GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

Análisis de Diagnósticos y Tratamientos de Migrañas mediante Procesamiento de Lenguaje Natural

Autor: Juan Sánchez-Blanco Gómez

Directora: Dra. Cristina Puente Águeda
Co-Directora: Dra. Elena García García

Madrid

05/06/2022

C.Puente

Agradecimientos

A mis tutores Cristina, Elena y Rafael por su ayuda, paciencia y dedicación. A mi hermano, mis padres y a toda mi familia por su apoyo continuado. A todos los docentes y compañeros de la escuela de los que tanto he aprendido.

ANÁLISIS DE DIAGNÓSTICOS Y TRATAMIENTOS DE MIGRAÑAS MEDIANTE PROCESAMIENTO DE LENGUAJE NATURAL

Autor: **Juan Sánchez-Blanco Gómez**

Director: Dra. Cristina Puente Águeda
Dra. Elena García García

RESUMEN DEL PROYECTO

Las migrañas afectan a una gran parte de la población y afectan a la calidad de vida de todas las personas que las padecen. Nuestro estudio examina la teoría sobre el impacto de las migrañas en los pacientes y en la sociedad, y busca identificar patrones comunes a los pacientes. Analizando más de 5800 historias clínicas de pacientes a través de procesado del lenguaje natural, nuestro estudio delimita ciertos grupos de pacientes que sufren los mismos síntomas, y que son afectados por los mismos agravantes y detonantes. Para el análisis de los textos y su representación numérica se han evaluado dos técnicas que componen el estado del arte en este campo de Inteligencia Artificial: el Word2Vec y el Transformer. La clusterización de los datos se analiza en profundidad mediante técnicas de selección del número óptimo de grupos, y representaciones de de datos de alta dimensionalidad.

Palabras clave — Migrañas, Procesado del Lenguaje Natural, Clustering, Transformer, Word2Vec

1 Introducción

El objeto de estudio consiste en el procesado y análisis de datos anonimizados sobre pacientes con migrañas de la Clínica Ilion, con el fin de identificar grupos de pacientes con patrones similares, de manera que sea posible identificar agravantes y detonantes de la condición específica de cada grupo.

Las migrañas son un problema de salud muy común y debilitante que afecta a muchas personas en todo el mundo. A menudo se tratan como si fueran un simple dolor de cabeza, pero las migrañas pueden ser mucho más que eso. Las migrañas pueden afectar la capacidad de una persona para realizar sus actividades diarias y pueden tener un impacto significativo en su calidad de vida. En algunos casos, las migrañas pueden ser tan intensas que la persona afectada no puede salir de la cama. Las migrañas también pueden causar otros problemas, como ansiedad y depresión.

Las migrañas pueden afectar a cualquier persona, independientemente de su edad, sexo o raza. Según la OMS, aproximadamente un 14% de la población mundial sufre de migrañas.

Las migrañas son responsables de una pérdida anual de más de 200 millones de días de trabajo en los Estados Unidos, con un coste estimado de $17 mil millones (Goldberg 2005). En el Reino Unido, se estima que las migrañas cuestan alrededor de £8,8 mil millones al año en costes médicos, pérdida de productividad y costes indirectos. Las migrañas también pueden tener un impacto negativo en la calidad de vida de las personas que las padecen. Se ha encontrado que las personas con migrañas tienen un riesgo 3 veces mayor de depresión y ansiedad (Peres et al. 2017).

2 Definición del proyecto

El desarrollo consiste en un análisis y procesado del lenguaje natural (NLP) inicial para extraer información de las fichas de pacientes y estructurar los datos. Se realizarán además diferentes estudios con algoritmos de Inteligencia Artificial, incluyendo el clustering de los distintos tipos de diagnósticos y dolencias entre otros.

El estudio examina también distintos métodos de representación numérica de textos, previo a realizar el clustering. Entro los métodos comprobados, se utilizan el algoritmo Word2Vec y la arquitectura de red neuronal denominada Transformer, basada en capas de neuronas de atención.

3 Descripción del modelo

El modelo abarca un conjunto de técnicas distintas desde la ingesta de datos textuales hasta los resultados finales. Gran parte de cualquier proyecto de análisis de texto se centra en el preprocesado y limpieza de los datos. Los textos se deben limpiar y filtrar extensamente, separar en palabras individuales y normalizar (extraer las raíces) antes de comenzar el análisis.

La manera de representar cada texto de forma numérica es un campo de estudio en constante desarrollo. Este estudio compara tres métodos de representación textual (vectorización), y las salidas de estas representaciones son utilizadas para entrenar un modelo de clustering K-Means. Los resultados son representados con distintas técnicas de proyección de datos de alta dimensionalidad.

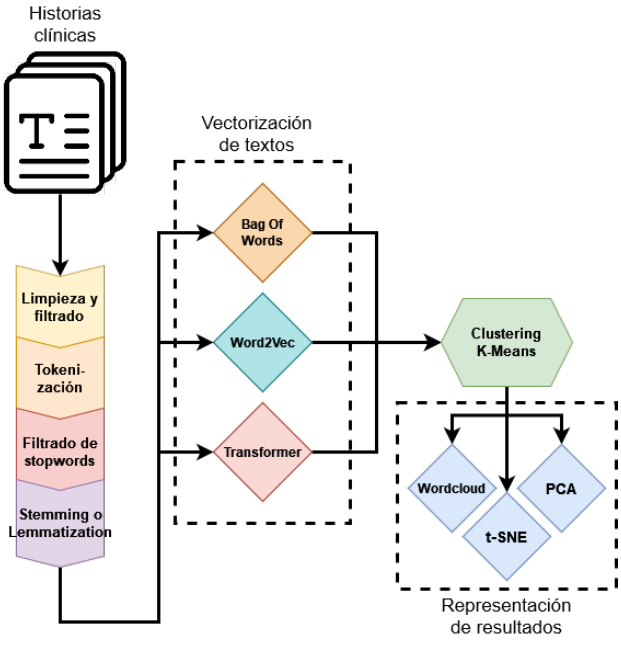


Figura 1: Resumen de Arquitectura

Una de las técnicas de vectorización utilizadas consiste en el Transformer. Los *transformers* son modelos de complejidad el-

evada que llevan días e incluso semanas o meses en entrenarse sobre sets de datos elevados. Por ello, para este estudio utiliza un modelo ya entrenado sobre textos similares a los observados en las historias clínicas de este desarrollo. El modelo en cuestión se trata de un desarrollo del Centro Nacional de Supercomputación, localizado en Barcelona. El modela ha sido entrenado sobre 2.172.491 documentos biomédicos en español, que componen más de 43M de frases en total.

4 Resultados

Utilizando las dos técnicas de representación vectorial más prometedoras: Word2Vec y el Transformer, se han entrenado dos modelos de clustering, y se han comprobado los grupos resultantes generados por cada arquitectura. Los resultados se representan en dos dimensiones con técnicas de proyección de alta dimensionalidad como t-SNE o PCA.

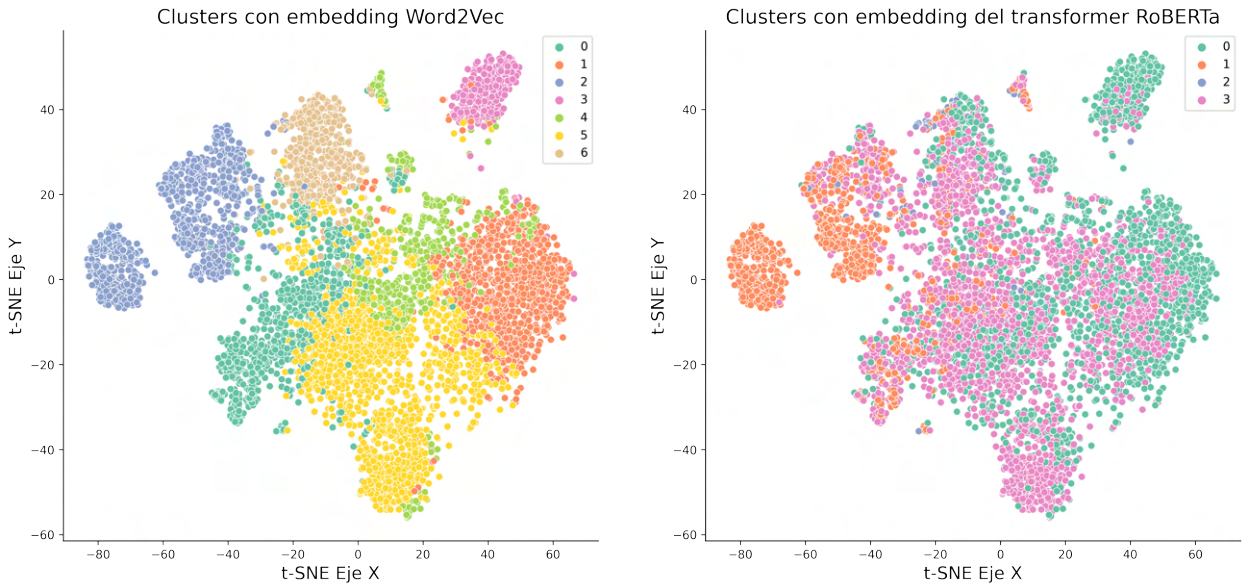


Figura 2: Representación mediante t-SNE de los clusters generados por cada modelo de vectorización

El análisis de los resultados ha permitido sacar la conclusión de que la codificación generada por el Word2Vec muestra resultados mucho más prometedores, consiguiendo delimitar los clusters de manera mucho más eficaz, y mostrando mejores contornos entre los distintos grupos.

Utilizando análisis de frecuencias de las palabras y representando los datos sobre wordclouds, se ha sacado una caracterización de cada uno de los grupos identificados. Se distinguen 7 grupos claramente delimitados, y la representación permite observar la permeabilidad entre cada grupo, así como cuales de ellos comparten más palabras similares entre ellos.

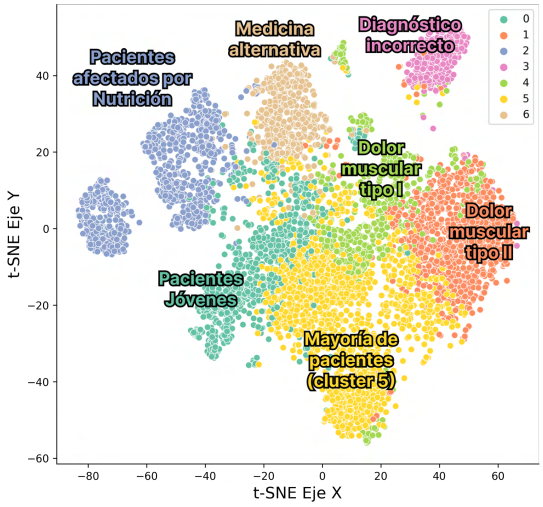


Figura 3: Scatterplot de los clusters generados por Word2Vec identificados y categorizados

5 Conclusiones

Este trabajo ha sentado las bases de los distintos grupos de personas que sufren migrañas. Los detonantes y agravantes de las migrañas no se comprenden del todo, y este estudio ha buscado delimitar distintos grupos de pacientes con síntomas y efectos similares dentro de la patología de la migraña.

Evaluando los distintos métodos de codificación, no cabe duda de que los resultados del Word2Vec son superiores. Los transformers, que formar parte del estado del arte en NLP, son muy poderosos, pero como cualquier otra red neuronal, requieren una cantidad de datos muy elevada para comenzar a mostrar resultados prometedores.

Los resultados arrojados por la codificación del Word2Vec así como los bajos tiempos de ejecución del entrenamiento del modelo lo convierten en el algoritmo ideal para este aplicativo, teniendo además en cuenta que la continuación de este estudio consiste en desarrollar una aplicación móvil capaz de albergar el modelo generado.

Bibliografía

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR*.

Goldberg, Lawrence D. (2005). "The cost of migraine and its treatment". In: *American Journal of Managed Care* 11 (SUPPL. 2). ISSN: 10880224.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of Workshop at ICLR*.

Peres, Mario Fernando Prieto et al. (2017). "Anxiety and depression symptoms and migraine: a symptom-based approach research". In: *Journal of Headache and Pain* 18 (1). ISSN: 11292377. DOI: 10.1186/s10194-017-0742-1.

Van der Maaten, Laurens and Geoffrey Hinton (2008). "Viualizing data using t-SNE". In: *Journal of Machine Learning Research*. ISSN: 15324435.

ANALYSIS OF MIGRAINE DIAGNOSIS AND TREATMENT THROUGH NATURAL LANGUAGE PROCESSING

Author: **Juan Sánchez-Blanco Gómez**

Supervisor: Dr. Cristina Puente Águeda
Dr. Elena García García

ABSTRACT

Migraine headaches affect a large proportion of the population and affect the quality of life of all sufferers. Our study examines theory on the impact of migraines on patients and society, and seeks to identify patterns common to patients. By analysing more than 5800 patient records through natural language processing, our study delineates certain groups of patients who suffer from the same symptoms, and who are affected by the same aggravating factors and triggers. For the analysis of the texts and their numerical representation, two techniques that make up the state of the art in this field of Artificial Intelligence have been evaluated: Word2Vec and Transformer. The clustering of the data is analysed in depth by means of techniques of selection of the optimal number of groups, and representations of high dimensional data.

Keywords — Migraines, Natural Language Processing, Clustering, Transformer, Word2Vec

1 Introduction

The object of the study consists of processing and analysing anonymised data on migraine patients from the Ilion Clinic, in order to identify groups of patients with similar patterns, so that it is possible to identify aggravating factors and triggers of the specific condition of each group.

Migraines are a very common and debilitating health problem that affects many people around the world. They are often treated as if they were a simple headache, but migraines can be much more than that. Migraines can affect a person's ability to perform daily activities and can have a significant impact on their quality of life. In some cases, migraines can be so intense that the affected person cannot get out of bed. Migraines can also cause other problems, such as anxiety and depression.

Migraine headaches can affect anyone, regardless of age, gender or race. According to the WHO, approximately 14% of the world's population suffers from migraines. Migraines are responsible for an annual loss of more than 200 million work days in the United States, at an estimated cost of $17 billion (Goldberg 2005). In the United Kingdom, migraines are

estimated to cost around $8.8 billion per year in medical costs, lost productivity, and indirect costs. Migraines can also have a negative impact on the quality of life of sufferers. People with migraines have been found to have a 3 times higher risk of depression and anxiety (Peres et al. 2017).

2 Project definition

The development consists of an initial analysis and natural language processing (NLP) to extract information from patient records and structure the data. Different studies will also be carried out with Artificial Intelligence algorithms, including clustering of the different types of diagnoses and ailments among others.

The study also examines different methods of numerical representation of texts, prior to clustering. Among the methods tested, the Word2Vec algorithm and the neural network architecture called Transformer, based on layers of attention neurons, are used.

3 Model description

The model encompasses a range of different techniques from textual data ingestion to final results. Much of any text analysis project focuses on preprocessing and cleaning of the data. Texts must be extensively cleaned and filtered, separated into individual words, and normalised (root extracted) before analysis can begin.

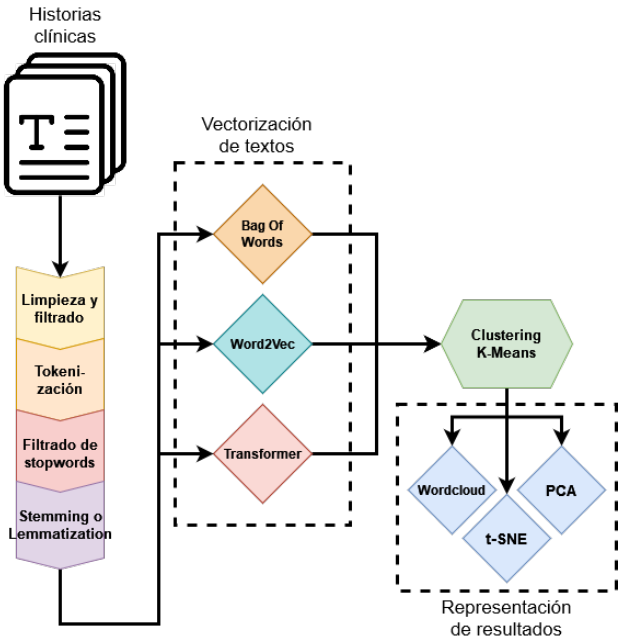


Figure 1: Architecture diagram

How to represent each text numerically is a constantly developing field of study. This study compares three methods of textual representation (vectorisation), and the outputs of these representations are used to train a K-Means clustering model. The results are represented with different high-dimensional data projection techniques.

One of the vectorisation techniques used is the Transformer. Transformers are highly complex models that take days or even weeks or months to train on large datasets. Therefore, for this study, it uses a model already trained on texts similar to those observed in the medical records of this development. The model in question is a development of the National Super-computing Centre, located in Barcelona. The model has been trained on 2,172,491 biomedical documents in Spanish, comprising more than 43M sentences in total.

4 Results

Using the two most promising vector representation techniques: Word2Vec and the Transformer, two clustering models have been trained, and the resulting groups generated by each architecture have been tested. The results are represented in two dimensions with high dimensionality projection techniques such as t-SNE or PCA.

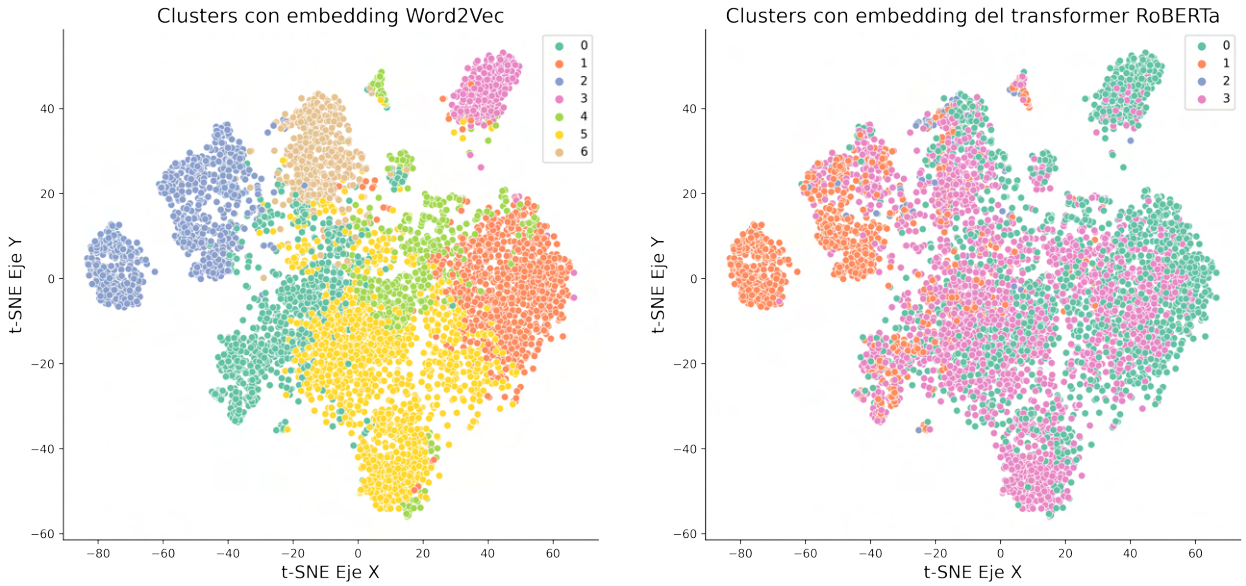


Figure 2: t-SNE representation of the clusters generated by each vectorisation model.

The analysis of the results has led to the conclusion that the coding generated by Word2Vec shows much more promising results, delimiting the clusters much more efficiently, and showing better contours between the different groups.

Using word frequency analysis and representing the data on wordclouds, a characterisation of each of the identified clusters has been obtained. Seven clearly delimited groups can be distinguished, and the representation allows to observe the permeability between each group, as well as which of them share more similar words between them.

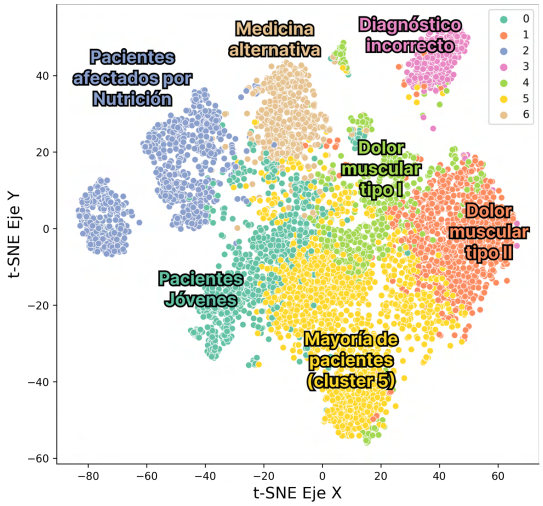


Figure 3: Scatterplot of Word2Vec-generated clusters identified and categorised.

5 Conclusions

This work has laid the groundwork for distinct groups of migraine sufferers. The triggers and aggravating factors of migraines are not fully understood, and this study has sought to delineate distinct groups of patients with similar symptoms and effects within migraine pathology.

Evaluating the different coding methods, there is no doubt that the Word2Vec results are superior. Transformers, which are part of the state of the art in NLP, are very powerful, but like any other neural network, they require a very large amount of data to begin to show promising results.

The results obtained by the Word2Vec coding as well as the low execution times of the model training make it the ideal algorithm for this application, taking into account that the continuation of this study consists of developing a mobile application capable of hosting the generated model.

References

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR*.

Goldberg, Lawrence D. (2005). "The cost of migraine and its treatment". In: *American Journal of Managed Care* 11 (SUPPL. 2). ISSN: 10880224.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of Workshop at ICLR*.

Peres, Mario Fernando Prieto et al. (2017). "Anxiety and depression symptoms and migraine: a symptom-based approach research". In: *Journal of Headache and Pain* 18 (1). ISSN: 11292377. DOI: 10.1186/s10194-017-0742-1.

Van der Maaten, Laurens and Geoffrey Hinton (2008). "Viualizing data using t-SNE". In: *Journal of Machine Learning Research*. ISSN: 15324435.

Índice general

1 Introducción **5**
- 1.1 Objetivos del estudio . . . 6
  - 1.1.1 Objetivo General . . . 6
  - 1.1.2 Objetivos Específicos . . . 6
- 1.2 Motivación . . . 8
- 1.3 Metodología . . . 9
  - 1.3.1 Sujeto de estudio . . . 9
  - 1.3.2 Análisis de Datos . . . 9
  - 1.3.3 Plan de Trabajo . . . 10
  - 1.3.4 Recursos empleados . . . 11
    - 1.3.4.1 Equipos y material . . . 11
    - 1.3.4.2 Herramientas software . . . 12
    - 1.3.4.3 Librerías de Python . . . 12
- 1.4 Estructura . . . 14

2 Estado de la cuestión **15**
- 2.1 Estado actual del estudio de las migrañas . . . 15
- 2.2 Procesamiento del Lenguaje Natural . . . 18
  - 2.2.1 Campos actuales de estudio del NLP . . . 18
  - 2.2.2 Aplicaciones prácticas del NLP . . . 19
  - 2.2.3 Procesado de textos . . . 20
    - 2.2.3.1 Tokenización . . . 20
    - 2.2.3.2 Flitrado de stop words . . . 21
    - 2.2.3.3 Stemming & Lemmatisation . . . 21
    - 2.2.3.4 Vectorization . . . 21
- 2.3 Representación de textos . . . 22
  - 2.3.1 Word embeddings . . . 22
    - 2.3.1.1 Word2Vec . . . 22
    - 2.3.1.2 Aplicaciones de los embeddings . . . 23
  - 2.3.2 Transformers . . . 23
    - 2.3.2.1 BERT . . . 24
    - 2.3.2.2 DistilBERT . . . 25
    - 2.3.2.3 RoBERTa & DistilRoBERTa . . . 25
- 2.4 Inteligencia Artificial Aplicada a medicina . . . 26
  - 2.4.1 Aplicaciones de la Inteligencia Artificial al campo de las migrañas 26

3 Desarrollo **28**
- 3.1 Preprocesado de Datos . . . 28
  - 3.1.1 Datos en crudo . . . 28
    - 3.1.1.1 Limpieza de los datos . . . 28
    - 3.1.1.2 Filtrado de historias clínicas . . . 29
    - 3.1.1.3 Eliminación de stop words . . . 30
    - 3.1.1.4 Stemming & Lemmatisation . . . 32
- 3.2 Representación de textos . . . 34

3.2.1 Bag Of Words . . . 34
3.2.2 Word2Vec . . . 35
3.2.2.1 Representación de palabras con t-SNE . . . 38
3.2.3 Transformers . . . 41
3.2.3.1 Selección del transformer . . . 41
3.2.3.2 Implementación . . . 43
3.2.3.3 Representación de palabras con t-SNE . . . 44
3.2.4 Visualización de textos . . . 48
3.2.4.1 Previsión de resultados . . . 51
3.3 Clustering de historias clínicas . . . 52
3.3.1 Reducción de dimensiones . . . 52
3.3.2 K-Means . . . 58
3.3.2.1 Definición del Pipeline . . . 58
3.3.3 Número óptimo de clusters . . . 60
3.3.3.1 Método Silhouette . . . 60
3.3.3.2 Método Elbow . . . 62
3.3.3.3 Selección de valores óptimos . . . 63
3.3.4 Resultados del clustering . . . 65

4 Resultados y Conclusiones **71**
4.1 Características de los clusters extraídos . . . 71
4.1.1 Clusters de codificado Word2Vec . . . 71
4.1.2 Clusters de codificado del Transformer . . . 76
4.1.3 Comparación de técnicas . . . 78
4.2 Futuras líneas de trabajo . . . 81
4.2.1 Aumento del set de datos . . . 81
4.2.2 Aplicación de diferentes algoritmos . . . 81
4.2.3 Continuación del estudio en el campo de las migrañas . . . 82

Bibliografía **84**

Anexos **88**
Objetivos de desarrollo sostenible . . . 88
Recursos adicionales . . . 90

Índice de Tablas

1 Ejemplo de codificado BOW de historias clínicas 34
2 Palabras más similares a "migraña" según modelo Word2Vec 37
3 Configuración de los modelos de clustering 65
4 Librerías Python 3.8 usadas durante el desarrollo 90

Índice de figuras

1 Variables registradas en *Migraine Buddy* 27
2 Distribución de Historias Clínicas por búsqueda de *keywords* 29
3 Wordcloud preliminar de historias clínicas 31
4 Wordcloud final de historias clínicas 33
5 Scatterplot de 400 palabras del Word2Vec a través de t-SNE 39
6 Word2Vec: Scatterplot del embedding de palabras similares a "migraña", "dolor" y "fiebre" 40
7 Transformer: Scatterplot del embedding de palabras 45
8 Comparación de Embeddings con t-SNE: Word2Vec vs. RoBERTa fine-tuned Transformer 46
9 t-SNE de los embeddings de las historias clínicas 49
10 Densidad de los embeddings de las historias clínicas (KDE) 50
11 Mapas de calor de correlación entre variables 54
12 Algoritmo PCA: Ratio de varianza explicada según número de variables . 55
13 Mapas de calor de correlación entre variables tras aplicar PCA 57
14 Silhouette scores para cada número de clusters 61
15 Medida de "WCSS" para cada número de clusters (Elbow method) . . . 63
16 Tamaño de los clusters tras aplicar K-Means 66
17 Resultados gráficos del clustering sobre codificación Word2Vec 68
18 Resultados gráficos del clustering sobre codificación del Transformer . . . 69
19 Wordclouds de los clusters generados del codificado con Word2Vec 73
20 Word2Vec: Scatterplot t-SNE de los resultados del clustering - Etiquetado con perfiles 75
21 Wordclouds de los clusters generados del codificado con Transformer . . . 77
22 Comparación de resultados de clustering proyectados sobre t-SNE 79
23 Comparación de resultados de clustering proyectados con PCA 79
24 Diagrama de flujo de los clusters de Word2Vec (izquierda) y el transformer (derecha) 80

Listados de código

1 Limpieza de stop words en Python 30
2 Ejemplo de salida de limpieza de stop words 31
3 Lematización de frases en Python (simplificado a una frase) 32
4 Ejemplo de salida de la lematización de palabras 33

5 Implementación de Word2Vec en Python (simplificado) 36
6 Ejemplo del uso del modelo “roberta-base-biomedical-clinical-es” 42
7 Implementación del transformer (simplificado) 43
8 Implementación de un Pipeline de aprendizaje (simplificado) 59

1 Introducción

Las migrañas son un problema de salud muy común y debilitante que afecta a muchas personas en todo el mundo. A menudo se tratan como si fueran un simple dolor de cabeza, pero las migrañas pueden ser mucho más que eso. Las migrañas pueden afectar la capacidad de una persona para realizar sus actividades diarias y pueden tener un impacto significativo en su calidad de vida. En algunos casos, las migrañas pueden ser tan intensas que la persona afectada no puede salir de la cama. Las migrañas también pueden causar otros problemas, como ansiedad y depresión.

Las migrañas pueden afectar a cualquier persona, independientemente de su edad, sexo o raza. Según la OMS, aproximadamente un 14% de la población mundial sufre de migrañas. Las migrañas son responsables de una pérdida anual de más de 200 millones de días de trabajo en los Estados Unidos, con un coste estimado de $17 mil millones (Goldberg 2005). En el Reino Unido, se estima que las migrañas cuestan alrededor de £8,8 mil millones al año en costes médicos, pérdida de productividad y costes indirectos. Las migrañas también pueden tener un impacto negativo en la calidad de vida de las personas que las padecen. Se ha encontrado que las personas con migrañas tienen un riesgo 3 veces mayor de depresión y ansiedad (Peres et al. 2017).

El objeto de estudio consiste en el procesado y análisis de datos anonimizados sobre pacientes con migrañas de la Clínica Ilion, con el fin de identificar grupos de pacientes con patrones similares, de manera que sea posible identificar agravantes y detonantes de la condición específica de cada grupo. El desarrollo consistirá en un análisis y procesado del lenguaje natural (NLP) inicial para extraer información de las fichas de pacientes y estructurar los datos. Se realizarán además diferentes estudios con algoritmos de Inteligencia Artificial, incluyendo el clustering de los distintos tipos de diagnósticos y dolencias entre otros.

El uso de técnicas de Inteligencia Artificial para aplicaciones médicas es un área de estudio que se encuentra en su infancia. Unido a esto, el desarrollo de nuevos modelos de *Machine Learning*, capaces de detectar patrones complejos y analizar lenguaje natural ha causado el desarrollo de un nuevo campo de estudio en el que todavía hay mucho por aprender. Combinado con la amplia experiencia y conocimiento de especialistas, la utilización de

estas técnicas en el ámbito médico puede proporcionar información valiosa en la búsqueda de diagnósticos, tratamientos, síntomas, etc.

1.1 Objetivos del estudio

1.1.1 Objetivo General

Este estudio busca identificar variables de interés en la evolución del paciente con migrañas, con el fin de mejorar el diagnóstico y tratamiento de una patología que afecta a una gran parte de la población, y a fecha de escritura de este estudio sigue sin tener cura definitiva.

1.1.2 Objetivos Específicos

- Explorar patrones de comportamiento y evolución, en función de los distintos abordajes terapéuticos y los planes de nutrición establecidos. Se deben identificar:
  - El diagnóstico y el tratamiento actuales para la migraña.
  - Los factores de riesgo y complicaciones de la migraña..
  - Los diferentes tipos de migraña.
  - La efectividad de los tratamientos actuales para la migraña.
  - Las necesidades no cubiertas de los pacientes con migraña.
  - Identificar las últimas investigaciones sobre la migraña y las tendencias futuras en el tratamiento de la migraña.
- Desarrollar modelos algorítmicos que permitan definir tipos de pacientes con migrañas en función de las variables identificadas. Los pasos definidos para este objetivo se detallan a continuación:
  - Identificar las variables relevantes para el estudio, y seleccionar un conjunto de datos que contenga dichas variables relevantes.
  - Preprocesar los datos seleccionados.

  - Explorar y visualizar los datos preprocesados.
  - Seleccionar un modelo de aprendizaje automático.
  - Ajustar el modelo seleccionado a los datos preprocesados.
  - Evaluar y optimizar el rendimiento del modelo ajustado.
  - Predecir el tipo de pacientes con migrañas en función de las variables identificadas.
  - Identificar las limitaciones del modelo y proponer mejoras.
- Ayudar a futuros estudios a destacar patrones que desencadenan en la mejora de pacientes con migrañas dentro de cada tipología definida. (ej. Pacientes del grupo X muestran una mejora notable al cambiar su dieta, mientras que el grupo Y debe tener cuidado con su exposición solar.)

1.2 Motivación

La principal motivación del estudio es contribuir al avance científico del tratamiento de las migrañas y a la búsqueda continua de mejoras en la calidad de vida de las personas que las padecen. Se busca ofrecer una perspectiva analítica de los datos, y aprender sobre la experiencia y conocimiento de expertos en la materia tratada. Manteniendo siempre la privacidad de los datos sensibles y siguiendo la normativa vigente en materia de protección de datos.

Tal y como se expuso en la introducción, las migrañas afectan a un porcentaje elevado de la población (Peters 2019), y no tiene a día de hoy solución permanente. Se busca con esta investigación poder contribuir al avance de un campo de estudio que tiene la capacidad de impactar la vida de tantas personas.

El estudio servirá además como fuente de información para el posterior desarrollo del TFM de María José Medina Hernández, el cual consistirá en una aplicación que buscará utilizar los patrones reconocidos anteriormente para informar al paciente de su estado, la influencia de sus hábitos y dieta, y predecir futuros ataques basados en mediciones por medio de sensórica.

1.3 Metodología

Se trata de un estudio descriptivo de carácter exploratorio, retrospectivo, utilizando herramientas de Inteligencia Artificial, que consistirá en el procesamiento del lenguaje natural de las notas de consulta de profesionales que tratan a pacientes con migrañas. Para ello primero se realizará un análisis exploratorio de la base de datos, de manera que se pueda determinar qué clase de información sería posible extraer mediante esta técnica.

A continuación, se desarrollarán modelos basados en Inteligencia Artificial como *clustering* y *transformers* (sección 2.3.2) orientados a representación de textos, y se extraerán conclusiones a partir de los resultados de estos. Entre estos modelos cabría la posibilidad de desarrollar uno capaz de detectar casos de pacientes que hayan sido erróneamente diagnosticados con migrañas o identificar los diferentes desencadenantes que afectan a cada paciente de forma individualizada.

El objetivo será para ambos casos detectar variables comunes y diferenciadas entre distintos tipos de pacientes. Por último, se evaluará la viabilidad de modelos más complejos basados en *Deep Learning* con el mismo fin, en particular sistemas recomendadores que puedan ayudar a prevenir la migraña o a predecir un futuro ataque.

1.3.1 Sujeto de estudio

La población objetivo serán pacientes de la clínica Ilion, con amplia experiencia en el tratamiento integral del paciente con migraña, que hayan sido diagnosticados con cualquiera de las modalidades de esta enfermedad y que hayan realizado algún tratamiento debido a esta. Se recurrirá a archivos de historias clínicas previamente existentes en los que los datos estarán anonimizados mediante la sustitución de los nombres de paciente y de los profesionales por códigos identificativos.

1.3.2 Análisis de Datos

Se analizarán las notas de consulta de profesionales que tratan a pacientes con migraña. Para ello se utilizará un enfoque basado en el procesamiento del lenguaje natural, utilizando Python como leguaje de programación y ejecutando el código directamente en el servidor de la clínica. Se comenzará con un análisis exploratorio de la base de datos,

utilizando métodos de procesamiento del lenguaje natural como *tokenization*, eliminación de palabras de paso y normalización mediante *stemming* y *lemmatization*.

A continuación, a partir de los resultados del análisis exploratorio, se evaluará qué información sería posible extraer a partir de los datos, por ejemplo, la detección de pacientes que han sido erróneamente diagnosticados o la identificación de desencadenantes de ataques de migraña, y en función de esto se desarrollarán diferentes modelos de Machine Learning basados en técnicas como el *clustering* y el análisis de sentimientos.

Una vez desarrollados los modelos, se evaluarán utilizando diferentes métricas según su función. Se extraerán conclusiones a partir de sus resultados y se planteará el camino a seguir para investigaciones futuras.

1.3.3 Plan de Trabajo

A continuación se enumeran los pasos seguidos durante la realización del estudio:

1. **Contacto con la Clínica Ilion, firma de contratos y confidencialidad de datos** según indicaciones del Departamento Legal y de Protección de Datos.

2. **Gestión del acceso al servidor**, con acceso únicamente al dataset a explorar, previamente anonimizado completamente.

3. **Preprocesado inicial de datos**. El preprocesado consistirá en la separación de los distintos campos de los datos y limpieza general de los datos, asegurando siempre la anonimización y agregación de los mismos.

4. **Análisis preliminar y exploratorio de los datos**. Este análisis obtendrá estadísticas generales sobre el estado actual, incluyendo por ejemplo la cantidad de pacientes, la duración del tratamiento, el nº de visitas medio, el tiempo entre visitas, etc.

5. **Pre-procesado de datos**. Incluye la limpieza y estructuración del texto mediante técnicas de normalización del texto y eliminación de stopwords o palabras de paso que no aportan información. El detalle completo de los métodos de procesar el texto se encuentran la sección 2.2.3.

   (a) Separación individual de todas las palabras del texto mediante la identificación

de las separaciones entre palabras: espacios, signos de puntuación o caracteres de escape (como `\n` y `\t` por ejemplo). Este proceso se denomina *tokenization* (sección 2.2.3.1).

(b) Eliminar las palabras de paso o *stopwords* que no aportan significado propio para la interpretación por parte del modelo. Esto incluye las preposiciones, conjunciones y otras palabras sin relevancia para el análisis (sección 2.2.3.2).

(c) Normalización de las palabras del texto. Dentro de las técnicas de normalización se distinguen dos principales: stemming y lematización (sección 2.2.3.3).

6. **Análisis exploratorio de resultados de NLP**. Tras estructurar el texto se podrán realizar análisis estadísticos como si se tratara de una fuente de datos numérica.

7. **Desarrollo de modelos de Inteligencia Artificial**. Modelos de enfoque clásico como clustering y análisis de sentimiento. En particular, para clustering, se utilizarán los métodos de k-means y knn. Por último, se realizará un estudio de la viabilidad de modelos basados en deep-learning, especialmente aplicados a sistemas recomendadores.

8. **Análisis de resultados**. Análisis completo de los resultados y conclusiones que podemos extraer del estudio y los modelos desarrollados.

9. **Futuros pasos**. Los futuros pasos incluyen la publicación de uno o varios artículos científicos sobre los resultados obtenidos.

1.3.4 Recursos empleados

Dentro de los recursos empleados se contemplan tres tipos:

1.3.4.1 Equipos y material

- Servidor de la clínica Ilion, será utilizado para el almacenaje de los datos de manera segura y respetando la normativa en materia de protección de datos. El acceso al servidor se realizará con una conexión encriptada. La gestión del servidor corresponde a responsables de la clínica.

- Portátil y PC, utilizados para el desarrollo de los modelos de procesado y de ML, procesando solo datos ya agregados y anonimizados.

1.3.4.2 Herramientas software

La mayor parte del desarrollo se llevará a cabo utilizando el lenguaje de programación Python, se trata de un lenguaje multiparadigma con una amplia documentación en materia de Inteligencia Artificial y ciencia de datos (*Data Science*). Específicamente, el proyecto hará uso de Python 3.8. Existen versiones más recientes de Python (3.9 y 3.10), pero la versión 3.8 es ampliamente considerada como una de las versiones más estables del lenguaje para el cual la mayoría de las bibliotecas tienen compatibilidad completa, y la versión aún recibe soporte a largo plazo (LTS) y actualizaciones de seguridad.

El desarrollo se llevará a cabo principalmente en dos herramientas: Jupyter Notebooks y Visual Studio Code.

Jupyter Notebooks forma parte del "Proyecto Jupyter". Los cuadernos de Jupyter proporcionan una interfaz basada en web que puede ejecutar fragmentos de código y mostrar resultados en una plataforma interactiva, haciendo uso de *iPython*. *iPython* es una shell interactiva de Python que ofrece un entorno de desarrollo mucho más potente que el intérprete de Python estándar. *iPython* proporciona una serie de características útiles que no se encuentran en el intérprete de Python estándar, como la posibilidad de ejecutar código en bloques y mostrar sus salidas.

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Es gratuito y de código abierto. Incluye soporte para la depuración, control de versión, historia del código, refactorización, etc. Permite desarrollar en una variedad de distintos lenguajes de programación, e instalar extensiones de la comunidad para mejorar su capacidad en base al lenguaje que se busque programar y depurar. Para el ámbito de este estudio se ha utilizado VS Code como editor de scripts de Python, y como gestor de entornos Python.

1.3.4.3 Librerías de Python

Las librerías de Python más relevantes para el desarrollo de este estudio se detallan a continuación:

Numpy Esta librería agrega soporte para una amplia gama de operaciones matemáticas con matrices multidimensionales. El código base de la librería está escrito en Python y en C, un lenguaje de programación de bajo nivel que brinda eficiencia y optimización (The NumPy community 2022). Numpy ofrece velocidades comparables a las operaciones realizadas en el lenguaje MATLAB, un lenguaje escrito específicamente con cálculos matriciales en mente.

Pandas La librería Pandas está diseñada para el análisis y la manipulación de datos en forma de series y tablas de datos (*DataFrame*). Similar a Numpy, está altamente optimizado para el rendimiento y parcialmente escrito en C (McKinney & Pandas Development Team 2022).

NLTK Kit de herramientas de lenguaje natural ("*Natural Language ToolKit*") es un grupo de librerías dedicadas al procesamiento del lenguaje natural (principalmente en idioma inglés, con soporte creciente para otros idiomas) (Bird, Klein, & Loper 2010). Ofrece soporte para muchos campos de estudio e implementa funcionalidades básicas necesarias para cualquier estudio de NLP, incluidos, entre otros, análisis sintáctico, tokenización, stemming, etiquetado de PoS, etc.

SpaCy Potente biblioteca de procesamiento del lenguaje natural que implementa modelos complejos que forman parte del estado del arte actual. Hace uso de bibliotecas de cómputo de redes neuronales como Tensorflow y PyTorch para ejecutar estos modelos de análisis de texto especializados.

Scikit-Learn Una librería de Python para el aprendizaje automático construida sobre SciPy (librería especializada en cómputo de álgebra lineal, integración, procesado de señales, etc.). Ofrece una amplia gama de algoritmos de aprendizaje supervisados y no supervisados para una variedad de tareas de análisis de datos y aprendizaje automático (Scikit-learn developers 2022). Ofrece soporte para modelos de aprendizaje supervisados y no supervisados. Este estudio utilizará modelos de clustering no supervisados para encontrar grupos de pacientes potenciales que compartan diagnósticos similares.

Gensim Gensim es una librería de Python para modelado de temáticas (topic modeling), indexación de documentos y cálculo de similitudes con grandes corpus de texto

(Řehůřek et al. 2022). Implementa varios modelos de representación vectorial de textos, como Word2Vec, para traducir palabras en matrices de datos.

La lista completa de librerías de Python y las versiones utilizadas se pueden encontrar en la tabla 4 en la sección de anexos.

1.4 Estructura

El estudio se llevará a cabo construyendo una base y revisando el conocimiento existente sobre el procesamiento del lenguaje natural. Tras la revisión inicial del estado del arte, el documento cubrirá la metodología utilizada durante la fase de desarrollo y procesamiento de datos. Finalmente, los resultados y conclusiones serán analizados y documentados. El diseño completo del estudio se puede encontrar en la tabla de contenido.

2 Estado de la cuestión

La sección actual detalla el estado del arte en los diferentes campos relevantes a este estudio, tanto en materia de procesamiento del lenguaje natural (Sección 2.2) como en el estudio de la migraña (Sección 2.1).

2.1 Estado actual del estudio de las migrañas

La migraña es una de las principales causas de discapacidad y sufrimiento en todo el mundo, siendo clasificada como la sexta causa de años perdidos de calidad de vida por discapacidad a nivel mundial (Maria Teófila Vicente-Herrero et al. 2020) (Peters 2019). En EEUU, una de cada 6 personas se ven afectadas por esta enfermedad que, contrariamente a lo que ocurre con las enfermedades crónicas, suelen ser personas jóvenes y sanas de mediana edad. La mayor prevalencia suele encontrarse entre los 18 y 44 años, en el que, según los estudios, el 17,9% de los sujetos experimentó una migraña en los 3 meses anteriores (Peters 2019).

La migraña es una enfermedad prevalente también en la edad infantil. En EEUU, aproximadamente el 5% de los niños son diagnosticados con esta enfermedad, aumentando la prevalencia en la adolescencia. Es, además, la sexta enfermedad más incapacitante a nivel mundial en el grupo de edad de 10 a 14 años, y la quinta entre los 15 y los 19 años. Añadido al impacto clínico, asociado al dolor incapacitante que conllevan los episodios, está el impacto social dado que afecta en el entorno escolar y educativo, tanto en faltas de asistencia como en rendimiento académico, incluso en relaciones sociales. La atención médica de estos pacientes supone también un alto impacto económico para las familias (Hornik et al. 2020).

Según resultados del estudio "Eurolight", llevado a cabo en 10 países europeos con 9247 pacientes, la migraña es una enfermedad que ni se diagnostica ni se trata adecuadamente. Es bajo el porcentaje de personas que consultan a los médicos, y los fármacos específicos se usan de manera inadecuada, sin supervisión médica (Maria Teófila Vicente-Herrero et al. 2020). Los expertos, consideran que en Europa el impacto social y económico de la migraña se relaciona con la duración de las crisis y con un control inadecuado de las mismas, repercutiendo en la calidad de vida, la productividad laboral, y el elevado

consumo de recursos sanitarios (Maria Teófila Vicente-Herrero et al. 2020).

Las nuevas terapias farmacológicas, biológicas y dispositivos de neuromodulación están ofreciendo resultados positivos, siendo seguras y bien toleradas en adultos. Sin embargo, en población pediátrica la evidencia científica es aún limitada (Hornik et al. 2020).

Son muchos los casos de pacientes que padecen la enfermedad que no consultan a un especialista, o bien no logran un alivio adecuado después de consultar al mismo, existiendo una necesidad no satisfecha en el cuidado de los síntomas de la migraña (Maria Teófila Vicente-Herrero et al. 2020). Según Wells, Beuthin, & Granetzke 2019, más del 50% de los pacientes con episodios severos de migraña acceden a tratamientos de medicina integrativa o complementaria, sin informar a su médico especialista, por lo que no es habitual encontrar registros en la historia clínica que permitan demostrar la efectividad de estas terapias en el manejo de la enfermedad. Entre ellos se encuentran los suplementos alimenticios como las vitaminas B2, D, E, C, B6, coenzima Q10, butterbur, ácido fólico, omega-3, boswella, jengibre o guaraná. La "AHS, American Headache Society" y la "AAN, American Academy of Neurology", incluyen en la guía publicada para la prevención de la migraña los siguientes suplementos con su correspondiente grado de evidencia: butterbur (nivel A, pero fue retirado por hepatotoxicidad), feverfew y magnesio (nivel B-probablemente efectivo), CoQ10 (nivel C, posiblemente efectivo).

En relación a la terapia manual, incluyendo la revisión de Wells, Beuthin, & Granetzke 2019 incluye las diferentes modalidades de esta (terapia física, osteopatía, masaje, tratamiento manipulativo, ejercicio terapéutico, ...), encontrando beneficios positivos sobre la percepción del dolor, la frecuencia de los episodios y la discapacidad, si bien no parece ser un tratamiento con efectos superiores al farmacológico.

Aunque se han producido avances significativos en la comprensión de los complejos mecanismos implicados en la patofisiología de la migraña, todavía se carece de un tratamiento eficaz. En la actualidad, sólo una minoría de pacientes experimenta un alivio completo del dolor en un plazo de dos horas y esto sólo ocurre en un porcentaje de los ataques. Además, su uso frecuente se asocia a cefalea por exceso de medicación. Por último, la interrupción del tratamiento por problemas de tolerabilidad o contraindicaciones como consecuencia de una enfermedad cardiovascular o un riesgo importante dificulta aún más la eficacia de la terapia (Lipton et al. 2013). Por todo ello, la mayoría de los tratamientos

actuales buscan prevenir los ataques de migrañas evitando los desencadenantes personales o aminorar los síntomas resultantes. Entre los desencadenantes más comunes se encuentran el estrés, el despertar abrupto, las elecciones dietéticas, los factores medicinales u otros cambios fisiológicos o emocionales. Dado que varían según el paciente, se aconseja llevar un diario de los ataques de migraña para identificarlos y evitarlos.

En el "Estudio sobre el Impacto Emocional de la Enfermedad Crónica" (2021) llevado a cabo por la Plataforma de Organizaciones de Pacientes (POP) se indica que la intensidad con la que se siente el impacto emocional de la enfermedad crónica es de 7,4 sobre 10 de media en su peor momento. Además, identifica las limitaciones más frecuentes y con el peor impacto como las asociadas a síntomas depresivos: cansancio y fatiga (88%), tristeza (el 70%), problemas de sueño (71%) y apatía (67%). Además, un 48% experimenta depresión. En segundo lugar en frecuencia e impacto se encuentran los síntomas relacionados con estrés (64%) y ansiedad (60%). Por último, los asociados a miedo (44%) y angustia (47%). La migraña, que se considera enfermedad crónica, tiene por síntomas un gran número de los mencionados anteriormente, por lo que podemos afirmar que el efecto sobre la salud mental de los pacientes es importante.

Las migrañas tienen un efecto notable sobre la salud de la población y por tanto suponen un coste a tener en cuenta para la sanidad. En el año 2012 se registraron en España un total de 12.705 procesos de incapacidad temporal asociados a cefaleas, con un total de 137.481 días perdidos y un coste económico global estimado en 7.582.605,92 €. De este coste, 2.957.216,31 € corresponden al Instituto Nacional de la Seguridad Social y 4.625.389,61 € a las entidades colaboradoras (Mutuas de Accidentes de Trabajo y Enfermedades Profesionales de la Seguridad Social). En conjunto, representan el 0,0560021% del coste total de la prestación económica de incapacidad temporal en España para todos los procesos, estimado en 5.076,15 millones de euros (María Teófila Vicente-Herrero et al. 2014).

2.2 Procesamiento del Lenguaje Natural

Los orígenes del procesamiento del lenguaje natural (NLP por sus siglas en inglés) se remontan a 1940, durante el desarrollo de las primeras técnicas de traducción automática (MT: "Machine Translation"), principalmente entre inglés y ruso, y posteriores desarrollos de MT orientados al idioma chino (Russell & Hutchins 1987).

Durante la década de los 60, se presentaron iteraciones de IAs primitivas basadas en Q-A (question & answer), como BASEBALL (Green Jr 1961), ELIZA (1966) y LUNAR (1971) (Woods 1978), en las cuales los usuarios podían lanzar una serie de preguntas sobre las cuales la IA estuviera entrenada, y ésta era capaz de devolver una respuesta coherente. En el caso de LUNAR, uno de los objetivos era utilizar la Inteligencia Artificial como interfaz entre el ser humano y una base de datos de gran tamaño (Hendrix et al. 1978), sin necesidad del uso de lenguajes estructurados de programación.

A principios de 1980 comenzó a surgir una corriente académica dedicada al estudio del lenguaje mediante técnicas matemáticas ("grammar theory"), buscando la capacidad de detectar lógica, significado, buscar las creencias e intenciones del escritor, y sacar énfasis y temática de textos. A finales de la década empezaron a surgir las primeras herramientas de uso general capaces de parsear y procesar textos como "SRI's Core Language Engine" (Covington & Alshawi 1994) y "Alvey Natural Language Tools" (Briscoe et al. 1987).

Durante la década de los 90 se puso gran énfasis al estudio estadístico y probabilístico del lenguaje (Manning & Schütze 1999), realizando mapas de calor y correctores de texto. Otro tema de gran interés resultó ser la simplificación y síntesis de textos (Mani & Maybury 1999), buscando las partes del texto con mayor cantidad de información relevante.

2.2.1 Campos actuales de estudio del NLP

Nuevas areas recientes de estudio que han surgido del campo de NLP son, entre otros: El Análisis de Opinión (sentiment analysis), "Speech to Text" y viceversa, Reconocimiento de Entidades Nombradas (NER), Detección de emociones, etc.

El análisis de opinión busca extraer de todo el texto una puntuación final sobre la opinión (sentiment) del autor con respecto a un determinado tema, ya sean las reseñas de una

tienda, la crítica de una novela, o los comentarios de una publicación en las redes sociales. El análisis de opinión convencional (Yi et al. 2003) hace uso de bases de datos de palabras y expresiones que hacen referencia a opiniones positivas o negativas, de manera que realiza una estimación en base a los patrones previamente suministrados. Nuevos modelos de análisis de opinión hacen uso de técnicas de Deep Learning (redes neuronales profundas) para clasificar opiniones sin necesidad de generar una base de datos de términos negativos y positivos.

Las técnicas de "Speech to Text" consisten en el reconocimiento del lenguaje hablado para redactar un texto. Estas técnicas aúnan modelos de reconocimiento auditivo basados en redes neuronales o SVMs (Support Vector Machines), y utilizan análisis probabilístico de NLP para el rellenado de huecos en casos de dudas, o para el rellenado de los signos de puntuación necesarios.

El reconocimiento de entidades nombradas (NER) tiene un uso muy necesario en internet, ya que para algoritmos como el buscador de Google o la clasificación de "trending topics" en Twitter (Ritter et al. 2011), estos algoritmos deben saber de qué persona o asunto se está hablando, aunque se escriba en distintos idiomas, esté mal escrito, e incluso en instancias en las que no se menciona directamente a la persona de la que se está hablando. Antiguamente las páginas webs tenían que redactar una lista de palabras o términos de búsqueda para asistir al motor de búsqueda de Google en su clasificación de las páginas más relevantes.

Finalmente, la detección de emociones es similar al campo de estudio de análisis de opinión, pero se utiliza en mayor medida para clasificar las publicaciones en redes sociales en 6 grupos dependiendo de su emoción predominante (Bhargava, Y. Sharma, & S. Sharma 2016).

2.2.2 Aplicaciones prácticas del NLP

El procesamiento del lenguaje natural es popular en aplicaciones tales como la traducción automática, la detección de correos electrónicos fraudulentos y spam, extracción de información, generación de resúmenes ("summarization") y aplicaciones de pregunta y respuesta (Q-A).

El mundo se encuentra cada vez más digitalizado, con acceso inmediato a todo tipo de

datos desde nuestros hogares con una simple búsqueda. Una de las principales barreras de acceso a esta información es el idioma. La función principal de la traducción automática es convertir frases de un idioma a otro como lo hacen por ejemplo Google Translate o DeepL, y el principal desafío no es cambiar cada palabra de un idioma a otro, sino preservar el significado aun cuando las reglas de los distintos idiomas requieren entender el contexto y reorganizar el orden de las palabras. Esta es una lucha constante que mejora con cada iteración y cada nuevo artículo que se publica sobre ella, pero es infinitamente compleja, única para cada idioma y en algunos casos ilógica para la máquina realizando la traducción.

Las aplicaciones capaces de sintetizar textos, generar resúmenes y extraer información ayudan a identificar datos y entidades relevantes: temas, sitios, eventos, fechas, precios, etc. La identificación clara de los datos más relevantes puede ayudar a mejorar la eficiencia y eficacia de los motores de búsqueda o modelos IA basados en entradas de texto. Para la extracción de información, históricamente se han utilizado modelos de recuento de palabras y la aplicación de modelos multinomiales (Mccallum & Nigam 2001), previa limpieza de las palabras que no aportan información relevante.

2.2.3 Procesado de textos

Las herramientas de preprocesado se basan en una serie de pasos para extraer la información más relevante de cada texto analizado, buscando reducir la carga de procesamiento que conlleva analizar todo el texto en bruto, sin que esta reducción de carga suponga una reducción significativa de información. Estas técnicas incluyen, entre otras, la segmentación de frases, la tokenización, el stemming, la lematización, la eliminación de *'stopwords'*, etc. (Khurana et al. 2017)

2.2.3.1 Tokenización

La tokenización es el proceso de dividir una cadena de texto en *tokens* más pequeños. La división se puede realizar en diferentes sub-unidades del texto: palabras, frases, párrafos, páginas, etc. Estos *tokens* también pueden incluir números, puntuación u otros fragmentos de texto que ùeden contribuir a la comprensión. La tokenización es un paso de preprocesamiento común en la mayoría de las aplicaciones de procesamiento del lenguaje natural (NLP).

2.2.3.2 Flitrado de stop words

La eliminación o filtrado de *stop words* es el proceso de eliminar palabras que no aportan significado relevante a un texto. Esto se puede hacer eliminando palabras que no son esenciales para el significado del texto, como "un" y "el", o eliminando palabras que no son relevantes para los temas de interés específicos que se analizan. Estas palabras suelen ser palabras de alta frecuencia, por lo que es importante filtrarlas para reducir el tamaño de un documento de texto y mejorar la eficiencia del procesamiento de texto.

2.2.3.3 Stemming & Lemmatisation

Stemming es el proceso de reducir una palabra a su forma base o raíz, la parte de la palabra que es común a todas sus formas derivadas. Por ejemplo, la raíz de la palabra "hablando" es "habl", y la raíz de la palabra "hablé" también es "habl". Esto se hace para normalizar las palabras para la comparación, ya que las palabras con la misma raíz a menudo tienen el mismo significado, o se refieren a un término similar. Stemming es un proceso simple de cortar sufijos y prefijos, con un conjunto muy simple de instrucciones, valorando la eficiencia sobre la precisión.

La lematización es también un proceso de reducción de una palabra a su forma base. La lematización es un proceso mucho más laborioso con muchas más reglas que busca tener en cuenta todas las formas irregulares de las palabras y normas gramaticales de un idioma. Esto se hace eliminando las terminaciones y devolviendo la palabra a su forma de diccionario, que puede ser diferente de su forma original, como "hablando" que se convierte en "hablar". Este proceso requiere encontrar el lema, o la forma de diccionario, de una palabra, lo que puede ser un proceso laborioso si el objetivo es cubrir todas las palabras de un idioma. Este proceso valora la precisión sobre la eficiencia, y existen muchas variaciones en el conjunto de instrucciones y la cantidad de complejidad, por lo que no existe una solución definitiva para ningún idioma.

2.2.3.4 Vectorization

La vectorización es el proceso de representar texto sobre vectores numéricos, de manera que pueda ser usada por un modelo de aprendizaje automático, encontrar similitudes entre textos, buscar temas en común (*topic modeling*), etc. El detalle completo sobre la representación numérica de textos se encuentra en la sección 2.3.

2.3 Representación de textos

La representación numérica de textos se puede afrontar de varias maneras: la forma convencional es crear un vector para cada palabra en el texto y luego representar cada frase o documento como un vector de la suma de los vectores de sus palabras (Bag Of Words). Este proceso se puede refinar aún más ponderando los vectores de cada palabra según su importancia en el texto, o usando métodos más sofisticados, como la generación de *"word embeddings"*.

2.3.1 Word embeddings

Los *"word embeddings"* son un tipo de representación de palabras que permite que las palabras con un significado similar tengan una representación similar. Esta representación distribuida de texto es posiblemente uno de los mayores avances avances en NLP. Los *"word embeddings"* se suelen crear haciendo uso de redes neuronales llamadas transformers (sección 2.3.2), que ofrecen un rendimiento impresionante en la velocidad de procesamiento y aprendizaje profundo en los problemas más desafiantes del procesamiento del lenguaje natural.

Los *"word embeddings"* se pueden interpretar de manera similar a las representaciones vectoriales de palabras en una capa oculta de un modelo de lenguaje basado en redes neuronales. La capa oculta en el modelo de lenguaje es una representación vectorial continua (continua en el espacio vectorial de esa capa) de las palabras de entrada, y se busca que esta representación vectorial capture las relaciones sintácticas y semánticas entre las palabras. Generalmente, los *"word embeddings"* se generan a partir de grandes cantidades de datos de texto sin etiquetar.

2.3.1.1 Word2Vec

Los modelos de *"embeddings"* de palabras más populares son los modelos Word2Vec introducidos por Mikolov et al. 2013. Hay dos tipos principales de modelos Word2Vec: el modelo *"continuous bag of words"* (CBOW) y el modelo skip-gram.

El modelo CBOW predice la palabra actual dado el contexto de las palabras que la rodean, mientras que el modelo skip-gram predice las palabras circundantes dada la

palabra actual. El modelo CBOW suele ser más rápido de entrenar que el modelo skip-gram, pero el modelo skip-gram es más preciso para palabras poco frecuentes.

Tanto el modelo CBOW como el skip-gram se entrenan utilizando un softmax jerárquico, un tipo de arquitectura de red neuronal eficiente para entrenar generadores de *embeddings* de palabras.

2.3.1.2 Aplicaciones de los embeddings

Una vez que se aprenden los *embeddings* de palabras, se pueden utilizar para distintas tareas de aprendizaje. La forma más común de usar *embeddings* de palabras es inicializar los pesos de una red neuronal para una tarea de procesamiento del lenguaje natural. Por ejemplo, los *embeddings* de palabras se pueden usar para inicializar los pesos de una red de memoria a corto plazo (LSTM) para el etiquetado del "*Part of Speech*" (PoS), capaz de identificar la función gramatical de cada palabra.

Los *embeddings* de palabras también se pueden usar como entradas en un modelo tradicional de aprendizaje automático. Por ejemplo, los modelos Word2Vec se pueden usar para entrenar un modelo de regresión, clasificación o aprendizaje no supervisado. En el caso de este estudio, se aplicará un algoritmo de clustering (aprendizaje no supervisado) para clasificar los textos en grupos.

Finalmente, los *embeddings* de palabras se pueden usar para crear un modelo de similitud de palabras, donde el objetivo es predecir cómo de similares son dos palabras. Esto puede ser útil para encontrar sinónimos o palabras que se utilizan en contextos similares.

2.3.2 Transformers

Los modelos Transformer son una clase de modelos de aprendizaje automático que se han convertido en muy populares en los últimos años. Estos modelos se caracterizan por su capacidad de aprender representaciones de texto de alta calidad a partir de datos de texto sin etiquetar. Los modelos Transformer han demostrado ser particularmente efectivos en tareas de procesamiento del lenguaje natural, como el análisis de sentimientos, la traducción automática y el resumen de texto.

El modelo Transformer se entrena usando una red neuronal compuesta exclusivamente

por capas de atención. Una capa de atención es un tipo de capa de red neuronal que ayuda al modelo a centrarse en la información relevante mientras filtra la información de menor importancia. Esto se hace ponderando los datos de entrada según su importancia para la tarea en cuestión. Por ejemplo, en la traducción automática, la capa de atención se usa para centrarse en las partes relevantes del texto de entrada con el fin de producir una traducción precisa.

El modelo está compuesto por un codificador y un decodificador. El codificador toma una secuencia de palabras y las convierte en un vector de longitud fija (*embedding*). Luego, el decodificador toma el vector e intenta generar de nuevo la misma secuencia de palabras. A medida que se entrena la red neuronal, el codificador aprende a codificar la información más valiosa para transmitirle al decodificador todo lo necesario para reconstruir la entrada.

2.3.2.1 BERT

BERT es un modelo de Transformer parcialmente pre-entrenado que se introdujo en 2018 (Devlin, M. Chang, et al. 2018). Fue desarrollado por el equipo de Inteligencia Artificial de Google y se basa en la arquitectura del transformer (codificador, decodificador). El modelo se puede ajustar en una variedad de tareas distintas enseñándole nuevos textos afinados a la causa que se busca darle al modelo.

Se ha demostrado que BERT supera a otros modelos de última generación en una variedad de tareas. En un estudio, se descubrió que BERT logró una *accuracy*[1] del 95,63% (Sun et al. 2019) en el análisis de sentimientos de un conjunto de datos de 50 000 reseñas de películas de IMDB (Maas et al. 2011). Esto es significativamente más alto que el modelo de última generación anterior, que logró una *accuracy* por debajo del 90%. En conclusión, BERT es una poderosa herramienta para la representación de texto, siempre que sea ajustado correctamente (*fine-tuning*). Se ha demostrado que supera a otros modelos de última generación en una variedad de tareas.

[1]La *accuracy* es una métrica utilizada en problemas de clasificación que se utiliza para indicar el porcentaje de predicciones correctas. Lo calculamos dividiendo el número de predicciones correctas por el número total de predicciones.

2.3.2.2 DistilBERT

Desarrollado por un equipo de investigadores de *Google AI* (Sanh et al. 2019), es una versión destilada de BERT, un popular modelo parcialmente pre-entrenado para el procesamiento del lenguaje natural (NLP). DistilBERT es más pequeño y más rápido que BERT, pero conserva gran parte de su poder predictivo. El modelo se entrena utilizando una técnica de destilación conocida como "*knowledge distillation*", que implica entrenar un modelo de menor tamaño para imitar la salida de un modelo más grande.

DistilBERT mantiene el 97% de la *accuracy* original de BERT y consigue reducir el tamaño y la complejidad originales del modelo en más de un 40%, sus ejecuciones tardan de media un 60% menos. El modelo $BERT_{BASE}$ contiene 110M de hiperparámetros (Devlin, M. Chang, et al. 2018), existiendo una variante $BERT_{LARGE}$ de 340M de hiperparámetros. En el caso de DistilBERT, el modelo cuenta con 66M hiperparámetros.

2.3.2.3 RoBERTa & DistilRoBERTa

El modelo RoBERTa ("*Robustly Optimized BERT Pre-training Approach*") es una mejora con respecto al modelo BERT, siendo sometido a tiempos de entrenamiento mucho más prolongados y utilizando muchos más textos de entrenamiento. El modelo presenta una *accuracy* mucho mayor. Fue presentado por el equipo de Inteligencia Artificial de Facebook (Liu et al. 2019) basándose en el artículo de BERT y buscando mejorar las prestaciones del modelo. El modelo RoBERTa tiene también un tiempo de entrenamiento mucho más rápido que BERT.

DistilRoBERTa sigue el mismo procedimiento de entrenamiento y destilación que DistilBERT, pero se aplica sobre el modelo de RoBERTa. Según sus desarrolladores, el modelo tiene 6 capas, 768 dimensiones y 12 cabezas, con un total de 82M de parámetros (en comparación con los 125M de parámetros del $RoBERTa_{BASE}$). De media, DistilRoBERTa es el doble de rápido que $RoBERTa_{BASE}$.

2.4 Inteligencia Artificial Aplicada a medicina

El procesamiento del lenguaje natural ya se utiliza en distintos ámbitos de medicina (Khurana et al. 2017). El proyecto "Linguistic String Project-Medical Language Processor" es uno de los proyectos a mayor escala en el campo de la medicina (Chi et al. 1985) (Sager et al. 1995), su objetivo es facilitar la recuperación de información específica de los textos en respuesta a las consultas de los investigadores, para buscar así posibles signos o síntomas, dosis de medicamentos, etc. con el objetivo de identificar los posibles efectos secundarios de cualquier medicamento.

Por otro lado, la Biblioteca Nacional de Medicina en los Estados Unidos está desarrollando el proyecto "The Specialist System" (McGray et al. 1987) (McCray & Nelson 1995). Se busca que funcione como herramienta de extracción de información para las bases de conocimiento biomédicas, en particular los abstracts de Medline. El léxico se ha creado utilizando el MeSH (Medical Subject Headings), Dorland's Illustrated Medical Dictionary y diccionarios de inglés general.

2.4.1 Aplicaciones de la Inteligencia Artificial al campo de las migrañas

Existen algunas aplicaciones para la migraña que buscan mejorar la calidad de vida de los usuarios que padecen esta enfermedad. Sin embargo, la mayoría de estas aplicaciones se centran en el seguimiento de los síntomas más que en la predicción de futuros dolores de cabeza. De entre ellas, la más destacada es *Migraine Buddy*, una aplicación desarrollada por Bluebird con la ayuda de neurólogos que permite a los usuarios registrar sus síntomas para entender sus propios desencadenantes e identificar los mejores alivios para ellos.

Entre los síntomas que se pueden seguir con esta aplicación se encuentran la frecuencia y la duración de la migraña, la localización del dolor y su intensidad. También permite registrar la toma de medicamentos. La siguiente tabla muestra un resumen de todas las variables que permite registrar la aplicación:

Figura 1: Variables registradas en *Migraine Buddy*

Feature	Options	Feature	Options	
Type of headache	Migraine	Relief Method	Dark room	Hot Shower
	Tension-type		Sleep	Cold Shower
	Cluster		Yoga/Meditation	Drink Water
	Postdrome		Staying Inside	Heating Pad
	Prodrome		Ice bags	Other
			Food	Cafeine
Sleep	Start time	Symptoms	Throbbing Pain	Dizziness
	End time		Pulsatile Pain	Congestion
Weather			Stabbing Pain	Insomni
Intensity	1-10 scale		Fatigue	Anxiety
Initial pain	Head location map		Nausea	Odor Sensitivity
Medication	Maxalt		Vomit	Heat
	Sumatriptan		Light Sensitivity	Blurred Vision
	Topiramate		Noise Sensitivity	Mood Changes
	Naproxen		Cervical Pain	Diharrea
	Zomig			
	Paracetamol			
	Ibuprofen			
	Other			

Esta aplicación se centra en el reconocimiento de los diferentes desencadenantes individuales que pueda tener cada paciente y los tratamientos que surten más efecto antes que en la predicción de futuros ataques. Por lo tanto, a conocimiento de los autores, la predicción de migrañas utilizando Inteligencia Artificial es un campo en el que queda aún queda mucho por desarrollar.

3 Desarrollo

Este apartado detalla la metodología seguida durante la fase de desarrollo del estudio, y los resultados encontrados a lo largo del proyecto.

3.1 Preprocesado de Datos

El set de datos completo consta de 38.944 historias clínicas anonimizadas, en las cuales tanto la identidad de los especialistas como la de los pacientes se encuentran en formato numérico y parametrizados de manera anónima, para evitar infringir en la confidencialidad de los pacientes. Por esta razón, todos los datos que sean mostrados en el estudio presente serán agregados, mostrando conjuntos anónimos de datos.

3.1.1 Datos en crudo

Los datos se descargan ya anonimizados desde el servidor de la Clínica Ilion en formato `.xlsx` de Excel. Son importados en Python mediante la librería de especializada en tablas llamada *Pandas* detallada en la sección 1.3.4.3. Una vez se han leído correctamente los datos del Excel, se procede a manipular los datos de entrada.

3.1.1.1 Limpieza de los datos

El primer paso de cualquier estudio que trata con textos es realizar una limpieza preliminar de los textos para ser procesados correctamente. En el caso de Python, esta limpieza se hace a través de expresiones regulares (RegEx). A continuación se muestra un ejemplo de expresión regular utilizada para la limpieza de textos, que consiste en buscar caracteres y símbolos no deseados:

Consulta RegEx: `[^a-zA-Z.\s:ñÑáéíóúÁÉÍÓÚ]`

Su funcionamiento es el siguiente:

- " ^ " Indica que la expresión busca cualquier caracter que **no se incluya** en la expresión regular.
- " $a - z$ " Coincide con un carácter en el rango entre a (índice 97) y z (índice 122)

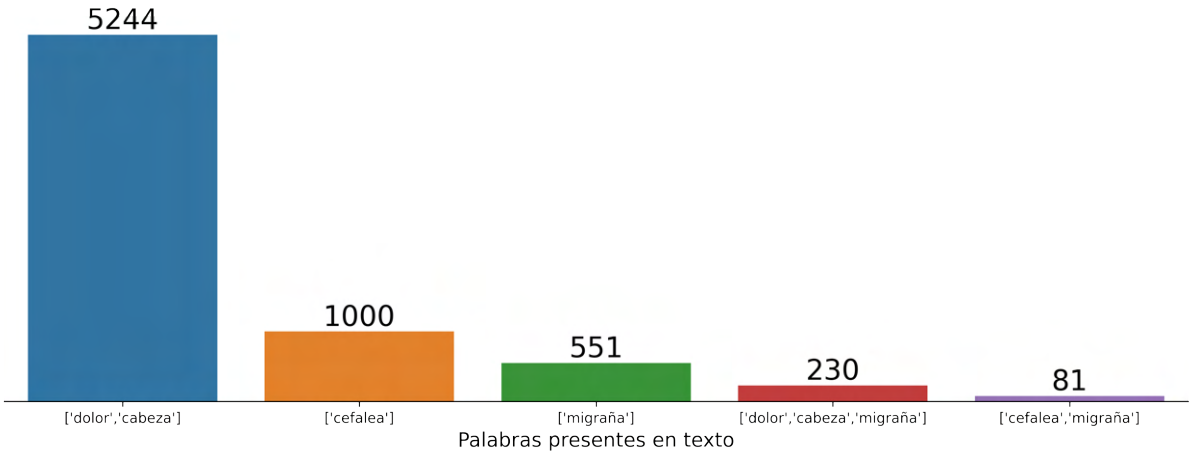
- “ $A - Z$ ” Coincide con un carácter en el rango entre A (index 65) y Z (index 90)
- “ . ” Coincide con el carácter ”.” (index 46_{10})
- “ $\backslash s$ ” Cualquier carácter de espacio en blanco (equivalente a [`\r,\n,\t,\f,\v`])
- “ `:ñÑáéíóúÁÉÍÓÚ` ” Coincide con un carácter en esa lista.

Como vemos en el desglose anterior, la consulta RegEx busca todos los caracteres que no se encuentren listados anteriormente, y una vez son identificados se procede a eliminarlos del set de datos utilizando la función `replace()` propia de las cadenas de caracteres en Python, limpiando así las frases y facilitando su comprensión.

3.1.1.2 Filtrado de historias clínicas

En segundo lugar, se deben filtrar las historias clínicas relativas a pacientes que padecen (o muestran síntomas de) migrañas. De las 38.944 historias clínicas, solo una fracción de ellas pertenecen a pacientes que sufren migrañas. Para identificar a estos pacientes, se han realizado varias búsquedas de palabras clave dentro de las historias clínicas:

Figura 2: Distribución de Historias Clínicas por búsqueda de *keywords*



Como se puede observar en la fig. 2, se reduce sustancialmente el set de datos con respecto a las 38.944 historias clínicas (HCs) que originalmente formaban parte del set de datos. Al filtrar historias clínicas de pacientes que cumplen las características que busca el estudio, no importa si se mezclan en el dataset algunas HCs de pacientes que no sufren migrañas, ya que al realizar el clustering de los datos, estos datos *outliers*[2] no tendrán

[2]Un outlier es un valor atípico en un conjunto de datos. Los valores atípicos pueden deberse a errores

un impacto significativo en la formación de clusters.

3.1.1.3 Eliminación de stop words

Siguiendo la definición de filtrado de stopwords detallada en la sección 2.2.3.2, se utiliza un corpus ya creado de stop words en español para realizar este filtrado. El corpus proviene de NLTK ("*Natural Language ToolKit*"), y contiene 313 palabras que carecen de información relevante.

Seleccionando 12 stop words en español de manera aleatoria se encuentran palabras como: `['de','la','que','el','en','y','a','los','del','se','las','por']`

El uso de esta librería en Python para la limpieza de cadenas es muy simple, y sigue la estructura mostrada en el listado de código 1:

Listado de cógido 1: Limpieza de stop words en Python

```
# Corpus de stopwords de Librería NLTK
from   nltk.corpus import stopwords

# Se carga la lista de 313 palabras:
spanish_stopwords = stopwords.words('spanish')

def filtrarStopWords(vectorDePalabras, vectorDeStopWords):
    # La función utiliza una 'list comprehension' nativa de Python para
     recorrer de manera eficiente la lista de palabras y filtrar
    aquellas que se encuentren en la lista de stopwords
    return [palabra for palabra in vectorDePalabras if palabra not in
    vectorDeStopWords]
```

Podemos probar la salida de la función anterior sobre una cadena simple, y comprobar que las palabras que carecen de significado son filtradas correctamente. El listado de código 2 muestra el resultado de aplicar la función `filtrarStopWords()` sobre la frase "salí a pasear por la tarde", que ya se encuentra tokenizada en distintas palabras:

de medición, errores de entrada o simplemente valores extremos. En estadística, se suele usar el término "outlier" para referirse a un valor atípico que se aleja significativamente del resto de los valores en el conjunto de datos.

Listado de cógido 2: Ejemplo de salida de limpieza de stop words

```
filtrarStopWords(['salí','a','pasear','por','la','tarde'],
    spanish_stopwords)
>>> ['salí','pasear','tarde']
```

Tras filtrar las stopwords se puede empezar a realizar un análisis preliminar del contenido del dataset. La mejor manera de ver gráficamente las palabras de mayor frecuencia dentro del set de datos es con un *wordcloud*. La fig. 3 muestra las palabras de mayor frecuencia, así como **los pares de palabras** que aparecen juntos con mayor frecuencia dentro de las historias clínicas seleccionadas.

Figura 3: Wordcloud preliminar de historias clínicas

Es importante volver a mencionar que algunas palabras aparecen repetidas ya que se muestran conjuntos de palabras y **pares de palabras** (llamados *bigram*). Por ello, la palabra "dolor" aparece una vez de manera individual, y vuelve a aparecer en el bigram ("dolor","cabeza").

Observando la fig. 3 se perciben ciertas palabras redundantes debido a sus múltiples formas: "dolor" y "dolores" por ejemplo aparecen en grande. El próximo paso es normalizar las palabras utilizando stemming o lemmatization (sección 2.2.3.3).

3.1.1.4 Stemming & Lemmatisation

Para este estudio se ha utilizado *lemmatisation* en lugar de *stemming*, debido a ciertas ventajas que proporciona la lematización. La lematización busca preservar el significado de la palabra, mientras que el *stemming* solo trata de encontrar la raíz de la palabra. La lematización también es más precisa que el *stemming*, ya que utiliza un diccionario para encontrar la forma base de la palabra, en vez de basarse en una lista reducida de normas que se aplican por igual a todas las palabras. Por esta razón, la lematización puede manejar palabras compuestas de forma más eficiente que el *stemming*.

La implementación de la lematización en Python se muestra simplificada en el listado de código 3 mediante el uso de la librería de SpaCy, y utilizando el pipeline[3] NLP de idioma Español llamado "`es_core_news_sm`". Existen variaciones de este pipeline optimizados para uso en GPU, o con mayor tamaño, aumentando la robustez pero también los tiempos de computación.

Listado de cógido 3: Lematización de frases en Python (simplificado a una frase)

```
import spacy

def lemmatize(frase):
    # Devuelve la versión lematizada de una frase.
    nlp = spacy.load("es_core_news_sm") # Pipeline NLP en Español optimizado para uso en la CPU.
    doc = nlp(frase)
    # Devuelve el lemma en el caso de que la palabra no esté vacía:
    return [(w.lemma_).strip() for w in doc if (w.lemma_).strip()]
```

Podemos probar la salida de la función anterior sobre una cadena simple y comprobar que las palabras a la salida se encuentran correctamente transformadas a su raíz común. El listado de código 4 muestra el resultado de aplicar la función `lemmatize()` sobre la frase "Tengo un dolor en las cervicales y también me duele la cabeza. Llevo con migrañas desde Enero":

[3]Un *Pipeline* o " *Natural Language Pipeline*" es una serie de pasos que se requieren para procesar datos de lenguaje natural para que puedan ser utilizados por un algoritmo de aprendizaje automático. Los pasos en un Pipeline de procesamiento del lenguaje pueden incluir preprocesamiento, tokenización, lematización y eliminación de stopwords, etc.

Listado de cógido 4: Ejemplo de salida de la lematización de palabras

```
lemmatize('Tengo un dolor en las cervicales y también me duele la
    cabeza. Llevo con migrañas desde Enero')
>>> ['tener', 'uno', 'dolor', 'en', 'el', 'cervical', 'y', 'también', '
    yo', 'doler', 'el', 'cabeza', '.', 'llevar', 'con', 'migraña', '
    desde', 'Enero']
```

Una vez se ha realizado la limpieza completa del texto, se puede pintar un segundo *wordcloud* sin que existan palabras similares repetidas como ocurría con el caso de "dolor" y "dolores". La fig. 4 muestra el *wordcloud* de frecuencias de la totalidad de los datos tras realizar la limpieza completa:

Figura 4: Wordcloud final de historias clínicas

En la figura 4 se comprueba que la palabra "dolores" desaparece, así como cualquier otra palabra en plural y otras palabras derivadas de la forma original. Una vez los datos han sido preprocesados, el estudio continúa a la siguiente fase: la representación numérica de cada texto.

3.2 Representación de textos

Antes de aplicar ningún algoritmo de aprendizaje automático es necesario codificar los textos de manera que puedan ser evaluados en condiciones iguales, permitiendo comparar unos textos con otros y realizar operaciones matemáticas con ellos.

3.2.1 Bag Of Words

El *bag of words* o BOW es la manera más primitiva de codificar textos. Consiste en asignar un vector a cada palabra del set de datos y representar cada documento como un vector de la suma de los vectores de las palabras.

Una de las ventajas es que es muy simple de usar y comprender. No requiere de un gran procesamiento del texto, ya que se basa en un simple conteo de palabras. Otra ventaja es que es un método muy eficiente (en términos de tiempo y cómputo) para trabajar con textos grandes. También se trata un método robusto, ya que es poco sensible a la variación en el lenguaje y la gramática. Por último, este método produce una representación de los textos que es fácilmente manipulable por los algoritmos de machine learning.

La figura 1 muestra la codificación de 5 historias clínicas seleccionadas de manera aleatoria, y muestra solo 10 de las palabras más frecuentes del set de datos:

Tabla 1: Ejemplo de codificado BOW de historias clínicas

Nº de HC[4]	dolor	haber	tener	ser	hacer	cabeza	bien	año	ir	dia
23940	1	3	1	0	3	0	0	4	2	0
19197	3	1	2	3	1	1	1	0	0	2
3822	5	0	0	0	0	1	2	0	0	0
13074	2	0	1	0	0	1	0	0	0	0
26485	5	1	4	6	1	0	3	3	1	1

La codificación mostrada en la tabla 1 es el resultado de aplicar un `CountVectorizer()` de `scikit-learn` sobre las frases del set de datos.

El método de codificación con BOW presenta varias limitaciones. Una de las principales limitaciones de usar "Bag of Words" es que no tiene en cuenta el orden de las palabras en un texto. Esto es un problema porque el orden de las palabras a menudo es importante para el significado de un texto. Por ejemplo, la frase "el perro mordió al hombre" tiene

un significado diferente a "el hombre mordió al perro". *Bag of Words* también tiene problemas para capturar el significado de las frases más largas, ya que con tantas palabras se diluye el contenido total. Si se extendiera el tamaño de cada documento a un libro entero la suma de todas las palabras termina siendo tan grande que es prácticamente imposible discernir significado, temática o contexto de un texto.

Otra desventaja es que no se tiene en cuenta la gramática del texto. Esto puede ser un problema si el texto contiene frases complicadas o si se necesita analizar el texto para extraer información más detallada. Finalmente, este método también puede ser muy **costoso en términos espacio de almacenamiento**. Se necesita un gran espacio de almacenamiento para almacenar todas las palabras del vocabulario y el tiempo de procesamiento puede ser muy lento si el vocabulario es muy grande. Esto se debe a que cada texto debe tener un registro o columna por cada palabra[5] presente en la totalidad del set de datos. La codificación de las casi 6000 historias clínicas seleccionadas para el estudio contiene un total de 26.269 lemas distintos identificados.

3.2.2 Word2Vec

Word2Vec es un algoritmo de aprendizaje automático que toma un corpus de texto y genera un modelo de espacio vectorial. El modelo genera un conjunto de vectores, cada uno de los cuales representa una palabra del corpus **en un determinado contexto** de tal manera que: la palabra "dolor" en el contexto "siento un dolor profundo" y en el contexto "tiene un dolor leve" generarán dos codificaciones completamente distintas debido al contexto de la palabra. Los vectores se generan de manera que estén cerca en el espacio vectorial las palabras que se usan juntas con frecuencia en el texto, y separados si las palabras rara vez se utilizan en un mismo contexto. La documentación de Word2Vec y sus distintas implementaciones se encuentra detallado en la sección 2.3.1.1.

Word2Vec es capaz de otorgar un vector numérico a **cada palabra**. En este caso, Word2Vec se usa para traducir **cada frase completa** en vectores numéricos, aplicando el algoritmo a cada palabra en la frase y posteriormente creando un vector que se corresponde a la media de todos los vectores de palabras. Usando esta representación numérica

[5]La Real Academia Española distingue alrededor de 93.000 lemas distintos en el español, lo cual nos indica que un set de datos suficientemente grande debería tener tantas columnas como palabras en el idioma, **por cada texto presente en el set de datos**.

de cada vector, los datos pueden pasar a través de un algoritmo de clustering tradicional para clasificar frases por su contenido y contexto.

El listado de código 5 muestra una simplificación de la implementación de un modelo Word2Vec en Python acorde con la documentación de la librería `gensim`:

Listado de cógido 5: Implementación de Word2Vec en Python (simplificado)

```
# Constantes relativas al modelo W2V
SIZE_VECTORS = 500
WINDOW = 10
EPOCHS = 10
MIN_COUNT = 5

def trainWord2Vec(corpus,vector_size,window,epochs,min_count):
    from gensim.models import Word2Vec
    w2v = Word2Vec(corpus,
                   vector_size=vector_size,
                   window=window,
                   epochs=epochs,
                   min_count=min_count)
    return w2v

coprus = [
    ['Esto','es','una', 'frase'],
    ['Esto','es','otra','frase'],
    ['Una','ultima','frase'],
]

modelo = trainWord2Vec(corpus,SIZE_VECTORS,WINDOW,EPOCHS,MIN_COUNT)
```

Observando los parámetros de entrada (mostrados en el listado de código 5) del entrenamiento de un modelo Word2Vec, se distinguen 5 entradas del modelo:

Corpus: Lista de listas de palabras, donde cada lista de nivel inferior representa una frase del set de datos. Son las palabras sobre las que se va a entrenar el modelo.

Vector size: Tamaño del vector de codificación resultante. Si el `vector_size = 200` esto significa que cada palabra tendrá asignado un vector de 200 posiciones, y por lo tanto cada frase (calculada como la media de los vectores de sus palabras) también

tendrá longitud 200.

Window: Distancia máxima entre la palabra actual y las palabras contiguas utilizadas para generar contexto. Una venana de 10 indica que se tendrán en cuenta un máximo de 10 palabras posteriores y anteriores a la palabra actual para generar el vector resultante.

Epochs: Cantidad de veces que el modelo deberá recorrer el set de datos durante el aprendizaje.

Min. Count: Frecuencia mínima que una palabra específica debe aparecer en el set completo de datos para ser tenida en cuenta en la codificación. Ayuda a evitar palabras mal escritas o que no pertenecen al contexto relevante.

Tras entrenar un modelo de Word2Vec, se puede extraer información valiosa del propio modelo. Por ejemplo, el modelo permite mostrarle palabras presentes en el set de datos, y extraer un conjunto de palabras que más se asemejan a la palabra suministrada. La similitud entre palabras en Word2Vec viene definida por la distancia entre los vectores de dichas palabras en el espacio vectorial de embedding.

Tabla 2: Palabras más similares a "migraña" según modelo Word2Vec

Palabra	**Similitud**	**Palabra**	**Similitud**
jaqueca	0.741033	intenso	0.519610
aura	0.614441	menopausia	0.518515
vomito	0.610178	sien	0.512068
coincidir	0.607044	sintoma	0.509503
cefalea	0.604163	intensidad	0.509436
periodo	0.601950	mareo	0.508103
nausea	0.596306	jaquecas	0.501877
episodio	0.592682	hemorragia	0.501768
durar	0.573850	paracetamol	0.497956
fiebre	0.565255	naproxeno	0.495527
regla	0.561586	ojo	0.493749
vertigo	0.548945	frecuencia	0.486672
relacionar	0.546201	ultimo	0.485606
pildora	0.544630	dolor	0.485525
cefalear	0.540013	tension	0.484657

Cabe destacar que las palabras más similares según esta clasificación Word2Vec **no necesariamente indican palabras de igual significado**, sino palabras que **se utilizan en**

un mismo contexto, es decir, se rodean de las mismas palabras de la misma forma con cierto grado de probabilidad.

3.2.2.1 Representación de palabras con t-SNE

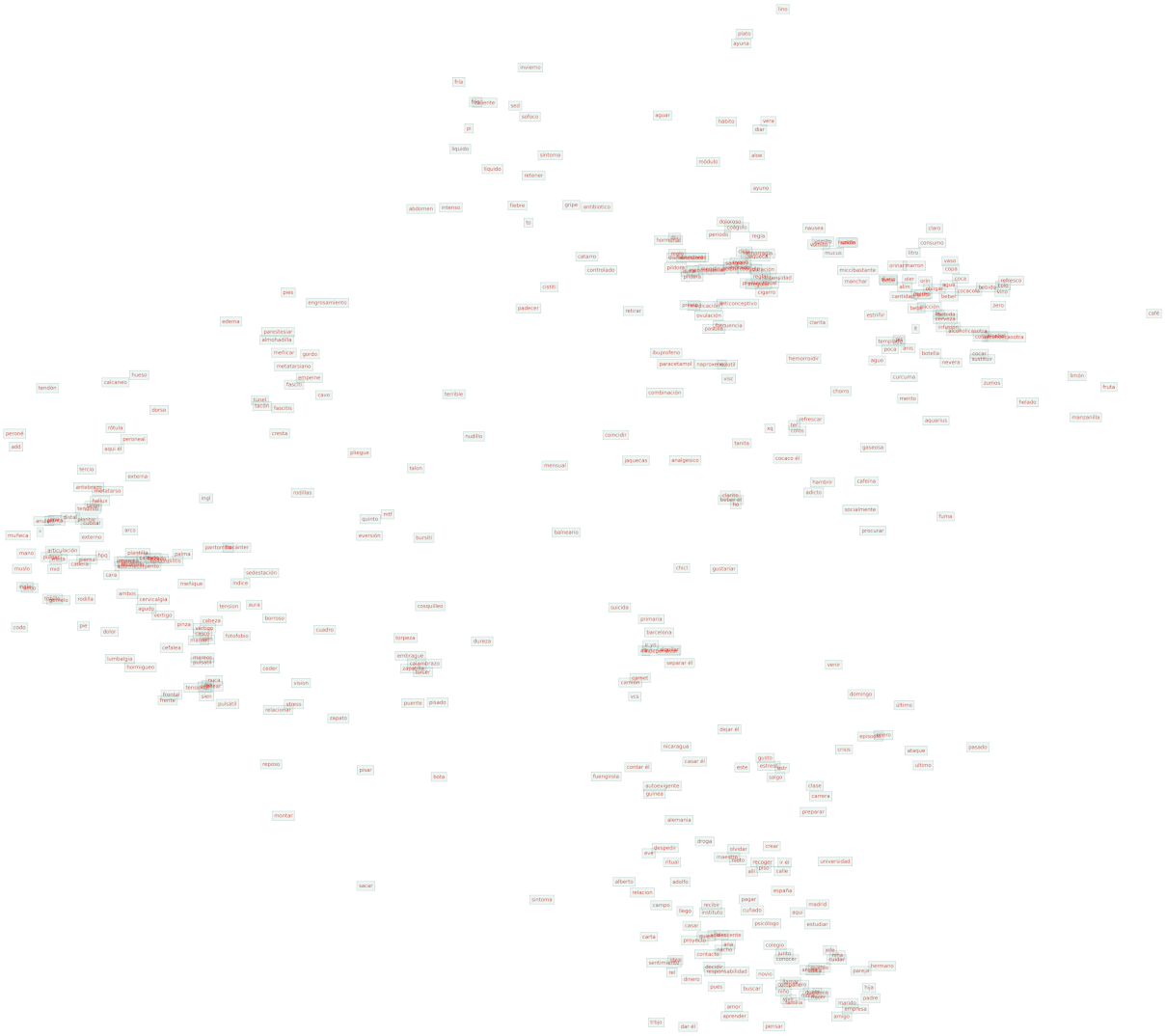
Utilizando los embeddings de las palabras presentes en el vocabulario, es posible realizar representaciones de la cercanía de palabras sobre 2 dimensiones. Es posible representar conjuntos de datos de dimensiones más altas en un espacio bidimensional utilizando t-SNE. El *embedding* de vecinos estocásticos distribuidos en T (t-SNE) es un algoritmo de aprendizaje automático para la visualización (Van der Maaten & Hinton 2008). Es una técnica de reducción de dimensionalidad no lineal que es particularmente adecuada para visualizar conjuntos de datos de alta dimensión. El algoritmo se basa en un modelo probabilístico del conjunto de datos llamado "*t-distributed stochastic neighbor embedding model*". El modelo define una distribución de probabilidad sobre las posibles asignaciones de los datos de alta dimensión a un espacio de menor dimensión. El mapeo se representa como un conjunto de puntos en el espacio de dimensiones inferiores, y la probabilidad de que un punto de datos se asigne a un punto en el espacio de dimensiones inferiores viene dada por una distribución gaussiana centrada en ese punto. El algoritmo t-SNE busca un mapeo que maximice la probabilidad de los datos bajo el "*t-distributed stochastic neighbor embedding model*".

El algoritmo t-SNE tiene varias ventajas sobre otras técnicas de reducción de dimensionalidad. Primero, es muy adecuado para visualizar conjuntos de datos con muchas variables. En segundo lugar, el algoritmo puede preservar la estructura local del conjunto de datos, lo cual es importante para visualizar conjuntos de datos con una estructura compleja. En tercer lugar, t-SNE es relativamente eficiente y se puede aplicar a conjuntos de datos con millones de puntos. Finalmente, el algoritmo es altamente flexible y se puede usar para una variedad de tareas, como visualizar conjuntos de datos con diferentes niveles de detalle o para visualizar conjuntos de datos con múltiples vistas.

Usando t-SNE, incluso si las matrices de las frases originales son objetos de 500 dimensiones, los datos se pueden visualizar de una manera mucho más eficiente que otras técnicas de reducción de dimensionalidad como PCA.

La figura 5 representa una selección de 400 palabras de alta frecuencia del set de datos:

Figura 5: Scatterplot de 400 palabras del Word2Vec a través de t-SNE



La representación de tantas palabras es dificil de interpretar, y eso que se han seleccionado solo 400 de las casi 8000 palabras registradas en el vocabulario del Word2Vec. Para interpretar los resultados mejor, se han seleccionado tres términos: "migraña", "dolor" y "fiebre", y se han seleccionado un conjunto de palabras similares a cada uno de éstos términos, de la misma manera que se ha realizado este cálculo de similitud en la tabla 2. Tras seleccionar estas palabras se pinta solo este conjunto más acotado.

La figura 6 representa una ampliación de la fig. 5 seleccionando palabras cercanas a los términos "migraña", "dolor" y "fiebre". Se muestran en la figura circuladas las tres secciones o *clusters* de palabras formados alrededor de los tres términos seleccionados:

Figura 6: Word2Vec: Scatterplot del embedding de palabras similares a “migraña”, “dolor” y “fiebre”

Observando la fig. 6 se empieza a entender el funcionamiento del Word2Vec, que otorga codificaciones muy similares a palabras que se suelen encontrar en un mismo contexto: “fiebre” con “resfriado” y “gripe” por ejemplo. Entendemos como contexto las demás palabras que rodean a una palabra concreta. Por esta interpretación en los textos observados el modelo ha determinado que estas palabras dentro de un mismo grupo se suelen encontrar rodeadas de las mismas palabras (mismo contexto).

Es importante destacar que esta representación es una aproximación en 2 dimensiones de vectores de 500 dimensiones, por lo que palabras muy similares pueden aparecer en lugares opuestos del plano ya que no se puede representar en dos dimensiones toda la información del embedding. t-SNE tiene esta limitación, aunque nos da un indicio muy bueno de si el modelo Word2Vec ha extraído información valiosa de las historias clínicas.

A diferencia del método BOW analizado en la sección 3.2.1, la representación de texto generada por Word2Vec muestra indicios de codificar información valiosa de cada frase correctamente, por lo que se utilizarán los embeddings generados por este modelo para

entrenar un modelo de aprendizaje no supervisado de clustering, con el fin de encontrar esos grupos potenciales de pacientes con casos similares. Antes de comenzar con el clustering, se debe estudiar un último método de representación de textos: *transformers*, que forman parte del estado del arte actual en representación de textos y una gran cantidad de otras áreas dentro de NLP.

3.2.3 Transformers

Los *Transformers* son una clase de modelos de aprendizaje automático utilizados para el procesamiento del lenguaje natural (NLP). Su popularidad radica en su capacidad de aprender la representación de los datos de entrada de forma auto-suficiente, con una arquitectura *encoder - decoder*, es decir, la primera mitad de la red neuronal (el *encoder*) realiza el embedding de la frase, y la segunda parte (el *decoder*) intenta recrearla y generar la misma frase que había a la entrada. Esta estructura permite que el modelo aprenda por su cuenta a codificar los embeddings, y con el propio error de recreación de la frase la red neuronal entera aprende a codificar este texto de la manera más eficiente para que su segunda mitad (que le propaga el error por medio de *backpropagation*[6]) sea capaz

Los *Transformers* se basan en el algoritmo de aprendizaje automático "*Attention*" (capa de atención), que les permite concentrarse en los datos de entrada relevantes para la tarea en cuestión. Esto hace que los *transformers* sean particularmente eficaces para tareas de procesamiento del lenguaje natural. Las capas de *Attention* no solo tienen un uso en NLP, modelos dedicados al análisis de imágenes y videos (*computer vision*) también tienen como estado del arte modelos basados en capas de atención. Información detallada sobre los *transformers* se puede consultar en la sección 2.3.2.

3.2.3.1 Selección del transformer

Los *transformers* son modelos de complejidad elevada que llevan días e incluso semanas o meses en entrenarse sobre sets de datos elevados. Por ello, para este estudio se va a utilizar un modelo ya entrenado sobre textos similares a los observados en las historias clínicas de este desarrollo. El modelo en cuestión se llama "`PlanTL-GOB-ES/roberta-base-biomedical-clinical-es`" (Carrino et al. 2021) y se trata de un desar-

[6]Backpropagation es un algoritmo de entrenamiento supervisado que permite a la red neuronal ajustar sus pesos y umbrales de forma iterativa para minimizar el error de predicción, traspasando el error cometido en la última capa de la red hacia las capas anteriores de manera gradual.

rollo del Centro Nacional de Supercomputación, localizado en Barcelona. El modelo se ha re-entrenado sobre el modelo $RoBERTa_{BASE}$ detallado en la sección 2.3.2.3, este proceso de reentrenamiento se denomina "*Fine-tuning*". *Fine-tuning* es el proceso de ajustar los parámetros de un modelo *transformer* para mejorar el rendimiento en una **tarea específica**, mediante el uso de sets de datos más enfocados a esa tarea específica.

Es importante destacar que este modelo fue creado con otra tarea en mente, el modelo busca realizar un rellenado de huecos en nuevas frases en base a observaciones pasadas. El modela ha sido entrenado sobre 2.172.491 documentos en español, que componen más de 43M de frases en total (Carrino et al. 2021). Su misión es llegar a igualar las prestaciones de modelos biomédicos entrenados en inglés como BioBERT (Lee et al. 2020), y superar el ámbito de la medicina a modelos que si entienden español pero son de uso general, como mBERT (Devlin, M. W. Chang, et al. 2019) y BETO (Cañete et al. 2020). El listado de código 6 muestra un ejemplo de la implementación del modelo `roberta-base-biomedical-clinical-es` utilizando la librería de Python *Hugging Face*:

Listado de cógido 6: Ejemplo del uso del modelo "roberta-base-biomedical-clinical-es"

```
Entrada: "El paciente sufre [MASK] desde la niñez."

Salida:
{
    {
        "token_str": " epilepsia",
        "sequence": " El paciente sufre epilepsia desde la niñez."
        "score": 0.09317338466644287,
      },
      {
        "token_str": " escoliosis",
        "sequence": " El paciente sufre escoliosis desde la niñez."
        "score": 0.04213542118668556,
      },
      {
        "token_str": " depresión",
        "sequence": " El paciente sufre depresión desde la niñez."
        "score": 0.03921464830636978,
      },
      {
        "token_str": " migrañas",
```

```
        "sequence": " El paciente sufre migrañas desde la niñez."
        "score": 0.035249773412942886,
      },
      {
        "token_str": " psoriasis",
        "sequence": " El paciente sufre psoriasis desde la niñez."
        "score": 0.03453437238931656,
      }
}
```

3.2.3.2 Implementación

El modelo, al igual que decenas de otros modelos *transformers*, son de acceso público a través de la librería de *Hugging Face*, y permite modificar las prestaciones de los modelos y utilizarlos con distintas aplicaciones.

El modelo seleccionado, al igual que la mayor parte de los *transformers*, consta de dos partes: el codificador que genera la representación del texto y la segunda parte que realiza la predicción mostrada en la sección de código 6. Las librería de `simpletransformers` permite extraer un modelo del repositorio público de *Hugging Face*, y utilizarlo para una variedad de aplicaciones, en el caso de este estudio se ha utilizado la porción del codificador del modelo para generar una representación numérica de los textos. El modelo ha sido entrenado para fijarse específicamente en relaciones y patrones en el campo biomédico, por lo que se espera que la codificación de textos generada mantenga un nivel de información relevante muy elevado.

Listado de cógido 7: Implementación del transformer (simplificado)

```
import numpy as np
import pandas as pd
from simpletransformers.language_representation import
    RepresentationModel
from simpletransformers.config.model_args import ModelArgs

model_args = ModelArgs(max_seq_length=512) # Secuencia máxima de tokens
     que admite el modelo

model = RepresentationModel(
    model_type = "roberta",
```

```
10      model_name = "PlanTL-GOB-ES/roberta-base-biomedical-clinical-es",
11      args = model_args,
12      use_cuda = True # Activa el uso de la GPU para realizar operaciones
13  )
14
15  # Ejemplo del formato de las frases de entrada al modelo
16  frases = [
17      "El paciente muestra síntomas de...",
18      "Lleva con dolor de cabeza durante...",
19      "Su dieta consiste de...",
20  ]
21
22  word_embeddings = model.encode_sentences(
23                                      frases,
24                                      combine_strategy="mean", # Modo de
        sumarización de los embeddings de cada palabra para generar el
        embedding de la frase.
25                                  )
```

El listado de código 7 se muestra una simplificación de cómo implementar el modelo mostrado en Python. La línea 8 genera un objeto "`RepresentationModel()`" que descarga el modelo del repositorio de *Hugging Face* y lo prepara para realizar representaciones de texto. La librería de `simpletransformers` permite implementar modelos del repositorio para diferentes tareas de manera simplificada. Estas tareas incluyen: `Text Classification`, `Text Representation`, `Question-Answer model`, etc.

La entrada del modelo consiste en una lista de distintas frases o textos que se buscan codificar. La salida consiste en un vector de longitud 768 que representa numéricamente sobre un espacio vectorial el texto original. El set de datos consta de 5839 textos seleccionados en relación a las migrañas, cada texto resultará en un vector de 768, por lo que la salida completa de este modelo resultará en un dataframe de tamaño: `(5839 x 768)`.

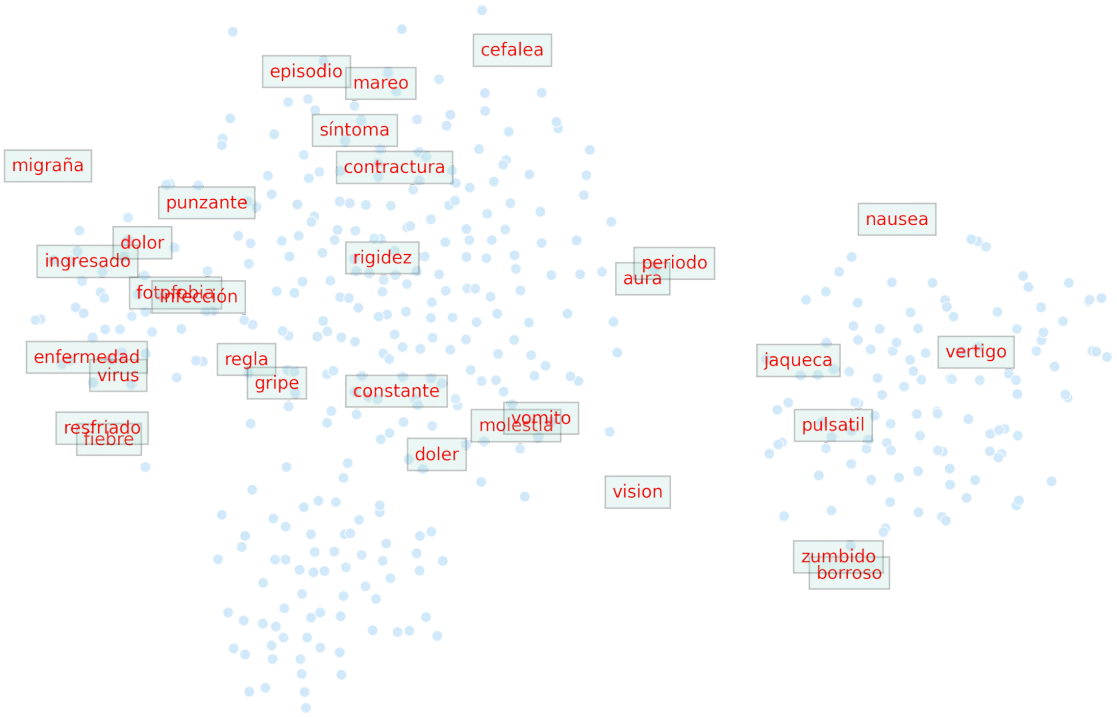
3.2.3.3 Representación de palabras con t-SNE

Los resultados de la representación de textos se pueden observar de manera similar a como se ha realizado en Word2Vec. El modelo acepta como entrada palabras individuales, y genera un embedding de esta palabra ausente de contexto. Mediante el uso de t-SNE, es posible representar en 2 dimensiones una aproximación de las agrupaciones de palabras,

y las distancias entre distintas palabras.

La figura 7 muestra un scatterplot[7] de las mismas 400 palabras seleccionadas en la sección del Word2Vec (3.2.2.1), y se han seleccionado para representar textualmente solo los términos cercanos a las palabras “migraña”, “dolor” y “fiebre”, las mismas que en Word2Vec.

Figura 7: Transformer: Scatterplot del embedding de palabras



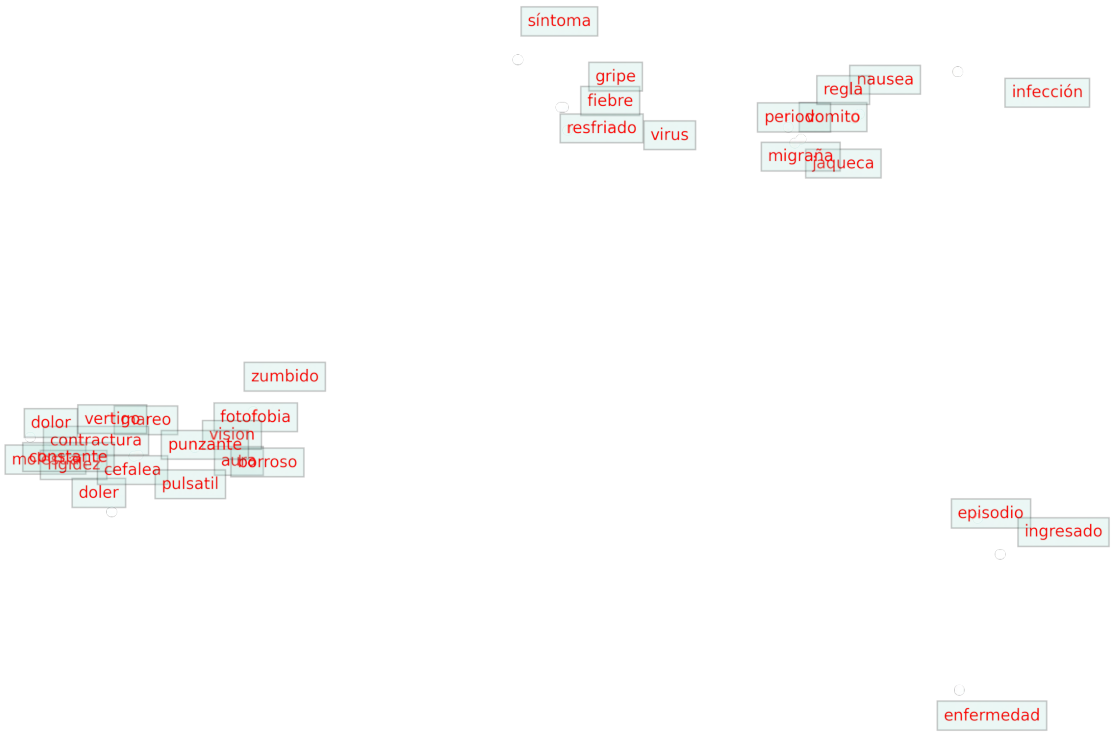
Se observa en la figura 7 que el modelo ha agrupado las palabras en distintos grupos, en base lo que el modelo ha considerado que es más importante para codificar las palabras y conservar la mayor cantidad de información posible. Existen similitudes con la codificación de Word2Vec, por ejemplo las palabras “resfriado” y “fiebre” se encuentran muy cerca.

Otras partes de la codificación difieren de lo observado en Word2Vec. La palabra “migraña” se encuentra muy cerca de las palabras “dolor” y “punzante”, pero la palabra “jaqueca” muy alejada.

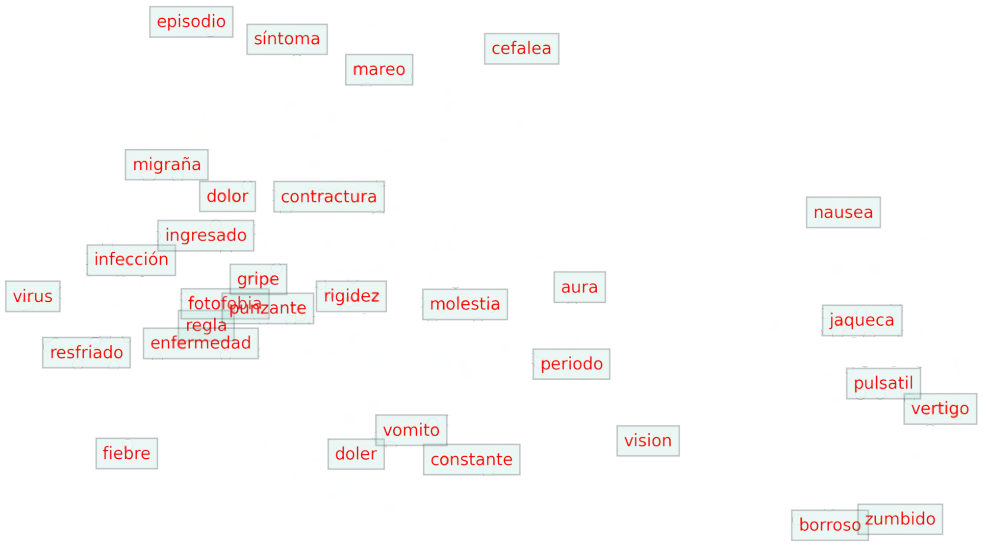
[7]Un scatterplot es un tipo de gráfico que se utiliza para representar datos en forma de puntos. Cada punto en el gráfico representa un valor en el conjunto de datos, y se coloca en el gráfico en función de su valor en el eje X y en el eje Y.

Figura 8: Comparación de Embeddings con t-SNE: Word2Vec vs. RoBERTa fine-tuned Transformer

(a) Word2Vec embedding



(b) Transformer embedding



La figura 8 muestra una comparación de las palabras agrupadas por cada uno de los modelos de representación textual. Como se ha mencionado en la sección 3.2.2.1, la representación con t-SNE es una aproximación en 2 dimensiones de vectores de 500 dimensiones en el caso de Word2Vec, y de 768 dimensiones en el caso de RoBERTa. Por esta razón, palabras muy similares pueden aparecer en lugares opuestos del plano ya que no se puede representar en dos dimensiones toda la información del embedding.

3.2.4 Visualización de textos

En las secciones previas se ha mostrado la **codificación de palabras sin contexto**, pero se busca codificar textos enteros de historias clínicas. Utilizando la implementación mostrada en los listados de código 5 y 7, se han codificado las 5839 historias clínicas. Antes de comenzar con el estudio de clustering (sección 3.3) es prudente representar las codificaciones para sacar expectativas iniciales. Una vez más, para la representación de datos de alto número de dimensiones se ha utilizado t-SNE (sección 3.2.2.1).

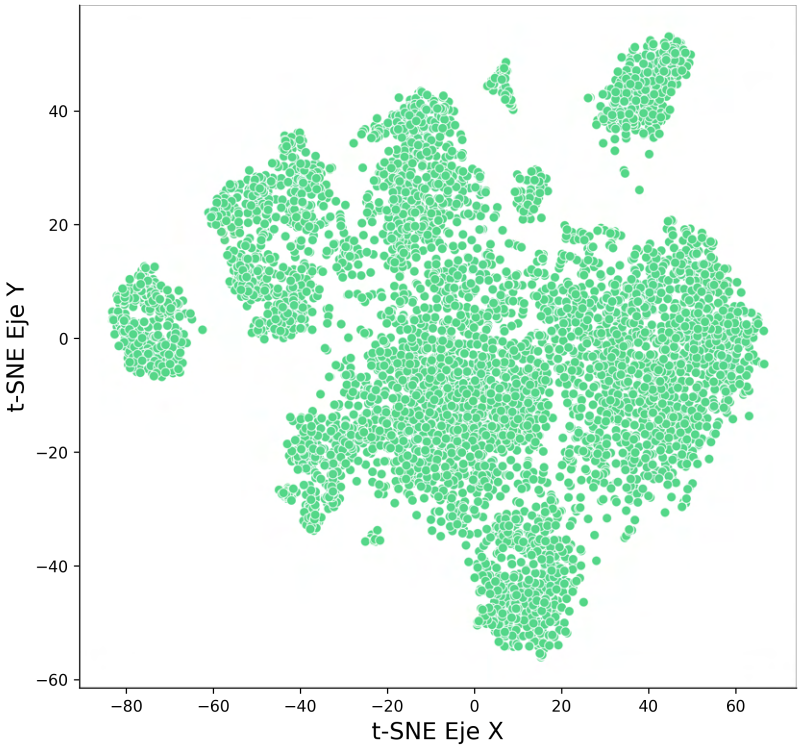
La figura 9 muestra un scatterplot de las historias clínicas sobre dos dimensiones (gracias a t-SNE) de las historias clínicas seleccionadas para el estudio. Se muestran lado a lado los *embeddings* generados por Word2Vec y por el *transformer* RoBERTa. Se busca que las representaciones numéricas de los textos se encuentren distribuidas de manera que sea fácil distinguir grupos claramente delimitados, ya que indica que los resultados de aplicar un algoritmo de clustering serán más eficaces.

La figura 9 muestra la posición de los *embeddings* de los textos, pero no muestra las distribuciones, es decir, si ciertas zonas tienen una acumulación mucho mayor de textos que otras. Por ello, se debe utilizar un KDE plot. Un KDE plot ("*Kernel Density Estimation plot*") es un tipo de gráfico de densidad de kernel utilizado para visualizar la distribución de un conjunto de datos. En lugar de agrupar los datos en contenedores de tamaño fijo, como es el caso de los histogramas, el KDE plot utiliza un kernel para estimar la densidad de puntos en cada ubicación. Nos permite representar sobre dos dimensiones la distribución de puntos.

La figura 10 muestra las densidades de los embeddings sobre los mismos ejes que la figura 9, utilizando KDE. Se busca identificar si existen varios máximos locales que inddican posibles centroides de los clusters que se quieren extraer, para realizar una estimación del número óptimo de clusters, así como observar si los datos se encuentran dispersos a concentrados en ciertas zonas.

Figura 9: t-SNE de los embeddings de las historias clínicas

(a) Word2Vec embedding



(b) Transformer embedding

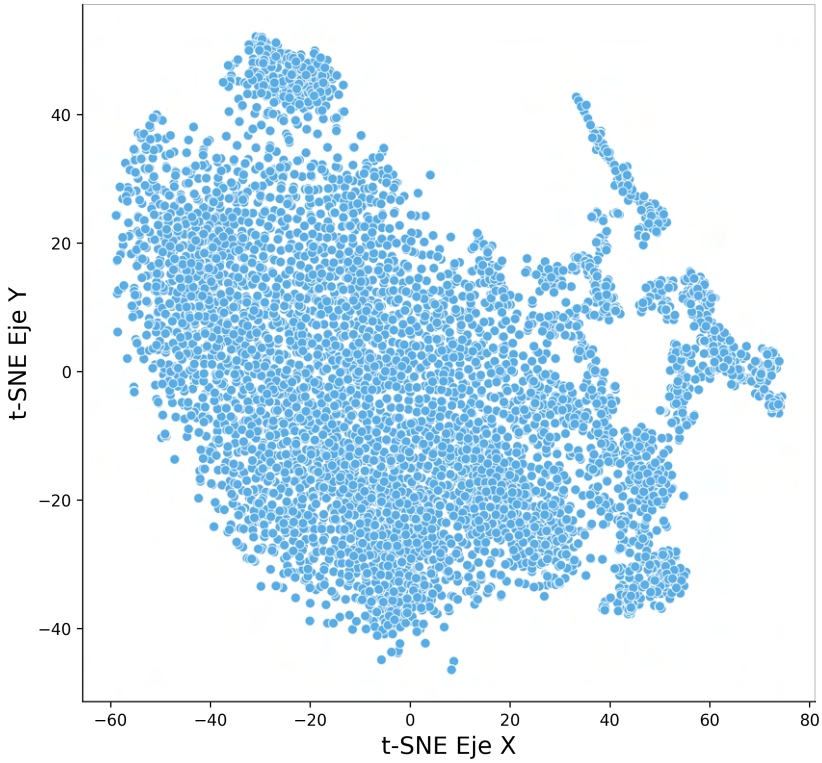
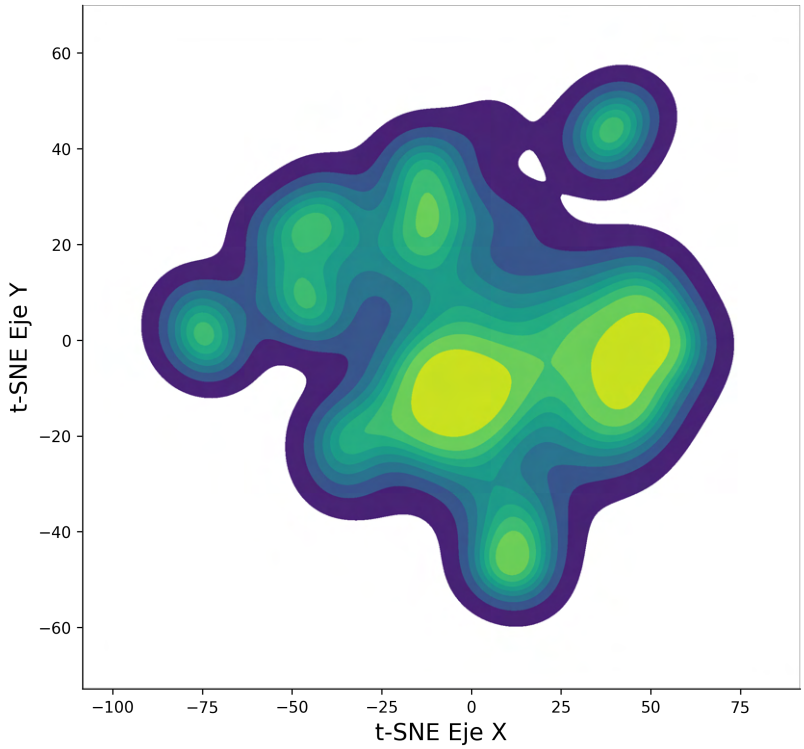
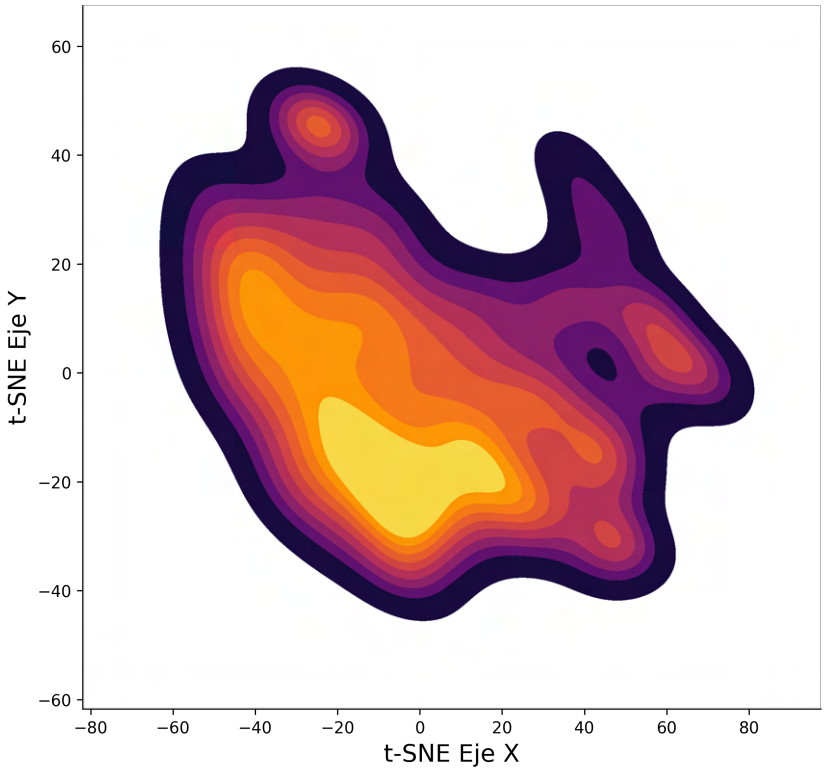


Figura 10: Densidad de los embeddings de las historias clínicas (KDE)

(a) Word2Vec embedding



(b) Transformer embedding



Observando las visualizaciones mostradas en las figuras 9 y 10, parece a priori que Word2Vec ha generado embeddings con una distribución más separada en distitnos grupos o clusters. Este resultado es esperable teniendo en cuenta que **el modelo Word2Vec se ha entrenado específicamente sobre este grupo de textos**, mientras que el modelo RoBERTa se ha pre-entrenado sobre un grupo mucho más amplio de textos (que no incluyen exclusivamente historias clínicas relativas a migrañas), y por tanto es posible que genere embeddings similares para todo el set de datos, ya que tratan exclusivamente sobre un grupo muy reducido de temas comparado con los datos sobre los que ha sido entrenado.

3.2.4.1 Previsión de resultados

Los KDE representados sobre la figura 10 nos dan un indicio sobre el número de clusters que debemos buscar, aunque no se deben tomar estas representaciones como una verdad absoluta, ya que sumarizan una parte muy pequeña de del set de datos de alto número de dimensiones.

La figura 10a muestra datos separados y organizados alrededor de 7 u 8 clusters distintos, y parece haber una separación clara entre los distintos clusters. Parece haber dos clusters centrales que agrupan un gran número de historias clínicas comparado con los demás grupos.

Por otro lado, la figura 10b muestra datos mucho más agrupados, existen 4 máximos locales, y uno de ellos agrupa la gran mayoría de los datos. No parece que el transformer, entrenado sobre un amplio repertorio de textos médicos, sea capaz de distinguir claramente entre distintos tipos de historias clínicas dentro del ámbito específico de las migrañas.

3.3 Clustering de historias clínicas

Una vez los datos de las historias clínicas han sido limpiados, procesados y codificados, es momento de encontrar los distintos grupos que componen el set de datos. Para ello, se ha utilizado un algoritmo de aprendizaje no supervisado llamado clustering. Clustering es un método de aprendizaje automático no supervisado que busca agrupar objetos en clusters basados en la similitud entre los distintos puntos. Se busca agrupar los datos de manera que los objetos en un cluster sean más similares entre sí que a los objetos en otros clusters. Clustering se puede usar para explorar y comprender los datos cuando no se tiene ya una salida en mente. En este caso no se saben de antemano los grupos que deben salir, por lo que gracias al clustering se es capaz de delimitar esos grupos.

Hay muchos algoritmos de clustering diferentes, y no hay un algoritmo que sea el mejor para todos los casos. El tipo de datos y el objetivo del clustering determinarán qué algoritmo es el más adecuado. Algunos algoritmos de clustering populares son K-Means, Clustering jerárquico y DBSCAN. Para los requisitos específicos de este estudio se utilizará K-Means. El algoritmo de clustering K-Means es un método de análisis de clusters que tiene como objetivo dividir n observaciones en k clusters en los que cada observación pertenece al clusters con punto medio (centroide) más cercano. Es un algoritmo de aprendizaje automático no supervisado que se utiliza para agrupar puntos de datos en un número predefinido de grupos, de forma que se minimice la suma de cuadrados de cada cluster (*within cluster sum of squares* - WCSS), es decir, la varianza intra-cluster. El algoritmo funciona inicializando aleatoriamente los centroides de los grupos y luego asignando cada punto de datos al grupo que tiene el centroide más cercano. Tras este paso, los centroides se actualizan para que sean la nueva media de todos los puntos de datos asignados a ese grupo. Este proceso se repite hasta que los centroides ya no cambian (converge el modelo) o se alcanza un número predefinido de iteraciones.

3.3.1 Reducción de dimensiones

Antes de comenzar a aplicar el algoritmo de clustering es importante comprobar, debido a la alta dimensionalidad de los embeddings, si es necesario mantener tantas dimensiones, o si se pueden reducir sin afectar a la varianza explicada. Si existen correlaciones claras entre distintas distintas columnas, es posible eliminar una de esas columnas conservando

un porcentaje elevado de la información codificada en ambas columnas, y permitiendo tiempos de procesamiento mucho menores. La forma más gráfica de mostrar posibles correlaciones entre las columnas es por medio de un mapa de calor representando todas las columnas.

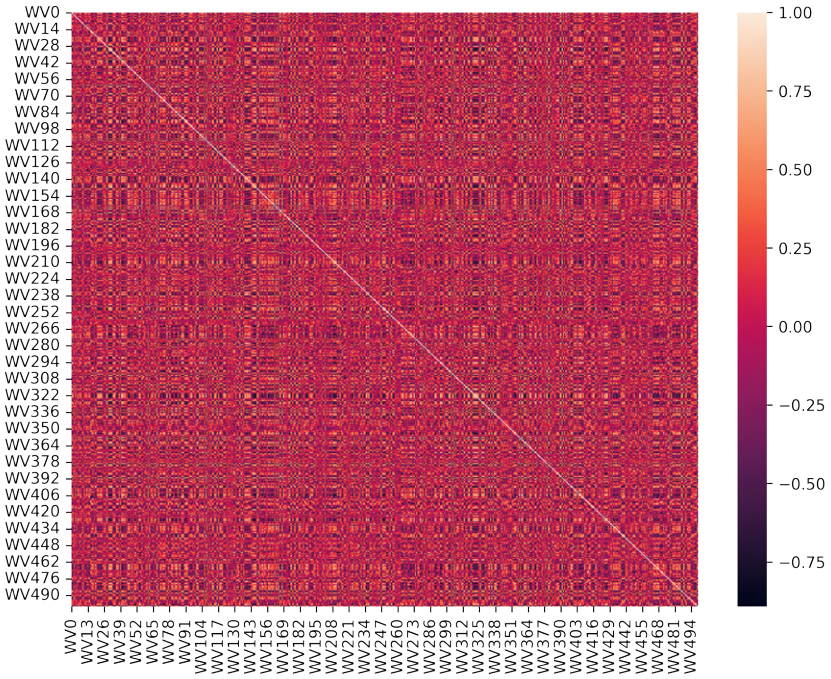
La figura 11 muestra dos mapas de calor representando por un lado las correlaciones entre las 500 variables generadas por Word2Vec, y por otro lado las correlaciones entre las 768 variables generadas por el transformer RoBERTa.

Es difícil discernir entre las distintas variables representadas en los mapas de calor de la figura 11, pero se saca una conclusión evidente: existen correlaciones altas entre variables de ambos embeddings, por lo que de debe estudiar el uso de métodos de reducción de dimensionalidad como PCA.

El análisis de componentes principales (PCA) es una técnica de reducción de dimensionalidad utilizada para simplificar la representación de los datos. PCA transforma el conjunto de datos de entrada en una forma que puede ser utilizada para identificar y visualizar patrones en los datos. PCA es una técnica no supervisada, lo que significa que no se utiliza ninguna etiqueta de clase o variable objetivo para realizar la transformación. En su lugar, PCA se basa en la correlación entre los atributos del conjunto de datos. PCA se puede utilizar para reducir la dimensionalidad de un conjunto de datos de alta dimensionalidad, reducir el ruido en los datos y para visualizar los datos en un espacio de dimensionalidad reducida.

Figura 11: Mapas de calor de correlación entre variables

(a) Variables del Word2Vec



(b) Variables del Transformer

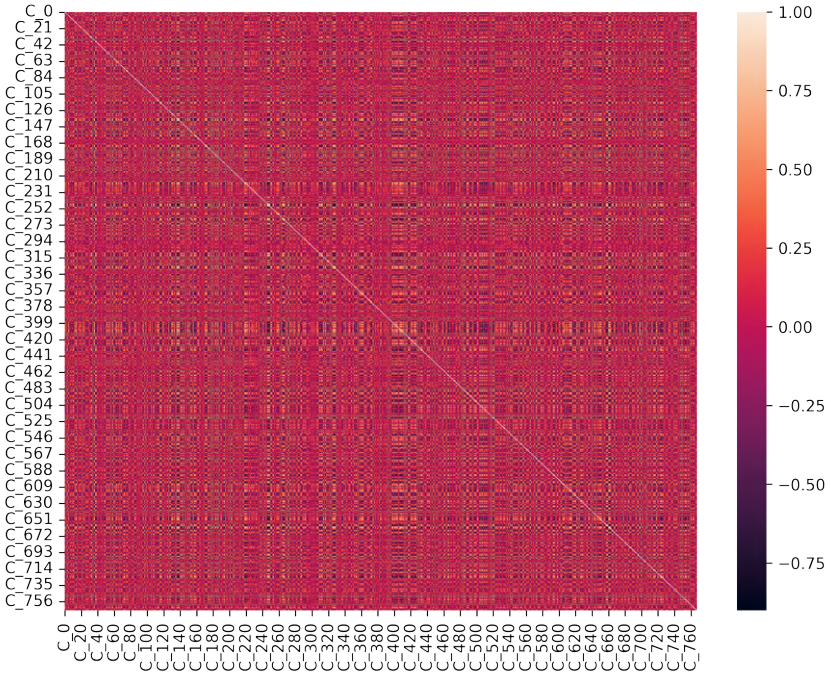
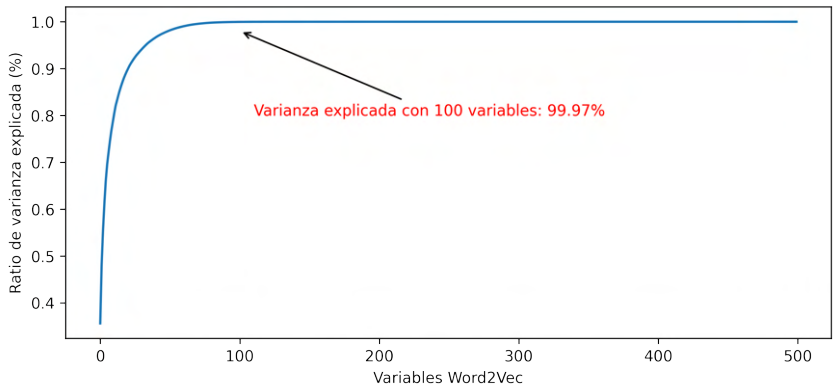
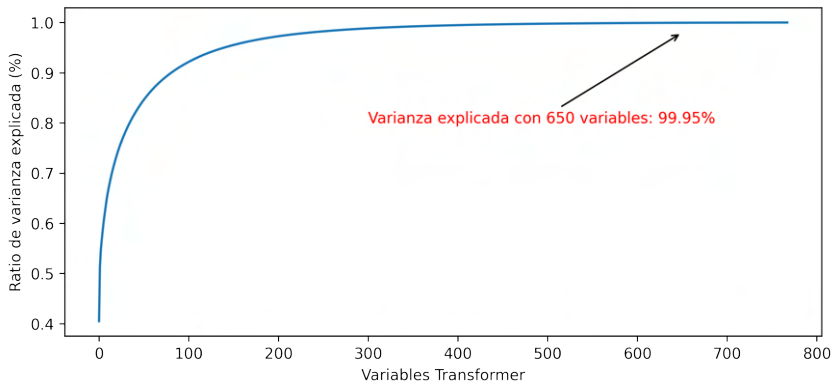


Figura 12: Algoritmo PCA: Ratio de varianza explicada según número de variables

(a) Variables del Word2Vec



(b) Variables del Transformer



Los resultados de aplicar PCA permiten sacar una serie de conclusiones. La figuras 12a y 12b muestran el ratio de varianza explicada (donde 1 representa el 100%) al incluir cada una de las variables sintéticas generadas por PCA, mostradas en el eje X.

Por un lado, la figura 12a muestra el embedding de 500 variables generado por Word2Vec puede ser resumido en 100 variables sin sacrificar más de un 0.03% de varianza explicada. Como se detalla en la sección 3.2.2 y en el listado de código 5, el número de dimensiones del Word2Vec se han elegido a mano, y es posible que el modelo no haya necesitado las 500 variables que se han forzado para codificar toda la información presente en el set de datos. Para los embeddings del Word2Vec, **es conveniente aplicar PCA** y reducir la dimensionalidad a 100, 1/5 del tamaño original del embedding.

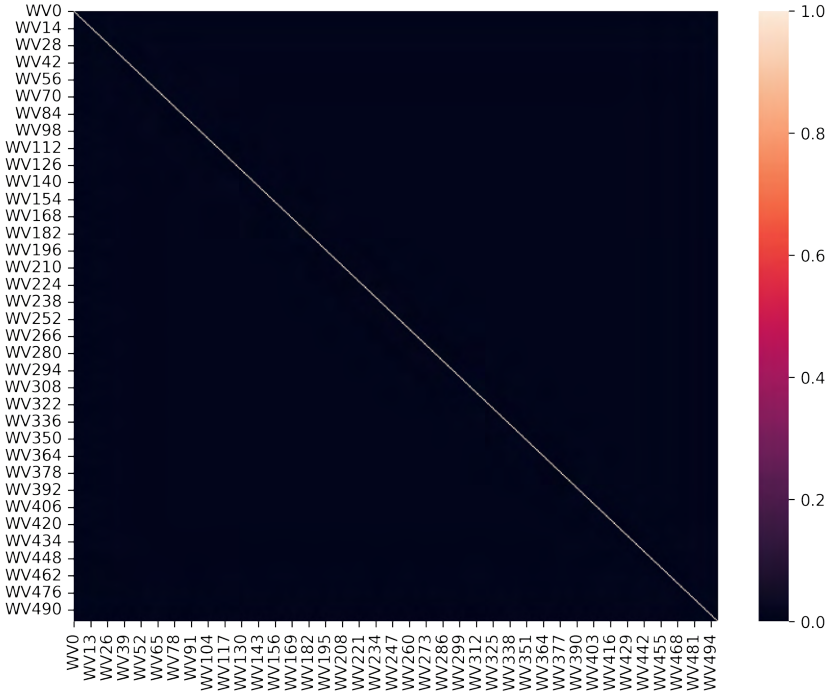
La figura 12b cuenta una historia distinta. El transformer hace uso completo de sus variables para codificar la información. De hecho, utilizando 650 variables PCA (de las

768 variables originales) aún no se explica la misma varianza que se consigue extraer en Word2Vec con solo 100 variables. Esto es de esperar, ya que el modelo ha sido entrenado sobre un set de datos recopilado por el Centro Nacional de Supercomputación de Barcelona, compuesto de 2.172.491 documentos en español (Carrino et al. 2021). La cantidad de datos es tan elevada que han sido necesarias todas las dimensiones del embedding para guardarla correctamente.

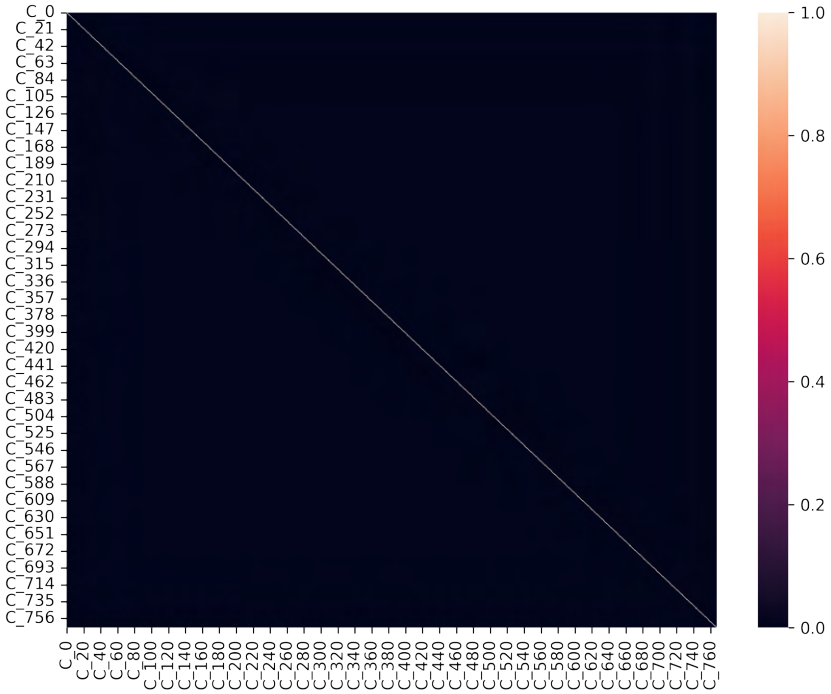
A pesar de que sea necesario mantener todas las dimensiones, es prudente aplicar PCA sobre el conjunto completo, con el fin de eliminar la correlación entre variables. La figura 13 muestra el mapa de calor correspondiente a las correlaciones de los embeddings del set de datos una vez se ha aplicado PCA. Se comprueba gráficamente que se ha eliminado todo tipo de correlaciones entre las variables iniciales.

Figura 13: Mapas de calor de correlación entre variables tras aplicar PCA

(a) Variables del Word2Vec



(b) Variables del Transformer



3.3.2 K-Means

Una ves se ha decidido el número de variables a las que se busca reducir la dimensionalidad, es el momento de aplicar el algoritmo de K-Means. Para este estudio se ha utilizado la implementación de K-Means desarrollada en la librería `scikit-learn`. La librería ofrece funcionalidad para desarrollar el modelo, el preprocesado de los datos por medio de escalado y PCA, y permite extraer métricas para evaluar la efectividad del algoritmo entrenado.

3.3.2.1 Definición del Pipeline

La librería `scikit-learn` permite generar objetos `Pipeline` o tubería que agrupan una serie de pasos por los que los datos deben pasar, simplificando la creación y entrenamiento del algoritmo de clustering.

En el aprendizaje automático, el procesamiento de datos se puede dividir en varias etapas: limpieza de datos, transformación de datos y aprendizaje o análisis de datos. Cada una de estas etapas requiere diferentes técnicas y herramientas. Un *pipeline* de aprendizaje automático es una secuencia de etapas de procesamiento de datos en el que cada etapa transforma los datos de entrada en datos de salida para que la siguiente etapa pueda utilizarlos. Un beneficio de usar un *pipeline* de aprendizaje automático es que permite a los desarrolladores de software reutilizar código y reducir el número de errores. Los *pipelines* de aprendizaje automático también hacen que el procesamiento de datos sea más reproducible.

Los pasos que serán aplicados en el *pipeline* son:

- Escalado de los datos con media cero y desviación típica $\sigma = 1$ por medio del objeto `StandardScaler` implementado en la librería.
- Algoritmo PCA con el fin de eliminar correlaciones y reducir dimensionalidad donde sea necesario, en base a las conclusiones extraídas en la sección 3.3.1.
- Algoritmo de clustering de K-Means, con un número definido de clusters. Este número de clusters debe ser seleccionado como parte del estudio. Esta selección es realizada en la sección 3.3.3.

El listado de código 8 muestra una simplificación de la definición de un pipeline ajustado a las necesidades del estudio. Como se ha mencionado anteriormente, la selección del número de dimensiones y del número de clusters debe ser seleccionado con anterioridad. Se debe generar un Pipeline individual para cada uno de los dos sets de datos: los embeddings generado por Word2Vec y por RoBERTa.

Listado de cógido 8: Implementación de un Pipeline de aprendizaje (simplificado)

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster       import KMeans
from sklearn.pipeline      import Pipeline

DIMENSIONES_REDUCIDAS = 100
NUMERO_DE_CLUSTERS = 8

# Lista de pasos aplicados en el pipeline.
# Cada paso se define como una tupla: (nombre, algoritmo)
pasos_del_pipeline = [
    ('escalado',  StandardScaler()),
    ('pca',       PCA(n_components  = DIMENSIONES_REDUCIDAS)),
    ('kmeans',    KMeans(n_clusters = NUMERO_DE_CLUSTERS)),
]

# Pipeline de datos
pipeline_modelo = Pipeline(steps=pasos_del_pipeline)

# Cómo aplicar el pipeline
# Entrenamiento
pipeline_modelo.fit(dataset)

# Predicción de clusters
clusters_resultantes = pipeline_modelo.predict(dataset)
```

Una vez se tiene definido el Pipeline, se debe ajustar el número óptimo de clusters para cada uno de los sets de datos.

3.3.3 Número óptimo de clusters

Se debe determinar una forma de encontrar el valor óptimo de clusters en los cuales delimitar las historias clínicas. Para ello existen dos métodos: el *Elbow method* y el *Silhouette method*.

3.3.3.1 Método Silhouette

Para elegir el número óptimo de clusters, el algoritmo se ejecuta en muchos valores diferentes de clusters y se calcula el *Silhouette Score*. El *silhouette score* es una medida de cómo de cerca está cada punto en un cluster con respecto a los grupos vecinos y, por lo tanto, cómo de bien agrupados están los datos. La puntuación se calcula usando la distancia media dentro del grupo (a) y la distancia media del grupo más cercano al suyo (b) para cada muestra:

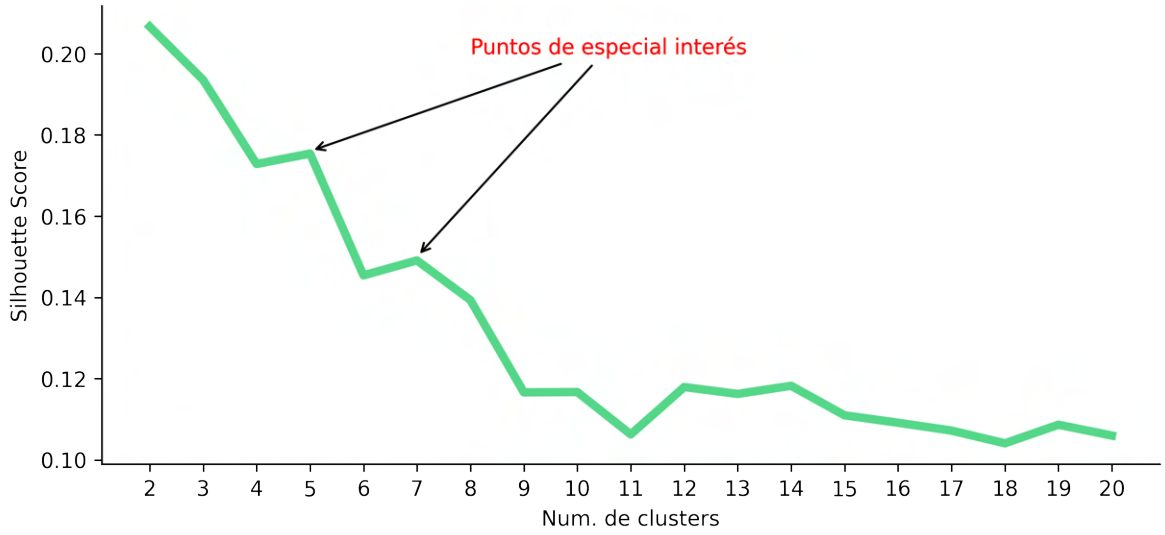
$$\text{Silhouette Score} = \frac{(b-a)}{max(a,b)}$$

El *silhouette score* es una medida de cómo de bien definido está cada grupo. Un *silhouette score* alto significa que el grupo está bien definido (es decir, hay una pequeña distancia entre los puntos del grupo y una gran distancia entre los puntos de los grupos vecinos). Un *silhouette score* bajo significa que el cluster no está bien definido (es decir, hay una gran distancia entre los puntos del propio cluster y no hay separación con respecto a los puntos de los clusters vecinos).

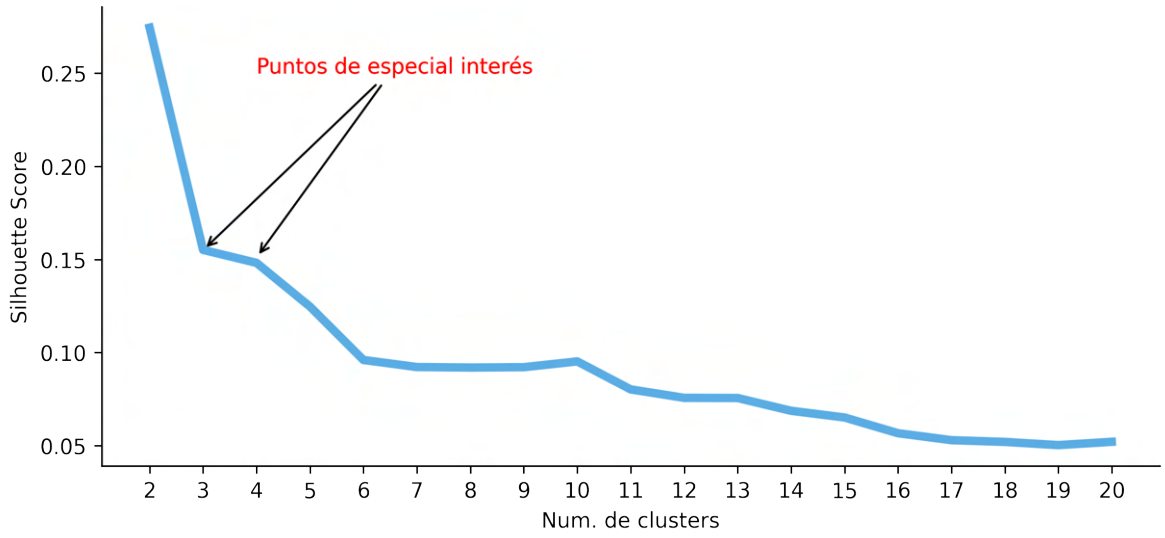
Iterando sobre el conjunto de datos con diferentes valores del número de clusters, en la figura 14 se observan a los siguientes resultados relativos al *silhouette score* (eje Y) para cada número de clusters (eje X):

Figura 14: Silhouette scores para cada número de clusters

(a) Aplicado a set de datos Word2Vec



(b) Aplicado a set de datos Transformer



3.3.3.2 Método Elbow

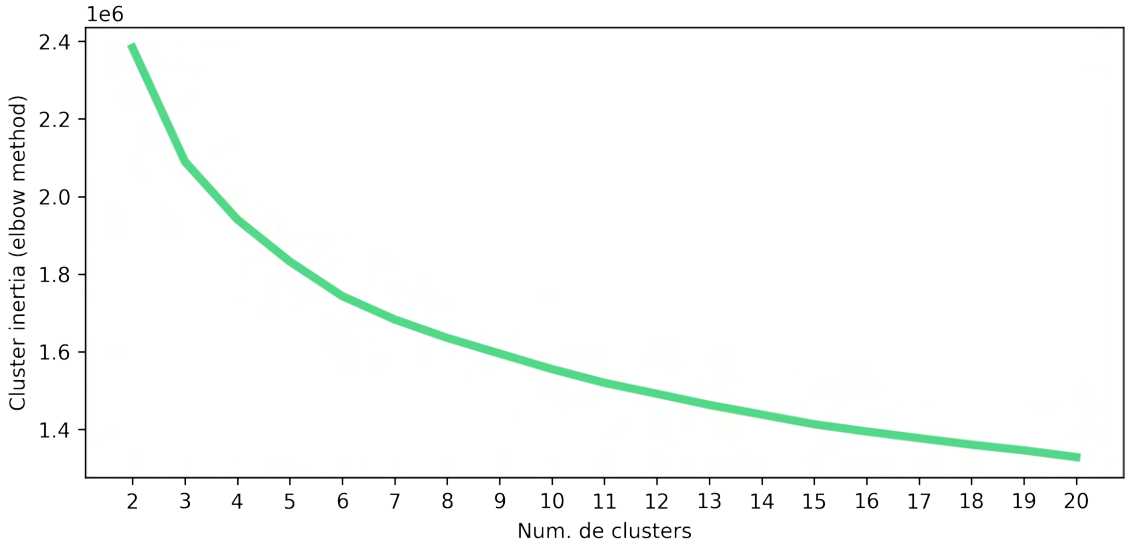
Otro método para encontrar el número óptimo de clusters es a través del método *Elbow*. El método *Elbow* analiza la suma de cuadrados dentro del cluster (*Within Cluster Sum-of-Squares* - WCSS o *inertia*) y encuentra el punto en el que el WCSS comienza a disminuir a un ritmo más lento. Este punto es el codo de la curva WCSS y es el número óptimo de clusters.

La curva de WCSS mide la varianza entre los clusters. A medida que se aumenta el número de clusters, la suma de cuadrados entre ellos siempre disminuye, ya que los datos se están agrupando más estrechamente. Sin embargo, si se aumenta el número de clusters más allá de cierto punto, la WCSS comenzará a disminuir a un ritmo más lento, ya que los datos comenzarán a agruparse en clústeres más pequeños y más específicos. El "codo" en la curva de WCSS es el punto en el que la suma de cuadrados comienza a frenar su caida, lo que indica que ese es el número óptimo de clusters.

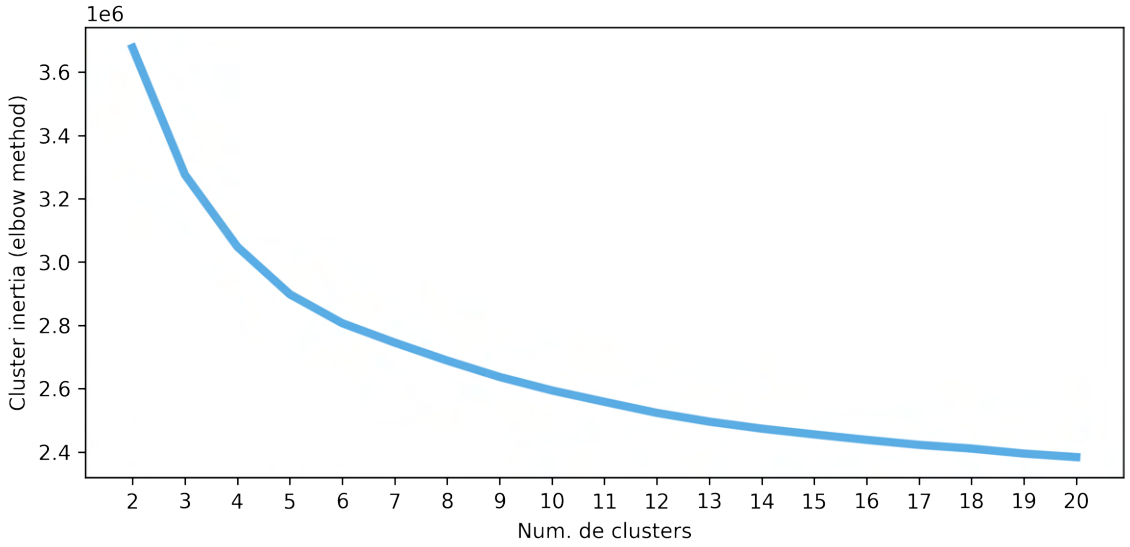
Iterando nuevamente sobre el conjunto de datos con diferentes valores de la cantidad de clusters, sobre la figura 15 se muestran los siguientes resultados de WCSS:

Figura 15: Medida de "WCSS" para cada número de clusters (Elbow method)

(a) Aplicado a set de datos Word2Vec



(b) Aplicado a set de datos Transformer



3.3.3.3 Selección de valores óptimos

Tras valorar los dos métodos de selección del número óptimo de clusters en conjunción con los demás datos observados durante el estudio, se debe llegar a una conclusión.

Echando un primer vistazo a la figura 15, las gráficas de WCSS no parecen mostrar un "codo" evidente en las curvas, sino que muestran un intervalo de números entre los cuales se puede encontrar el valor óptimo.

La teoría del *silhouette score* indica que se debe seleccionar un número de clusters con un alto silhouette score, pero no se debe dejar que esta métrica nuble nuestra intuición. Comenzando con la figura 14a relativa a los datos del Word2Vec, claramente el máximo se encuentra creando únicamente 2 clusters, pero observando la distribución de los datos representada en la figura 10a, se identificaban al menos 7 grupos delimitados de historias clínicas. Por ello, **para el caso de Word2Vec, se seleccionarán 7 clusters en total**. Si a la hora de visualizar los textos de cada cluster se determina que ciertos grupos carecen de distinción entre ellos, se reducirá el número de clusters a 5.

En el caso de los datos generados en el transformer, la selección de un valor óptimo es menos evidente que en el caso del Word2Vec. La figura 14b relativa a los *silhouette scores* muestra un máximo absoluto en 2, de manera similar a lo observado en el modelo previo. Teniendo una vez más en cuenta las densidades observadas en la figura 10b, se distinguen entre 3 y 4 grupos delimitados de datos. La figura 15b a su vez respalda la selección de 4 clusters, ya que a partir de 4 clusters se comienza a formar el "codo" de la curva, mientras que con 3 clusters se ve claramente que no se encuentra en ese "codo".

3.3.4 Resultados del clustering

Agrupando toda la información recopilada en las secciones anteriores, la tabla 3 muestra las prestaciones de cada uno de los modelos que serán entrenados:

Tabla 3: Configuración de los modelos de clustering

	Datos Word2Vec	**Datos Transformer**
Algoritmo	K-Means	K-Means
Nº de clusters	7	4
Dimensiones originales	500	768
Dimensiones reducidas	100	768
Tiempo de entrenamiento	409 ms ± 12.9 ms	1570 ms ± 48.9 ms

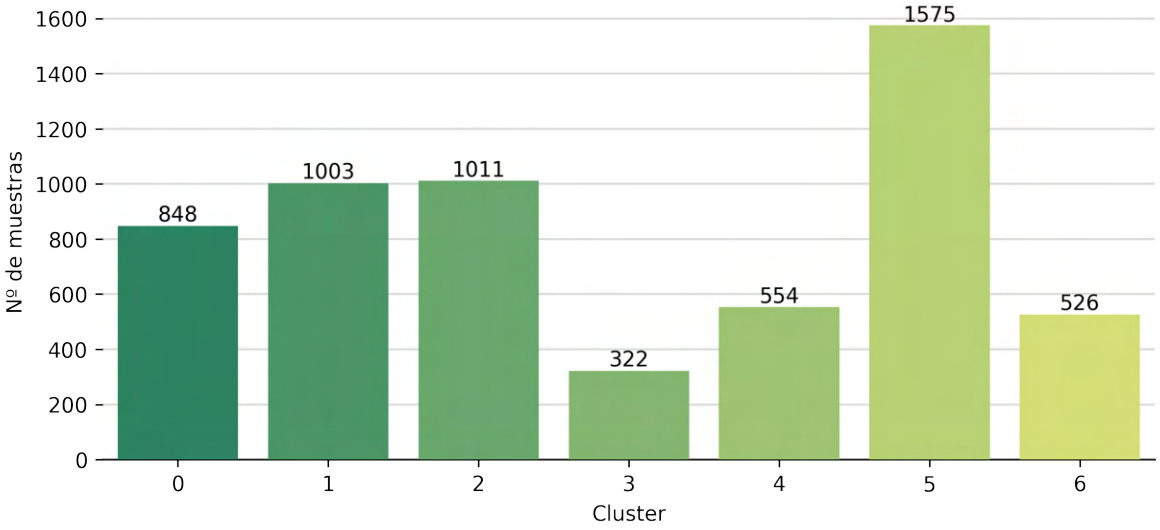
Es necesario estudiar también la forma en la que se agrupan los datos entre los clusters, es posible que uno de los clusters agrupe la gran mayoría de los datos, dejando los demás grupos con un número muy reducido. Si esto ocurre, los datos no se han conseguido separar claramente, por lo que es posible que la codificación del embedding no sea la correcta para este estudio.

La figura 16 muestra las distribuciones de los datos dentro de cada cluster, aplicado a los datos codificados con Word2Vec (16a) y a los datos codificados con el transformer (16b). En el caso del transformer, la gráfica 16b muestra que uno de los clusters contiene una proporción de datos mucho menor que los demás. Se deberá estudiar más adelante si es prudente eliminar este cluster o conservarlo porque separa datos completamente distintos a los demás grupos.

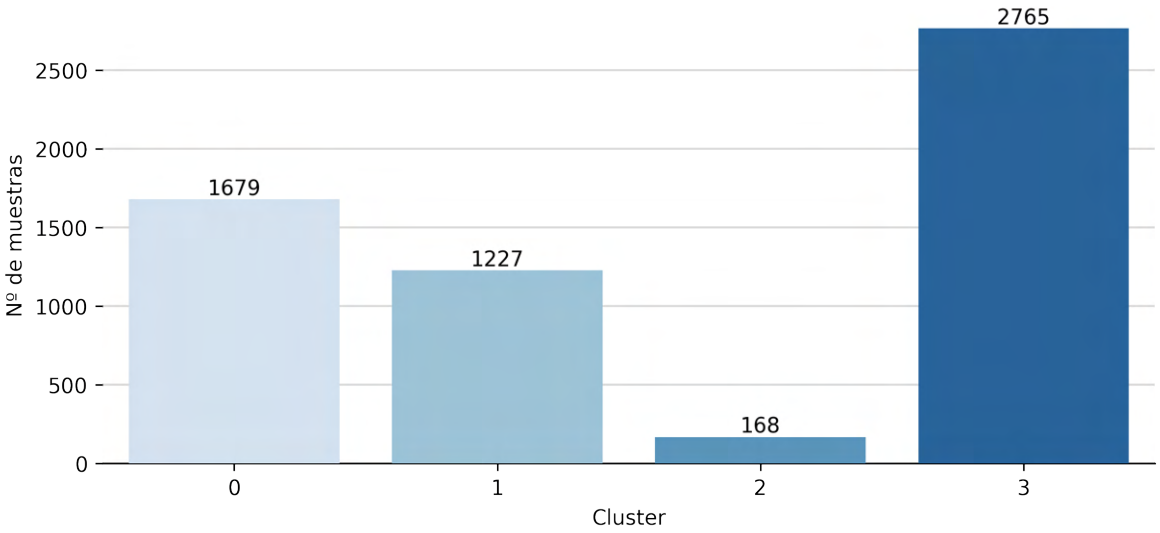
En cuanto al Word2Vec, la figura 16a muestra cuatro clusters con un número elevado de muestras. Se muestran también otros tres clusters con un menor número de muestras, pero no se destaca ninguno solo que contenga un número muy reducido de muestras. Se tendrán que evaluar los resultados para sacar conclusiones sobre el número adecuado de clusters.

Figura 16: Tamaño de los clusters tras aplicar K-Means

(a) Set de datos Word2Vec (7 clusters)



(b) Set de datos Transformer (4 clusters)



Utilizando la representación de datos con t-SNE generada en la sección 3.2.2.1, es posible pintar ahora los clusters individuales sobre esta proyección en dos dimensiones. Gracias a estas gráficas se puede visualizar fácilmente la densidad de los datos y las relaciones entre los clusters. También se pueden ver fácilmente la presencia de outliers.

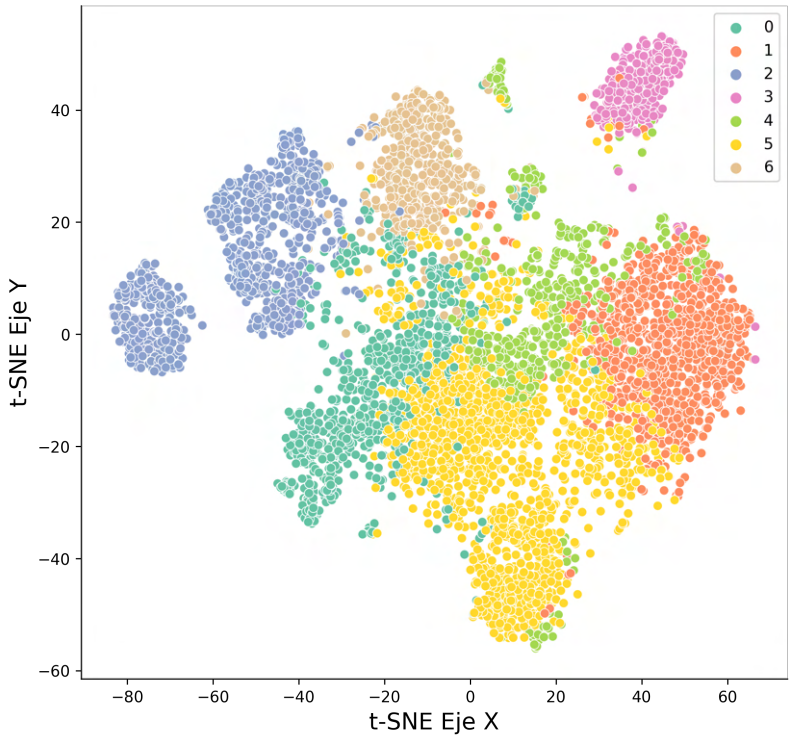
La figura 17 muestra los resultados del clustering sobre los datos codificados con word2vec. Por un lado, la figura 17a representa un scatterplot donde cada punto representa una historia clínica, y el color representa el cluster al que pertenece. La segunda figura (17b) muestra los mismos ejes que la figura anterior, pero representando las densidades los puntos de cada cluster mediante el uso de KDE (como se explicó en la sección 3.2.4 sobre visualización de los textos).

Las figuras 18a y 18b muestran las mismas representaciones que en el caso anterior pero relativas a la codificación del transformer. Se observa claramente que los datos del transformer son más difíciles de separar en clusters correctamente.

Si los clusters fueran perfectos, las representaciones KDE mostrarían cada uno con un solo máximo, y las fronteras entre los datos claramente definidas. Esto no es posible representarlo en dos dimensiones, ya que se trata de una proyección de datos de alta dimensionalidad. Las gráficas mantienen consistencia con los diagramas de barras 16a y 16b en los números asignados a cada uno de los clusters, permitiendo obtener una imagen completa de cada cluster.

Figura 17: Resultados gráficos del clustering sobre codificación Word2Vec

(a) Scatterplot sobre proyección t-SNE



(b) Distribuciones KDE de cada cluster

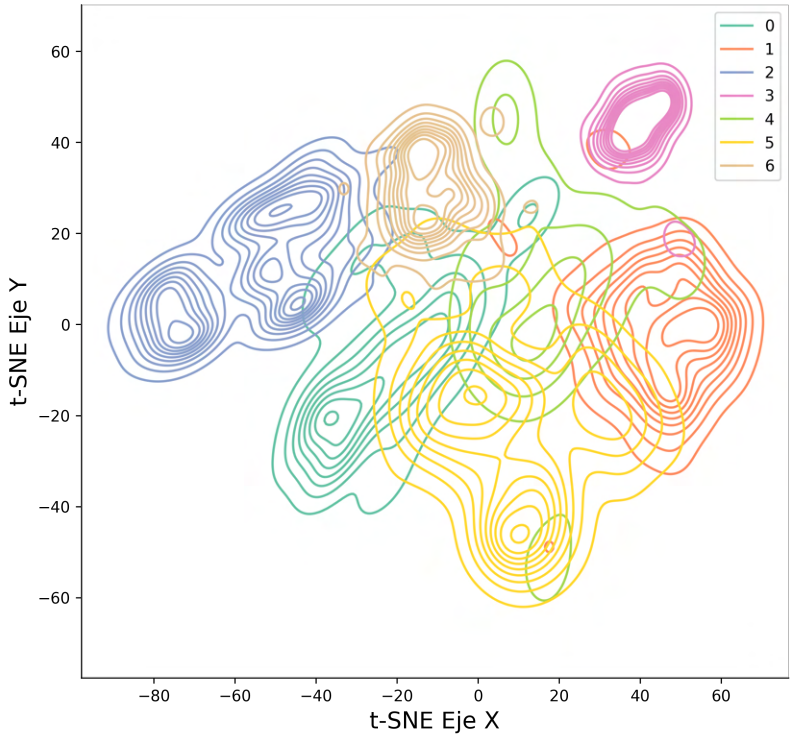
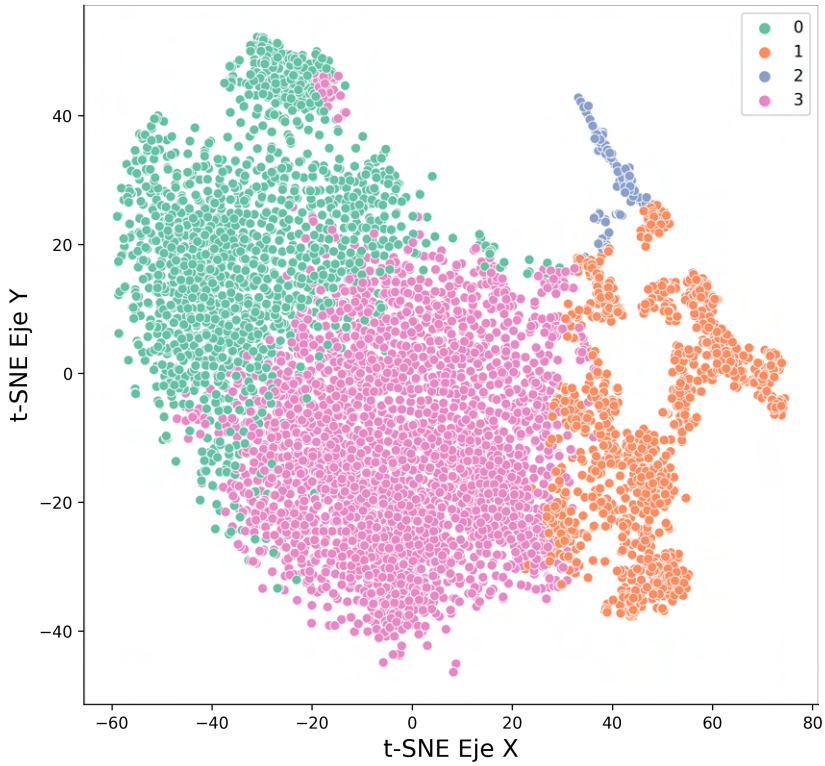
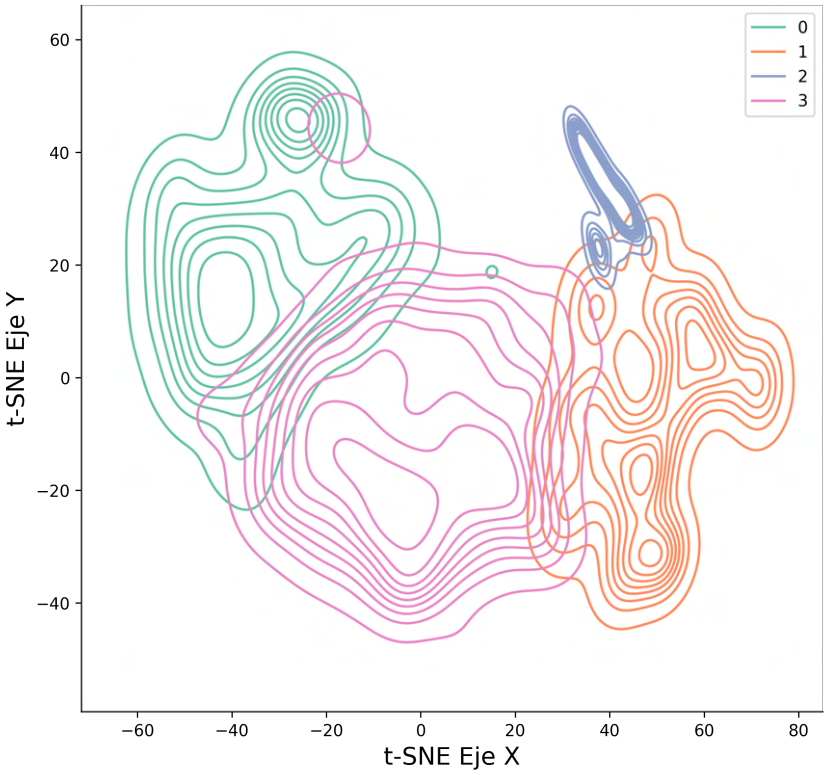


Figura 18: Resultados gráficos del clustering sobre codificación del Transformer

(a) Scatterplot sobre proyección t-SNE



(b) Distribuciones KDE de cada cluster



Las características individuales de los clusters, así como el contenido de cada uno será explorado en profundidad en la sección 4.1 de los resultados y conclusiones del estudio.

4 Resultados y Conclusiones

El desarrollo de este estudio no se encuentra completo sin realizar un análisis de los resultados obtenidos. Esta sección detalla las conclusiones extraídas a lo largo del estudio, y las líneas futuras de estudio.

4.1 Características de los clusters extraídos

Para que el clustering sea de utilidad para este estudio, se debe dar un nombre y un conjunto de rasgos característicos a cada grupo. Para lograr esto, se pueden obtener frecuencias de palabras de cada uno de los grupos para encontrar las palabras más características de cada grupo. Para presentar estos datos, la forma más eficiente es mediante el uso de una nube de palabras o *wordcloud*. Un *wordcloud* es una representación gráfica de las palabras más comunes en un texto dado. Cuanto más común es una palabra, más grande aparece en la nube de palabras.

4.1.1 Clusters de codificado Word2Vec

La figura 19 muestra los wordclouds de cada uno de los 7 clusters generados durante el análisis y desarrollo del clustering. La mayoría de ellos muestran de manera prominente las palabras "dolor" y "cabeza", como es de esperar, pero cada uno de ellos muestra particularidades que muestran indicios de que realmente se han separado las historias clínicas por características particulares de grupo, que son perceptibles al representarlos.

El cluster 0 mostrado en la figura 19a, compuesto de 848 textos, se trata de un grupo representando mayoritariamente a historias clínicas pediátricas, refiriéndose a los síntomas de jóvenes con dolor de cabeza y síntomas de migraña. Se distinguen las palabras "padre", "madre" e "hijo" (que agrupa las palabras hijo e hija por la normalización de palabras). Otras palabras destacadas de este grupo son "miedo" y "ansiedad". Jóvenes que sufren esta patología de manera recurrente suelen sufrir ansiedad y otros efectos adversos en su salud mental al no poder dar explicación clara a su dolor.

Otro cluster fácil de identificar es el cluster 2, mostrado en la figura 19a. Este cluster de 1011 textos muestra pacientes cuyos síntomas se encuentran relacionados con su dieta. Se muestran de manera prominente las palabras "tomar" y "comida". Además se distinguen

una serie de alimentos en la figura, entre ellos pan, café, leche, etc. Muchos pacientes y expertos reportan ciertos alimentos como detonantes de un episodio de migrañas, y es importante registrar los hábitos alimenticios de cada paciente para identificar estos alimentos que desencadenan migrañas con mayor frecuencia de lo habitual.

A primera vista los clusters 1, 3 y 4 (figuras 19b, 19d y 19e respectivamente) parecen todos estar relacionados con dolor muscular, pero se distingue uno de ellos como outlier: el cluster 3. Observando la distribución de los datos en la figura 17a se confirma esta sospecha. Los pacientes del cluster 3 fueron identificados inicialmente como pacientes con migrañas, pero su dolor de cabeza resultó ser causado por dolores musculares. Por esta razón, este grupo se encuentra claramente separado del resto de datos sobre la proyección t-SNE, ya que a parte del dolor de cabeza guardan poca relación con el resto de pacientes del estudio. Es importante seleccionar también a estos pacientes mal identificados, ya que en situaciones similares en el futuro, se podrá distinguir con mayor facilidad la patología real del paciente con herramientas de Inteligencia Artificial como esta.

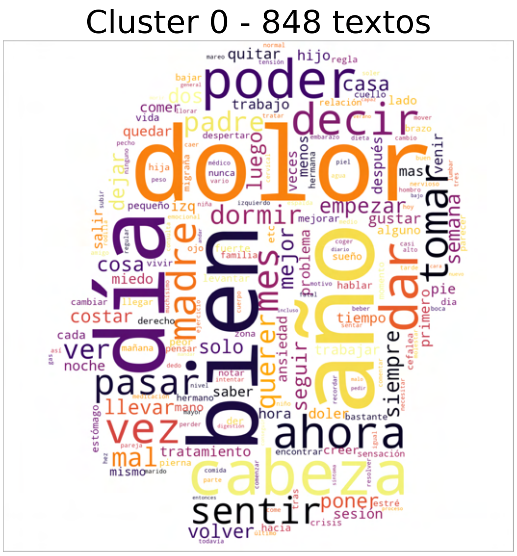
Volviendo a los clusters 1 y 4 (figuras 19b y 19e respectivamente), su diferencia radica en las zonas del cuerpo en las cuales sufren esos dolores musculares y ese dolor de cabeza. Analizando los textos, probablemente la mayoría de estos pacientes fueron al fisioterapeuta como parte de su tratamiento. Observando la distribución de estos dos clusters en la figura 17a se observa la cercanía de estos dos grupos, indicando que se trata de personas con síntomas similares, pero el cluster 3 muestra principalmente pacientes con dolor de cuello, hombros y resto de la parte superior del cuerpo, mientras que el cluster 4 se centra en dolor de espalda, lumbares, y pies. Aun así, no cabe duda de que existe un cierto solape entre estos dos grupos.

El cluster 6, mostrado en la figura 19g muestra claros indicios de representar pacientes que han acudido a tratamientos de medicina alternativa o tradicional, se muestran términos relativos a distintos tratamientos holísticos, y a partes del cuerpo de especial relevancia durante estos tratamientos. 526 pacientes se agrupan dentro de esta categoría.

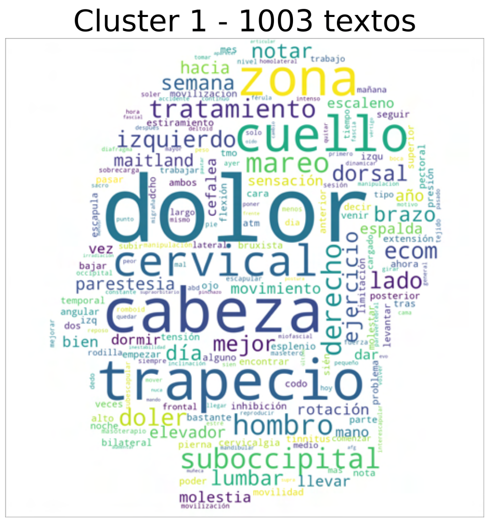
Finalmente, el cluster 5 es el grupo de mayor tamaño, con 1575 historias clínicas de pacientes distintos. Este grupo parece tener síntomas variados, y sufren migrañas principalmente durante el día, posiblemente debido en parte a la exposición solar. Sufren también dolores musculares, mareos, y dolor de cabeza al levantarse por la mañana.

Figura 19: Wordclouds de los clusters generados del codificado con Word2Vec

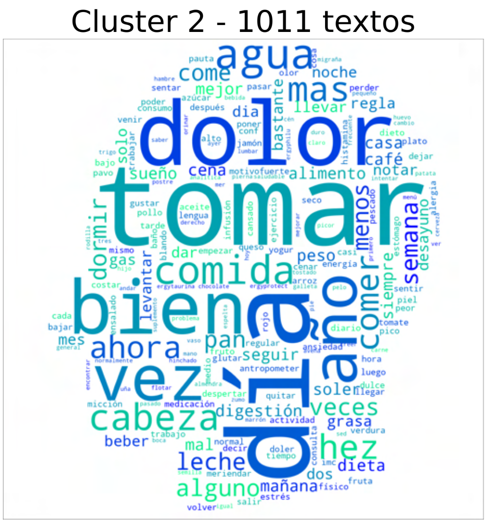
(a) Word2Vec Cluster 0



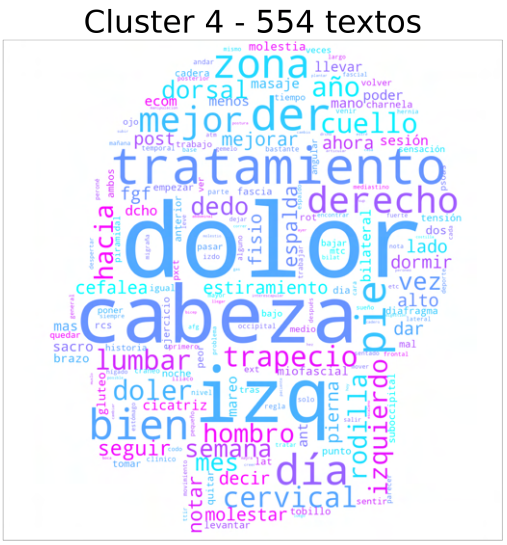
(b) Word2Vec Cluster 1



(c) Word2Vec Cluster 2



(d) Word2Vec Cluster 3

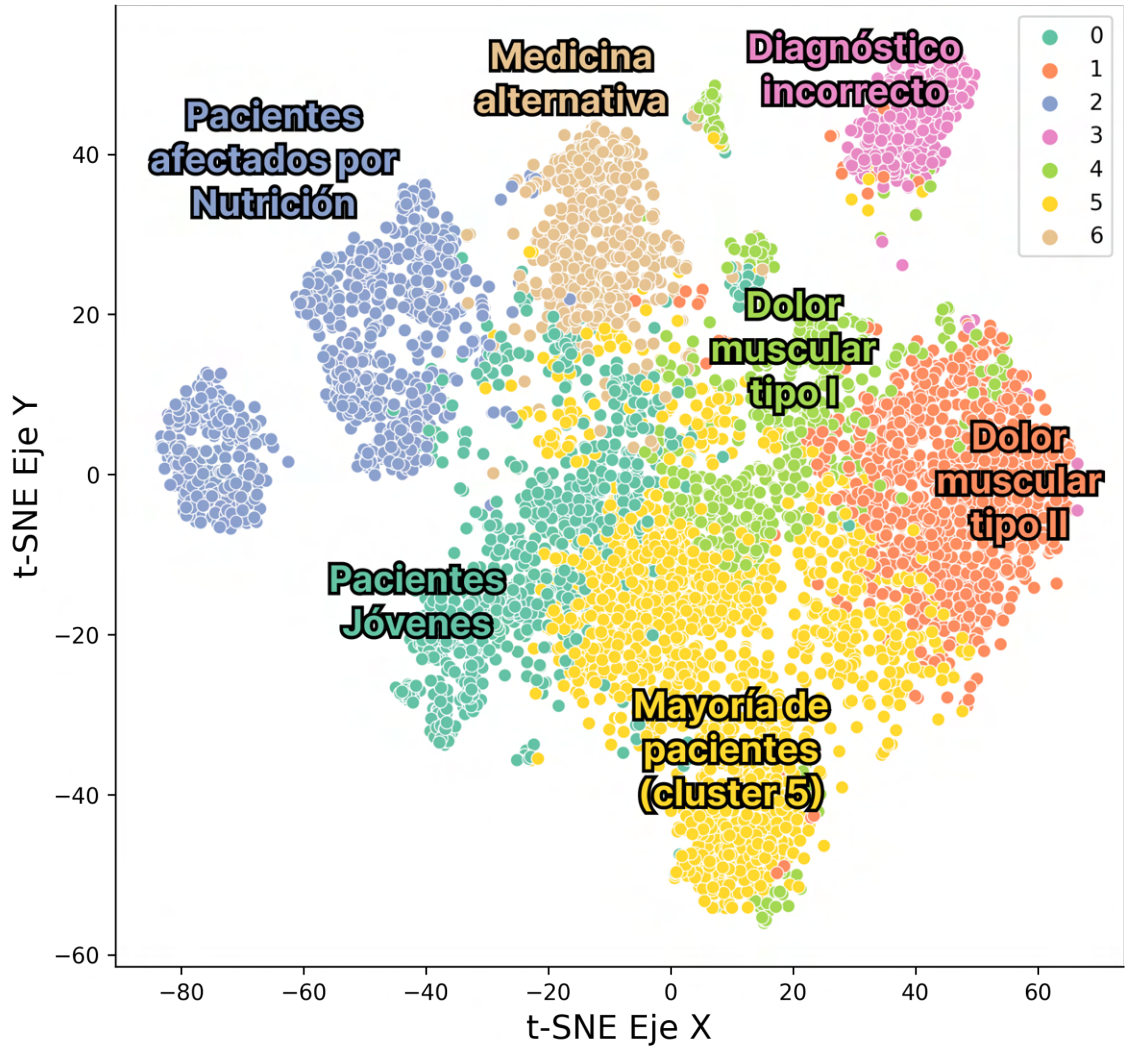


(e) Word2Vec Cluster 4

(f) Word2Vec Cluster 5

(g) Word2Vec Cluster 6

Figura 20: Word2Vec: Scatterplot t-SNE de los resultados del clustering - Etiquetado con perfiles



La figura 20 muestra en el contexto de un scatterplot las etiquetas correspondientes a las descripciones de cada cluster mencionadas anteriormente. Se comprueba gráficamente el solape entre los dos clusters relativos a dolores musculares, la separación clara del cluster designado como pacientes con un primer diagnóstico incorrecto, etc.

4.1.2 Clusters de codificado del Transformer

Los resultados de los 4 clusters generados a partir del embedding del transformer se muestran en la figura 21. Algunos de los clusters muestran una similitud con los que han sido observados en el caso anterior con Word2Vec.

Para empezar, el cluster 1, mostrado en la figura 21b muestra rasgos muy similares al grupo de historias clínicas 2 en el caso de Word2Vec, relativo a pacientes afectados por su nutrición. Se muestran en gran tamaño palabras como "dieta", "comida" y ciertos alimentos como la leche y el pan.

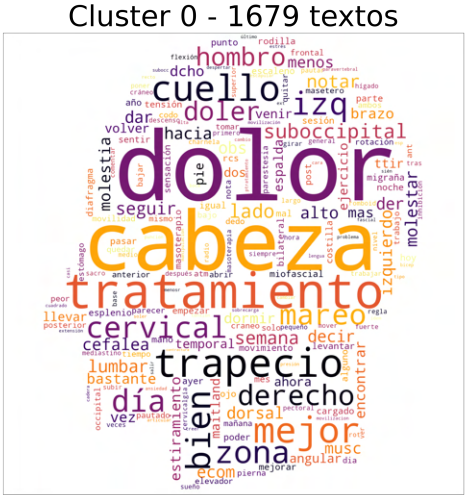
El cluster 2 (figura 21c) parece agrupar datos similares al cluster 6 en Word2Vec, relativo a pacientes que han buscado tratamientos de medicina alternativa para paliar los efectos de las migrañas. Se repiten con mucha frecuencia términos de especial interés en el campo de la medicina tradicional china y otras prácticas de medicina alternativa, incluyendo la acupuntura.

Los clusters 0 y 3 (figuras 21a y 21d respectivamente) agrupan la gran mayoría de los datos, como ya se mostraba en el diagrama de barras mostrado de la figura 16. El cluster 0 agrupa todo tipo de pacientes con dolores musculares, tanto los que han sido diagnosticados correctamente como los que no. No distingue tampoco entre distintos tipos de dolor muscular, al agrupar tal cantidad de textos.

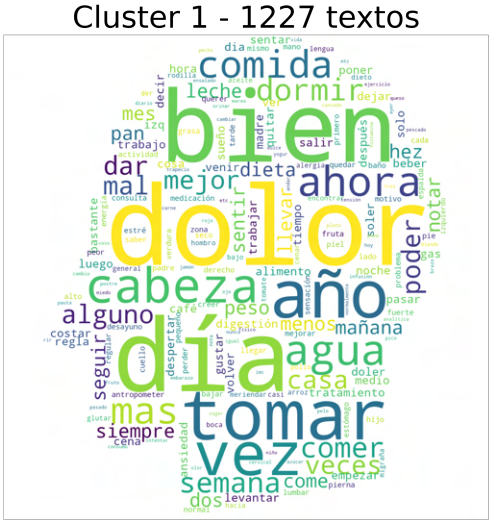
El cluster 3 a su vez, con el mayor número de textos (2765) agrupa todo tipo de pacientes con migrañas. El transformer no ha sido capaz de delimitar claramente características de historias clínicas del set de datos. Esto era esperar, ya que el modelo seleccionado de RoBERTa se ha pre-entrenado sobre un grupo mucho más amplio de textos que no incluyen exclusivamente historias clínicas relacionadas con las migrañas, y ha aprendido a generar embeddings mucho más amplios y genéricos.

Figura 21: Wordclouds de los clusters generados del codificado con Transformer

(a) Transformer Cluster 0



(b) Transformer Cluster 1



(c) Transformer Cluster 2

(d) Transformer Cluster 3

4.1.3 Comparación de técnicas

Para comparar la capacidad de ambos modelos en igualdad de condiciones, en la figura 22 se han plasmado los clusters de ambos modelos sobre la misma proyección t-SNE generada con los datos del Word2Vec (explicada en la sección 3.2.2.1) en la que se observan claramente los distintos grupos de pacientes. La figura siguiente (fig. 23) plasma los mismos datos, representados sobre dos dimensiones utilizando el algoritmo PCA en lugar de t-SNE.

Se observa claramente en ambas figuras la superioridad de la codificación generada por el Word2Vec para generar los clusters. Como se ha mencionado anteriormente, esto se debe a que el transformer ha sido pre-entrenado sobre un grupo de datos mucho más amplio. Futuras líneas de trabajo deberían estudiar el entrenamiento de un transformer específico a historias clínicas de pacientes con migrañas, si se llega a tener una cantidad de datos suficiente para ello. Este estudio se ha realizado sobre casi 6000 documentos, y el modelo de $RoBERTa_{BASE}$ fue entrenado por Facebook AI sobre más de 160 Gb de datos de texto, y el modelo ha sido afinado (*fine-tuned*) sobre otros 2.172.491 documentos médicos en español (Carrino et al. 2021), por lo que si se quiere generar un modelo eficaz se necesitan muchos más datos. Esto no debe desilusionar al lector, ya que el Word2Vec ha mostrado resultados muy satisfactorios sobre el set de datos actual, por lo que se saca como conclusión que cada modelo tiene sus puntos fuertes y débiles, y deben ser utilizados en situaciones distintas.

También se ha analizado el cruce de datos entre los clusters generados de un modelo y otro. Para ello, la forma más visual de representar el flujo de datos entre unos clusters y otros es un un diagrama *Sankey* o diagrama de flujo, mostrado en la figura 24.

Figura 22: Comparación de resultados de clustering proyectados sobre t-SNE

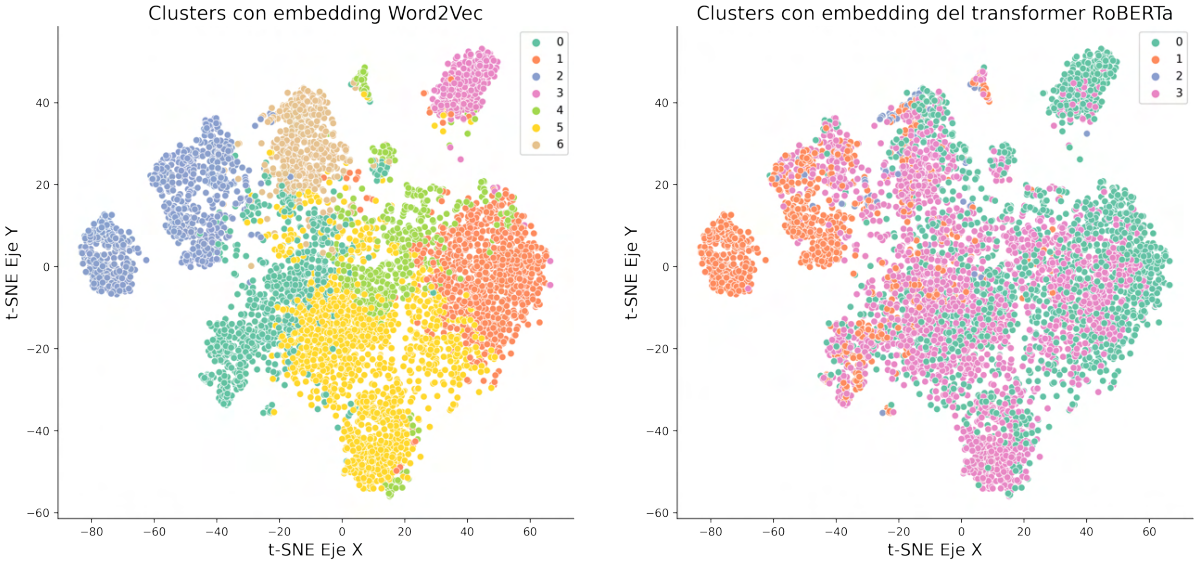


Figura 23: Comparación de resultados de clustering proyectados con PCA

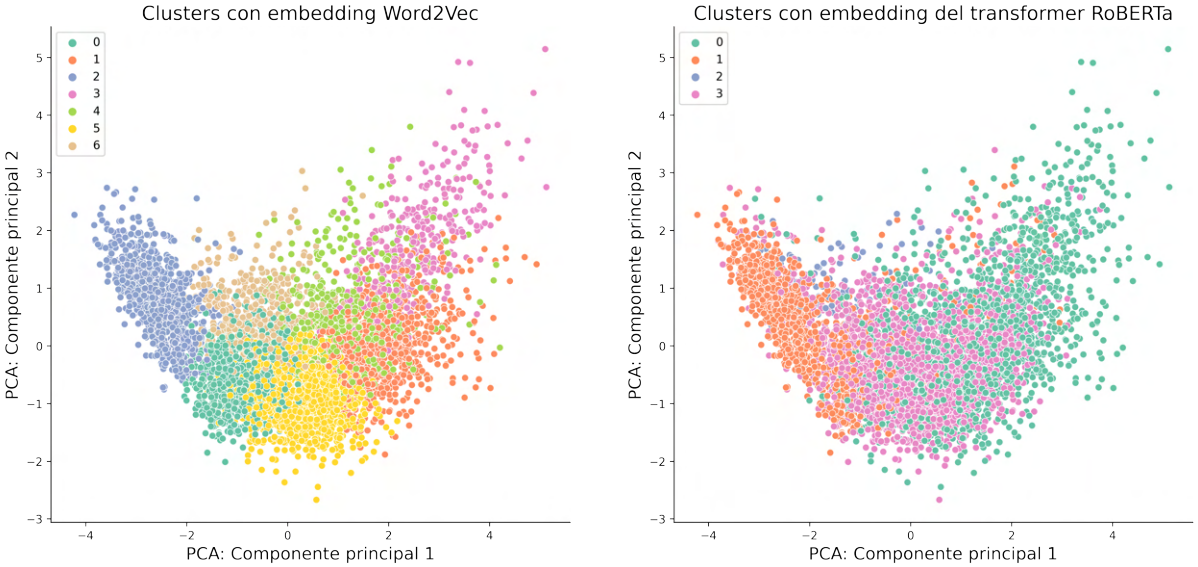
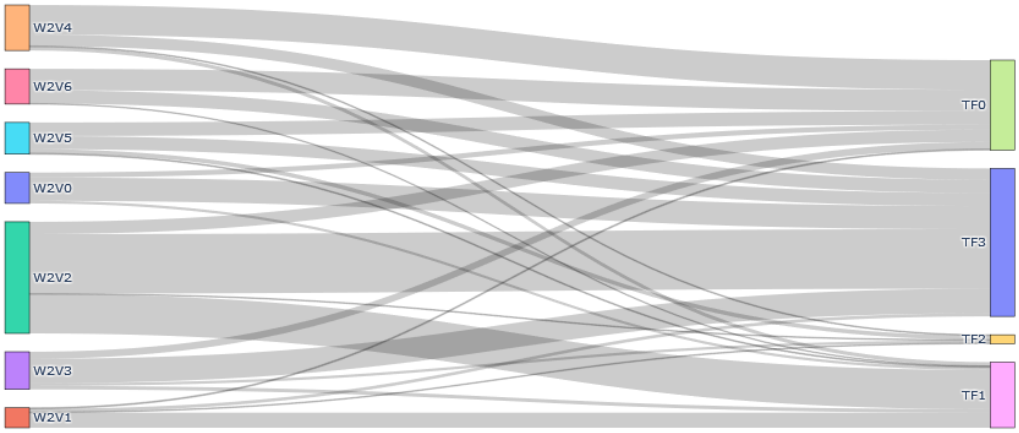


Figura 24: Diagrama de flujo de los clusters de Word2Vec (izquierda) y el transformer (derecha)



No cabe duda de que los resultados del Word2Vec son superiores. Los transformers, que formar parte del estado del arte en NLP, son muy poderosos, pero como cualquier otra red neuronal, requieren una cantidad de datos muy elevada para comenzar a mostrar resultados prometedores.

Los resultados arrojados por la codificación del Word2Vec así como los bajos tiempos de ejecución del entrenamiento del modelo lo convierten en el algoritmo ideal para este aplicativo, teniendo además en cuenta que la continuación de este estudio consiste en desarrollar una aplicación móvil capaz de albergar el modelo generado.

4.2 Futuras líneas de trabajo

Las futuras líneas de trabajo derivadas de este proyecto se pueden dividir en los siguientes apartados.

4.2.1 Aumento del set de datos

El conjunto de datos del estudio se puede expandir tanto vertical como horizontalmente dependiendo del alcance de los estudios futuros. La expansión vertical de los datos implicaría agregar datos similares para expandir las capacidades del modelo sin la necesidad de recurrir a técnicas de sobre-muestreo sintético.

Una expansión horizontal del conjunto de datos implicaría buscar datos con un alcance diferente en mente, este podría ser un nuevo alcance geográfico, incluyendo datos de pacientes en distintas localidades, o expandiendo el set de datos a otro tipo de patologías y estudiando los conjuntos resultantes de manera similar a como se ha realizado con pacientes de migrañas.

En cualquier caso de expansión del conjunto de datos, se deben tener en cuenta los tiempos de procesamiento. La implementación de un transformer requiere el uso de una GPU[8] para realizar tareas en tiempos razonables, y aun así los tiempos de procesamiento pueden aumentar exponencialmente al aumentar los datos. Para mejorar este tiempo de procesamiento, nuevos estudios deberían buscar optimizar este proceso, utilizar una máquina más potente o estudiar el uso de métodos de procesamiento distribuido.

4.2.2 Aplicación de diferentes algoritmos

Este estudio ha hecho uso extenso de diferentes algoritmos de aprendizaje automático, pero a la hora de realizar clustering solo se ha probado a utilizar K-Means. Existen una gran variedad de algoritmos de clustering alternativos a K-Means, que ofrecen prestaciones distintas y cuyos clusters se forman de manera distinta. Dentro de esta lista de algoritmos se incluyen. entre otros:

[8]Una GPU es una unidad de procesamiento especializada utilizada para mejorar el rendimiento de las aplicaciones que requieren cálculos matemáticos intensivos. Las GPUs se encuentran en las tarjetas gráficas de los ordenadores y en algunos procesadores de sobremesa y portátiles. Algunas GPUs se pueden usar para acelerar tareas no gráficas, como el cálculo científico y el procesamiento de imágenes y video.

- AGNES & DIANA (Clustering Jerárquico)
- Affinity propagation
- BIRCH (Árbol de Clustering)
- DBSCAN
- Fuzzy C-Means (Clustering borroso)
- K-Medoids
- PAM (Partitioning Around Medoids)
- SOM (Self-Organizing Maps)
- Spectral Clustering

Además de estudiar otros algoritmos de clustering, se podría probar a utilizar distintos modelos transformers. En este estudio se ha seleccionado el modelo "`PlanTL-GOB-ES/roberta-base-biomedical-clinical-es`", pero existen una gran variedad de distintos modelos que se entrenan a diario y se ajustan cada vez más a los casos de uso específicos de un estudio.

Si se capturara un conjunto de datos suficientemente grande, lo más efectivo sería entrenar un modelo transformer específico para la representación numérica de historias clínicas relativas a migrañas.

4.2.3 Continuación del estudio en el campo de las migrañas

Como se menciona al comienzo del estudio, este trabajo buscaba sentar las bases de los distintos grupos de personas que sufren migrañas. Los detonantes y agravantes de las migrañas no se comprenden del todo, y este estudio ha buscado delimitar distintos grupos de pacientes con síntomas y efectos similares dentro de la patología de la migraña.

Una vez se han sentado estas bases y se han encontrado una serie de grupos, se continuará el estudio integrando esta clasificación de pacientes en una aplicación, y registrando los síntomas de los pacientes, así como sus hábitos y los momentos en los cuales empiezan a sufrir dolores de cabeza (registrando también la intensidad del dolor). Gracias a este

estudio propuesto, se tendrá no solo una visión "post-hoc" de los eventos que sufren los pacientes, sino que se podrá comenzar a predecir posibles eventos "ex-ante", es decir, antes de que ocurran.

Bibliografía

Bhargava, Rupal, Yashvardhan Sharma, & Shubham Sharma (2016). "Sentiment Analysis for Mixed Script Indic Sentences". In: *2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016*. DOI: 10.1109/ICACCI.2016.7732099.

Bird, Steven, Ewan Klein, & Edward Loper (2010). "Natural Language Processing with Python, Analyzing Text with the Natural Language Toolkit". In: *Language Resources and Evaluation*. ISSN: 1574-020X. DOI: 10.1007/s10579-010-9124-x.

Briscoe, Ted et al. (1987). "A Formalism and Environment for the Development of a Large Grammar of English." In: *Proceedings of the 10th international joint conference on Artificial intelligence*.

Cañete, José et al. (2020). "Spanish Pre-Trained BERT Model and Evaluation Data". In: *PML4DC at ICLR 2020*.

Carrino, Casimiro Pio et al. (2021). "Biomedical and Clinical Language Models for Spanish: On the Benefits of Domain-Specific Pretraining in a Mid-Resource Scenario". In: *CoRR* abs/2109.03570. URL: https://arxiv.org/abs/2109.03570.

Chi, E. C. et al. (1985). "Database Of Computer-structured Narrative: Methods Of Computing Complex Relations." In: *Proceedings - Annual Symposium on Computer Applications in Medical Care*. ISSN: 01954210.

Covington, Michael A. & Hiyan Alshawi (1994). "The Core Language Engine". In: *Language*. ISSN: 00978507. DOI: 10.2307/416341.

Devlin, Jacob, Ming Wei Chang, et al. (2019). "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1.

Devlin, Jacob, Ming-Wei Chang, et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR*.

Goldberg, Lawrence D. (2005). "The cost of migraine and its treatment". In: *American Journal of Managed Care* 11 (SUPPL. 2). ISSN: 10880224.

Green Jr, B. F. (1961). "Baseball: an automatic question-answerer". In: *Papers presented at the western joint IRE-AIEE-ACM computer conference*.

Hendrix, Gary G. et al. (1978). "Developing a Natural Language Interface to Complex Data". In: *ACM Transactions on Database Systems*.

Hornik, Christoph P. et al. (2020). "Development of a Prospective Real-World Data Clinical Registry of Children and Adolescents With Migraine". In: *Headache* 60 (2). ISSN: 15264610. DOI: 10.1111/head.13714.

Khurana, Diksha et al. (2017). "Natural Language Processing: State of The Art, Current Trends and Challenges". In: *CoRR*.

Lee, Jinhyuk et al. (2020). "BioBERT: A pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36 (4). ISSN: 14602059. DOI: 10.1093/bioinformatics/btz682.

Lipton, Richard B. et al. (2013). "Examination of unmet treatment needs among persons with episodic migraine: Results of the American Migraine prevalence and prevention (AMPP) study". In: *Headache* 53 (8). ISSN: 00178748. DOI: 10.1111/head.12154.

Liu, Yinhan et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *ArXiv*.

Maas, Andrew L. et al. (2011). "Learning Word Vectors for Sentiment Analysis". In: *Association for Computational Linguistics*.

Mani, Inderjeet & Mark T. Maybury (1999). "Advances in Automatic Text Summarization". In: *Computational Linguistics*.

Manning, Christopher D. & Hinrich Schütze (1999). "Foundations of Statistical Natural Language Processing". In: *The MIT Press*.

Mccallum, Andrew & Kamal Nigam (2001). "A Comparison of Event Models for Naive Bayes Text Classification". In: *AAAI 1998*.

McCray, A. T. & S. J. Nelson (1995). "The representation of meaning in the UMLS". In: *Methods of Information in Medicine* 34 (1-2). ISSN: 00261270. DOI: 10.1055/s-0038-1634592.

McGray, Alexa T. et al. (1987). "Role Of Lexical Knowledge In Biomedical Text Understanding." In: *Proceedings - Annual Symposium on Computer Applications in Medical Care*. ISSN: 01954210.

McKinney & the Pandas Development Team (2022). *Pandas: powerful Python data analysis toolkit*. URL: https://pandas.pydata.org/docs/pandas.pdf.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of Workshop at ICLR*.

Peres, Mario Fernando Prieto et al. (2017). "Anxiety and depression symptoms and migraine: a symptom-based approach research". In: *Journal of Headache and Pain* 18 (1). ISSN: 11292377. DOI: 10.1186/s10194-017-0742-1.

Peters, Golden L. (2019). "Migraine overview and summary of current and emerging treatment options". In: *The American journal of managed care* 25 (2). ISSN: 19362692.

Řehůřek et al. (2022). *Gensim: Topic modelling for humans*. URL: https://radimrehurek.com/gensim/auto_examples/index.html#documentation.

Ritter, Alan et al. (2011). "Named Entity Recognition in Tweets: An Experimental Study". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Russell, Robert A. & W. J. Hutchins (1987). "Machine Translation: Past, Present, Future". In: *The Modern Language Journal*. ISSN: 00267902. DOI: 10.2307/328480.

Sager, N. et al. (1995). "Medical language processing: Applications to patient data representation and automatic encoding". In: *Methods of Information in Medicine* 34 (1-2). ISSN: 00261270. DOI: 10.1055/s-0038-1634579.

Sanh, Victor et al. (2019). "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108*.

Scikit-learn developers (2022). *scikit-learn: Machine Learning in python*. URL: https://scikit-learn.org/stable/modules/classes.html.

Sun, Chi et al. (2019). "How to Fine-Tune BERT for Text Classification?" In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 194–206. ISSN: 16113349. DOI: 10.1007/978-3-030-32381-3_16.

The NumPy community (2022). *Numpy Library Documentation*. URL: https://numpy.org/doc/stable/.

Van der Maaten, Laurens & Geoffrey Hinton (2008). "Viualizing data using t-SNE". In: *Journal of Machine Learning Research*. ISSN: 15324435.

Vicente-Herrero, Maria Teófila et al. (2020). "Tratamiento sintomático en migraña. Farmacos utilizados y variables relacionadas. Resultados de la encuesta europea Trabajo y Migraña". In: *Revista de la Sociedad Española del Dolor* 27. ISSN: 1134-8046. DOI: 10.20986/resed.2020.3744/2019.

Vicente-Herrero, María Teófila et al. (2014). "El coste de la incapacidad temporal por cefaleas en España". In: *Neurología Argentina* 6 (4). ISSN: 18530028. DOI: 10.1016/j.neuarg.2014.05.003.

Wells, Rebecca Erwin, Justin Beuthin, & Laura Granetzke (2019). "Complementary and Integrative Medicine for Episodic Migraine: an Update of Evidence from the Last 3 Years". In: *Current Pain and Headache Reports* 23 (2). ISSN: 15343081. DOI: 10.1007/s11916-019-0750-8.

Woods, W.A. (1978). “Semantics and Quantification in Natural Language Question Answering”. In: *Advances in Computers*.

Yi, J. et al. (2003). “Sentiment analyzer: extracting sentiments about a given topic using natural language processing techniques”. In: *Proceedings - IEEE International Conference on Data Mining, ICDM*. ISSN: 15504786. DOI: 10.1109/ICDM.2003.1250949.

Anexo

Objetivos de desarrollo sostenible

Los objetivos de desarrollo sostenible (ODS) son un conjunto de 17 objetivos globales que fueron establecidos por la Asamblea General de las Naciones Unidas en 2015. Fueron establecidos por la Asamblea General de las Naciones Unidas en 2015 y reemplazaron a los Objetivos de Desarrollo del Milenio, que vencían en ese año. Tienen como objetivo mejorar la calidad de vida de todas las personas alrededor del mundo, mientras se protege el medio ambiente. Los ODS abarcan una amplia gama de temas, como la pobreza, la educación, la igualdad de género, la salud, el agua limpia y la energía limpia.

Cada objetivo tiene un conjunto específico de metas que deben alcanzarse para 2030. Se espera que los ODS ayuden a guiar los esfuerzos de desarrollo en todo el mundo durante la próxima década y más allá, y que sirvan como un marco común para medir el progreso. Cada uno de nosotros debe poner de su parte para alcanzar estos objetivos como sociedad.

Los objetivos de desarrollo sostenible son esenciales, porque establecen un marco para que las naciones trabajen juntas para mejorar la calidad de vida de las personas en todo el mundo, sin sacrificar el ambiente que nos rodea. Estos objetivos tienen el potencial de transformar nuestra economía y nuestra sociedad para hacerlas más sostenibles. También nos ayudarán a proteger el medio ambiente y asegurar que nuestros recursos naturales sean utilizados de manera sostenible y no se agoten en futuras generaciones.

El objetivo de este estudio es mejorar la calidad de vida de los pacientes con migrañas mediante la detección de sus síntomas con Inteligencia Artificial. Esto se ajusta al objetivo de desarrollo sostenible de **Salud y Bienestar**, y busca ayudar a estos pacientes a tener una mejor calidad de vida, clasificando sus síntomas y pudiendo tratarlos de manera efectiva una vez se hayan encontrado los detonantes y agravantes.

El objetivo general de Salud y Bienestar se subdivide en una serie de objetivos específicos, con metas tangibles en cuanto a fecha y métricas. Entre ellas se incluyen: Reducir las tasas de mortalidad materna y neonatal mundial a menos de 70 por cada 100 000, reducir la tasa de mortalidad neonatal a menos de 12 por cada 1000, poner fin a las epidemias de SIDA, tuberculosis, malaria y enfermedades tropicales desatendidas y combatir la hepatitis, las enfermedades transmitidas por el agua y otras enfermedades transmisibles y asegurar el acceso público y universal a servicios de salud en todo el mundo, entre otros.

La calidad de vida de los pacientes con migrañas puede mejorar significativamente si se les proporciona un tratamiento eficaz para sus migrañas, ya que se trata de una patología crónica sin aparente cura a fecha de redacción de este estudio. Reducir los síntomas de manera eficaz puede ayudar a mejorar la vida de millones de personas en todo el planeta.

Este estudio también avala por un acceso universal y de bajo coste a servicios de sanidad. Mediante el desarrollo de herramientas de Inteligencia Artificial de uso abierto se ha podido llevar a cabo este estudio sobre el trabajo de muchas otras personas en todo el planeta, y juntos somos capaces de crear soluciones accesibles a cualquier persona con un dispositivo móvil.

El desarrollo de este estudio toca también otro amplio rango de objetivos de desarrollo sostenible, incluyendo el objetivo 9 en materia de innovación y uso de Inteligencia Artificial. El desarrollo de herramientas de software e Inteligencia Artificial sostenible es crucial para cumplir este objetivo, porque pueden ayudar a automatizar tareas y procesos, haciéndolos más eficientes y reduciendo la necesidad de mano de obra para labores repetitivas y de mala calidad para el trabajador. Además, la IA puede ayudar a identificar patrones y tendencias que se pueden utilizar para mejorar la infraestructura y los procesos de distintas industrias, incluyendo en el campo de la medicina. Finalmente, la IA puede ayudar a desarrollar productos y servicios nuevos e innovadores, que pueden ayudar a promover el crecimiento económico y el desarrollo.

Recursos adicionales

Tabla 4: Librerías Python 3.8 usadas durante el desarrollo

Library	**Version**	**Homepage**
fastparquet	0.8.1	fastparquet.readthedocs.io
gensim	4.1.2	radimrehurek.com/gensim
googletrans	4.0.0rc1	github.com/ssut/py-googletrans
h5py	3.6.0	h5py.org
joblib	1.1.0	joblib.readthedocs.io
nltk	3.7	nltk.org
numpy	1.22.3	numpy.org
pandas	1.4.1	pandas.org
pypdf2	1.26.0	pypdf2.readthedocs.io
scikit-learn	1.0.2	scikit-learn.org
spacy	3.2.3	spacy.io
tensorflow	2.8.0	tensorflow.org
torch	1.11.0	pytorch.org
transformers	4.17.0	github.com/huggingface/transformers