Learning with R*.

Mhata, Nelao, et al(2023). *Prevalence of depression, anxiety and burnout in medical students at the University of Namibia*. South African Journal of Psychiatry. P. 29.

Korte, Michaela et al. (2023). *The same but different. Multidimensional assessment of depression in students of natural science and music*. Health Psychology Research. P.11.

Hosseinzadeh Kasani, Payam et al. (2023). *Evaluation of nutritional status and clinical depression classification using an explainable machine learning method*. Frontiers in Nutrition.

Ishtiaq, Muhammad et al. (2023). *Detecting Depression on Social Platforms Using Machine Learning*.

Lee, Kwang-Sig & Ham, Byung-Joo. (2022). *Machine Learning on Early Diagnosis of Depression*. Psychiatry Investigation. P. 19.

Tiwari, Shikha et al. (2023). *Machine and Deep Learning Technique for Depression Detection Using EEG Data*.

Carrizosa, Emilio et al. (2021). *On Clustering Categories of Categorical Predictors in Generalized Linear Models*. Expert Systems with Applications.

Tongbram, Simon et al. (2021). *Segmentation of image based on k-means and modified subtractive clustering*. Indonesian Journal of Electrical Engineering and Computer Science.

Akhbardeh, Farhad & Demirel, Hasan. (2018). *Coronary Stenosis Measurements Using K-Means Clustering*.

Ozdemir, Ozer & Cerman, Simgenur. (2021). *Performance Comparison with Hierarchical and Partitional Clustering Methods*. WSEAS TRANSACTIONS ON COMMUNICATIONS. P. 20.

Zhou, Xiao et al. (2022). *Tour-Route-Recommendation Algorithm Based on the Improved AGNES Spatial Clustering and Space-Time Deduction Model*. ISPRS International Journal of Geo-Information. P. 11.

Shariful, Islam. (2023). *Student Mental health dataset*. Kaggle. https://www.kaggle.com/datasets/shariful07/student-mental-health. (Conjunto de datos)

Capítulo 8

Aplicando un algoritmo minimax al problema de coloreo de grafos

Octavio Mendoza[1], Antonio Pérez[1], Guillermo De Ita[1]

[1] Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
octaviomen12@gmail.com, antonio.pe.v@gmail.com, deitaluna63@gmail.com

Resumen. El estudio de la teoría de grafos permite tener una abstracción de distintos problemas que ayuda a proponer algoritmos para su solución. En este trabajo se aplican distintos conceptos y definiciones sobre grafos que permiten diseñar e implementar un algoritmo basado en el método minimax y que será aplicado al problema del coloreo de grafos (PCG). Los principios del PCG proponen usar un número mínimo de colores para que todos los vértices tengan un color y de forma que los vértices adyacentes tengan un color diferente.

Palabras Clave: Heuristica, minimax, teoría de grafos, cuatro coloreo, Kempe.

1 Introducción

Desde mediados del siglo XIX la comunidad matemática empezó a analizar y a plantear posibles soluciones para asignar colores a los vértices de los grafos planares, a este se le conoce como problema de coloreo de grafos (PCG) aplicado a grafos planares (Kaveh, 2022). El problema del coloreo de un grafo planar con a lo más cuatro colores se enunció en forma de conjetura en 1852.
En 1879 A.B Kempe propuso un teorema donde solo eran necesarios cuatro colores para colorear cualquier grafo plano. Kempe demostró que el problema se puede restringir a la consideración de "mapas planos normales" en donde todas las caras son polígonos conectados (Robertson, 1996). Kempe utilizó la fórmula de Euler para mapas cúbicos (Tomé, 2017) para probar que: todo mapa tiene al menos una región con a lo más cinco regiones vecinas, es decir, cada mapa contiene al menos un triángulo, un cuadrado o un pentágono (Tomé, 2017). Al realizar pruebas y buscar un contraejemplo para el teorema de Kempe se presentaron algunos fallos, entre los que se observaba que no todos los ejemplos cumplían por completo el cuatro coloreo, principalmente se observó en la condición de la existencia de contraejemplo de grafos que requerían más de cuatro colores para su coloreo
A pesar de este inconveniente, los ánimos por resolver este problema no han caído, a la fecha se siguen investigando distintos métodos que permitan obtener la solución al cuatro coloreo de grafos planares. Existen distintas propuestas para solucionar el PCG en grafos planares con cuatro colores, entre ellas podemos encontrar algunas heurísticas, las cuales son técnicas desarrolladas de tal manera que son capaces de encontrar soluciones aproximadas a distintos problemas más rápido que los métodos exactos.

En este trabajo se presenta el diseño y aplicación de una heurística basada en el método minimax para el problema de coloreo de grafos. El método minimax aplica la técnica poda alfa-beta que puede dar la mejor solución posible a cualquier grafo, sin importar sus características.

2 Historia del arte

La teoría de grafos tiene una tradición en el área de la matemática discreta, y ha sido de alta utilidad en la modelación de problemas de muy diversa índole, el PCG ha sido uno de los problemas más importantes hasta la fecha y se han realizado distintas propuestas para resolverlo.

Appel, Haken y Koch presentan una propuesta donde introducen los procesos de reducibilidad y descarga, que produce una lista de 1,936 configuraciones inevitables, cada una de las cuales se demuestra reducible con la ayuda de una computadora (Tomé, 2017). Modificando consecutivamente el algoritmo de descarga, proponen un nuevo conjunto de 1,482 configuraciones inevitables, comprobando su reducibilidad mediante un programa realizado por Koch. Este programa requirió de 1.200 horas de cálculo en una IBM 360. Appel y Haken completaron la demostración del teorema de los 4 colores en 1976, tras seis años de análisis del programa implementado, el cual tenía más de 100 páginas de código (Appel, 1977).

Otras propuestas para resolver el PCG, se tiene el trabajo de Muhammad Naeem cuyo objetivo era encontrar el número cromático para una familia de grafos con configuraciones simétricas, usando solamente tres colores (Naeem, 2023). De manera similar Ziwen Huang y su equipo proponen algunas condiciones que satisfacen el cuatrocoloreo sobre grafos formados por ciclos de tres vértices y ciclos de cinco vértices que son adyacentes entre sí (Huang, 2022). En ambos trabajos se observa la propuesta de algoritmos exactos, pero otro enfoque que se le puede dar para resolver el PCG es el uso de heurísticas.

Por ejemplo, en el trabajo de Taha Mostafaie y colaboradores se presenta el uso de distintas metaheurísticas para resolver el PCG y reconocer cual de todas es la que mejor resuelve este problema (Mostafaie, 2022). Otro enfoque es considerar hiper-heurísticas, Nasser R. Sabar y colaboradores realizaron investigación de una hiper-heurística aplicada al PCG para obtener el número cromático de cualquier grafo en tiempos bajos (Sabar, 2011).

3 Funcionamiento de la heurística minimax

En las ciencias computacionales se estudian y diseñan diversos algoritmos que ayudan a resolver un problema, entre ellos se presentan las heurísticas que son algoritmos enfocados en conseguir respuestas aproximadas en corto tiempo. Para el caso del PCG, una heurística se define como una función h que trabajará sobre los nodos de un grafo para mapearlos al campo de los reales que representarán uno de los colores que se les puede asignar (Edelkamp, 2012).

La heurística minimax consiste de dos procesos, uno encargado de maximizar las soluciones, y otro encargado de evaluar todas las soluciones dadas por el proceso de maximización y eliminar aquellas soluciones cuya evaluación tenga un valor por debajo de un umbral establecido. A pesar que el algoritmo minimax suele aplicarse para juegos contra un adversario, en este trabajo se diseñará una heurística minimax para dar solución al problema de coloreo de grafos planos.
La propuesta de la heurística minimax es la siguiente:

```
def

minimax_coloring(G){

colors = {}

   vertices = sorted(G.nodes(), key=lambda x: G.degree[x],

reverse=True)     for vertex in vertices{

      used_colors = set(colors.get(neighbour) for neighbour in

graph.neighbors(vertex))        available_colors = set(range(len(vertices))) -

used_colors        if available_colors{            color = min(available_colors)

colors[vertex] = color}        else{            color = len(vertices)

colors[vertex] = color}}     return colors}
```

En este algoritmo se usan los grados y los vecinos de todos los vértices del grafo para reconocer qué colores no pueden ser dados a un vértice. Al inicio se les asigna un color único a cada uno de los vértices del grafo, en cada iteración se reconocen los vértices adyacentes, se comparan sus colores asignados y en caso de que no sean adyacentes, su color puede ser el mismo cambiando la asignación de color.

4 Poda Alfa-Beta

La técnica poda Alfa-Beta se utiliza comúnmente en la resolución de problemas representados mediante árboles. Su objetivo es reducir el número de nodos existentes en un árbol y eliminar aquellos que no cumplan cierta condición y que por lo tanto no necesitan ser considerados en el resultado final.
Para resolver el problema de coloreo de grafos planos, la poda alfa-beta, que es común aplicarlo en un juego entre dos jugadores, en este caso, un jugador realiza el coloreo y el otro jugador verifica si el coloreo es válido. En el contexto del coloreo de grafos, la técnica poda alfa-beta se puede aplicar de la siguiente manera:

- Se construye el árbol de búsqueda, donde cada nodo representa un estado del coloreo parcial del grafo.
- Se define una función de evaluación que asigna un valor a cada nodo del árbol, representando la calidad del coloreo.
- Se realiza una búsqueda en el árbol de manera recursiva, evaluando los nodos de acuerdo a la función de evaluación y aplicando la poda alfa-beta.
- Durante la búsqueda se mantiene un valor alfa que representa la mejor puntuación que el jugador maximizador ha encontrado hasta el momento, mientras que el valor beta representa la mejor puntuación que el jugador minimizador ha encontrado hasta el momento.
- En cada nodo, se evalúan los movimientos posibles y se podan aquellos que no mejorarán el resultado final. Si se encuentra un nodo que no necesita ser explorado debido a que no afectará el resultado final (según los valores de alfa y beta), se realiza la poda y se retorna al valor actual.
- El algoritmo continúa explorando los nodos restantes y actualizando los valores de alfa y beta según corresponda.
- Finalmente, se retorna el mejor resultado obtenido durante la búsqueda.

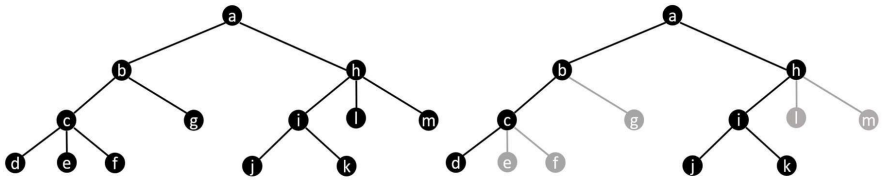


Figura 1. Poda alfa-beta

5 Algoritmo genético

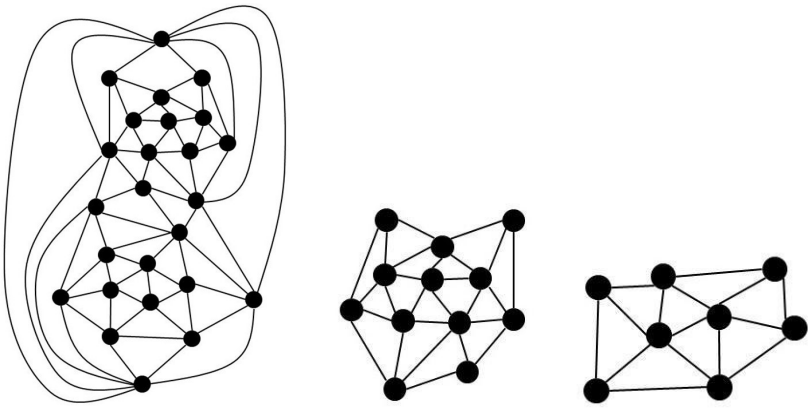
Un tipo de heurísticas comúnmente utilizada en los problemas de optimización son los algoritmos genéticos (AG). Los AG se inspiran en el proceso de evolución por generaciones. Simulan el cruce entre los mejores individuos de una población, cada uno es solución a un problema, de esté cruce se obtiene un individuo completamente nuevo que será miembro de una nueva generación si cumple ciertas condiciones de selección. Este proceso se repite hasta cumplir con un valor de aptitud deseado o una generación límite, y puede necesitar un tiempo de ejecución alto (Iñesta, 2012). En este trabajo se compara el comportamiento de un AG enfocado al PCG contra el algoritmo minimax propuesto para el mismo problema.

6 Comparación de heurísticas.

Ambos algoritmos; minimax y AG, tendrán como entrada la lista de adyacencia de los vértices del grafo. Mientras que el AG realiza una gran cantidad de iteraciones hasta llegar a diez mil generaciones, el algoritmo minimax consta solamente de un ciclo for por cada vértice en el grafo.
Se puede observar que el algoritmo genético tiene una complejidad de O(g (nm + n)) donde g es una función lineal, mientras que para el algoritmo minimax se tiene una complejidad de O(log(n)). Esto dice que existe una gran mejora sobre la rapidez de ejecución del algoritmo minimax sobre el algoritmo genético.

7 Implementación del algoritmo y resultados.

Una vez explicados los conceptos necesarios, presentaremos los resultados obtenidos de implementar en Python el algoritmo minimax propuesto en este trabajo. Cómo datos de entrada al programa se crearon listas de adyacencia que representarán los siguientes grafos.



(a) (b) (c)

Figura 2. Grafos de prueba

Uno de estos grafos, figura(a), es el que Heawood usó como contraejemplo al teorema del cuatro-coloreo propuesto por Kempe.
Los grafos de la figura b y c son grafos que, por su configuración, sus vértices que no son externos forman ruedas impares de grado 5 las cuales al estar unidas por aristas son mas complicadas de colorear utilizando el menor número cromático posible
Los resultados obtenidos se presentan en la siguiente tabla donde se comparan el tiempo requerido y número cromático obtenido por las heurísticas propuestas en este trabajo.

	Algoritmo Genético		Algoritmo minimax	
	Número cromático	Tiempo de ejecución	Número cromático	Tiempo de ejecución
Figura a	8 a 6	30 minutos	4	5 segundos
Figura b	5	25 minutos	4	1 segundo
Figura c	6	15 minutos	4	1 segundo

Tabla 1. Resultados de pruebas de las heurísticas

De los resultados obtenidos se observa que el AG requiere una gran cantidad de tiempo y no siempre dará el mejor resultado posible, es más, suele dar números cromáticos muy grandes. Mientras que el algoritmo minimax ha obtenido el número cromático mínimo. Esto se comprobó al modificar y extender el grafo de la figura (a), formándose un nuevo grafo planar pero donde el algoritmo minimax encontró la solución usando cinco colores, pero cuyo número cromático fue obtenido en cuatro segundos, esta prueba se realizó para comprobar que nuestra propuesta minimax es una heurística y no un algoritmo exacto.

8 Conclusiones y discusión

A pesar de que ambos algoritmos expuestos son heurísticas, se muestra la diferencia en sus tiempos de ejecución. Comparado con el algoritmo genético que también fue implementado por los autores, la heurística minimax otorga un resultado más aceptable, pues ronda entre los cuatro y cinco colores para el coloreo de grafos planares, en comparación con los ocho colores obtenidos con el algoritmo genético, ambos considerando los mismos grafos de entrada.

Se debe recordar que minimax es una heurística implementada para juegos contra adversarios, por eso su funcionamiento sobre un árbol no estará optimizado a la perfección para este tipo de problemas. Sin embargo, la heurística minimax con la técnica poda alfa-beta aprovecha la propiedad de que los jugadores alternan sus movimientos para explorar el árbol de búsqueda de manera eficiente.

Referencias

Kaveh, Ali. (2022). Introduction to Graph Theory and Algebraic Graph Theory.

Yang, Hong, Naeem, Muhammad y Qaisar, Shahid. (2023). On the P3 Coloring of Graphs. Symmetry. P. 15.

Huang, Ziwen, Ma, Jianqing y Zhang, Xiaoxia. (2022). A sufficient condition for planar graphs to be DP -4-colorable. Quaestiones Mathematicae. PP. 1-18.

Mostafaie, Taha & Modarres Khiyabani, Farzin & Navimipour, Nima. (2019). A systematic study on meta-heuristic approaches for solving the graph coloring problem. Computers & Operations Research. P. 120.

Sabar, Nasser & Ayob, Masri & Qu, Rong & Kendall, Graham. (2011). A graph coloring constructive hyper-heuristic for examination timetabling problems. Applied Intelligence - APIN. 37. PP. 1-11.

Tomé, C. (2017, April 24). Una historia que comienza en 1852. Cuaderno De Cultura Científica. https://culturacientifica.com/2017/04/26/teorema-los-cuatrocolores-1-una-historia-comienza-1852/

Tomé, C. (2017, mayo 10). El teorema de los cuatro colores (2): el error de Kempe y la clave de la prueba — Cuaderno de Cultura Científica. Cuaderno de Cultura Científica. https://culturacientifica.com/2017/05/10/teorema-los-cuatrocolores-2-error-kempe-la-clave-la-prueba/

Tomé, C. (2017, May 24). Tras más de un siglo de aventura... ¿un ordenador resuelve el problema? Cuaderno De Cultura Científica. https://culturacientifica.com/2017/05/24/teorema-los-cuatro-colores-3-tras-massiglo-aventura-ordenador-resuelve-problema/

Robertson, N., Sanders, D., Seymour, P., & Thomas, R. (1996). A new proof of the four-colour theorem. Electronic research announcements of the American Mathematical Society, 2(1), PP. 17-25.

Appel, K., Haken, W., & Koch, J. (1977). Every planar map is four-colorable. Part II: Reducibility. Illinois Journal of Mathematics, 21(3), P. 491-567.

Edelkamp, Stefan & Schrödl, Stefan. (2012). Heuristic Search - Theory and Applications. P. 16-19.

Iñesta, J., Calera, J., Carrasco, R., & Pérez-Francisco, A. (2002). Genetic Algorithms for Surface Simplification. En Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). PP. 1235-1242.

Capítulo 9

Answer Set Programming: Solucionador de Conecta 4

Octavio Mendoza[1], Antonio Pérez[1], Fernando Zacarias[1], Rosalba Cuapa[2]

[1] Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
[2] Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias Arquitectura

octaviomen12@gmail.com, antonio.pe.v@gmail.com, fzflores@yahoo.com.mx, rosalbacc2005@yahoo.com.mx

Resumen. La abstracción matemática es una herramienta de la cual nos podemos apoyar para la representación y resolución de problemas, pero con un enfoque distinto al que se acostumbra al momento de programar. Apoyada de este tipo de abstracción, la programación lógica basada en ASP se ha convertido en un paradigma establecido para la representación y el razonamiento del conocimiento, en particular, cuando se trata de resolver problemas complejos. La combinación única de ASP de un lenguaje de modelado simple pero rico con tecnología de resolución de alto rendimiento ha llevado a un creciente interés en ASP tanto en la academia como en la industria. En este artículo, presentamos un juego conocido como conecta 4, que demuestra el uso de un solucionador de programas lógicos comercial llamado "Clingo", como una forma fácil y expresiva de modelar estos problemas de restricciones y encontrar soluciones que satisfagan las restricciones del juego. Adicionalmente, se implementará una heurística con el algoritmo conocido como minimax para calcular el mejor movimiento posible.

Palabras Clave: Conecta 4, Clingo, Answer Set Programing, Programación lógica, Minimax, Heurística.

1 Introducción

La teoría de juegos (Osborne & Rubinstein 1996) hace contribuciones a diversos campos de la inteligencia artificial. En particular, existe una conexión natural con los sistemas basados en agentes. Cuando queremos diseñar un agente inteligente capaz de comportarse inteligentemente en algún entorno, entonces necesitamos proporcionarle a este agente el conocimiento suficiente sobre este entorno. La lógica computacional proporciona soluciones, con un nivel de abstracción suficiente que la naturaleza de su fundamento mismo en lógica permite. Por otro lado, la programación lógica ha evolucionado, lo que ha permitido el desarrollo de propuestas alternativas para la creación de agentes lógicos racionales, como la Programación lógica basada en ASP

(programación lógica basada en el cálculo de conjuntos de respuestas, en inglés – answer set programming) (Gelfond M. and Lifschitz V. 1988). Por lo tanto, necesitamos un lenguaje de modelado que proporcione un marco bien definido, general y riguroso para expresar este conocimiento, junto con alguna forma precisa y bien entendida para manipular conjuntos de oraciones del lenguaje que nos permita sacar inferencias, responder consultas, y actualice tanto la base de conocimientos como el comportamiento deseado del programa. Por lo tanto, sustentaremos nuestra propuesta sobre la semántica del modelo estable que se utiliza actualmente en la programación lógica y que se conoce como Answer Set Programming – Clingo (Potassco project 2000).

En los últimos años, el paradigma de agente que utiliza la programación lógica ha cobrado un papel importante en la definición de las tendencias de la investigación moderna, influyendo en un amplio espectro de disciplinas como la lógica computacional, la filosofía, la programación lógica, la programación basada en conjuntos de respuestas, entre otras ((Dyoub, Costantini and Gasperis 2018) (Gelfond 2004). El paradigma de agente ha invadido todos los subcampos de la informática.

El agente propuesto tiene una base de conocimiento conformada por las siguientes reglas: Primero, los movimientos posibles. Cuando se hace un movimiento, se cambia el turno del jugador. Segundo, la aplicación del algoritmo minimax se utiliza para evaluar posiciones examinando recursivamente todos los posibles movimientos futuros hasta una cierta profundidad y devolviendo la puntuación del mejor movimiento posible. La función heurística minimax se utiliza para evaluar posiciones que han alcanzado la profundidad máxima y, por lo tanto, no se evalúan más.

En la segunda sección se describen los preliminares involucrados en este artículo para el desarrollo de conecta 4, en la tercera el estado del arte con las aplicaciones de ASP es la resolución de problemas en la jerarquía de polinomios, del apartado 4 al 7 se genera el desarrollo del algoritmo, pasando por el desarrollo de la propuesta, el pseudocódigo, su implementación en Python y en Clingo.

2 Preliminares

2.1 Conecta 4

Es un juego de estrategia para dos jugadores que se juega en un tablero vertical que consta de siete columnas y seis filas. Los jugadores se turnan para dejar caer discos de colores (generalmente rojo y amarillo) desde la parte superior del tablero, y el objetivo es conectar cuatro de tus propios discos vertical, horizontal o diagonalmente antes que tu oponente. El juego se gana cuando un jugador coloca con éxito cuatro de sus discos en una fila, y el juego termina en empate si ninguno de los jugadores puede lograrlo después de que se hayan llenado todos los espacios en el tablero. Conecta 4 es un juego popular tanto para niños como para adultos, y se puede jugar tanto en persona como en línea.

El juego fue creado por la empresa de juguetes y juegos estadounidense Milton Bradley (ahora parte de Hasbro) en 1974. Fue inventado por Howard Wexler y Ned Strongin, quienes se inspiraron en un juego similar llamado "Captain's Mistress", que se jugaba en un tablero mucho más grande.
El nombre original del juego era "Connect Four" y rápidamente se convirtió en un juego popular en las familias y en un éxito de ventas para Milton Bradley. En los años siguientes, el juego se comercializó en todo el mundo bajo diferentes nombres, como "Four in a Row" y "Plot Four". La popularidad del juego creció aún más cuando apareció en la película de 1984 "WarGames", en la que el protagonista jugaba una partida de Conecta 4 contra un ordenador. Desde entonces, el juego se ha convertido en un clásico y ha sido disfrutado por millones de personas en todo el mundo.
La evaluación de todos los movimientos posibles se realiza mediante la evaluación de las celdas vacías, todos los posibles movimientos se generan encontrando la celda vacía superior en cada columna o encontrando la celda vacía encima de la celda ocupada superior en una columna. Una vez realizado un movimiento, se cambia el turno del jugador. La heurística aplica el algoritmo minimax (Schiffel & Thielscher 2009) que se utiliza para evaluar posiciones examinando recursivamente todos los posibles movimientos futuros hasta una cierta profundidad y devolviendo la puntuación del mejor movimiento posible. La función heurística del algoritmo minimax se utiliza para evaluar posiciones que han alcanzado la profundidad máxima y, por lo tanto, no se evalúan más.
Para saber cuál es el mejor movimiento dentro de todas las opciones posibles, la función heurística evalúa las posiciones horizontales, verticales y diagonales al buscar una cantidad especifica de piezas consecutivas del mismo tipo en una fila. Si se encuentra una ganancia, a la puntuación se le agrega un valor 100 para el jugador 1 o se le quita 100 para el jugador 2. Si no se encuentra ninguna ganancia, la puntuación se establece en 0.

2.2 Minimax

El algoritmo Minimax (Schiffel & Thielscher 2009) es un método utilizado en la inteligencia artificial para tomar decisiones óptimas en situaciones de juego de dos jugadores con información perfecta, como el ajedrez, las damas, el Conecta 4, entre otros. El algoritmo funciona generando un árbol de búsqueda que representa todas las posibles jugadas y respuestas a esas jugadas, desde el estado actual del juego hasta un estado final. El algoritmo determina el mejor movimiento que un jugador puede hacer en cada nivel del árbol de búsqueda, asumiendo que el oponente también está jugando de manera óptima. En el nivel superior del árbol, el algoritmo considera todas las posibles jugadas del jugador actual y simula la respuesta óptima del oponente a cada jugada. Luego, evalúa cada uno de estos posibles resultados para determinar cuál es la mejor opción para el jugador actual, utilizando una función de evaluación heurística. Este proceso se repite recursivamente para todos los niveles del árbol de búsqueda, alternando entre el jugador actual y el oponente, hasta que se alcanza un estado final del juego.

El algoritmo Minimax, entonces, devuelve el mejor movimiento que puede hacer el jugador actual en el nivel superior del árbol de búsqueda de todos los posibles movimientos que serán evaluados.

2.3 Suma cero

El juego de suma cero es un tipo de juego en el que el beneficio de un jugador se produce a expensas del otro jugador. En otras palabras, la suma de las ganancias y pérdidas de los jugadores es siempre cero.

Un ejemplo de juego de suma cero es el juego de "piedra, papel o tijera". Si un jugador gana, el otro pierde. La suma de las ganancias y pérdidas siempre será cero.

Los juegos de suma cero son comunes en la teoría de juegos y pueden aplicarse a situaciones en las que dos jugadores tienen intereses opuestos. En algunos casos, como en una negociación, los jugadores pueden trabajar juntos para maximizar su ganancia colectiva. En otros casos, como en un conflicto militar, los jugadores deben trabajar para minimizar las pérdidas del otro jugador mientras maximizan su propia ganancia.

El juego Conecta 4 se considera un juego de suma cero porque cada jugador compite por alcanzar el objetivo de conectar cuatro fichas del mismo color en una fila, columna o diagonal. Si un jugador logra conectar cuatro fichas, gana el juego, mientras que el otro jugador pierde. Por lo tanto, el beneficio de un jugador se produce a expensas del otro jugador, y la suma total de las ganancias y pérdidas siempre será cero. Es importante destacar que, aunque el juego es de suma cero, esto no significa necesariamente que sea un juego de suma nula, es decir, que no haya beneficios externos o efectos secundarios para los jugadores o para terceros.

2.4 Poda alfa-beta para optimizar

El algoritmo de Conecta cuatro es de búsqueda de árboles utilizado en juegos de dos jugadores con información perfecta, como el ajedrez o el tres en raya. La idea principal detrás del algoritmo es que el jugador MAX intenta maximizar su puntuación, mientras que el jugador MIN intenta minimizar la puntuación de MAX.

Funciona explorando todos los posibles movimientos y estados del juego, construyendo un árbol de búsqueda. Cada nodo del árbol representa un estado del juego y cada arco representa un movimiento posible. La puntuación en cada nodo hoja es la puntuación final del juego. La puntuación se asigna de forma que los jugadores MAX intentan maximizar y los jugadores MIN intentan minimizar.

Se utiliza la técnica de búsqueda en profundidad para explorar todos los posibles movimientos. Sin embargo, la cantidad de nodos en el árbol de búsqueda crece exponencialmente con la profundidad del árbol, lo que hace que la exploración completa sea imposible para juegos complejos.

Para solucionar este problema, se utiliza la técnica de poda alfa-beta, que reduce el número de nodos explorados. El algoritmo de poda alfa-beta mantiene dos valores en cada nodo del árbol: alfa y beta. Alfa representa la puntuación mínima que MAX puede alcanzar al

explorar el subárbol izquierdo del nodo, mientras que beta representa la puntuación máxima que MIN puede alcanzar al explorar el subárbol derecho del nodo.
La poda alfa-beta funciona eliminando los subárboles que no necesitan ser explorados. Si el valor de beta en un nodo es menor o igual al valor de alfa en otro nodo hermano, entonces se puede eliminar todo el subárbol derecho del primer nodo, ya que MIN nunca elegiría esa rama, dado que MAX tiene una mejor opción disponible. De manera similar, si el valor de alfa en un nodo es mayor o igual al valor de beta en otro nodo hermano, se puede eliminar todo el subárbol izquierdo del primer nodo, ya que MAX nunca elegiría esa rama, dado que MIN tiene una mejor opción disponible.

3 Estado del arte

3.1 Jerarquía de polinomios

Una de las aplicaciones de ASP es la resolución de problemas en la jerarquía de polinomios. La jerarquía de polinomios es una clasificación de los problemas de complejidad computacional que se basa en la cantidad de términos de un polinomio en una determinada variable. Los problemas de menor complejidad son aquellos que pueden resolverse en tiempo polinómico (es decir, su complejidad es O(n^k), donde n es el tamaño de la entrada y k es una constante), mientras que los problemas de mayor complejidad son aquellos que requieren tiempo exponencial para su resolución.
Para resolver problemas en la jerarquía de polinomios con ASP, es necesario modelar el problema en términos de restricciones lógicas y utilizar un solucionador de ASP para encontrar una solución que satisfaga esas restricciones. El modelo debe reflejar la estructura del problema, incluyendo sus variables y relaciones, y puede incluir restricciones adicionales para mejorar la eficiencia del solucionador.

3.2 Satisfactibilidad y ASP

Tanto los algoritmos basados en satisfactibilidad como los basados en ASP pueden ser utilizados para medir inconsistencias en un conjunto de reglas o restricciones. Sin embargo, hay algunas diferencias en la forma en que estos dos tipos de algoritmos abordan el problema de la medición de inconsistencias.
Los algoritmos basados en satisfactibilidad booleana se enfocan en encontrar una asignación de valores a las variables que satisfagan un conjunto de restricciones dadas. En la medición de inconsistencias, esto significa encontrar una asignación de valores que haga que un conjunto de reglas o restricciones se vuelva inconsistente. Los algoritmos basados en satisfactibilidad pueden ser muy eficientes en encontrar una solución, especialmente para problemas de satisfactibilidad de restricciones NP-completos.
Por otro lado, los algoritmos basados en ASP (programación de conjuntos de respuesta) se enfocan en encontrar un conjunto mínimo de reglas o restricciones que, si se eliminan, hacen que el conjunto restante sea consistente. Esto se conoce como el conjunto de respuesta. Los algoritmos basados en ASP son muy útiles en problemas donde el objetivo es eliminar la menor cantidad posible de reglas o restricciones para lograr consistencia. También pueden

ser eficientes en la medición de inconsistencias, especialmente en problemas donde las restricciones están altamente relacionadas.
En términos generales, los algoritmos basados en satisfactibilidad son más adecuados para la medición de inconsistencias en problemas grandes y complejos, donde encontrar una solución es el objetivo principal. Los algoritmos basados en ASP son más adecuados para la medición de inconsistencias en problemas donde se busca eliminar la menor cantidad de restricciones para lograr consistencia, especialmente en problemas con restricciones altamente relacionadas.

4 Desarrollo de la propuesta

Una vez presentado el concepto de la función minimax, presentaremos la propuesta de un algoritmo minimax que será implementado en una IA de tal manera que se base en un agente racional, quien tomará decisiones y acciones óptimas en función de un conjunto de reglas y objetivos específicos. En este caso, el agente racional puede jugar al juego Conecta 4 de manera autónoma y tomar decisiones óptimas para ganar el juego.
En nuestro algoritmo implementaremos el agente racional para Conecta 4 utilizando el algoritmo minimax con poda alfa-beta para determinar la mejor jugada en función de la posición actual del tablero, la cual consiste en crear el árbol de todas las posibles soluciones, evalúa las mejores y "poda" todas aquellas cuyo valor de evaluación sea menor a un umbral deseado.

5 Pseudocódigo

El algoritmo propuesto es el siguiente:

```
minimax(estado_tablero, profundidad_arbol, alpha, beta, jugador_maximizador){
posiciones_validas = evaluar_posicion(estado_tablero) es_terminal =
es_nodo_terminal(estado_tablero) if (profundidad_arbol == 0 |
es_nodo_termina){ if (es_nodo_terminal){
        if movimiento_ganador(estado_tablero, IA){
        return(None,10000000)} elif
        movimiento_ganador(estado_tablero,IA){
        return(None,-10000000)}
                else {return(None,0)}}
        else{return(None, puntaje_posicion(estado_tablero,IA)} if
        (jugador_maximizador){
                valor = -infinito
                columna = seleccion_al_azar(localizacion_valida) for localizacion
                in localizacion_valida{ fila =
                calcular_fila_disponible(estado_tablero, localizacion) copia_tablero
```

```
                = estado_tablero.copy() tirar_en_posicion(copia_tablero, fila,
                localizacion, IA)
                        nueva_puntuacion = minimax(copia_tablero, profundidad-1,
                alpha,      beta, False)[1]} if
                        nueva_puntuacion > valor{ valor
                        = nueva_puntuacion columna =
                        localizacion}
                        alpha = max(valor, alpha) if
                        alpha >= beta{break}}}
        else{ valor = infinito columna =
        seleccion_al_azar(localizacion_valida) for localizacion in
        localizacion_valida{
                fila = calcular_fila_disponible(estado_tablero, localizacion)
                copia_tablero = estado_tablero.copy()
                tirar_en_posicion(copia_tablero, fila, localizacion, Jugador)
                        nueva_puntuacion = minimax(copia_tablero, profundidad-1,
                alpha,              beta, True)[1]
                if nueva_puntuacion < valor{ valor
                        = nueva_puntuacion
                        columna = localizacion}
alpha = min(valor, beta)
    if alpha >= beta{break}}}}
```

El pseudocódigo anterior implementa el algoritmo minimax con poda alfa-beta para un juego en un tablero representado por la variable "estado_tablero".

El objetivo del algoritmo es buscar el mejor movimiento que un jugador puede realizar en un juego, explorando todas las posibles secuencias de movimientos que se pueden dar. El algoritmo trabaja evaluando los movimientos de cada jugador y determinando cuál es el movimiento que maximiza el beneficio para un jugador, mientras que minimiza el beneficio para el otro jugador.

En este algoritmo, el jugador que desea maximizar el beneficio se conoce como el "jugador maximizador" y el jugador que desea minimizar el beneficio se conoce como el "jugador minimizador". El algoritmo funciona evaluando el estado actual del tablero, y a partir de él, construyendo un árbol de posibles jugadas para el jugador maximizador y el jugador minimizador.

Para construir este árbol, el algoritmo se llama a sí mismo recursivamente para explorar cada uno de los posibles movimientos que se pueden hacer en cada nivel del árbol. El algoritmo evalúa el beneficio que se puede obtener de cada movimiento y elige el movimiento que maximiza el beneficio para el jugador maximizador, y minimiza el beneficio para el jugador minimizador.

La poda alfa-beta es una técnica de optimización que se utiliza para reducir la cantidad de nodos que el algoritmo debe explorar para buscar el mejor movimiento. Esto se logra mediante la evaluación de los nodos del árbol en orden, de tal manera que si se encuentra un nodo que no mejorará la solución actual, se poda todo el subárbol asociado a ese nodo, ahorrando tiempo de procesamiento.
El algoritmo se ejecuta recursivamente, evaluando cada nivel del árbol de posibles jugadas hasta alcanzar la profundidad deseada. Una vez que se alcanza la profundidad deseada o se llega a un nodo terminal, es decir, un nodo que representa un estado en el que ya no se pueden hacer más movimientos, el algoritmo evalúa la posición actual del tablero.
El algoritmo devuelve una tupla con dos valores: el primer valor es la columna en la que se debe hacer el movimiento, y el segundo valor es la puntuación asociada a ese movimiento.

6 Implementación en Clingo

En la implementación en Clingo se debe implementar el paradigma ASP, donde debemos crear líneas de código formadas por una preposición seguida por las condiciones que permitirán que se cumpla. Para poder empezar un juego de conecta cuatro debemos declarar las partes esenciales del juego, que son un tablero, los jugadores, las fichas y las celdas donde es posible tirar. En conecta cuatro tenemos un tablero que es una matriz de tamaño 7x6 donde las piezas se colocarán respecto a una posición definida por un par ordenado X, Y. Nuestro tablero será creado con un predicado de nombre cell que nos permite saber dónde se pueden mover las fichas de acuerdo con las dimensiones anteriormente especificadas.

```
cell(1..7, 1..6).
```

El siguiente paso es otorgar un turno al jugador, el cual cambiará cada vez que se realice un movimiento por jugador. Usamos currentPlayer(_)para indicar si es que el jugador actual es el que está realizando el movimiento, si se cumple dicha condición se le otorgará un valor booleano True y en caso contrario será False. Cada jugador será reconocido por fichas, “X” en el caso del jugador humano y “O” en el caso de la inteligencia artificial y solo podrán tirar en aquellas celdas evaluadas como vacías. Para reconocer las casillas vacías se usa un predicado con dos condiciones, empty(X, Y) será verdadero solo cuando no haya ningún valor dentro de dicha celda y sea una celda dentro del rango (X,Y). empty(X, Y) :- cell(X, Y), not value(X, Y, _).
Una vez definidas las bases del juego, debemos definir las acciones que se pueden realizar sobre estas mismas. Los movimientos son la parte más vital de un juego de conecta cuatro, pues dependiendo del movimiento que haga nuestro contrincante nosotros podemos realizar un movimiento nos puede acercar a la victoria o bloquear al contrincante. Las acciones que realizamos ante un movimiento es pensar en varios posibles movimientos, evaluar cual de estos es el mejor y una vez decidido cuál realizar entonces procedemos a realizarlo. Para cada una de estas acciones se crea un predicado con sus condiciones respectivas, empezando

por los posibles movimientos, en lugar de realizar un movimiento sobre toda la matriz lo realizaremos sobre un vector el cual se moverá solamente entre las columnas del eje X dejando estable el movimiento sobre una de las seis filas del eje Y, es decir, se escogerá un valor X bajo dos condiciones,

Este código define el estado del juego e implementa un algoritmo minimax con una función heurística para evaluar la posición. El estado del juego está representado por un predicado cell(X, Y) que define la posición de cada celda en el tablero, y los predicados value(X,Y,P) y empty(X,Y) que definen el valor (color de la pieza) de una celda y si está vacía, respectivamente. El jugador actual está representado por el predicado currentPlayer(P). Los posibles movimientos son generados por el predicado move(X), que busca celdas vacías en la fila inferior y en la fila superior de cualquier pieza existente. El predicado evalúa la puntuación de una posición score (X, Score), que primero comprueba si el jugador actual puede ganar con el movimiento X y asigna una puntuación de 100 o 100 según corresponda. Si no hay una ganancia inmediata, el algoritmo llama minimax para evaluar recursivamente la posición con una profundidad más baja. La puntuación de la posición es el máximo o mínimo de las puntuaciones de los posibles movimientos, dependiendo del jugador. El predicado makeMove(X)actualiza el estado del juego colocando una parte del jugador actual en la celda X. El predicado undoMove(X)deshace el movimiento quitando la pieza de la celda X. La función heurística está definida por el predicado heuristic(Score), que verifica las ganancias horizontales, verticales y diagonales y asigna una puntuación de 100 o -100 según corresponda. Si no hay victoria, la puntuación es 0.

7 Conclusiones

El modelado de este problema utilizando el lenguaje de programación lógica Clingo mediante un agente racional ha demostrado su poder expresivo, simple y con características novedosas en términos de representación de conocimiento y razonamiento sobre acciones, permitiendo codificar incluso problemas de planificación difíciles con precondiciones alternativas de acciones.

Una experiencia que resulta de este trabajo es que el modelado de Conecta 4 con el algoritmo minimax en Clingo, tiene las ventajas que proporciona la programación lógica ASP, ya que es muy diferente a la clásica programación imperativa en donde se le debe decir exactamente que hacer al programa.

El algoritmo minimax crea un árbol bastante amplio de posibles jugadas, pero gracias a la poda Alfa-Beta se logra mejorar en gran medida este algoritmo y de este modo el agente realiza decisiones inteligentes para poder ganarle al usuario. Aun así, no es infalible, por lo que como trabajo futuro se pueden agregar ciertas restricciones para aumentar el poder de este algoritmo. Además de la posibilidad de expandir este juego para más de 2 jugadores, es decir, la generalización del problema a conecta “k”, para “k” mayor que 4. Para este apartado la implementación en Clingo será la más optima porque permite reutilizar los mismos predicados expuestos anteriormente y agregarles ligeras modificaciones de tal

manera que se generé un conecta “k” que no necesitará de un gran procesamiento ni tiempo de cómputo.

Por lo tanto, el paradigma ASP es una gran herramienta para resolver problemas, a pesar de tener un estilo de programación más parecido posee un costo computacional mucho menor al método de programación convencional y sobre todo existen lenguajes que nos permiten implementarlo que no son tan demandantes y no requieren de un espacio tan grande en la memoria y almacenamiento de una computadora.

Referencias

Documentation. (s/f). Potassco.org. Recuperado el 22 de agosto de 2023, de https://potassco.org/doc/

Networks. (s/f). Cornell.edu. Recuperado el 22 de agosto de 2023, de https://blogs.cornell.edu/info2040/2015/09/21/solving-connect-four-with-game-theory/

Clare, L. (2018, septiembre 11). *How to win Connect 4*. Base Camp Math. https://basecampmath.com/how-to-win-connect-4

Hahn, S., Sabuncu, O., Schaub, T., & Stolzmann, T. (2023). Clingraph: A system for ASP-based visualization. En *arXiv [cs.AI]*. http://arxiv.org/abs/2303.10118

Wang, C., Yu, Z., McAleer, S., Yu, T., & Yang, Y. (2023). ASP: Learn a universal neural solver! En *arXiv [cs.LG]*. http://arxiv.org/abs/2303.00466

Kuhlmann, I., Gessler, A., Laszlo, V., & Thimm, M. (2023). Comparison of SAT-based and ASP-based algorithms for inconsistency measurement. En *arXiv [cs.AI]*. http://arxiv.org/abs/2304.14832

Amendola, G., Cuteri, B., Ricca, F., & Truszczynski, M. (2022). Solving problems in the polynomial hierarchy with ASP(Q). En *Logic Programming and Nonmonotonic Reasoning* (pp. 373–386). Springer International Publishing.

Wotawa, F., & Kaufmann, D. (2022). Model-based reasoning using answer set programming. *Applied Intelligence*, *52*(15), 16993–17011. https://doi.org/10.1007/s10489-022-03272-2

Gelfond, M., & Lifschitz, V. (1988). The Stable Model Semantics For Logic Programming. En R. Kowalski & K. Bowen (Eds.), *5th Conference on Logic Programming* (pp. 1070–1080). MIT Press.

Machuca, M. (2013, agosto 25). *Conecta Cuatro*. aboutespanol; AboutEspañol. https://www.aboutespanol.com/conecta-cuatro-2077685

Potassco Project. (*2000*). The Potsdam Answer Set Solving Collection. Tools for Answer Set Programming developed at the University of Potsdam. Potassco.org. Recuperado el 22 de agosto de 2023, de https://potassco.org/

Answer set programming and agents. (2018). *The Knowledge Engineering Review*, *33*, 1–30.

Answer Set Programming and the Design of Deliberative Agents. the Lecture Notes in Computer Science book series (Vol. 3132). (2004). Michael Gelfond.

Automated Theorem Proving for General Game Playing. (2009). En *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence IJCAI-09*. Publisher AAAI Press.

Capítulo 10

Sobre la tratabilidad de la clase 2μ-3MON

Sonia Navarro Flores[1], Carlos Guillén Galván[2]

[1] Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias Físico Matemáticas
[2] Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias Físico Matemáticas

sonianavarroflores91@gmail.com, carlosguillen.galvan@gmail.com

Resumen. En el presente trabajo se exponen diversos resultados sobre el uso del ancho hiper-arbóreo de hipergrafos para buscar instancias de Problemas de Satisfacción de Restricciones donde estos problemas sean tratables. Además, presentamos diversos hipergrafos que provienen de la clase sintáctica 2μ-3MON así como un análisis de su ancho hiper-arbóreo y las ideas que tenemos para poder determinar si algún ancho hiper-arbóreo puede garantizar la tratabilidad de la clase.

Palabras Clave: Problemas de Satisfacción de Restricciones, Hipergrafos, Ancho fraccional hiper-arbóreo.

1 Introducción

Los Problemas de Satisfacción de Restricciones (PSR) consisten en una cantidad finita de variables con dominio finito y una cantidad finita de restricciones sobre los valores que las variables pueden tomar de manera simultánea. La importancia de estos problemas yace en que problemas importantes de inteligencia artificial, investigación de operaciones y consultas de bases de datos son PSR. En general los PSR son problemas NP-completos y en ciencias de la computación se han desarrollado diversos métodos de búsqueda, así como heurísticas para poder resolverlos de forma eficiente (ver Tsang, 2014).

Para los problemas que involucran grafos, se sabe que de manera frecuente la aciclicidad de los grafos garantiza la tratabilidad del problema. Por esta razón se introdujeron diversas medidas que pretenden capturar la aciclicidad de un grafo, siendo el ancho arbóreo la medida óptima (ver Downey y Fellows, 2013). Cabe mencionar que hay muchos problemas que no involucran grafos en su enunciado original, pero se pueden modelar a través de grafos.

Para los problemas modelados con grafos se sabe que tener ancho arbóreo acotado es equivalente a ser tratable (ver Downey y Fellows, 2013). Sin embargo, la mayoría de los PSR se modelan mejor a través de hipergrafos por lo cual se han definido algunos anchos

hiper-arbóreos con la finalidad de conocer parámetros bajo los cuales los PSR son parámetro fijo tratables (ver Gottlob et al, 2014; Grohe y Marx, 2014).

El problema SAT es un problema muy importante en las ciencias computacionales. Este problema ha recibido mucha atención pues tiene aplicaciones en diversas áreas como en la verificación de sistemas de tiempo real e incrustados, y en el problema de planificación en inteligencia artificial.

Aunque el problema SAT es NP-completo, usando la teoría de la complejidad parametrizada se han identificado diversas clases de fórmulas para las cuales el problema SAT se resuelve en tiempo polinomial. Por ejemplo, hay un teorema de dicotomía del problema SAT que dice que la clase de todas las subfórmulas permitidas en una fórmula dada es o bien polinomial o bien NP-difícil dependiendo de las características de la fórmula dada. El lector interesado en más ejemplos puede ver Szeider (2003), así como de Haan y Szeider (2014). Otra aproximación de estudio consiste en tratar al problema SAT como un PSR y modelar las fórmulas a través de hipergrafos. Este último enfoque es el que trataremos en este trabajo.

2 Preliminares

2.1 Problemas de Satisfacción de Restricciones

Los Problemas de Satisfacción de Restricciones (PSR) se componen de una cantidad finita de variables $x_1, x_2, \dots, x_n$ donde cada variable x_i tiene asociado un dominio finito, y un conjunto de restricciones $\{C_{x_i,\dots,x_k} | i, \dots, k \in 1,2,\dots,n\}$ tal que $C_{x_i,\dots,x_k}$ es el conjunto de todas asignaciones permitidas para las variables $x_i, \dots, x_k$. Para resolver el problema se debe encontrar una asignación para todas las variables de forma que se satisfagan todas las restricciones. Los PSR en general son problemas NP-completos (ver Tsang, 2014).

A cada PSR se le puede asociar un hipergrafo de la siguiente manera, las variables del PSR se toman como vértices y un conjunto de vértices es una hiperarista si hay una restricción que considera exactamente ese conjunto de variables.

Ejemplo (Gottlob et al, 2014). Resolver un crucigrama se puede ver como un PSR si tomamos cada posible letra como una variable, el dominio de cada variable es el alfabeto y las restricciones son que las columnas verticales y horizontales deben formar palabras.

1	2	3	4	5		6
7				8	9	10
11	12	13		14		15
16		17		18		19
20	21	22	23	24	25	26

Figura 1. Crucigrama

En el crucigrama mostrado en la Figura 1, las variables son $x_1, x_2, \ldots, x_{26}$. El dominio de cada variable es el alfabeto del español {a, b, c,…,z}. Las restricciones son las cuatro palabras horizontales y las cuatro palabras verticales que se deben formar. Así, con las variables y las restricciones podemos formar el hipergrafo asociado a este problema.

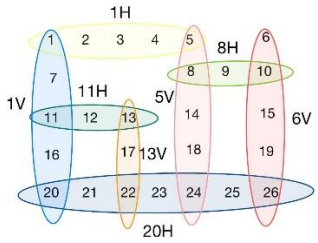


Figura 2. Hipergrafo asociado al crucigrama

2.2 El Problema SAT

Para definir el problema SAT mencionaremos algunas definiciones sobre fórmulas booleanas.

Una variable booleana es cualquier símbolo al que se le pueden asociar los valores 0 y 1. Las fórmulas booleanas se forman con las constantes 0 y 1, variables booleanas y conjunciones, disyunciones y negaciones de las anteriores. Además, decimos que una fórmula booleana es satisfactible si existe una asignación para sus variables que hace verdadera a la fórmula. El problema SAT consiste en decidir si cualquier fórmula booleana libre de cuantificadores es satisfactible.

Una literal es o bien una variable booleana o bien la negación de ella. Una cláusula es una disyunción de literales. Las cláusulas tienen la forma $C = l_1 \vee l_2 \vee \ldots \vee l_n$. Finalmente, decimos que una fórmula F está en su forma normal conjuntiva si es la conjunción de un conjunto de cláusulas. Las fórmulas en su forma normal conjuntiva tienen la forma $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$. La razón por la que presentamos las fórmulas FNC es porque cualquier fórmula booleana libre de cuantificadores se puede expresar en su forma normal conjuntiva y esta forma nos permite tratar al problema SAT como PSR. Para esto, se considera a las variables de la fórmula como las variables del problema, el dominio de cada variable es el conjunto {0,1} y las restricciones son las cláusulas de la fórmula.

Una fórmula pertenece a la clase 2µ-3MON si al escribirla en su FNC cada clausula tiene a lo más tres variables, ninguna variable aparece negada y cada variable puede ocurrir en a lo más dos clausulas. La siguiente fórmula es un ejemplo de una fórmula en 2µ-3MON, $F = (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee x_5)$. Por lo tanto, los hipergrafos asociados a esta clase sintáctica son hipergrafos donde cada hiperarista contiene a lo más tres vértices y cada vértice pertenece a lo más a dos hiperaristas.

2.3 Hipergrafos

En esta sección presentaremos conceptos de hipergrafos y descomposiciones arbóreas que son importantes para seguir el texto.

Definición. Un hipergrafo es una pareja H = (V(H), E(H)) donde V(H) es el conjunto de vértices y E(H) contiene subconjuntos no vacíos de V(H). Donde los elementos de E(H) se llaman hiperaristas y cada vértice debe pertenecer a alguna hiperarista. En adelante identificaremos a los hipergrafos con conjuntos de hiperaristas.

Dado un hipergrafo H, el grafo primal de H es el grafo cuyos vértices son los vértices de H y un par de vértices están relacionados en G si comparten una hiperarista en H. Además, el tamaño de un hipergrafo se define como |H|=|V(H)|+|E(H)|*|V(H)|.

Ejemplo. Sea H el hipergrafo que cumple que V(H)={0, 1, 2, 3, 4, 5, 6, 7, 8} y E(H)={{0, 1, 2}, {0, 3, 4}, {5, 6, 7}, {1, 5, 8}, {2, 3, 6}, {4, 7, 8}}. El grafo primal de H consiste de las aristas {0, 1}, {0, 2}, {0, 3}, {0, 4}, {1, 2}, {1, 5}, {1, 8}, {2, 3}, {3, 4}, {3, 6}, {4, 7}, {4, 8}, {5, 6}, {5, 7}, {5, 8}, {6, 7}, {7, 8}. A continuación, presentamos visualizaciones de H (vea Figura 3) su grafo primal (vea Figura 4).

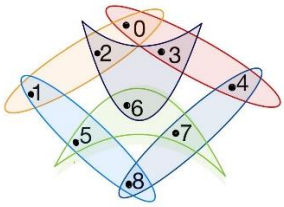


Figura 3. Hipergrafo H

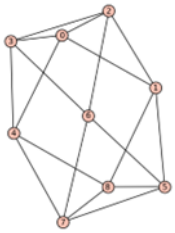


Figura 4. Grafo primal de H

2.4 Anchos asociados a hipergrafos

En esta subsección presentamos diversos anchos que se han definido con el objetivo de medir la aciclicidad de los hipergrafos.

Definición. Una descomposición arbórea de un hipergrafo H es una pareja $\langle T, (B_u)_{u \in N(T)} \rangle$ donde T=(N(T), E(T)) es un árbol y cada nodo u ∈ N(T) tiene asociado un conjunto de vértices $B_u \subseteq V$ que satisface las siguientes condiciones:

1) para cada e ∈ E(H) existe u ∈ N(T) tal que $e \subseteq B_u$,
2) para cada v ∈ V(H), el conjunto {u ∈ N(T) | v ∈ B_u} es conexo.

A los conjuntos B_u les llamamos bolsas y en adelante identificaremos a u con B_u. El último punto nos dice que, si fijamos un vértice de H, la colección de todas las bolsas que lo contienen forman un subgrafo conexo de T.

El ancho de una descomposición arbórea $\langle T, (B_u)_{u \in N(T)} \rangle$ es el tamaño de la bolsa más grande menos uno, es decir máx{|B_u|-1 : u ∈ $N(T)$}. El ancho arboreo de H es el mínimo ancho de todas sus descomposiciones arbóreas.

Ejemplo. El árbol que presentamos a continuación (vea Figura 5) es una descomposición arbórea de H, lo encontramos usando SageMath. El árbol consiste en un camino de 5 nodos, las bolsas de T son los conjuntos {0, 2, 3, 4, 6}, {0, 1, 2, 4, 6}, {0, 1, 4, 6, 8}, {1, 4, 5, 6, 8} y {4, 5, 6, 7, 8}.

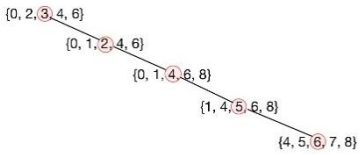


Figura 5. Árbol de descomposición de H

El ancho arbóreo de un hipergrafo es la generalización inmediata del ancho arbóreo en grafos. Para clases de hipergrafos de tamaño acotado, similar a como ocurre con grafos el tener ancho arbóreo acotado garantiza tratabilidad (Grohe, 2007). Sin embargo, existen clases de hipergrafos con tamaño no acotado tales que, aunque tienen ancho arbóreo no acotado sí son tratables. Como el ancho arbóreo no captura la aciclicidad de los hipergrafos de forma precisa cuando trabajamos con clases de hipergrafos de tamaño no acotado, se definieron otros anchos más generales que aproximan mejor la aciclicidad de los hipergrafos (Gottlob et al, 1999; Gottlob et al, 2014; Grohe y Marx, 2014).

Ahora vamos a definir las nociones de cubierta por aristas y cubierta fraccional para hipergrafos pues son necesarias para hablar de descomposiciones generalizadas y fraccionales.

Definición. Dado un hipergrafo H y conjunto de vértices A ⊆ V(H), una cubierta por aristas de A en H es una función c : E(H) →{0,1}de forma que para cada v ∈ A existe una hiperarista que contiene a v tal que c(e)=1.

Definición. Dado un hipergrafo H y un conjunto de vértices A ⊆ V(H), una fraccional cubierta por aristas de A en H es una función c : E(H) →[0,1] de forma que para cada v ∈ A, la suma de los pesos de las hiperaristas a las que pertenece v es al menos 1, es decir $\sum_{v \in e} c(e) \geq 1$.

El peso de c es la suma de los valores que asigna a todas las hiperaristas, es decir $\sum_{e \in E(H)} c(e)$.

Una descomposición hiper-arbórea generalizada de un hipergrafo H es una tercia $\langle T, (B_u)_{u \in N(T)}, (c_u)_{u \in N(T)} \rangle$ donde:

1) $\langle T, (B_u)_{u \in N(T)} \rangle$ es una descomposición arbórea de H,
2) para cada u ∈ N(T), c_u es una cubierta por aristas para B_u en H.

Ejemplo. Para este ejemplo enumeramos las hiperaristas del hipergrafo H que mencionamos en los ejemplos anteriores. e_1={0, 1, 2}, e_2={0, 3, 4}, e_3={5, 6, 7}, e_4={1, 5, 8}, e_5={2, 3, 6}, e_6={4, 7, 8}. Una descomposición hiper-arbórea generalizada consiste en el camino compuesto por las bolsas {0, 2, 3, 4, 5, 6, 7} y {0, 1, 2, 4, 5, 7, 8}, con sus respectivas cubiertas por aristas {e_2, e_3, e_5} y {e_1, e_4, e_6}.

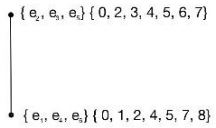


Figura 6. Descomposición hiper-arbórea de H

Una descomposición hiper-arbórea es una descomposición hiper-arbórea generalizada que además cumple una tercera propiedad que se incluyó para que el computar una descomposición generalizada sea tratable.

Una descomposición hiper-arbórea fraccional de un hipergrafo H es una tercia $\langle T, (B_u)_{u\in N(T)}, (c_u)_{u\in N(T)}\rangle$ donde:

1) $\langle T, (B_u)_{u\in N(T)}\rangle$ es una descomposición arbórea de H,
2) para cada u ∈ N(T), c_u es una cubierta fraccional por aristas para B_u en H.

El ancho de una descomposición hiper-arbórea, de una descomposición hiper-arbórea generalizada, y de una descomposición hiper-arbórea fraccional es el máximo peso de todas las cubiertas por aristas c_u. El ancho hiper-arbóreo de H (hw(H)), el ancho hiper-arbóreo generalizado de H (ghw(H)) y el ancho hiper-arbóreo fraccional de H (fhw(H)) se definen como el mínimo ancho de todas sus respectivas descomposiciones.

Se sabe que determinar el ancho hiper-arbóreo de un hipergrafo es NP-completo sin embargo, dado un k fijo, se puede determinar en tiempo polinomial si hw(H)≤ k ($Gottlob\ et\ al,$1999; $Gottlob\ et\ al,$2000). Desafortunadamente, para los anchos fraccional y generalizado, determinar si fhw(H)≤ k o ghw(H)≤ k es NP-completo incluso para k=2 (Gottlob et al, 2021). Además, se sabe que para cualquier hipergrafo H, fhw(H) ≤ ghw(H) ≤ hw(H).

Un problema computacional es Parámetro Fijo Tratable (PFT) bajo algún parámetro fijo k si el problema se puede resolver en tiempo $f(k)*n^{O(1)}$ donde n es el tamaño de la entrada y f es una función que depende solamente de k. En particular, los problemas polinomiales son Parámetro Fijo Tratables para cualquier k.

Gracias a diversos trabajos (Grohe, 2001; Grohe et al, 2011; Gottlob et al, 1999, Gottlob et al, 2000; Grohe, 2017) sabemos que los PSR son PFT si tomamos como parámetro el ancho arbóreo, el ancho hiper-arbóreo, el ancho hiper-arbóreo generalizado y el ancho hiper-arbóreo fraccional.

Derivado de lo anterior, se han realizado grandes esfuerzos por identificar propiedades estructurales sobre hipergrafos que garanticen que determinar si el ancho es acotado, es un problema tratable (Fischl, 2018). Nuestro interés es contribuir en ese esfuerzo y por eso estamos estudiando el ancho hiper-arbóreo de hipergrafos que provienen de la clase sintáctica 2μ-3MON.

3 Hipergrafos y sus anchos

En esta sección presentamos algunos hipergrafos y sus anchos. Como nos interesa estudiar la clase sintáctica 2μ-3MON, todos nuestros hipergrafos cumplen que sus hiperaristas tienen a lo más 3 vértices y cada vértice incide en a lo más 2 hiperaristas. El ancho arbóreo lo calculamos en SageMath, cabe mencionar que el algoritmo que usa este programa no es óptimo y con hipergrafos de más de 30 vértices el programa se tarda más de dos horas en hacer el cómputo. Para calcular el ancho hiper-arbóreo usamos el programa que viene en el siguiente repositorio: https://github.com/daajoe/detkdecomp

A nosotros nos interesa saber si la clase 2μ-3MON es parámetro fijo tratable con el ancho hiper-arbóreo fraccional. Como nos enfrentamos a una subclase de un problema NP-completo y para cada $k \geq 2$, determinar si el ancho fraccional de un hipergrafo está acotado por k también es un problema NP-completo, la mayoría de los cálculos que hemos hecho son sobre hipergrafos de a lo más 33 vértices. Esta clase de hipergrafos, a pesar de ser muy restrictiva se desconoce a la fecha cuál es un ancho fraccional (incluso su ancho arbóreo). El software conocido para determinar el ancho arbóreo para grafos presenta una explosión combinatoria para un número reducido de vértices e hiperaristas. En la dirección de la determinación del ancho fraccional incluimos una visualización del grafo primal de cada hipergrafo. Notamos que aún cuando estos hipergrafos no tienen más de 33 vértices su visualización se vuelve confusa.

Los hipergrafos que presentamos a continuación se construyeron tratanto de alcanzar el máximo ancho posible con ese número de vértices e hiperaristas. Como podemos observar el máximo ancho arbóreo que hemos alcanzado es 8 y el máximo ancho hiperarboreo que hemos alcanzado es 6. El siguiente paso en nuestro proyecto es encontrar hipergrafos con ancho arbóreo 9 y ancho hiper-arbóreo 7, para eso hemos definido varios hipergrafos con 33 vértices y 22 hiperaristas pero hasta ahora no hemos alcanzado los anchos deseados.

El hipergrafo H_1 determinado por el conjunto de hiperaristas {{0, 1, 2}, {0, 3, 4}, {1, 5, 6}, {3, 7, 8 }, {5, 9, 10}, {7, 9, 14}, {11, 12, 13 }, {9, 7, 14}, {6, 8, 11}, {4, 10, 12}}. Ningún otro hipergrafo con 15 vértices y 10 hiperaristas tiene ancho arboreo más grande que H_1. Su ancho arbóreo es 6 y su ancho hiper-arbóreo es 3.

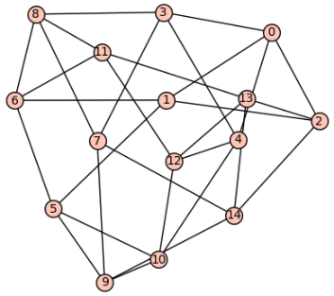


Figura 7. Grafo primal de H_1

El hipergrafo H_2 determinado por el conjunto de hiperaristas {{0, 1, 2}, {0, 3, 4}, {5, 6, 7}, {5, 8, 9}, {1, 10, 11}, {3, 12, 13}, {14, 15, 16 }, {17, 18, 19}, {20, 10, 14}, {12, 15, 17}, {6, 11, 13}, {2, 8, 18}, {4, 9, 16}, {19, 20, 7}}. Ningún otro hipergrafo con 21 vértices y 14 hiperaristas tiene ancho arbóreo más grande que H_2. Su ancho arbóreo es 7 y su ancho hiper-arbóreo es 4.

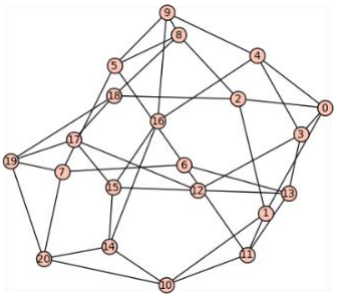


Figura 8. Grafo primal de H_2

El hipergrafo H_3 determinado por el conjunto de hiperaristas {{0, 1, 2}, {0, 3, 6}, {1, 4, 7}, {2, 5, 8}, {6, 9, 15}, {3, 10, 16}, {4, 12, 18 }, {7, 11, 17}, {5, 13, 20}, {8, 14, 19}, {9, 12, 19}, {11, 15, 20}, {10, 13, 18}, {14, 16, 17}}. H_3 tiene 21 vértices y 14 hiperaristas. Su ancho arbóreo es 7 y su ancho hiper-arbóreo es 5.

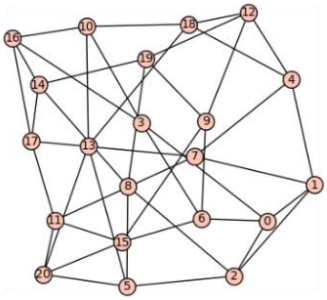


Figura 9. Grafo primal de H_3

El hipergrafo H_4 determinado por el conjunto de hiperaristas {{0, 1, 2}, {0, 3, 4}, {1, 3, 5}, {6, 7, 8}, {6, 9, 10}, {7, 11, 12}, {9, 13, 14}, {11, 15, 16}, {17, 18, 19}, {20, 21, 22}, {23, 24, 25}, {17, 20, 26}, {13, 18, 23}, {2, 8, 21}, {4, 12, 14}, {10, 15, 24}, {16, 19, 22}, {5, 25, 26}}. Ningún otro hipergrafo con 27 vértices y 18 hiperaristas tiene ancho arbóreo más grande que H_4. Su ancho arbóreo es 8 y su ancho hiper-arbóreo es 5.

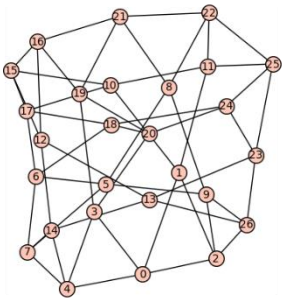


Figura 10. Grafo primal de H_4

El hipergrafo H_5 determinado por el conjunto de hiperaristas {{0, 1, 2}, {0, 3, 4}, {1, 5, 6}, {2, 7, 8}, {3, 11, 12}, {4, 9, 10}, {5, 13, 14}, {6, 15, 16}, {7, 17, 18}, {8, 19, 20}, {9, 21, 22}, {15, 23, 24}, {19, 25, 26}, {16, 18, 22}, {10, 23, 26}, {14, 21, 25}, {11, 13, 20}, {12, 14, 17}}. H_5 tiene 27 vértices y 18 hiperaristas. Su ancho arbóreo es 7 y su ancho hiper-arbóreo es 5.

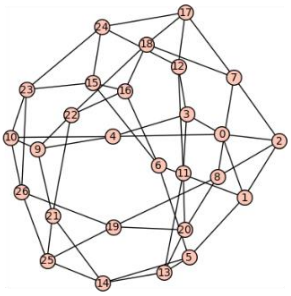


Figura 11. Grafo primal de H_5

El hipergrafo H_6 determinado por el conjunto de hiperaristas {{0, 1, 2}, {0, 3, 4}, {1, 5, 6}, {2, 7, 8}, {3, 11, 12}, {4, 9, 10}, {5, 13, 14}, {6, 15, 16}, {7, 17, 18}, {8, 19, 20}, {9, 21, 22}, {15, 23, 24}, {19, 25, 26}, {10, 27, 28}, {14, 29, 80}, {20, 31, 32}, {16, 21, 32}, {22, 26, 30}, {11, 29, 31}, {12, 13, 25}, {17, 23, 27}, {18, 24, 28}}. H_6 tiene 33 vértices y 22 hiperaristas. Su ancho arbóreo es 6 y su ancho hiper-arbóreo es 5.

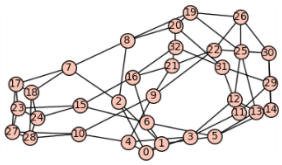


Figura 12. Grafo primal de H_6

El hipergrafo H_7 determinado por el conjunto de hiperaristas {{0, 1, 2}, {0, 3, 4}, {1, 5, 6}, {2, 7, 8}, {3, 11, 12}, {4, 9, 10}, {5, 13, 14}, {6, 15, 16}, {7, 17, 18}, {8, 19, 20}, {9, 21, 22}, {15, 23, 24}, {19, 25, 26}, {10, 27, 28}, {14, 29, 80}, {20, 31, 32}, {16, 21, 32},

{22, 26, 30}, {11, 29, 31}, {12, 13, 25}, {17, 23, 27}, {18, 24, 28}}. H_7 tiene 33 vértices y 22 hiperaristas. Su ancho arbóreo es 8 y su ancho hiper-arbóreo es 5.

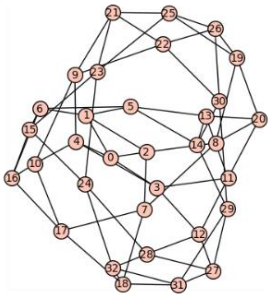


Figura 13. Grafo primal de H_7

El hipergrafo H_8 determinado por el conjunto de hiperaristas {{0, 1, 2}, {0, 3, 4}, {1, 5, 6}, {2, 7, 8}, {3, 11, 12}, {4, 9, 10}, {5, 13, 14}, {6, 15, 16}, {7, 17, 18}, {8, 19, 20}, {10, 21, 22}, {12, 23, 24}, {14, 25, 26}, {16, 27, 28}, {18, 29, 80}, {20, 31, 32}, {9, 27, 17}, {11, 13, 29}, {15, 19, 21}, {22, 25, 30}, {23, 26, 31}, {24, 28, 32}}. H_8 tiene 33 vértices y 22 hiperaristas. Su ancho arbóreo es 8 y su ancho hiper-arbóreo es 6.

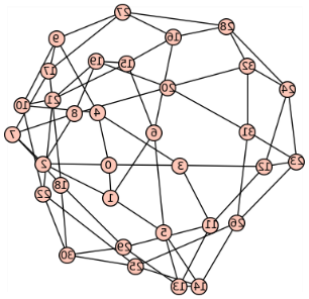


Figura 14. Grafo primal de H_8

El hipergrafo H_9 determinado por el conjunto de hiperaristas {{0, 1, 2}, {0, 3, 4}, {1, 5, 6}, {2, 7, 8}, {3, 11, 12}, {4, 9, 10}, {5, 13, 14}, {6, 15, 16}, {7, 17, 18}, {8, 19, 20}, {10, 21, 22}, {12, 23, 24}, {13, 23, 31}, {14, 25, 26}, {22, 27, 28}, {24, 29, 80}, {26, 31, 32}, {9, 15, 29}, {18, 21, 32}, {16, 19, 27}, {20, 28, 30}, {11, 17, 25}}. H_9 tiene 33 vértices y 22 hiperaristas. Su ancho arbóreo es 7 y su ancho hiper-arbóreo es 5.

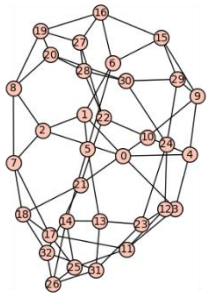


Figura 15. Grafo primal de H_9

A la hora de construir hipergrafos, hemos notado que, si queremos que nuestros hipergrafos sean cada vez más grandes, se vuelve cada vez más difícil construirlos si vamos definiendo las hiperaristas una por una. Motivados por esto, estamos buscando formas de definir clases de hipergrafos donde el conjunto de vértices sea cada vez más grande.

Siguiendo esa idea definimos la siguiente clase infinita de hipergrafos, que tiene ancho arbóreo acotado y por lo tanto es tratable. Para cada número natural n, sea H_n de tal forma que $V(H_n) = \{1, 2, \ldots, 3n\}$ y como hiperaristas tomamos todos los conjuntos de la forma $\{3k+1, 3k+2, 3k+4\}$ con $k<n-1$, los conjuntos de la forma $\{3k+2, 3k+3, 3k+6\}$ $k<n-1$, además agregamos los conjuntos $\{3n-2, 3n-1, 1\}$ y $\{3n-1, 3n, 3\}$. Observamos que H_n tiene 2n hiperaristas. La clase $C=\{H_n \mid n \in N\}$ es una clase infinita de hipergrafos.

Fijamos n número natural. Ahora construiremos una descomposición arbórea para H_n. Para cada $i<n$, sea u_i el conjunto (bolsa de vértices) $\{1+i, 2+i, 3+i, 4+i, 5+i, 6+i\}$ y sea u_n el conjunto $\{3n-2, 3n-1, 3n, 1, 2, 3\}$. Sea T el árbol cuyos nodos son los u_i y para cada $i<n$, $\{u_i, u_{i+1}\}$ son adyacentes en T. Observamos que las hiperaristas de la forma $\{3k+1, 3k+2, 3k+4\}$ y $\{3k+2, 3k+3, 3k+6\}$ están contenidas en la bolsa $\{1+3k, 2+3k, 3+3k, 4+3k, 5+3k, 6+3k\}$. Además, las hiperaristas $\{3n-2, 3n-1, 1\}$ y $\{3n-1, 3n, 3\}$ están contenidas en la bolsa $\{3n-2, 3n-1, 3n, 1, 2, 3\}$. Además, cualquier vértice $3j+i$ con $i<3$ solamente pertenece a las bolsas u_{j-1}, u_j, y u_{j+1} que forman un subgrafo conexo de T. Así, T es una descomposición arbórea de H_n y su ancho es 5. Por lo tanto, la clase C es de ancho arbóreo acotado y por lo tanto tratable.

4 Conclusiones

Observando el comportamiento del ancho hiper-arbóreo en los ejemplos analizados, nuestra conjetura es que ninguno de los anchos es acotado sobre la clase 2µ-3MON. Como se ha tratado con problemas NP-completos, los algoritmos conocidos que calculan el ancho arbóreo y el ancho hiper-arbóreo no son de utilidad cuando se tratan hipergrafos muy grandes. Por esta razón los hipergrafos que hemos estudiado hasta ahora en su mayoría tienen a lo más 33 vértices. Como es muy difícil definir hipergrafos muy grandes

enlistando todas sus hiperaristas, estamos buscando formas inductivas y ordenadas de definir hipergrafos sobre conjuntos arbitrarios de vértices. Además, continuaremos definiendo hipergrafos específicos y calculando sus anchos puesto que nos interesa identificar las propiedades estructurales de los hipergrafos que nos permiten minimizar o maximizar su ancho hiper-arbóreo fraccional.

Referencias

Courcelle, B., & Engelfriet, J. (2012). "*Graph structure and monadic second-order logic: a language-theoretic approach"* (Vol. 138). Cambridge University Press.

de Haan, R., & Szeider, S. (2014, July). "Fixed-parameter tractable reductions to SAT". In *International Conference on Theory and Applications of Satisfiability Testing* (pp. 85-102). Cham: Springer International Publishing.

Downey, R. G., & Fellows, M. R. (2013). "*Fundamentals of parameterized complexity"* (Vol. 4). London: springer.

Fischl, W., Gottlob, G., & Pichler, R. (2018, May). "General and fractional hypertree decompositions: Hard and easy cases". In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (pp. 17-32).

Gottlob, G., Greco, G., & Scarcello, F. (2014). "Treewidth and hypertree width". *Tractability: Practical Approaches to Hard Problems*, *1*, 20.

Gottlob, G., Lanzinger, M., Longo, D. M., Okulmus, C., & Pichler, R. (2020, September). The hypertrac project: Recent progress and future research directions on hypergraph decompositions. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research* (pp. 3-21). Cham: Springer International Publishing.

Gottlob, G., Lanzinger, M., Pichler, R., & Razgon, I. (2021). "Complexity analysis of generalized and fractional hypertree decompositions". *Journal of the ACM (JACM)*, *68*(5), 1-50.

Gottlob, G., Leone, N., & Scarcello, F. (1999, May). "Hypertree decompositions and tractable queries". In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (pp. 21-32).

Gottlob, G., Leone, N., & Scarcello, F. (2000). A comparison of structural CSP decomposition methods. *Artificial Intelligence*, *124*(2), 243-282.

Grohe, M. (2007). "The complexity of homomorphism and constraint satisfaction problems seen from the other side". *Journal of the ACM (JACM)*, *54*(1), 1-24.

Grohe, M., & Marx, D. (2014). "Constraint solving via fractional edge covers". *ACM Transactions on Algorithms (TALG)*, *11*(1), 1-20.

Guillén Galván, C., Lemuz López, R., & Ayaquica Martínez, I. O. (2013). "Model Counting in the 2mu-3MON Syntactic Class". *Computación y Sistemas*, *17*(4), 501-513.

Szeider, S. (2003, May). "On fixed-parameter tractable parameterizations of SAT". In *International Conference on Theory and Applications of Satisfiability Testing* (pp. 188-202). Berlin, Heidelberg: Springer Berlin Heidelberg.

Tsang, E. (2014). "*Foundations of constraint satisfaction: the classic text"*. BoD–Books on Demand.

Índice de autores

Nombre del Autor	Nacionalidad	
Aaron Ramírez Martínez	Mexicana	
Ana Laura Lezama Sánchez	Mexicana	
Antonio Pérez Vazquez	Mexicana	
Carlos Guillén Galván	Mexicana	
Erick Barrios González	Mexicana	
Erika Bonfil Barragán	Mexicana	
Fernando Zacarias Flores	Mexicana	
Gabriela A. García Robledo	Mexicana	
Guillermo De Ita Luna	Mexicana	Editor
José A. Reyes Ortiz	Mexicana	
Leonardo D. Sánchez Martínez	Mexicana	
Luis Carlos Altamirano Robles	Mexicana	
Maricela Bravo	Mexicana	
Mario A. Cruz Miguel	Mexicana	
Meliza Contreras González	Mexicana	
Mireya Tovar Vidal	Mexicana	Editora
Nidia K Serafin Rojas	Mexicana	
Octavio Mendoza Gómez	Mexicana	
Pedro Bello López	Mexicana	Editor
Rosalba Cuapa	Mexicana	
Sonia Navarro Flores	Mexicana	

Compiladores

Mireya Tovar Vidal
Guillermo De Ita Luna
Pedro Bello López

Revisores

Ana Laura Lezama Sánchez
Cecilia Reyes Peña
Claudia Zepeda Cortés
Fernando Zacarias Flores
Guillermo De Ita Luna
Hilda Castillo Zacatelco
Jacobo Leonardo Gonzalez Ruiz
José Alejandro Reyes Ortiz
José de Jesús Lavalle Martínez
José Luis Carballido Carranza
José Raymundo Marcial Romero
Karina Rosales López
Leonardo Daniel Sánchez Martínez
Mario Rossainz López
Meliza Contreras González
Mireya Tovar Vidal
Pedro Bello López

Editores

Mireya Tovar Vidal
Guillermo De Ita Luna
Pedro Bello López

Aplicaciones en procesamiento de lenguaje natural y teoría de grafos
Coordinadores de la publicación:
Mireya Tovar Vidal
Guillermo De Ita Luna
Pedro Bello López
A partir de diciembre de 2023
está disposición en PDF en la página
de la Facultad de Ciencias de la Computación
de la Benemérita Universidad Autónoma de Puebla (BUAP)
https://ontologica.cs.buap.mx/icokg2023/libroICOKG2023.pdf
Peso del archivo: 6.50 MB