Tesina de Máster Universitario en Inteligencia Artificial Avanzada

MEDAID: Propuesta de modelo de clasificación híbrido ligero para MLP.

MEDAID: Proposal for efficient hybrid classification model for MLP.

Autor

José Alberto García Gutiérrez

Director académico

Dr. Jesus G. Boticario

Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia
Madrid, España

Convocatoria de junio 2020

This page was intentionally left blank

Agradecimientos

A mis padres y mi hermana por animarme siempre a reinventarme e ir un poco más allá. Así como al profesor Jesús González Boticario, mi tutor académico por confiar en mis desvaríos iniciales y guiar este trabajo, por todas las correcciones y aclaraciones que me ha hecho a lo largo de estos meses, que me han permitido madurar como profesional en este fascinante campo, a la vez que lo hacía este trabajo de investigación.

There are chords in the human heart – strange, varying strings – which are only struck by accident, which will remain mute and senseless to appeals the most passionate and earnest and respond at last to the slightest casual touch.

Charles Dickens, "The old curiosity shop", 1847.

This page was intentionally left blank

Abstract

In the modern medical ecosystem, the processing and indexing of data from different unstructured sources may be an essential task in some applications. For example, this can be applied to pre-classifying or discriminating the clinical documentation of a patient according to their care needs or clinical course. This can have important social and economic benefits and have a decisive influence on the quality of patient care. Traditional NLP methods are based on mapping handmade features or using word embedding methods over generic corpus requiring lots of time and lead to over-specified and incomplete features. There have been successful attempts to use deep learning in natural language processing in recent literature, however, the correct interpretation of a clinical document presents several difficulties: use of very specific terminology and complex technical and procedural glossaries, high contextual dependence (where the meaning of some parts of the document depends entirely on others), the spatial and temporal dispersion of the information and other less important issues such as the use of abbreviations, specialized units of measurement, or the inclusion of fragments written in colloquial language. The semantic models based on computing continuous distributed representations of words are too generalist to solve this problem.

To handle these drawbacks, two deep network architectures were elaborated: In one first approximation we work in an recommender system to supporting emergency physician to determine medical priority for patients by collecting available unstructured information and providing recommendations based on the application of CNN and word embedding methods. This approach turned out to be too ambitious, obtaining success rates of 60%, results in the average of other studies, but it allowed us to develop a prototype that served as a first approximation and detect the main problems. Motivated by the good initial results, we decided to implement our own architecture that we applied to the MSKCC dataset specialized in oncological documentation, using a hybrid approach where we incorporate a 3-stage model with a convolutional stage that focuses on basic semantics at the sentence level using a FastText word embedding semantic model trained on a specific corpus extracted from the different clinical repositories; and a recurrent Bi-LSTM-type network. The experimental results show that the proposed hybrid model (CNN + Bidirectional LSTM + Context Attention) has a classification accuracy between 5% and 10% higher than the methods collected so far in the state of the art reaching for some of the classes at rates recognition over 80%. This well-known data set was chosen for the many studies based on it, which provided us with a benchmark.

As an additional result, publishing a scientific paper will be included, gathering the most relevant results of the work for admission to a scientific journal or bulletin, with the purpose of releasing our results of the research to the international scientific community. Although, in order to test the already trained models and explore the different catalogs and data sets, a wrapper class has been developed to call the libraries implemented from a Python environment and a web demonstrator, which will be made public shortly.

Keywords: Natural Language Processing, Unsupervised Classification; Convolutional Neural Networks; Artificial Intelligence, Machine Learning, E-Health, Electronic Health Records; Named Entity Recognition.

Resumen

En el ecosistema médico moderno, el procesamiento y la indexación de datos de diferentes fuentes no estructuradas puede ser una tarea esencial en algunas aplicaciones como el triage de enfermos, la unificación de historias o la discriminación de la documentación clínica de un paciente de acuerdo con sus necesidades de atención o curso clínico. Esto puede tener beneficios sociales y económicos de importancia, e influir de forma determinante en la calidad de la atención al paciente. Los métodos tradicionales de PLN se basan en el mapeo de características construidas a mano o en el uso de métodos *word embedding* sobre corpus genéricos que requieren mucho tiempo y generalmente conducen a características sobreespecificadas e incompletas. Ha habido intentos exitosos de utilizar el aprendizaje profundo en el procesamiento del lenguaje natural en la literatura recuente, sin embargo, la interpretación correcta de un documento clínico presenta varias dificultades: uso de terminología muy específica y de glosarios técnicos y procedimentales complejos, gran dependencia contextual (donde el significado de unas partes del documento depende totalmente de otras), la dispersión espacial y temporal de la información y otras cuestiones menos importantes como el uso de abreviaturas, unidades de medida especializadas, dosificaciones o la inclusión de fragmentos escritos en lenguaje coloquial. Los modelos semánticos basados en el cálculo de representaciones distribuidas continuas de palabras simplemente son demasiado generalistas para ser efectivos.

Para abordar estos inconvenientes, en este trabajo se elaboraron dos arquitecturas de red profundas: inicialmente, se trabajó en un sistema abierto pensado para que permitiera a un médico de emergencias tratar a un nuevo paciente al recopilar la información no estructurada disponible y proporcionar recomendaciones basadas en la aplicación de un modelo predictivo. Este enfoque resultó ser demasiado ambicioso obteniendo tasas de acierto del 60%, resultados en el promedio de otros estudios, pero nos permitió desarrollar un prototipo que nos sirvió como primera aproximación y detectar los principales problemas. Motivados por los buenos resultados iniciales, decidimos implementar nuestra propia arquitectura que aplicamos sobre el dataset MSKCC especializado en documentación oncológica, utilizando un enfoque híbrido donde incorporamos un modelo de dos etapas con una etapa convolucional enfocada en la semántica básica a nivel local, usando un modelo semántico FastText entrenado sobre un corpus extraído de los diferentes repositorios clínicos; y una red recurrente de tipo Bi-LSTM, esta etapa implementará también un mecanismo de atención básico que nos permitirá focalizar la búsqueda seleccionando únicamente aquellas características de alto nivel más prometedoras. Este conocido conjunto de datos fue elegido por los numerosos estudios basados en él preexistentes, lo que nos proporcionó un punto de referencia y una base sólida para realizar comparaciones cruzadas.

Los resultados experimentales muestran que el modelo híbrido propuesto (CNN + *Bidirectional* LSTM + *Context Attention*) obtiene una precisión de clasificación entre 5% y 10% mejor que los métodos recopilados hasta ahora en el estado de la técnica llegando para algunas de las clases a tasas de reconocimiento superiores al 80%. Como resultado adicional se incluirá el envío de un artículo científico recogiendo los resultados más relevantes del trabajo para su admisión en una revista o boletín científico, con la finalidad de difundir a la comunidad internacional los resultados de nuestra investigación. Los estudios son elegidos por grupos de revisión especializados en el tema, mismos que determinarán su rigor científico para su publicación. Por último, para poder testar los modelos ya entrenados, así como explorar los diferentes catálogos y conjuntos de datos, se ha desarrollado un wrapper para llamar a las librerías implementadas desde un entorno Python, así como un demostrador web, los cuales se harán públicos en breve.

Palabras-Clave: Natural Language Processing, Unsupervised Classification; Convolutional Neural Networks; Artificial Intelligence, Machine Learning, E-Health, Electronic Health Records; Named Entity Recognition.

Tabla de contenido

Tabla de figuras

Lista de Abreviaturas

AI	Artificial Intelligence
CBoW	Continuous Bag of Words Model
CNN	Convolutional Neural Network
DL	Deep Learning
EHR	Electronic Health Records
ELMO	Embeddings from Language Models
FASTTEXT	Facebook Word2Vec Extension
GLOVE	Global Vectors for Word Representation
GRU	Gated Recurrent Unit
ICD	International Classification of Diseases of the World Health Organization
KNN	K-Nearest Neighbour
LSTM	Long-Short Term Memory Network
NIH	National Institute of Health of America
NIPS	Annual Conference on Advances in Neural Information Processing Systems
NLP	Natural Language Processing
ML	Machine Learning
MSKCC	Memorial Sloan Kettering Cancer Centre de New York
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
SVM	Support Vector Machine
t-SNE	T-distributed Stochastic Neighbour Embedding
UMLS	Unified Medical Language System
WE	Word Embedding Model

CAPÍTULO I

INTRODUCCIÓN

1. Introducción y objetivos

Las redes neuronales convolucionales (CNN) y las redes recurrentes de memoria a largo plazo (LSTM) se han ganado un lugar predominante en las tareas de procesamiento del lenguaje natural. El caso del lenguaje biomédico, sin embargo, ha sido especialmente difícil de abordar, por ser este ampliamente dependiente del contexto. Las CNN son simplemente, demasiado planas, e incapaces por sí solas de realizar una interpretación completa de estos textos, ya que estas ignoran en gran parte el entorno contextual de las palabras. Los textos clínicos sin estructura escritos en lenguaje natural son si embargo muy comunes, ya que la mayoría de los sistemas de gestión hospitalaria permiten la inclusión de campos de texto libre o han heredado esta clase de información de sistemas más antiguos. Este tipo de notas incluye información desestructurada y no anotada proveniente de: antecedentes médicos, descripción de patologías y síntomas, historial médico, información analítica, tratamiento farmacológico, o proveniente de pruebas diagnósticas o indicadores médicos. Por este motivo, el avance de este tipo de técnicas es imprescindible en el desarrollo de cualquier sistema que trabaje con datos de procedencia médica hoy en día. El procesamiento del lenguaje clínico adolece además de una serie de dificultades propias: el uso de terminología específica, de glosarios técnicos y vocabulario médico especializado, entre otras características, hacen fracasar modelos semánticos basados en incrustaciones de palabras, demasiado generalistas. Por otro lado, los mecanismos de atención, recientemente incorporados al estado del arte, permiten ganar profundidad contextual, pero son costosos cuando se aplican a grandes conjuntos de datos ya estos basan su funcionamiento en extraer meta-características que resultan en una buena capacidad predictiva lo que requiere de un procesamiento intensivo del conjunto de datos. En este capítulo abordaremos todos estos enfoques, sus ventajas y limitaciones, y la que creemos nueva ventana de oportunidad que se abre para los nuevos sistemas híbridos especializados en procesamiento del lenguaje biomédico.

1.1 Contexto y definición del problema

El Procesamiento del Lenguaje Médico (MLP) es un caso especial dentro del procesamiento del lenguaje natural especialmente difícil de abordar y que sigue abierto debido a que en este tipo de textos se mezcla un uso informal del lenguaje (con abundantes vulgarismos y expresiones coloquiales), con lenguaje altamente especializado; vocabulario dependiente de contexto; y entidades de significancia clínica o científica; junto con numerosos convencionalismos y abreviaturas. A esto, hay que sumar, que proceden en su mayoría, de fuentes total o parcialmente desestructuradas y, por tanto, con bajo o nulo nivel de anotado. Los métodos tradicionales de procesamiento del lenguaje natural se basan en mapeos de características hechos a mano que requieren mucho tiempo y conducen a características sobreespecificadas e incompletas. Por esta razón, estos métodos solo son aplicables a pequeños textos o conjuntos de textos con una temática muy definida donde podemos acotar mucho el tamaño de las matrices de términos. Sin embargo, la generación de automática de características es un aspecto intrínseco a las redes neuronales profundas y permiten que el modelo aprenda niveles crecientes de complejidad. Recientemente ha habido intentos exitosos de utilizar el aprendizaje profundo para el análisis de textos, principalmente mediante la utilización de redes neuronales convolucionales profundas

(Levy&Golberg, 2015; Lanzarini et al., 2017, Jiménez-Perera, 2017; Zang&Wallace, 2018, y otros) siendo este tipo de soluciones son ampliamente demandadas para uso sanitario. Actualmente, la implantación en los centros de atención sanitaria de la historia y registro digital del paciente ha aumentado drásticamente el número de registros clínicos existentes en formato digital. Y sin embargo, la aplicabilidad de las técnicas de procesamiento automático del lenguaje es todavía muy limitada. En gran parte, esto es debido a que la mayoría de los sistemas de gestión clínica permiten la inclusión de campos de texto libre. Este tipo de notas incluye información desestructurada y/o no anotada proveniente de un sinfín de situaciones clínicas: anotaciones de antecedentes médicos, descripción de patologías y síntomatología, información analítica, tratamiento farmacológico, o proveniente de pruebas diagnósticas o indicadores médicos, etc.

Los métodos tradicionales de procesamiento automático del lenguaje requieren mucha preparación ya que se basan en mapeos de características hechos por expertos a mano lo que hace que con frecuencia conduzcan a características sobreespecificadas e incompletas, por esta razón, estos métodos solo son aplicables a pequeños textos o conjuntos de textos con una temática muy definida donde se puede acotar muy bien el tamaño de las matrices de términos. En la bibliografía reciente ha habido intentos exitosos de utilizar el aprendizaje profundo para el análisis de pequeños textos médicos, principalmente mediante la utilización de redes neuronales o convolucionales profundas. Las CNN profundas tiene un punto clave a su favor y es que no se basan en una amplia ingeniería de características como sí que lo hacen la mayoría de los otros enfoques existentes. Las redes neuronales requieren menor conocimiento experto ya que automatizan la fase de generación de características y a la vez, permiten el apilado, lo que son capaces de aprender representaciones cade vez más generales, lo que hace que este enfoque sea robusto y flexible cuando se aplica a varios dominios. Por ello, muchos autores han adoptado con éxito métodos de aprendizaje profundo como un nuevo estándar de facto sobre análisis gramatical o clasificación de sentimientos (Bespalov et al., 2011; Zhou et al., 2013; Uysal et al., 2017; Chen et al., 2018; Zhao et al., 2018; y otros). Esta tendencia es provocada principalmente por el éxito de Word Embeddings (WE) y los modelos de aprendizaje profundo convolucional (CNN). A pesar de esto, no han obtenido los resultados esperados en aplicaciones de reconocimiento de lenguaje clínico sobre registros médicos reales, principalmente porque no son eficientes para capturar relaciones semánticas entre términos distantes, ni son buenos para distinguir relaciones temporales, donde el orden en que ocurren las entidades afectar sustancialmente el significado y el resultado de la clasificación. Las redes convolucionales recurrentes (RNN) pueden compensar parcialmente estas deficiencias al generar un apoyo de persistencia limitado para encontrar relaciones distantes dentro del alcance de varios párrafos o incluso de todo el documento. Sin embargo, no son eficientes en el análisis gramatical o la validación sintáctica. La combinación de estos dos tipos de redes parece surgir de manera natural.

1.2 Conjunto de datos

Como fuente de datos para los experimentos utilizaremos el dataset "*Personalized Medicine: Redefining Cancer Treatment*" liberado por el *Memorial Sloan Kettering Cancer Center* de New York[1] (MSKCC) con motivo de la *31ª Annual Conference on Advances in Neural Information Processing Systems*[2] de 2017. La elección de este conjunto de datos se ha realizado teniendo en cuenta su disponibilidad pública, su complejidad relativa y representativa del lenguaje biomédico, pero sobre todo debido a la disponibilidad de docenas de estudios basados en él que actúan como

[1] 1275 York Avenue New York NY 10065, USA.

[2] https://mitpress.mit.edu/books/advances-neural-information-processing-systems

un marco de referencia al diseñar experimentos y contrastar resultados. El conjunto de datos contiene un enorme conjunto de documentos oncológicos previamente anotados y clasificados por gen objetivo y mutación. La figura 1 nos permite hacernos una idea la complejidad del espacio de búsqueda, donde cada clase es el resultado de valorar la interrelación de decenas de términos. Una vez secuenciado, un tumor canceroso puede tener miles de mutaciones genéticas. Algunos de ellos ocurren naturalmente en todos nosotros sin afectar la función celular, mientras que otros ayudan al desarrollo del tumor y aumentan el riesgo de desarrollar el tumor, propagarlo a otras partes del cuerpo o limitar la respuesta al tratamiento. Actualmente, esta interpretación de las mutaciones genéticas se realiza de forma manual. Esta es una tarea que requiere mucho tiempo en la que un patólogo clínico debe revisar y clasificar manualmente cada una de las mutaciones genéticas con base en la evidencia de la literatura clínica de algunos artículos científicos. Para esta competencia, MSKCC pone a nuestra disposición una base de conocimiento con comentarios de expertos donde investigadores y oncólogos de clase mundial han anotado manualmente miles de mutaciones.

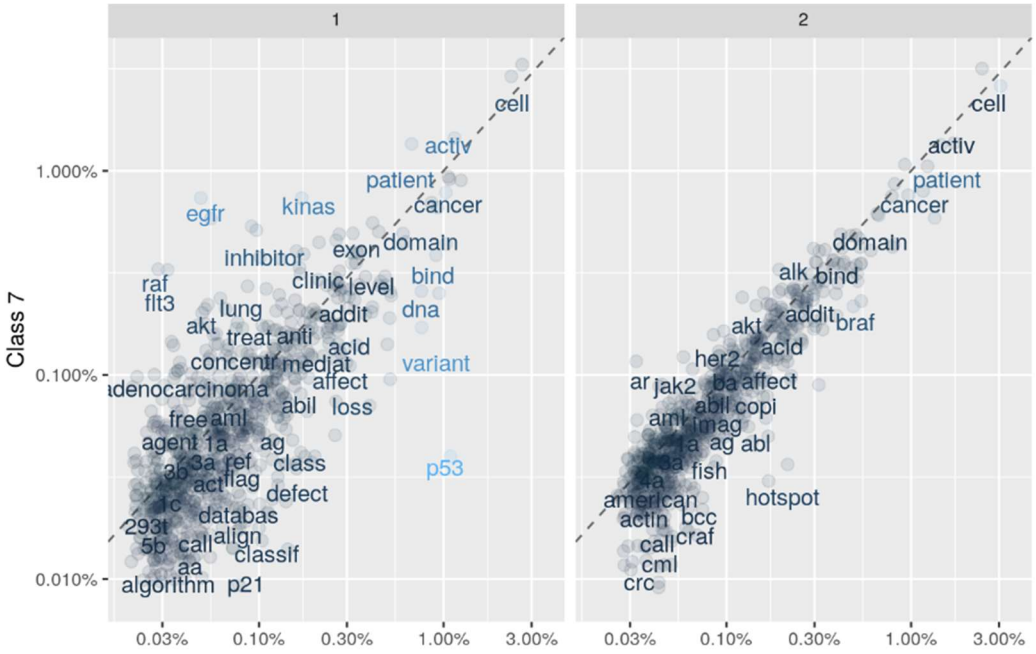

Figura 1. Ejemplo de distribución de los principales términos por representatividad en uno de los clústers.

El conjunto de datos se compone de 3321 documentos anotados de longitud variable (la distribución exacta de la longitud de los textos puede observarse en el gráfico incorporado como figura 2) conteniendo información de 264 expresiones de genes con 2996 variaciones diferentes. En función de estas variaciones, los documentos se clasifican en 9 clases diferentes numeradas de 1 al 9. El conjunto reserva un 30% de los datos para test. Los genes más frecuentes cuando comparamos los datos de entrenamiento y prueba son completamente diferentes, lo que nos indica que este campo por sí solo no tiene influencia en la clasificación. Además, los datos de prueba parecen contener significativamente más Genes diferentes y menos Genes con una alta frecuencia que los datos de entrenamiento. Por lo tanto, la diferencia en la frecuencia podría reflejar que los datos de entrenamiento no recogen toda la riqueza de los datos de prueba. En contraste, las variaciones más frecuentes entre ambos conjuntos son en gran medida idénticas; aunque, nuevamente, las frecuencias correspondientes son más bajas en los datos de prueba (por un factor de 5 a 10).

Acerca de la terminología científica y los términos clave encontrados, la mayoría de los artículos científicos tienen un estilo de lenguaje común que será razonablemente homogéneo en todos los archivos de texto. Las palabras como "resultado" o "artículo" serán frecuentes sin necesariamente contener ninguna señal para nuestro objetivo de predicción. Por lo tanto, definimos nuestras propias listas de palabras de parada y sinónimos en función del dataset que estamos abordando. Después de varias pruebas, generamos una lista de 400 palabras al incluir términos característicos del campo de investigación general que son tan omnipresentes que su alta frecuencia puede enmascarar términos realmente interesantes. Las palabras como "mutación", "cáncer" o "tumor" parecen ser demasiado generales para tener un poder distintivo aquí. Sería interesante obtener comentarios de personas con conocimiento de dominio sobre qué otros términos podrían eliminarse a priori del texto, supliremos esta carencia utilizando en el entrenamiento distintos glosarios médicos especializados.

También encontramos una serie de artefactos provenientes de la digitalización de documentos desde el papel. Ninguno de ellos tendrá un gran impacto individualmente, pero juntos podrían reducir la precisión del análisis: Los números de citas (como los utilizados, por ejemplo, en los papers científicos) se adjuntan a la palabra correspondiente. Ocasionalmente, también encontramos fragmentos provenientes de sintaxis XML frecuentes en contenidos descargados de la web. Otros campos filtrados incluyen: números de página, encabezados, los nombres de los autores y las afiliaciones o las referencias a centros de investigación. Parece haber diferencias significativas en la forma y la mediana de las distribuciones de longitud de la prueba. Las clases 8 y 9 requieren en promedio más texto, mientras que la clase 3 tiene los trabajos más cortos. Estas diferencias parecen responder a que algunas clases de textos contienen mayor cantidad de evidencia clínica.

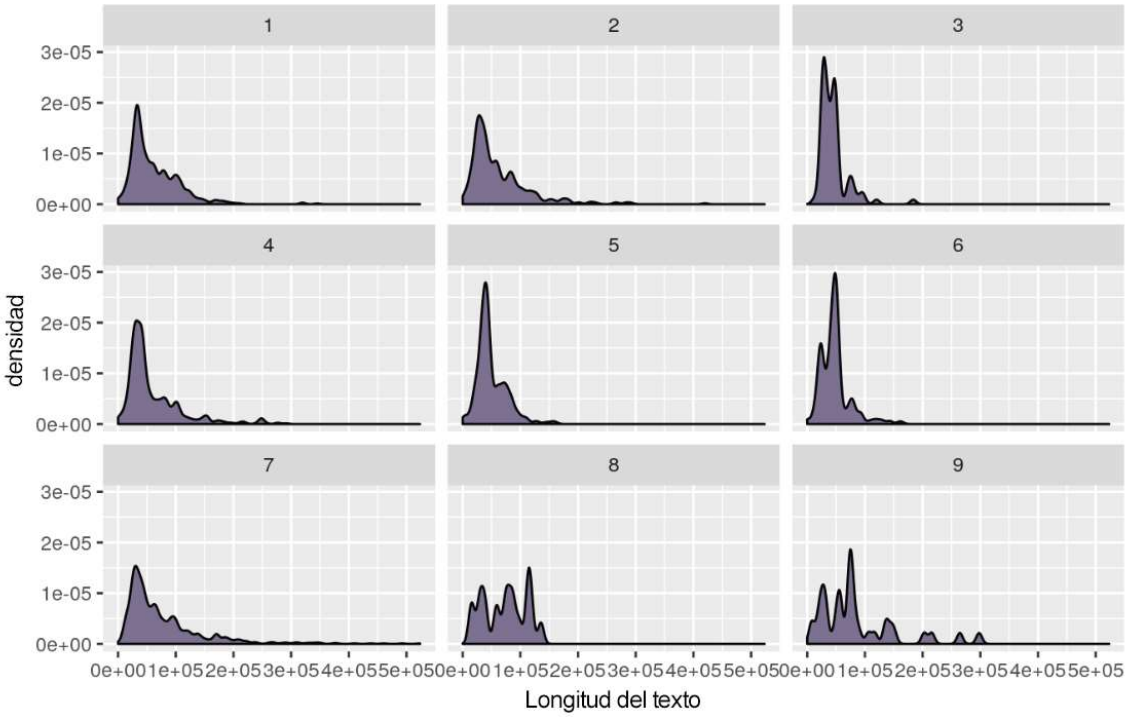

Figura 2.. La distribución de la longitud de los textos según longitud en las diferentes clases.

1.2.1 Consideraciones éticas

Todos los datos utilizados en este trabajo provienen del citado dataset que ha sido debidamente anonimizado y no contiene información personal de pacientes ni casos clínicos que pudiese llevar a su identificación. Todos estos datos han sido puestos a disposición de la comunidad científica por el MSKCC y por tanto ha pasado todos los controles de las autoridades competentes.

1.3 Hipótesis de trabajo y objetivos

Premisa: La estandarización de la labor médica hoy día permite la existencia de numerosos glosarios y diccionarios de terminología clínica. Comprender e interpretar correctamente esta terminología, sus relaciones, y la importancia del orden en que aparecen en un documento es crucial a la hora de entrenar un modelo para clasificar un documento clínico.

Hipótesis principal. H.1 Un modelo semántico no generalista entrenado específicamente en el tratamiento de terminología biomédica obtendrá más y mejor información que un modelo de incrustación generalista y esto permitirá a sistema clasificador basado en esta información tomar mejores soluciones y obtener predicciones mejores.

H.1.1 Aunque nuestro modelo seguirá necesitando de un modelo de incrustación de uso general (en nuestro caso usamos el Glove de 200 dimensiones para inglés americano) el peso principal de la clasificación no corresponderá a este, con lo podemos ser menos exigentes y no necesitaremos reentrenarlo. Lo que nos permite utilizar modelos más ligeros con el consiguiente ahorro de RAM y tiempo computacional.

Premisa: Las redes convolucionales son eficientes en el procesamiento de textos cortos y gramáticas simples, pero resultan demasiado planas para recoger la riqueza contextual de un texto largo donde existen dependencias temporales y espaciales a medio plazo. En la literatura asociada los enfoques híbridos parecen suplir estar carencias, en concreto, las redes recurrentes LSTM se han mostrado prometedoras a la hora de identificar series temporales y relaciones a larga distancia.

Hipótesis secundaria. H.2 El uso de un enfoque híbrido debería aumentar la precisión del modelo al aprovechar las características propias de cada tipo de red. La fase convolucional actuará explorando las características locales. Mientras que las fases recurrentes permitirán dotar a la red de una visión global y temporal.

H.2.1 Los modelos con arquitectura híbrida permiten retener más carga semántica con un menor uso de capas densas. La reducción en el número de capas densas debería redundar de nuevo en un menor tiempo de cómputo sin obtener necesariamente resultados de peor calidad.

Premisa: Los mecanismos de atención fueron introducidos en los algoritmos de procesamiento del lenguaje natural porque permiten que el método procese entradas más largas evitando que la red olvide características importantes que ya han aparecido en la secuencia.

Hipótesis secundaria. H.3 La adición de un mecanismo de atención también debe permitir acotar un conjunto más restrictivo de características relevantes, reduciendo la dimensión de las matrices de términos y permitiendo obtener mejores resultados con un menos uso de memoria.

Para verificar las hipótesis de trabajo, se han definido los siguientes objetivos específicos:

1.3.1 Objetivos funcionales.

Principales:

OF.1 Recopilar un corpus suficientemente amplio y representativo del conjunto de datos de estudio que tenga en cuenta la diversidad y particularidades del ámbito de aplicación.

OF.2 Preprocesar los datos para eliminar aquellas partes que no aportan información de valor, redundancias, información de formateo, cabeceras, sinónimos, unidades de medida y referencias sin significancia clínica.

OF.3 Entrenar un modelo Fasttext sobre el corpus elaborado en el punto anterior y realizar una comparativa de su rendimiento predictivo frente a otros modelos genéricos.

OF.4 Proponer un primer modelo de ensamblado y pruebas de aplicación a la clasificación rápida de documentos clínicos.

OF.5 Realizar un análisis profundo del estado de la cuestión e implementación de los algoritmos más destacables de referencia en la literatura asociada.

OF.6 Proponer un segundo modelo basado en un enfoque híbrido que aúne varios de los enfoques exitosos anteriores y comparativa contra los modelos de referencia aplicado específicamente sobre el dataset**.**

Secundarios:

OF.7 Desarrollar una interfaz web de demostración que permita ajustar manualmente los parámetros de algoritmo e interactuar con el modelo entrenado de forma que se obtengan resultados de clasificación para textos no pertenecientes a los conjuntos iniciales de entrenamiento / validación.

OF.8 Estudiar las posibilidades de aplicación del modelo a otros conjuntos de datos o dataset más amplios y otras líneas de aplicación futuras.

1.3.2 Objetivos metodológicos y operativos.

OM.1 Configuración de un entorno de pruebas basado en Python con soporte para las librerías TensorFlow y Keras sobre infraestructura virtualizada.

OM.2. Instalación del entorno de desarrollo integrado Jupyter facilitando el acceso por consola y la ejecución de código en remoto mediante interfaz web.

OM.3. Documentación del código fuente y de los distintos experimentos de forma que se garantice la reproducibilidad de estos.

OM.4. Extracción de las principales conclusiones obtenidas durante el desarrollo del trabajo e identificar posibles mejoras y líneas de trabajo futuro en el contexto de los resultados recogidos hasta la fecha por otros trabajos relacionados.

1.4 Metodología empleada

Con el objetivo final de evaluar nuestras hipótesis de trabajo, adoptamos un enfoque metodológico basado en la experimentación y el refinamiento sucesivo de hipótesis cuyo funcionamiento general se describe en los siguientes pasos:

1. Revisar el estado del arte para tener una visión clara del campo del procesamiento automático del lenguaje médico, detección exitosa de problemas abiertos, y los enfoques y líneas de investigación a seguir.

2. Seleccionar las fuentes de datos que se van a utilizar, detectando, cómo se deben extraer los datos de ellos y cómo se deben procesar estos datos para ser afectivos información.

3. Definir el alcance del trabajo, los objetivos a conseguir, requerimientos técnicos y funcionales y proponer un plan de trabajo.

4. Implementar las diferentes soluciones y prototipos, documentando adecuadamente el código para garantizar su reproducibilidad.

5. Llevar a término los distintos experimentos garantizando que no existen factores externos que influyen en los resultados y siguiendo rigorosamente el método científico para obtener los datos generados para luego analizar las mejores formas de interpretarlos.

6. Analizar los resultados de los diferentes enfoques contrastando fortalezas y debilidades de cada enfoque y determinando las aportaciones al campo y la medida en que se han validado nuestras hipótesis iniciales.

Para la consecución de los anteriores puntos se programa el trabajo de 6 hitos que necesariamente deben ejecutarse en forma secuencial ya que cada una necesita para comenzar el entregable acabado producto de la fase anterior:

1- Elaboración de la propuesta de investigación (2-4 semanas)

2- Lectura de documentación para elaborar el Estado del Arte (8-10 semanas)

3- Desarrollo software/hardware asociado al TFM y su desarrollo. Este punto incluye el estudio de posibles entornos de ejecución *Cloud*. (12-24 semanas)

4- Diseño de la experimentación con la que se pretende validar el TFM (24-32 semanas)

a. Limpieza y normalización de dataset y entrenamiento de un modelo FastText en corpus específico (bronco corpus + farmacológico).

b. Propuesta del primero modelo y comparativa de rendimiento contra los diferentes métodos de incrustación preentrenados.

c. Implementación de los principales algoritmos en el estado el arte y análisis de rendimiento.

d. Propuesta de arquitectura final, ajuste de los hiperparámetros y pruebas de rendimiento.

e. Prototipo funcional que permita testar el modelo y medir su posible aplicabilidad futura.

5- Redacción de la memoria del TFM (8-12 semanas)

6- Redacción y confección de paper científico recogiendo las principales conclusiones del trabajo. (4-8 semanas)

7- Preparación de la defensa del TFM, incluyendo la preparación de la presentación (1-2 semanas)

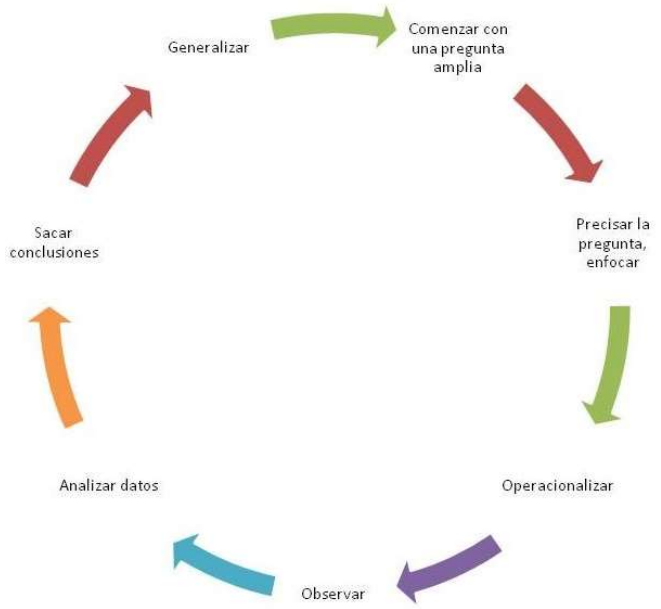

Figura 3. Conceptualización básica del ciclo de desarrollo en las metodologías de desarrollo ágil.

Esto no quiere decir que dichas fases no se realimenten de forma que la consecución de una fase pueda significar el replanteamiento o mejora de la anterior, siguiendo una metodología de mejora continua dentro de un desarrollo ágil basado en entregables cuyo funcionamiento e idea básica está expuesta en la figura 3 y se basa en un ciclo cerrado de seis etapas: Observación, Análisis, Generalización, Hipótesis, Experimentación y Refinamiento. Para conocer más detalles sobre la metodología ágil y basada en prototipos, el lector puede acudir a las fuentes **Highsmith, J., & Cockburn, A., 2001** y **Astels, D., 2003**.

1.5 Resultados esperados

Como meta del presente estudio nos proponemos la validación o descarte de las hipótesis de trabajo establecidas en el apartado 1.3, lo que nos permitirá comprobar si efectivamente el enfoque híbrido propuesto consigue ser útil para mejorar el rendimiento de los clasificadores clínicos a la vez que nos permite obtener un mejor conocimiento sobre las redes convolucionales y su aplicabilidad a textos de temática compleja, en este caso a textos dentro del área biomédica. Adicionalmente, como resultado del trabajo esperamos obtener un modelo exportable a otros datasets basado en una arquitectura con enfoque híbrido ligero con el que queremos conseguir resultados similares o mejores que los recogidos en el estado del arte tomando como métricas la precesión (*accuracy*) y la media de la tasa de acierto (*f1_score_avg*). Los resultados del trabajo deberán conllevar la consecución de los objetivos funcionales establecidos en las secciones anteriores (ver objetivos OF1 a OF8) en el marco de los objetivos metodológicos y operativos que nos marcamos en las secciones 1.4 y 1.5 (ver objetivos OM1 a OM4).

La consecución de estos objetivos, validaría nuestras hipótesis iniciales, y podríamos afirmar que la clasificación no supervisada de documentos clínicos desestructurados puede ser rentable utilizando una arquitectura de red neuronal híbrida CNN-RNN ligera entrenada sobre un corpus específico cuidadosamente seleccionado, lo que puede tener interés académico e importancia en la explotación de grandes volúmenes de este tipo de datos, siendo esta situación muy común, bien por provenir estos de aplicativos clínicos fuera de soporte, desde fuentes propietarias o de las que desconocemos el formato, con anterioridad a la implantación de los modernos estándares de interoperativilidad de sistemas clínicos, o en la indexación automática de documentación clínica como paso previo al etiquetado semántico.

Durante la búsqueda de estos objetivos se desarrollarán los siguientes productos y entregables:

- Detalle de las arquitecturas propuestas, modelos entrenados con los datos de entrada y métricas de rendimiento.
- Interfaz Python para interactuar con los diferentes modelos.
- Prototipo de aplicación web completamente funcional que permita el acceso en línea al modelo entrenado.
- Documentación asociada.

Como resultado adicional se incluirá el envío de un artículo científico recogiendo los resultados más relevantes del trabajo para su admisión en una revista o boletín científico, con la finalidad de difundir a la comunidad científica internacional los resultados de esta investigación en alguna de las ramas de conocimiento especializada en algoritmia, inteligencia artificial o procesamiento del lenguaje lo que nos permitirá validar su rigor científico, y luego de un análisis, su publicación.

1.5 Estructura y organización de la memoria

En este primer capítulo nos hemos centrado en la motivación del trabajo y en explicar su contexto, antecedentes, objetivos planteados como metas de manera primaria o secundaria y los resultados esperados.

El segundo capítulo es una actualización del estado de la cuestión desde el enfoque de las redes convolucionales aplicadas a la clasificación de textos biomédicos y sus diferentes elementos destacables desde el punto de vista investigador.

En el tercer capítulo estudiaremos en profundidad el dataset y todo lo que conlleva la preparación de los datos y los pasos previos a la experimentación.

A lo largo del cuarto capítulo**,** se detallarán los experimentos realizados y se expondrá en detalle los resultados obtenidos durante las distintas fases de experimentación y su repercusión en el estudio.

Para ya en el capítulo cinco centrarnos en el estudio de resultados y verificación de estos y su encuadre dentro del contexto del estado del arte.

Con todo lo anterior, en el capítulo seis realizaremos un análisis comparado de los resultados, basado en las evidencias encontradas y en el análisis crítico de los hallazgos.

Finalmente, el capítulo siete y el capítulo ocho presentan las principales conclusiones del estudio, discutiendo cuáles son sus aportaciones al campo, que aspectos son destacables desde el punto de vista investigador, y en qué grado se han validado las hipótesis iniciales junto a posibles líneas futuras de investigación o aspectos que se considera que merecerían ser objeto de futuras líneas de estudio.

Por otra parte, los distintos anexos contendrán otra información útil y de interés, el manual de uso del aplicativo, la estimación de costes y la planificación básica del desarrollo del proyecto y sus principales hitos, así como enlace al código fuente completo utilizado en la obtención de los diferentes experimentos.

CAPÍTULO II

CONTEXTO Y TRABAJOS RELACIONADOS.

2. Contexto del estudio

2.1 Background.

La recopilación de conjuntos de datos etiquetados en el dominio biomédico se justifica tanto por su escasa disponibilidad, como por lo difícil de su obtención, y su papel fundamental para hacer avanzar la medicina a través de métodos de aprendizaje automático. Los conjuntos de datos abiertos y de libre acceso en biomedicina ofrecen enormes beneficios en el bienestar de los pacientes, la economía y la sociedad en su conjunto. El valor de los datos aumenta con su flexibilidad para combinarse, reubicarse y compartirse (**Allen & Dreyer, 2019**). En los últimos años las redes neuronales de aprendizaje profundo y el uso de los mapas semánticos preentrenados (más conocidos como incrustaciones de palabras por su origen en el término inglés *word embeddings*) han tenido un papel destacado y constituyen el estado-del-arte del etiquetado automático de textos desestructurados. Sin embargo, este tipo de métodos tienen una desventaja obvia; dado que el entrenamiento se realiza de manera no supervisada, no hay garantías de que dichas relaciones recojan apropiadamente la riqueza semántica de textos. Aun así, se han aplicado con éxito redes neuronales a la extracción de entidades de naturaleza clínica en textos semi o parcialmente estructurados, pero limitados o enfocados únicamente a estudiar temáticas acotadas y muy específicas como la sintomatología en cáncer (**Strauss et al., 2012**; **Carrell et al., 2014**), la evolución de cuadros de afecciones del espectro autista (**Lingren et al., 2016**), la aplicación o ensayo de diversos tratamientos farmacológicos (**Savova et al., 2011**), o los hábitos en el consumo de alcohol o tabaco (**Uzuner et al., 2008; Wang et al., 2020**), entre otros.

Sin embargo, el abordaje más general de textos clínicos presenta aún retos que hacen difícil su abordaje como la extensa terminología clínica, los errores ortográficos o terminológicos, el uso continuo de sinónimos y términos ambiguos, y las distintas nomenclaturas médicas (**Nadkarni et al., 2011**, **Murff et al., 2011**). En este escenario, las redes convolucionales parecen poseer interesantes potenciales en autoorganización, permitiéndoles ser semiautónomas en el autodescubrimiento de la estructura de los datos cuando esta se desconoce; y, como todas las redes neuronales, sus capacidades de escalabilidad y robustez son muy buenas al no verse tan influenciado su rendimiento cuando aumenta el número de entradas como en otras familias de técnicas. A pesar de la eficiencia de las redes CNN y su sofisticado mecanismo de extracción de características, la incapacidad de incorporar información de contexto a media y larga distancia limita su uso. Dado que la información de contexto es muy importante en un problema de NLP, parece muy atractiva la idea de incorporar un mecanismo que aporte una visión contextual más amplia a los modelos CNN. (**Lai y col, 2015**) fueron los primeros en proponer la utilización de una red neuronal recurrente (RNN) para esta función, de esta manera, la red CNN puede dedicarse más a la interpretación de la sintaxis y la extracción de características mientras que la red RNN se centra en la semántica y crea relaciones a más larga distancia. El esquema híbrido, en cierto grado imita la forma de la percepción humana donde distintas áreas del cerebro se especializan en la extracción conceptual, la detección de patrones, para el procesamiento temporal o espacial o la abstracción matemática. En la figura 4 podemos ver algunos ejemplos clásicos de problemas de difícil abordaje computacional pues requieren de distintas etapas de percepción especializadas. La forma de unir y hacer colaborar estas distintas redes especializadas dentro de una arquitectura cohesionada es una tarea ardua y que requiere un enfoque distinto para cada problema abordado.

En las siguientes secciones veremos de forma muy breve las características más relevantes de cada una de estas redes y las distintas arquitecturas que se han propuesto tratando de obtener el mayor grado de sinergia entre estas distintas redes y extender su uso a escenarios cada vez más ambiciosos.

Figura 4. Las capacidades de abstracción y autoorganización de la percepción humana permiten que un conjunto de líneas pueda evocar una respuesta asociativa en forma de recuerdo o imágenes.

2.2 Redes Neuronales profundas

Los métodos de aprendizaje profundo se están volviendo cada vez más importantes para resolver varias tareas de procesamiento y han logrado resultados muy notables en diferentes áreas (**Krizhevsky et al., 2013**; **Herrera-Alonso, 2015**; **Jiménez-Perera, 2017**; **Karimi et al. 2017**), alcanzando resultados de vanguardia en los últimos años para la PLN en el área médica. Superan los enfoques de aprendizaje automático tanto basados en el diccionario como tradicionales que requieren un esfuerzo significativo en el procesamiento previo y la extracción artesanal de características (**Huang et al., 2014**; **Lisitsa & Zhilenkov, 2017**; **Gehrmann et al., 2018**). Estos métodos están comenzando a alcanzar una mayor precisión y habilidades semánticas (**Kim et al., 2014**; **Yang et al., 2016**; **Chen et al., 2017**) que los métodos de clasificación basados en el conocimiento experto en relación con diversas tareas biomédicas, como la anotación y clasificación automática de texto clínico. En otros métodos de aprendizaje tradicionales, como las máquinas de vectores de soporte (SVM) las características gramaticales están hechas a mano, en las redes de aprendizaje profundo estas características se aprenden de los datos sin procesar. El desafío es lograr una precisión suficientemente alta con un bajo esfuerzo computacional que haga que su explotación sea lucrativa en un entorno donde el rendimiento de otras técnicas es especialmente limitado debido al alto nivel de especialización, lo que dificulta la obtención de conceptos abstractos de alto nivel directamente de un texto sin formato. El aprendizaje profundo evita este problema aprendiendo representaciones complejas expresadas en términos de otras representaciones más simples. La Tabla 1 enumera algunas de los trabajos más significativos, ya que fueron los primeros en introducir ciertas mejoras que se incorporaron al estado del arte a medida que su uso se generalizó. Esta tabla muestra la innovación principal que realizaron, el tamaño del vocabulario base y el número de muestras en el conjunto de datos.

2.2.1 Redes Convolucionales (CNN)

Una red convolucional (CNN) es una red neutral multicapa que está biológicamente inspirada en la corteza visual de los animales. La arquitectura es particularmente útil en aplicaciones para procesar imágenes. El mecanismo principal de una CNN son las "convoluciones" o capas convolucionales. Las convoluciones funcionan como filtros capaces de abstracción y transformación que permiten extraer características de diferentes niveles y granularidad. Al principio era evidente que, al igual que la traducción, la rotación y la invariancia de escala de las imágenes, una CNN puede aprender las características locales de palabras o frases en diferentes lugares de textos y recientemente ha obtenido popularidad para las tareas de procesamiento del lenguaje natural (**Collobert & Weston, 2008**; **Levy & Goldberg, 2015**; **Cho et al., 2015**; **Zang & Wallace, 2015**; **Liu et al., 2018**). Un modelo CNN puede aprender simultáneamente una representación distribuida para cada palabra, esto es lo que llamamos incrustaciones de palabras (WE), junto con la función de probabilidad para secuencias de palabras basadas en las representaciones anteriores (**Bengio et al., 2013**; **Bojanowski et al, 2016**). Los métodos WE capturan las similitudes / regularidades entre las palabras (**Mikolov et al. 2013**; **Pennington et al., 2014**). Esta idea se ha extendido para obtener incrustaciones que capturan la semántica de secuencias de palabras más largas como frases, oraciones, o párrafos (**Yang et al., 2016**; **Hughes et al., 2017**).

Tabla 1. Tabla resumen de las obras más notables del estado del arte.

Publicación	Arquitectura	Año	#Muestras	Long. media (palabras)
Kim et al. [3]	CNN + pretrained word vectors	2014	variable	~ 25
Le and Mikolov [56]	Tree-Based CNN for paragraph level embedding	2014	11K	~ 25
Rocktaschel et al. [35]	CNN w/ Attention on long sentences	2015	570K	~ 25
Wu et al. [42]	CNN for patient characterisation	2015	403K	variable
Palangi et al. [27]	Hybrid LSTM-RNN model for unstructured text	2015	200K	unspecified
Yin et al. [21]	Multichannel and variable-size convolution	2016	~10K	~ 200
Yang et al. [31]	CNN w/ Hierarchy-based Attention	2016	~10K	~ 50
Wang et al. [34]	Hybrid CNN + RNN	2016	~10K	~ 50
Chen et al. [38]	Hybrid CNN + LSTM	2017	804K	~ 50
Krause et al. [28]	Multiplicative LSTM	2017	2M	~ 1000
Pappas&Popescu [40]	Multilingual Hierarchical Attention Network	2017	100K	516
Gehrmann et at. [41]	CNN w/ Attention for named entity recognition	2018	53K	~ 200

2.2.2 Redes Neuronales Recurrentes (RNN)

Las redes neuronales recursivas o recurrentes (RNN) son una de las arquitecturas fundamentales de las redes sobre las cuales se han construido las otras arquitecturas de aprendizaje profundo. La principal diferencia entre una red multicapa típica y una red recurrente es que, en lugar de alimentar completamente las conexiones, una red recurrente puede tener conexiones que realimentan las capas anteriores (o en la misma capa). Esta retroalimentación permite a los RNN mantener una memoria de entradas pasadas y modelar problemas en ese momento. Los RNN están formados por un amplio conjunto de arquitecturas. Uno de los tipos de RNN más conocidos y utilizados son las *Long Short-Term Memory* (LSTM). Las LSTM crean conexiones a través del tiempo, esas rutas son pesos de conexión que pueden cambiar en cada iteración permitiendo acumular información u olvidar borrando los estados anteriores. Las LSTM se desvían de las arquitecturas típicas de redes neuronales basadas en neuronas y en su lugar introducen el concepto de una célula de memoria. La celda de memoria puede retener su valor por un período de tiempo corto o largo en función de sus entradas, lo que le permite a la celda recordar lo que es importante y no solo el último valor que calculó. En 2014, Lanzarini et al. introdujeron una simplificación del LSTM que se llamó *Gated Recurrent Unit* (GRU). Este modelo tiene solo dos puertas lógicas: una puerta de actualización y una puerta de reinicio. La puerta de actualización indica la cantidad de contenido que se debe mantener en las celdas anteriores. La puerta de reinicio define cómo incorporar la nueva entrada con los contenidos anteriores de la celda, descartando la puerta de salida que está presente en el modelo LSTM. Para muchas aplicaciones, las GRU tienen un rendimiento comparable al de las LSTM, pero, al ser más simple, tiene menos pesos y una ejecución más rápida con menos memoria.

2.3 Modelos basados en incrustaciones de palabras

Existen muchos métodos diferentes para aprender incrustaciones de palabras a partir de un corpus, aunque actualmente, los tipos más comunes de incrustaciones de texto utilizados son los generados con: i) Word2Vec (**Mikolov et al., 2013**), que incluye modelos *skip-gram* y CBoW (*Continuous Bag of Words*), ii) Glove (**Pennington & Manning, 2014**), o iii) FastText (**Grave et al., 2017**), considerándose una versión más reciente y mejorada de los dos primeros. Los métodos de Word embedding fueron definidos por (**Bengio et al., 2003**). Las Word Embedding son incrustaciones aprendidas sin supervisión donde las características de la palabra se obtienen a partir de las activaciones de la primera capa (capa de embedding) de una red neural entrenada para producir una distribución de probabilidad sobre cómo se relacionan los distintos términos en un texto. El objetivo de estos métodos es realizar predicciones basadas en el contexto. Los métodos de incrustación de palabras tratan de mapear cada término del vocabulario con una representación densa, cuya dimensionalidad es típicamente mucho más baja que el tamaño del vocabulario, y cuyos elementos capturan lo semántica intrínseca de los datos del lenguaje.

2.1.1. Word2vec

Word2vec es un método de incrustación de palabras que utiliza redes neuronales para aprender el mapeo de palabras a un punto en un espacio vectorial basado en dos parámetros: el número de dimensiones del embedding (normalmente entre 50 y 500), y la longitud de la ventana de contexto (es decir, cuántas palabras antes y después de la palabra objetivo deben usarse como contexto para entrenar las incrustaciones de palabras, generalmente 5 o 10 palabras). Mientras que entrenar incrustaciones con más dimensiones generalmente requiere más datos de entrenamiento (cada dimensión debe capturar algún aspecto del significado). Doc2vec y Par2vec (**Le & Mikolov, 2014**) son variantes de Word2vec en las que los vectores representan documentos o párrafos en lugar de palabras, respectivamente.

2.1.2. Global Vectors for Word Representation (GloVe)

El modelo GloVe también aprende incrustaciones de palabras, pero a partir de una matriz de coincidencia de términos en lugar de una tarea de predicción de palabras (**Pennington et al., 2014**). Una matriz de concurrencia es una matriz V x V donde V es el tamaño del vocabulario. Cada entrada de la matriz corresponde al número de veces que los elementos de vocabulario ocurren juntos dentro de una ventana de contexto preespecificada, que se mueve por todo el cuerpo. Glove aprende incrustaciones de vectores para minimizar el error de reconstrucción entre las estadísticas de concurrencia predichas por el modelo y las estadísticas de concurrencia global observadas en el corpus de entrenamiento. El modelo consta de numerosos hiperparámetros que deben elegirse con criterio, incluida la dimensión de incrustación de vectores y el tamaño de la ventana de contexto. Los vectores de palabras estimados usando GloVe son conceptualmente similares a los creados mediante Word2vec (**Mikolov et al., 2016**) pero usa un modelo subyacente basado en conteo, más eficiente que el modelo basado en predicciones de Word2vec. Debido a que GloVe generalmente calcula estadísticas sobre ventanas de contexto más grandes que Word2vec, con esto permite capturar dependencias a más largo plazo.

2.4 Modelos híbridos para NLP

Palangi y col. (2015) propusieron un modelo híbrido LSTM-RNN para la clasificación de texto no estructurado que demuestra que un modelo híbrido puede funcionar mejor para la clasificación de documentos completos que usar un segundo nivel de convolución para codificar vectores de incrustación de párrafos para cada documento. A medida que el modelo lee hasta el final de la oración, la activación del tema (salida para la clase) se incrementa y la representación del estado oculto en la última palabra codifica la rica información contextual de toda la oración. LSTM-RNN es efectivo contra el ruido y puede ser robusto en situaciones en las que cada palabra en el documento no es igualmente importante y solo las palabras sobresalientes deben recordarse usando la memoria limitada. Este es un modelo muy simple que toma todas las palabras del documento de forma secuencial y el resultado final obtiene el *document embedding (mapa semántico del documento)*. El modelo no utiliza capas *max-pooling* para capturar información contextual global. Este modelo híbrido se mejoró más tarde para (**Chen et al., 2016**) y (**Wang et al. 2017**) obteniendo un mejor efecto de sinergia. En 2016, Yin et al. propusieron MV-CNN, una arquitectura que combina diversas versiones de mapas semánticos (*word embeddings)* pre-entrenadas y extrae características de frases de distinta granularidad con convoluciones de tamaño variable. El uso de múltiples *embeddings* de la misma dimensión a partir de diferentes conjuntos de vectores de palabras debería contener más información que se pueda aprovechar durante el entrenamiento. Esta inicialización multicanal puede ser de ayuda con palabras desconocidas en diferentes *embeddings*. Las palabras frecuentes pueden tener múltiples representaciones y las palabras menos usuales pueden ser formadas a partir de otras palabras. Luego, el modelo tiene dos conjuntos de capas de convolución y capas dinámicas de *pooling* seguidas de una capa completamente conectada con salida *softmax*. El documento describe dos mecanismos para mejorar el modelo. Uno se llama aprendizaje mutuo, que se implementa en este proyecto. Consiste en mantener el mismo vocabulario en todos los canales mientras que se ayudan mutuamente a ajustar los parámetros durante el entrenamiento. El otro truco que se utiliza es mejorar las *embeddings* con preentrenamiento al igual que un *autoencoder* utilizando estimación de contraste de ruido. Los diferentes conjuntos de *embeddings* utilizados para este modelo provienen de los mapas preentrenados Glove y Word2vec.

(Krause et al. 2016) propusieron la arquitectura *multiplicative*-LSTM[3], que es un RNN híbrido que le da al modelo más flexibilidad para elegir transiciones recurrentes para cada posible entrada, lo que lo hace más expresivo en la estimación de densidad autorregresiva. Este modelo se propone para el modelado secuencial del lenguaje, pero se eligió para este proyecto para ver cómo ayuda en las tareas de clasificación de texto. Los autores sostienen que los RNN[4] actuales tienen dificultades para recuperarse de los errores al predecir secuencias. Si el estado oculto RNN recuerda información errónea, entonces podría tomar más tiempo recuperarse y habrá un efecto de bola de nieve. Algunas soluciones propuestas, como la introducción de variables latentes y el aumento de la memoria, solo darán como resultado una distribución compleja intratable sobre los estados ocultos y la reinterpretación de las entradas almacenadas. El modelo multiplicativo proporciona más flexibilidad para las transiciones en este tipo de situaciones.

Por su parte (Zhou et al., 2015) presentaron el modelo C-LSTM, que combina una red convolucional CNN para capturar características locales de frases y una red recurrente RNN de tipo LSTM para capturar semántica de oraciones de forma global y temporal. Una capa LSTM aprende dependencias a largo plazo entre pasos de tiempo en series temporales y datos en secuencia. El modelo extrae correlaciones de alto nivel de características con ventanas más cortas en los modelos CNN que se proveen en forma de secuencia a la siguiente capa LSTM, lo que ayuda a capturar mayor semántica del documento para su clasificación. La salida de cada convolución (mapa de características únicas) tiene información semántica de toda la oración como una secuencia. A partir de estos mapas de características múltiples, sin utilizar capas de *max-pooling*, los vectores que representan la información posicional se agrupan o concatenan y se introducen en la red RNN como una secuencia.

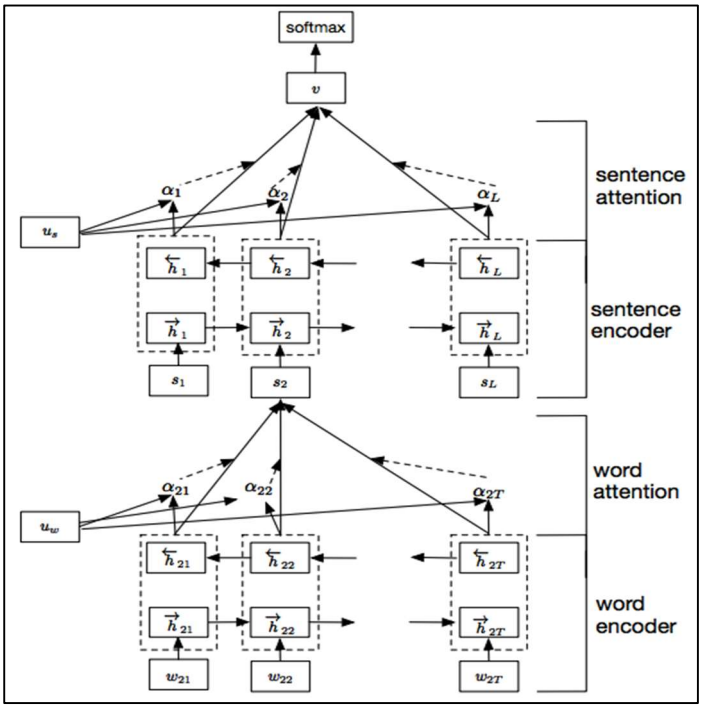

Figura 5. Arquitectura basada en un mecanismo de atención a dos niveles. (Yang et al., 2016)

Una desventaja importante de utilizar de vectores de contexto de longitud fija es la incapacidad del método para recordar secuencias más largas. Dada una secuencia larga, por ejemplo, un párrafo, un revisor humano puede resaltar esas ideas, términos o relaciones de términos que resultan esenciales en la interpretación semántica del mismo. Como seres humanos, podemos comprender rápidamente estas asignaciones entre varias partes de la secuencia de entrada y las partes correspondientes de la secuencia de salida. Sin embargo, no es tan sencillo para la red neuronal artificial detectar automáticamente estas asignaciones. Para paliar este problema, (Yang et al., 2016) propusieron una red con un mecanismo de atención a la que llaman HAN

[3] Una LSTM o Long short-term memory es una RNN que implementa celdas de memoria de forma que la red puede retener cierta información contextual durante un lapso de tiempo determinado.
[4] Una RNN o Recurrent neural network es una CNN que implementa conexiones de *feedback* de forma que sus salidas se encuentran ponderada por las salidas anteriores.

(Hierarchical Attention Network) pero primero veamos qué significa el mecanismo de atención. Para esto, echemos un vistazo al mecanismo de atención anticipada propuesto por (**Raffel et al., 2015**). El mecanismo de atención de Raffel permite dependencias directas entre los estados del modelo en diferentes momentos (la estructura básica de una arquitectura basada en atención puede verse en la figura 2). A partir de una instantánea, los vectores de pesos vistos en la secuencia de estado oculto se introducen en una función de aprendizaje que produce un vector de probabilidad, que a su vez se utiliza para encontrar el promedio ponderado de los estados ocultos de salida. Por lo tanto, el mecanismo de Atención se desarrolla para aprender estas asignaciones a través del Descenso de gradiente y la propagación inversa. Para mejorar aún más la flexibilidad de representación de tales modelos, se han introducido mecanismos de atención en los componentes de los modelos utilizados para la clasificación de texto. Un mecanismo de atención puede mejorar efectivamente la calidad de la clasificación identificando qué relaciones son importantes y manteniendo el foco en esa información. (**Pappas & Popescu-Belis, 2017**) también presentaron su propio mecanismo de atención, al que llamaron "Red de atención jerárquica multilingüe" abreviado HAL, en un enfoque más similar a nuestro trabajo, ya que lo usaron para limitar el número de características para estudiar, ya que adoptaron un enfoque multilingüe. HAL propone un mecanismo de atención en dos etapas para codificar documentos. La intuición detrás del modelo es que diferentes palabras y oraciones en un documento son diferencialmente informativas. Las representaciones de documentos se construyen a partir de cada oración y cada representación de oración se construye a partir de palabras. Cada oración en un documento puede considerarse como una secuencia de palabras que alimenta a una capa LSTM generativa, y nuevamente se sigue el mismo procedimiento para las oraciones con atención en la parte superior. Finalmente, a la salida existe una capa *softmax* para clasificar las etiquetas del documento.

Por último, (**Chen et al., 2017**) propusieron un modelo que es la combinación de 2 redes que colaborarán entre sí, una red convolucional, y una RNN para la categorización de texto de etiquetas múltiples, que ayuda a capturar la semántica global y local. Emplea un modelo CNN de forma muy similar a como planteaba (**Yoon et al., 2014**) para la clasificación de oraciones y un modelo RNN que devuelve la salida como secuencia clasificada para el etiquetado de esas palabras. El modelo se modifica para usarse para la clasificación de texto. La estructura RNN se usa para crear el sentence embedding (mapa semántico a nivel de oración) con la misma oración de entrada proporcionada para CNN. La salida del modelo CNN se alimenta al primer estado oculto del RNN lo que ayuda a una mejor recuperación de la información.

CAPÍTULO III

PREPARACIÓN DE LOS DATOS Y ENTRENAMIENTO DEL MODELO SEMÁNTICO.

3. Preparación de los datos y entrenamiento del modelo semántico.

El dataset de estudio recoge una serie de estudios científicos publicados en los años 2016 y 2017 referidos a la relación entre una determinada mutación genética y su implicación en el desarrollo de un determinado tipo de cáncer. Estos estudios han sido clasificados por personal experto según su significancia y evidencia clínica en 9 categorías. Por tanto, hay nueve clases diferentes en las que clasificar una mutación genética. Pero su distribución no es nada uniforme, habiendo clases mucho más sobre representadas. Podemos hacernos una idea de esta complejidad observando la distribución de los textos en la figura 6. Además de esto, no existe una clara separación temática entre los clústers. Las distintas categorías no atienden a un único criterio, sino que han sido separadas por expertos en el dominio del problema siguiendo diversos criterios clínicos y no terminológicos.

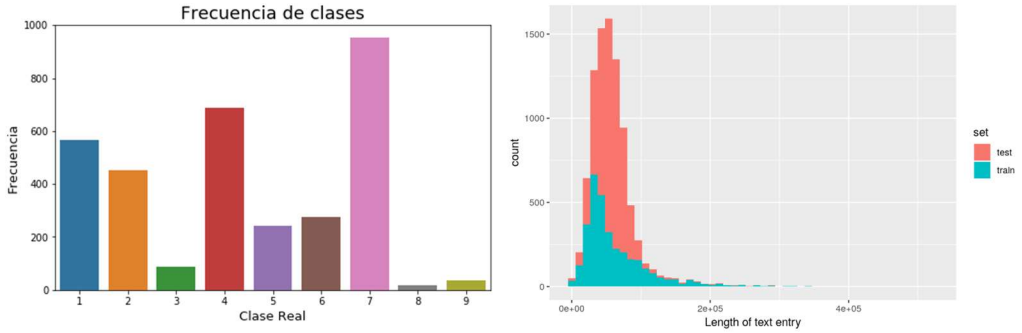

Figura 6. Distribución de clases en el dataset (izq) y de la longitud de las entradas para los subconjuntos de entrenamiento y test.

Esta no es una tarea simple, ya que la interpretación de la evidencia clínica es un reto incluso para los especialistas. Los conjuntos de datos de entrenamiento y de prueba se proporcionan a través de dos archivos diferentes. Uno (training / test_variants) proporciona información sobre las mutaciones genéticas, mientras que el otro (training / test_texts) proporciona la evidencia clínica (texto) que utilizaron los expertos humanos para clasificar las mutaciones genéticas. Ambos están vinculados a través del campo ID. La estructura interna del dataset es la siguiente:

texts *- Contiene la información del artículo.*

id (int) *- identificador único del estudio*

título (string) *– Título del artículo*

texto (text) *– el cuerpo completo del artículo*

variants *– Contains the principal cast/crew for titles*

tid (int) *- identificador único del artículo*

gen1 (string) *– el primer gen gen implicado en el estudio (si aplica)*

gen2 (string) *– el segundo gen gen implicado en el estudio (si aplica)*

real class (int) *- clasificación manual (solo en el conjunto de entrenamiento)*

3.1 Corrigiendo el desbalanceo de clases

Por simplicidad unificamos ambos conjuntos de datos y estudiamos el balanceo de las clases. Observando la distribución de clases vemos que existen clases con muy escasa representación. Concretamente las clases 3, 8 y 9 representan juntas a menos del 10% de los datos. Tenemos diversas estrategias para tratar de mejorar la situación (**Gonzalez et al., 2014**; **Castillo A., 2016**):

- Ajuste de Parámetros del modelo: Consiste en ajustar parámetros u métricas del propio algoritmo para intentar equilibrar a la clase minoritaria penalizando a la clase mayoritaria durante el entrenamiento. Ejemplos de esto son el ajuste de peso en árboles, o en regresión logística el parámetro ***class_weight=balanced***. No todos los algoritmos tienen estas posibilidades.
- Modificar el Dataset: podemos eliminar muestras de la clase mayoritaria para reducirlo e intentar equilibrar la situación. Tiene como "peligroso" que podemos prescindir de muestras importantes, que brindan información y por lo tanto empeorar el modelo. Entonces para seleccionar qué muestras eliminar, deberíamos seguir algún criterio. También podríamos agregar nuevas filas con los mismos valores de las clases minoritarias, por ejemplo, cuadriplicar nuestras 492 filas. Pero esto no sirve demasiado y podemos llevar al modelo a caer en ***overfitting***[5].
- Muestras artificiales: podemos intentar crear muestras sintéticas (no idénticas) utilizando diversos algoritmos que intentan seguir la tendencia del grupo minoritario. Según el método, podemos mejorar los resultados. Lo arriesgado de crear muestras sintéticas es que podemos alterar la distribución natural de esa clase y confundir al modelo en su clasificación.
- Balanced Ensemble Methods: Utiliza las ventajas de hacer ensamblado de métodos, es decir, entrenar diversos modelos y entre todos obtener el resultado final (por ejemplo, votando) pero se asegura de tomar muestras de entrenamiento equilibradas.

En nuestro caso y dado que el tamaño del conjunto no es muy grande optamos por la generación de una submuestra sintética. Para ello se generaron nuevas filas de las clases minoritarias utilizando el generador de muestras de *sklearn*[6] con probabilidad de mutación de 0.05. Además de eso ajustaremos la métrica de *Loss* para que penalice a las clases mayoritarias.

3.2 Limpieza y normalización

Cargamos todo el dataset a memoria y cargamos el contenido de los textos clínicos. Realizamos algunos preprocesamientos básicos como tokenización, minúsculas, etc. y construimos una lista de tokens (palabras).

```
stemmer = snowballstemmer.EnglishStemmer()
stop = stopwords.words('english')
stoplist = stemmer.stemWords(clean(stop))
stoplist = set(stoplist)
stop = set(sorted(stop + list(stoplist)))
def read_input(input_file):
    """This method reads the input file which is in gzip format"""
```

[5] Sobreentrenamiento o sobreadaptación.
[6] Scikit-learn: Machine Learning in Python at https://scikit-learn.org/stable/

```
    logging.info("reading file {0}...this may take a while".format(input_file)
)
    with gzip.open (input_file, 'rb') as f:
        for i, line in enumerate (f):

            if (i%100==0):
                logging.info ("read {0} tuples".format (i))
            yield gensim.utils.simple_preprocess(line)
documents = list(read_input (data_file))
logging.info ("Done reading data file")
```

Entrenando al modelo pasando una lista de listas. Después de construir el vocabulario, solo necesitamos llamar a *train* (...) para comenzar a entrenar el modelo. El entrenamiento en el conjunto de datos NIPS17 (**Kukovacec et al., 2018**) toma aproximadamente 45 minutos.

Eliminamos posibles columnas con valores nulos (\N, null o NaN) para después proceder al tokenizado.

```
ID
0     Cyclin-dependent kinases (CDKs) regulate a var...
1      Abstract Background  Non-small cell lung canc...
2      Abstract Background  Non-small cell lung canc...
3     Recent evidence has demonstrated that acquired...
4     Oncogenic mutations in the monomeric Casitas B...
5     Oncogenic mutations in the monomeric Casitas B...
6     Oncogenic mutations in the monomeric Casitas B...
7     CBL is a negative regulator of activated recep...
8      Abstract Juvenile myelomonocytic leukemia (JM...
9      Abstract Juvenile myelomonocytic leukemia (JM...
10    Oncogenic mutations in the monomeric Casitas B...
11    Noonan syndrome is an autosomal dominant conge...
12    Noonan syndrome is an autosomal dominant conge...
13    Noonan syndrome is an autosomal dominant conge...
14    Oncogenic mutations in the monomeric Casitas B...
15    Noonan syndrome is an autosomal dominant conge...
16    To determine if residual cylindrical refractiv...
17    Acquired uniparental disomy (aUPD) is a common...
18    Oncogenic mutations in the monomeric Casitas B...
19    Acquired uniparental disomy (aUPD) is a common...
20     Abstract Background  Non-small cell lung canc...
21    Oncogenic mutations in the monomeric Casitas B...
22    Oncogenic mutations in the monomeric Casitas B...
23    Recent evidence has demonstrated that acquired...
24    Recent evidence has demonstrated that acquired...
25    Recent evidence has demonstrated that acquired...
26     Abstract N-myristoylation is a common form of...
27    Heterozygous mutations in the telomerase compo...
28    Sequencing studies have identified many recurr...
29    Heterozygous mutations in the telomerase compo...
...                                                 ...
```

Limpiamos los datos de todo tipo de información de formateo, metadatos, contracciones, abreviaturas y todo el vocabulario superfluo. En Python hacemos esto definiendo unas sencillas funciones que utilizan expresiones regulares y la API de *BeautifulSoup*[7] para realizar esta tarea.

```
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)

def denoise_text(text):
    text = strip_html(text)
    text = remove_between_square_brackets(text)
```

[7] Github del proyecto y documentación técnica disponible en **https://pypi.org/project/beautifulsoup4/**

```
    return text

def replace_contractions(text):
    """Reemplazo las contracciones de cadena"""
    return contractions.fix(text)

def remove_non_ascii(words):
    """Elimino los caracteres non-ASCII de la lista de tokens""
"
    new_words = []
    for word in words:
        new_word = unicodedata.normalize('NFKD', word).enco
de('ascii', 'ignore').decode('utf-8', 'ignore')
        new_words.append(new_word)
    return new_words

def to_lowercase(words):
    """Convierto a minúscula"""
    new_words = []
    for word in words:
        new_word = word.lower()
        new_words.append(new_word)
    return new_words

def remove_punctuation(words):
    """Elimino signos de puntuación"""
    new_words = []
    for word in words:
        new_word = re.sub(r'[^\w\s]', '', word)
        if new_word != '':
            new_words.append(new_word)
    return new_words

def replace_numbers(words):
    """Reemplazo los números por su representación textual""
"
    p = inflect.engine()
    new_words = []
    for word in words:
        if word.isdigit():
            new_word = p.number_to_words(word)
            new_words.append(new_word)
        else:
            new_words.append(word)
    return new_words

def remove_stopwords(words):
    """Elimino las stop words"""
    new_words = []
    for word in words:
        if word not in stopwords.words('english'):
            new_words.append(word)
    return new_words

def stem_words(words):
    """Hago Stemming para quedarme con las raíces"""
    stemmer = LancasterStemmer()
    stems = []
    for word in words:
        stem = stemmer.stem(word)
        stems.append(stem)
    return stems

def lemmatize_verbs(words):
    """Lemmatize verbs"""
    lemmatizer = WordNetLemmatizer()
    lemmas = []
    for word in words:
        lemma = lemmatizer.lemmatize(word, pos='v')
        lemmas.append(lemma)
    return lemmas

def normalize(words):
    words = remove_non_ascii(words)
    words = to_lowercase(words)
    words = remove_punctuation(words)
    words = replace_numbers(words)
    words = remove_stopwords(words)
    return words
```

3.3 Reducción de la alta dimensionalidad.

Para convertir nuestros textos en vectores de enteros eliminamos todo el texto que no aporta información y después barremos cada documento buscando ocurrencias de entidades clínicas. Para realizar este tokenizado nos valemos del corpus *Biomedical entity Relation ONcology COrpus*, abreviado como BRONCO corpus (**Lee et al, 2017**) que contiene más de 400 variantes (mutaciones o deleciones) y sus relaciones con genes, enfermedades, y líneas celulares en el

contexto de la investigación de detección de cáncer y fármacos antitumorales. Las variantes y sus relaciones que se extrajeron manualmente de 108 artículos de texto completo. También se utilizaron los diccionarios ICD (Tyrer et al., 2011) y el corpus farmacológico DTC (siglas en inglés de *Drug Target Corpus*) propuesto por Roxana Danger en (Danger et al. 2010).

Procesando Dataset..

Fri Aug 02 03:19:33 CEST 2019: 50 lineas procesadas (1%).

Fri Aug 02 06:56:57 CEST 2019: 675 lineas procesadas (20%).

Fri Aug 02 08:12:49 CEST 2019: 1025 lineas procesadas (30%).

Fri Aug 02 15:01:23 CEST 2019: 2025 lineas procesadas (60%).

Fri Aug 02 21:44:28 CEST 2019: 2975 lineas procesadas (89%).

Sat Aug 03 00:26:30 CEST 2019: 3300 lineas procesadas (99%).

Normalización..

Lematización..

Tokenizado..

Identificación entidades (corpus bronco+ufal+icd)

Generación de muestras artificiales para balanceo clases

Done (miCorpus.ser).

Done (miDict.ser).

Done (training_unificado_tokens_corpus_filtered.csv).

Generando Corpus..

uterine cancer=16, Dacarbazine=10, doxorubicin=138, endoplasmic reticulum (ER)=46, chelating=51, blasts=1119, Mailing address=17, spreading=179, ibuprofen=12, carriers=1836, avoidance=144, Diet=42, fragile site=22, Selection=1122, goma=10, gold=132, venous=157, rare diseases=12, Funded=15, brachydactyly=14, Diabetes mellitus=16, truncated=1031, metformin=69, DNA fingerprint=11, stasis=1017, inhalation=13, straining=69, sufferer=15, oligodendrocyte=19, Obtain=20, Foot=18, rolo=1199, colorimetry=22, salting=12, root=1064, room=2520, H358=41, hemizygous=154, cluster membership=13, adenylate cyclase=38, et al=2351, Forb=67, antihistamine=10, lamellar=50, recurrence risk=15, flor=54, flow=2874, myelodysplastic syndromes=66, statistical methods=43, pellet=2340, pine=196, PET scan=156, pino=142, dramatic symptomatic improvement=20, differential count=20, chirality=19, cancer.=1951, reproducibility=173, Four=1174, Akt=1902, comorbidity=14, ECOG performance status=38, Estonia=10, carcinosarcoma=134, chemical reaction=12, T-cell count=18, seminal vesicles=12, gout=22, starting material=21, Intervention=14, chicken=924, cytosine arabinoside=11, childbirth=12, cholecystectomy=20, curation=69, muscle=894, flux=143, colonoscopy=12, quantitative=2556, dopaminergic=11, phosphatidylinositol-3,4,5-triphosphate=10, pito=2484, Y-linked=10, genetic screening=69, mitotic figure=69, B-RAF inhibitor=18, restraint=184, tonsils=145, effluent=12, covariates=129, lining of the uterus=14, chain-reaction=48, rib cage=14, celery=12, creation=150, non-fat=57, electrophoretic mobility=193, aureus=22, Grove=120, polymeric=10, Relapse=70, cell lines=2316, thigh=64, insulin-like growth factor 1 receptor=19, stability=1867, actual=2373, diaphragm=39, IGF-1=65, neurogenic=15, solubility=177, memory=126, transductant=21, revertant=23, trisomy 8=17, progesterone receptor=169, indefinite=68, Heterogeneity=120, progeny=80, nucleolus=39, surfactant=42, cardio-facio-cutaneous (CFC) syndrome=43, Histology=175, chronic lymphocytic leukaemia=11, Doxycycline=43, sex determination=10, funnel=13, preload=38, embryogenesis=185, vital stain=10, steroid=178, gamma knife=10, monochromatic=12, B-cell lymphomas=116, mental disorders=192, malignant effusions=16, eyebrows=14, detoxifying=22, cutis=69, anhidrosis=12, homeobox=123, fetal=2306, live birth=50, temperature-sensitive mutant=66, maxilla=22, Non-small-cell lung cancer=13, Concentration=123, Versus=18, performed=2875, Type=893, prevalent=2187, cytology=183, compact=150, possession=10, multivalent=17, ental=2622, neurofibromin=120, Gastric cancer=73, satellite cells=13, numerically=40, exploratory analysis=11, key features=51, methylation=1896, cloning site=62, once daily=130, intent-to-treat=10, Heterochromatin=12, DNA molecules=40, specimen=2184, Rodent=18, equal distribution=14, CD4=187, variable region=78, hepatocellular adenoma=15, CD8=123, Heart=46, contracted=20, regional control=13, sRNA=12,

primary cause=42, gene deletion=173, storage disease=13, Nutrient=13, kinesi=62, Liver metastases=18, capillaries=78, H441=14, endpoints=52, polymorphonuclear leukocyte=12, land development=20, subclinical=10, biologic activity=50, lymphopoiesis=15, therapeutics=908, new patient=13, mellar=52, S45A=12, Plasmodium=21, Penicillin=44, vesical=57, G13D=80, bone pain=11, anise=48, Status of=19, brown=127, vertigo=20, carnoso=17, Acute myeloid leukemia=56, correlated significantly=56, fascicles=12, standard deviation=1923, palate=44, laboratory test=78, pound=2312, cell pellet=181, Leydig cell tumor=40, carbonate=195, MCF7=168, translocation=1147, prolactin=12, length=2162, concerns=171, babe=10, intestinal=2397, DNA cloning=40, baby=128, gene translocation=52, acini=79, narrow therapeutic index=17, frayed=193, palmo=14, topology=73, Ptosis=14, being=2118, retinopathy=11, contents=183, optic nerve=45, tetrapeptide=11, contaminants=67, monoblast=16, vesicle=140, silencer=39, stream=2399, reperfusion=20, cyclosporine=18, papillary adenocarcinoma=14, COS=902, amphotericin B=13, COX=38, cultivation=18, Negative control=172, BCL2=170, basal layer=39, skin fibroblast=49, colitis=130, cecal=15, Metastases=41, Template=122, acetonitrile=59, bromo=882, trigeminal=10, baseline characteristics=38, phosphorus=44, bearing=1822, streak=74, deposits=40, dentation=20, chronic hypoxia=10, compass=981, CEBPA=70, paraplegia=12, energy transfer=47, bald=18, sebaceous=13, hypertelorism=48, spindle=1038, adenopathy=79, intensity=1019, somatic cells=69, anima=904, Microsatellite analysis=22, Enhancer=120, testicular cancer=11, predicate=183, deaminated=12, mama=111, Bullock=15, Argentina=12, lymphocytes=1820, mammography=60, Absence=77, platelets=150, Land=134, Estimate=61, latitude=42, examiner=12, cancer research=111, partition=114, childhood=806, gene targeting=42, microscope slides=40,

Obteniendo datos de salida en formato extendido..Done (training_unificado_tokens_corpus_filtered.csv).

Obteniendo datos de salida en formato binario..ID|Class|pito|Heart|contracted|Enhancer|uterine cancer|regional control|sRNA|primary cause|Y-linked|gene deletion|testicular cancer|Dacarbazine|predicate|genetic screening|deaminated|doxorubicin|endoplasmic reticulum (ER)|mama|chelating|mitotic figure|B-RAF inhibitor|blasts|storage disease|Bullock|Argentina|lymphocytes|mammography|Mailing address|spreading|ibuprofen|Absence|carriers|avoidance|platelets|restraint|Land|tonsils|Diet|effluent|covariates|lining of the uterus|Nutrient|kinesi|fragile site|Liver metastases|Estimate|Selection|capillaries|latitude|chain-reaction|H441|examiner|goma|cancer research|gold|venous|rare diseases|rib cage|Funded|celery|partition|brachydactyly|creation|non-fat|endpoints|childhood|electrophoretic mobility|Diabetes mellitus|gene targeting|microscope slides|truncated|metformin|aureus|DNA fingerprint|Grove|polymorphonuclear leukocyte|stasis|land development|inhalation|subclinical|polymeric|biologic activity|straining|lymphopoiesis|Relapse|therapeutics|new patient|cell lines|thigh|sufferer|mellar|S45A|Plasmodium|insulin-like growth factor 1 receptor|Penicillin|vesical|G13D|stability|actual|bone pain|anise|oligodendrocyte|diaphragm|Status of|IGF-1|neurogenic|brown|Obtain|Foot|rolo|vertigo|carnoso|solubility|Acute myeloid leukemia|correlated significantly|fascicles|standard deviation|palate|memory|transductant|revertant|trisomy 8|laboratory test|progesterone receptor|indefinite|colorimetry|Heterogeneity|pound|cell pellet|Leydig cell tumor|salting|root|progeny|nucleolus|carbonate|surfactant|cardio-facio-cutaneous (CFC) syndrome|MCF7|translocation|prolactin|length|Histology|concerns|babe|room|H358|intestinal|hemizygous|cluster membership|chronic lymphocytic leukaemia|adenylate cyclase|Doxycycline|et al|Forb|DNA cloning|baby|antihistamine|sex determination|gene translocation|lamellar|funnel|recurrence risk|acini|narrow therapeutic index|preload|embryogenesis|vital stain|flor|flow|steroid|frayed|palmo|gamma knife|topology|Ptosis|monochromatic|B-cell lymphomas|being|mental disorders|malignant effusions|retinopathy|contents|eyebrows|myelodysplastic syndromes|optic nerve|statistical methods|tetrapeptide|detoxifying|contaminants|monoblast|pellet|pine|PET scan|cutis|anhidrosis|vesicle|homeobox|fetal|pino|dramatic symptomatic improvement|silencer|live birth|stream|reperfusion|differential count|cyclosporine|papillary adenocarcinoma|chirality|temperature-sensitive mutant|cancer.|maxilla|reproducibility|COS|Four|Non-small-cell lung cancer|Akt|Concentration|Versus|amphotericin B|performed|COX|cultivation|Type|comorbidity|ECOG performance status|Negative control|BCL2|prevalent|basal layer|Estonia|carcinosarcoma|cytology|skin fibroblast|colitis|compact|possession|multivalent|cecal|Metastases|Template|ental|neurofibromin|acetonitrile|bromo|trigeminal|Gastric cancer|satellite cells|baseline characteristics|phosphorus|numerically|bearing|chemical reaction|exploratory analysis|streak|T-cell count|deposits|seminal vesicles|dentation|chronic hypoxia|key features|compass|methylation|gout|starting material|cloning site|Intervention|CEBPA|chicken|paraplegia|cytosine arabinoside|childbirth|energy transfer|once daily|intent-to-treat|Heterochromatin|cholecystectomy|DNA molecules|curation|muscle|specimen|flux|bald|sebaceous|Rodent|hypertelorism|colonoscopy|spindle|equal distribution|adenopathy|intensity|CD4|variable region|hepatocellular adenoma|somatic cells|quantitative|CD8|anima|dopaminergic|phosphatidylinositol-3,4,5-triphosphate|Microsatellite analysis

Done (training_final_T.csv).

Obteniendo clases en formato binario..ID|Class|pito|Heart|contracted|Enhancer|uterine cancer|regional control|sRNA|primary cause|Y-linked|gene deletion|testicular cancer|Dacarbazine|predicate|genetic screening|deaminated|doxorubicin|endoplasmic reticulum (ER)|mama|chelating|mitotic figure|B-RAF inhibitor|blasts|storage disease|Bullock|Argentina|lymphocytes|mammography|Mailing address|spreading|ibuprofen|Absence|carriers|avoidance|platelets|restraint|Land|tonsils|Diet|effluent|covariates|lining of the uterus|Nutrient|kinesi|fragile site|Liver metastases|Estimate|Selection|capillaries|latitude|chain-reaction|H441|examiner|goma|cancer research|gold|venous|rare diseases|rib cage|Funded|celery|partition|brachydactyly|creation|non-fat|endpoints|childhood|electrophoretic mobility|Diabetes mellitus|gene targeting|microscope slides|truncated|metformin|aureus|DNA fingerprint|Grove|polymorphonuclear leukocyte|stasis|land development|inhalation|subclinical|polymeric|biologic activity|straining|lymphopoiesis|Relapse|therapeutics|new patient|cell lines|thigh|sufferer|mellar|S45A|Plasmodium|insulin-like growth factor 1 receptor|Penicillin|vesical|G13D|stability|actual|bone pain|anise|oligodendrocyte|diaphragm|Status of|IGF-1|neurogenic|brown|Obtain|Foot|rolo|vertigo|carnoso|solubility|Acute myeloid leukemia|correlated significantly|fascicles|standard deviation|palate|memory|transductant|revertant|trisomy 8|laboratory test|progesterone receptor|indefinite|colorimetry|Heterogeneity|pound|cell pellet|Leydig cell tumor|salting|root|progeny|nucleolus|carbonate|surfactant|cardio-facio-cutaneous (CFC) syndrome|MCF7|translocation|prolactin|length|Histology|concerns|babe|room|H358|intestinal|hemizygous|cluster membership|chronic lymphocytic leukaemia|adenylate cyclase|Doxycycline|et al|Forb|DNA cloning|baby|antihistamine|sex determination|gene translocation|lamellar|funnel|recurrence risk|acini|narrow therapeutic index|preload|embryogenesis|vital stain|flor|flow|steroid|frayed|palmo|gamma knife|topology|Ptosis|monochromatic|B-cell lymphomas|being|mental disorders|malignant effusions|retinopathy|contents|eyebrows|myelodysplastic syndromes|optic nerve|statistical methods|tetrapeptide|detoxifying|contaminants|monoblast|pellet|pine|PET scan|cutis|anhidrosis|vesicle|homeobox|fetal|pino|dramatic symptomatic improvement|silencer|live birth|stream|reperfusion|differential count|cyclosporine|papillary adenocarcinoma|chirality|temperature-sensitive mutant|cancer.|maxilla|reproducibility|COS|Four|Non-small-cell lung cancer|Akt|Concentration|Versus|amphotericin B|performed|COX|cultivation|Type|comorbidity|ECOG performance status|Negative control|BCL2|prevalent|basal layer|Estonia|carcinosarcoma|cytology|skin fibroblast|colitis|compact|possession|multivalent|cecal|Metastases|Template|ental|neurofibromin|acetonitrile|bromo|trigeminal|Gastric cancer|satellite cells|baseline characteristics|phosphorus|numerically|bearing|chemical reaction|exploratory analysis|streak|T-cell count|deposits|seminal vesicles|dentation|chronic hypoxia|key features|compass|methylation|gout|starting material|cloning site|Intervention|CEBPA|chicken|paraplegia|cytosine arabinoside|childbirth|energy transfer|once daily|intent-to-treat|Heterochromatin|cholecystectomy|DNA molecules|curation|muscle|specimen|flux|bald|sebaceous|Rodent|hypertelorism|colonoscopy|spindle|equal distribution|adenopathy|intensity|CD4|variable region|hepatocellular adenoma|somatic cells|quantitative|CD8|anima|dopaminergic|phosphatidylinositol-3,4,5-triphosphate|Microsatellite analysis

Done (training_final_c.csv).

Input Sample:

0|FAM58A|Truncating Mutations|1|Cyclin-dependent kinases (CDKs) regulate a variety of fundamental cellular processes. CDK10 stands out as one of the last orphan CDKs for which no activating cyclin has been identified and no kinase activity revealed. Previous work has shown that CDK10 silencing increases ETS2 (v-ets erythroblastosis virus E26 oncogene homolog 2)-driven activation of the MAPK pathway, which confers tamoxifen resistance to breast cancer cells. The precise mechanisms by which CDK10 modulates ETS2 activity, ...

Output Sample:

FAM58A;Truncating Mutations;breast cancer;RAF;MAP;ERK;Thr;Lys;ERK1;Raf;MAPK;ATP;MCF7;ERK2;ETS2;MYC;AT;phospho;AR;Pro;MA;p21;serine;MYCN;MG132;IM;CDK;aberrant;abnormalities;acceptor;receptor;acceptor site;alternative splicing;anomaly;antibodies;autosomal;binding;blasto;blastoma;breast;buffer;cDNA;cervix;colon;cyclin;daughter;degeneration;degradation;degraded;delete;deleterious;deletion;denatured;de novo;distant tumor;syndrome;edges;electroporation;embryogenesis;endogenous;terminal;enriched;epilepsy;Escherichia coli;estrogen;receptor;female;fetal;mutation;gene;gene silencing;genesis;gene splicing;genome;glutathione;glycerol;growth;heart;hematic;hence;heparin;heterozygous;histone;homogeneous;homolog;homologous;homology;orizon;horizontal;hormone;hormonal therapy;human development;hybridization;immunoprecipitation;infect;infected;infection;in vitro;inhibitor;insertion;intellect;interaction;intron;isoform;kidney;kinase;knockdown;lapping;ligase;ligases;luminescence;lymph;lymphoblast;lysate;lysis;malformation;malformations;Mammal;mass spectrometry;migration;monomeric;monomer;mother;moth;mount;mRNA;mutant;mutated allele;Mutation;nasal;renal;nephroblastoma;nucleotide;nucleus;numb;oligonucleotide;oncogene;oncogen;overlapping

;overlap;paralogs;partner;paternity;Paternity;path;pathogen;pathogenesis;pathwa;pathway;PCR.;phenotype;phenyl;phosphate;phosphorylation;photoreceptor;plasmid;point mutation;Polyp;premature;pressor;proliferation;proline;protease;protein kinase;pseudogene;radioactivity;radiography;recombinant;regulation;regulatory;reporter gene;resistance;retardation;retina;retrotransposon;RNA expression;RNAi;rounds;round;RT-PCR;sequencing;severe;short-lived;siRNA;site;spectrometry;spectrum;splicing;stability;stage;staining;sting;stoma;stop codon;supernatant;syringomyelia;tamoxifen;target;tent;teres;tissue;total protein;transcription factor;transcription;transcript;transfection;transgenic;transposon;trophic;truncated;ubiquitin;urine;urogenital;uterus;virus;wild-type;X chromosome;X chromosome inactivation;X inactivation;X-linked dominant;X-linked;X-ray;yeast;

Mapeando entidades..

uterine cancer=16, Dacarbazine=10, doxorubicin=138, endoplasmic reticulum ER=46, chelating=51, blasts=1119, Mailing address=17, spreading=179, ibuprofen=12, carriers=1836, avoidance=144, Diet=42, fragile site=22, Selection=1122, goma=10, gold=132, venous=157, rare diseases=12, Funded=15, brachydactyly=14, Diabetes mellitus=16, truncated=1031, metformin=69, DNA fingerprint=11, stasis=1017, inhalation=13, straining=69, sufferer=15, oligodendrocyte=19, Obtain=20, Foot=18, rolo=1199, colorimetry=22, salting=12, root=1064, room=2520, H358=41, hemizygous=154, cluster membership=13, adenylate cyclase=38, et al=2351, Forb=67, antihistamine=10, lamellar=50, recurrence risk=15, flor=54, flow=2874, myelodysplastic syndromes=66, statistical methods=43, pellet=2340, pine=196, PET scan=156, pino=142, dramatic symptomatic improvement=20, differential count=20, chirality=19, cancer.=1951, reproducibility=173, Four=1174, Akt=1902, comorbidity=14, ECOG performance status=38, Estonia=10, carcinosarcoma=134, chemical reaction=12, T-cell count=18, seminal vesicles=12, gout=22, starting material=21, Intervention=14, chicken=924, cytosine arabinoside=11, childbirth=12, cholecystectomy=20, curation=69, muscle=894, flux=143, colonoscopy=12, quantitative=2556, dopaminergic=11, phosphatidylinositol-3,4,5-triphosphate=10, pito=2484, Y-linked=10, genetic screening=69, mitotic figure=69, B-RAF inhibitor=18, restraint=184, tonsils=145, effluent=12, covariates=129, lining of the uterus=14, chain-reaction=48, rib cage=14, celery=12, creation=150, non-fat=57, electrophoretic mobility=193, aureus=22, Grove=120, polymeric=10, Relapse=70, cell lines=2316, thigh=64, insulin-like growth factor 1 receptor=19, stability=1867, actual=2373, diaphragm=39, IGF-1=65, neurogenic=15, solubility=177, memory=126, transductant=21, revertant=23, trisomy 8=17, progesterone receptor=169, indefinite=68, Heterogeneity=120, progeny=80, nucleolus=39, surfactant=42, cardio-facio-cutaneous (CFC) syndrome=43, Histology=175, chronic lymphocytic leukaemia=11, Doxycycline=43, sex determination=10, funnel=13, preload=38, embryogenesis=185, vital stain=10, steroid=178, gamma knife=10, monochromatic=12, B-cell lymphomas=116, mental disorders=192, malignant effusions=16, eyebrows=14, detoxifying=22, cutis=69, anhidrosis=12, homeobox=123, fetal=2306, live birth=50, temperature-sensitive mutant=66, maxilla=22, Non-small-cell lung cancer=13, Concentration=123, Versus=18, performed=2875, Type=893, prevalent=2187, cytology=183, compact=150, possession=10, multivalent=17, ental=2622, neurofibromin=120, Gastric cancer=73, satellite cells=13, numerically=40, exploratory analysis=11, key features=51, methylation=1896, cloning site=62, once daily=130, intent-to-treat=10, Heterochromatin=12, DNA molecules=40, specimen=2184, Rodent=18, equal distribution=14, CD4=187, variable region=78, hepatocellular adenoma=15, CD8=123, Heart=46, contracted=20, regional control=13, sRNA=12, primary cause=42, gene deletion=173, storage disease=13, Nutrient=13, kinesi=62, Liver metastases=18, capillaries=78, H441=14, endpoints=52, polymorphonuclear leukocyte=12, land development=20, subclinical=10, biologic activity=50, lymphopoiesis=15, therapeutics=908, new patient=13, mellar=52, S45A=12, Plasmodium=21, Penicillin=44, vesical=57, G13D=80, bone pain=11, anise=48, Status of=19, brown=127, vertigo=20, carnoso=17, Acute myeloid leukemia=56, correlated significantly=56, fascicles=12, standard deviation=1923, palate=44, laboratory test=78, pound=2312, cell pellet=181, Leydig cell tumor=40, carbonate=195, MCF7=168, translocation=1147, prolactin=12, length=2162, concerns=171, babe=10, intestinal=2397, DNA cloning=40, baby=128, gene translocation=52, acini=79, narrow therapeutic index=17, frayed=193, palmo=14, topology=73, Ptosis=14, being=2118, retinopathy=11, contents=183, optic nerve=45, tetrapeptide=11, contaminants=67, monoblast=16, vesicle=140, silencer=39, stream=2399, reperfusion=20, cyclosporine=18, papillary adenocarcinoma=14, COS=902, amphotericin B=13, COX=38, cultivation=18, Negative control=172, BCL2=170, basal layer=39, skin fibroblast=49, colitis=130, cecal=15, Metastases=41, Template=122, acetonitrile=59, bromo=882, trigeminal=10, baseline characteristics=38, phosphorus=44, bearing=1822, streak=74, deposits=40, dentation=20, chronic hypoxia=10, compass=981, CEBPA=70, paraplegia=12, energy transfer=47, bald=18, sebaceous=13, hypertelorism=48, spindle=1038, adenopathy=79, intensity=1019, somatic cells=69, anima=904, Microsatellite analysis=22, Enhancer=120, testicular cancer=11, predicate=183, deaminated=12, mama=111, Bullock=15, Argentina=12, lymphocytes=1820, mammography=60, Absence=77, platelets=150, Land=134, Estimate=61, latitude=42, examiner=12, cancer research=111, partition=114, childhood=806, gene targeting=42, microscope slides=40, thought to occur=66,

Encontradas 861 entidades únicas.

Obteniendo datos de salida en formato extendido..Done (training_unificado_tokens_corpus_filtered.csv).

Obteniendo datos de entrenamiento en formato binario..Done (training_final_T.csv).

Obteniendo clases en formato binario..Done (training_final_c.csv).

Obteniendo datos de entrenamiento en formato binario..Done

Terminada generación ok.

Ahora utilizaremos el algoritmo *Hyperplot* en su implementación Python (**Heusser et al. 2017**) para crear distintas proyecciones en tres dimensiones que si bien se ven muy limitados por la enorme diferencia de dimensionalidad si nos ayudan a visualizar las distintas clases y su separabilidad en el número de clústeres deseado como puede apreciarse en la figura 7.

```
geo = hyp.plot(train, '.', n_clusters=9, legend=list(set(performance)))
```

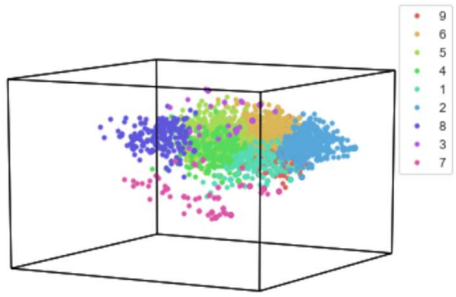

Figura 7. Representación en tres dimensiones del espacio de búsqueda.

Hypertools utiliza PCA[8] como método de reducción de dimensionalidad por defecto, pero existen otras técnicas más efectivas que además permiten conservar la coherencia de los clusters originales. Utilizamos el algoritmo *T-distributed Stochastic Neighbor Embedding* (**Maaten & Hinton, 2010**), abreviado como t-SNE, un algoritmo que funciona muy bien en entornos de alta dimensionalidad y que es muy utilizado en otras aplicaciones computacionales como las ciencias físicas o la bioinformática. El resultado en este caso es mucho mejor como se aprecia en la siguiente figura.

```
geo_tsne = hyp.plot(train, '.', group=performance, legend=list(set(performance)), reduce='TSNE', ndims=3)
```

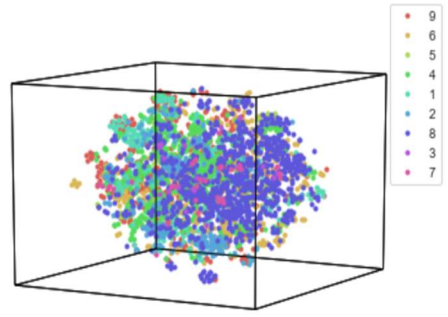

Figura 8. Representación en tres dimensiones utilizando la técnica t-SNE.

Ambos métodos obtienen proyecciones muy limitadas, pero dejan intuir cierta coherencia en las agrupaciones creadas. Observemos ahora la separación por clases, donde encontramos un serio desbalanceo en los datos lo que podría afectar a la precisión de nuestro clasificador. Todas las columnas se importaron inicialmente como cadenas. Tendremos que convertir los valores

[8] Método de análisis de componentes principales, de ahí sus siglas en inglés PCA.

realmente numéricos en tipos de datos numéricos, para que luego podamos manipular fácilmente los marcos de datos. También declaramos algunas funciones auxiliares.

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import log_loss
from sklearn.metrics import precision_score, recall_score, roc_auc_score
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from boruta import BorutaPy

# Funciones auxiliares
def zero_one_score(y, pred_y):
    total_errors =  np.sum( np.abs(y - pred_y) )
    return total_errors / len(y)
def run_predict_logistic_regression(X_train,Y_train,X_test,Y_test):
    clf = LogisticRegression()
    clf = clf.fit(X_train, Y_train)
    pred = clf.predict(X_test)
    print('Logistic 0-1 error. \n Training: ', zero_one_score(Y_train, clf.pre
dict(X_train)),
          '\n Test:', zero_one_score(Y_test, pred))

    return clf
def show_im(im):
    plt.imshow(im, cmap='gray')
    plt.show()

# Clasificador
def run_forests():
    print('random forest: \n')
    params = []
    scores = []

    for _ in range(5):
        max_features = np.random.randint(400,800)
        max_depth = np.random.choice([None, None, None, None, 30, 40, 60])
        forest = RandomForestClassifier(n_estimators=50,
                                        max_features=max_features,
                                        max_depth=max_depth)
        forest_fit = forest.fit(X_train, Y_train)
        pred = forest_fit.predict(X_test)
        print('\n pars:',dict(max_features=max_features,max_depth=max_depth))
        print('forest train:',zero_one_score(Y_train,forest_fit.predict(X_trai
n)), ' test: ',
                    zero_one_score(Y_test, pred))

        params.append( (max_features, max_depth) )
        scores.append( zero_one_score(Y_test, pred))

    print('best:', params[np.argmin(scores)])

class SKClassifier(object):
    def __init__(self, clf):
        self.clf = clf
        self.threshold = 0.0
        #tf.add_to_collection("threshold", self.threshold)

    def fit(self,X_train,y_train):
        self.clf = self.clf.fit(X_train,y_train)
        return self.clf

    def predict_proba(self,X):
        return self.clf.predict_proba(X)

    def predict_proba_with_loss(self, X, y):
```

```
        y_pred = self.predict_proba(X)
        loss = log_loss(y,y_pred)
        return y_pred, loss

    # smallest prob given to an actual catastrophe
    def threshold_from_data(self, X, y):
        y_bool = y == 1.   ## true if x is a catast
        y_pred = self.predict_proba(X)
        if np.count_nonzero(y) == 0:
            return np.max(y_pred)
        return np.min(y_pred[y_bool][:,1])    # TODO CHANGED FROM WILL CODE

    def metrics(self, X, y):
        metrics = {}
        y_pred_pair, loss = self.predict_proba_with_loss(X, y)
        y_pred = y_pred_pair[:,1]  ## From softmax pair to prob of catastrophe

        metrics['loss'] = loss
        threshold = self.threshold_from_data(X, y)
        metrics['threshold'] = threshold
        metrics['np.std(y_pred)'] = np.std(y_pred)
        denom = np.count_nonzero(y == False)
        num = np.count_nonzero(np.logical_and(y == False, y_pred >= threshold)
)
        metrics['fpr'] = float(num) / float(denom)

        if any(y) and not all(y):
            metrics['auc'] = roc_auc_score(y, y_pred)
            y_pred_bool = y_pred >= threshold
            if (any(y_pred_bool) and not all(y_pred_bool)):
                metrics['precision'] = precision_score(np.array(y, dtype=np.fl
oat32), y_pred_bool)
                metrics['recall'] = recall_score(y, y_pred_bool)

        return metrics

    def get_gaps(self,X,y):
        return np.abs(y - self.predict_proba(X)[:,1])

    def show_mistakes(self,X,y,k):
        gaps = self.get_gaps(X,y)
        index_gap = np.array([[i,gap] for i,gap in enumerate(gaps)])
        sorted_index_gap = index_gap[index_gap[:,1].argsort()]
        tail_index_gap = sorted_index_gap[-k:,:]

        xs = []
        probs = []
        for row in range(k):
            index,gap = tail_index_gap[row,:]
            x = X[int(index),:]
            xs.append(x)
            show_im(x.reshape(40,40))
            print('Contains:',y[int(index)] == 1.,'.Prob gap:',gap,' \n ')
        return xs
```

Debido a la alta dimensionalidad del problema realizaremos una selección de atributos para eliminar aquellos irrelevantes o menos representativos. Utilizaremos para ello la implementación para Python del algoritmo de Boruta (Kursa et al., 2010), un método basado en *Random Forest* para la selección de atributos en muestras de muy alta dimensionalidad.

```
BorutaPy(alpha=0.05,
    estimator=RandomForestClassifier(bootstrap=True, class_weight='balanced',
```

```
    criterion='gini', max_depth=5, max_features='auto',
    max_leaf_nodes=None, min_impurity_decrease=0.0,
    min_impurity_split=None, min_samples_leaf=1,
    min_samples_split=2, min_weight_fraction_leaf=0.0,
    n_estimators=384, n_jobs=-1, oob_score=False,
    random_state=<mtrand.RandomState object at 0x000001D84F4C6CF0>,
    verbose=0, warm_start=False),
    max_iter=100, n_estimators='auto', perc=100,
    random_state=<mtrand.RandomState object at 0x000001D84F4C6CF0>,
    two_step=True, verbose=2)

Iteration:  1 / 100
Confirmed:  0
Tentative:  861
Rejected:  0
Iteration:  25 / 100
Confirmed:  152
Tentative:  59
Rejected:  650
Iteration:  50 / 100
Confirmed:  159
BorutaPy finished running.
Iteration:  100 / 100
Confirmed:  163
Tentative:  10
Rejected:  676
Tentative:  39
Rejected:  663
Iteration:  75 / 100
Confirmed:  161
Tentative:  28
Rejected:  672
Iteration:  99 / 100
Confirmed:  163
Tentative:  22
Rejected:  676
```

3.4 Entrenando distintos modelos semánticos

Vamos a entrenar diferentes modelos semánticos y lo hacemos sobre un corpus conformado a partir del bronco corpus, el corpus farmacológico de (**R.Danger, 2010**) y una selección de los 500 términos con mayor representación en las muestras de entrenamiento, que se obtiene de forma proporcional o su prevalencia en cada una de las clases de estudio donde con anterioridad hemos realizado diferentes filtrados y limpieza de términos y se han resuelto sinónimos, abreviaturas e información de formateo. En total manejaremos un corpus de 3 millones de términos.

```
#model = gensim.models.Word2Vec (documents, size=500, window=8, min_count=20, workers=15)
#model.train(documents,total_examples=len(documents),epochs=10)
model = gensim.models.FastText(documents, size=250, window=5, min_count=25, workers=15)
model.train(documents,total_examples=len(documents),epochs=10)
```

```
2020-03-20 19:33:00,648 : INFO : collecting all words and their counts
2020-03-20 19:33:00,650 : INFO : PROGRESS: at sentence #0, processed 0 words, keeping 0 word types
2020-03-20 19:33:05,893 : INFO : collected 81541 word types from a corpus of 29476562 raw words and 3322 sentences
2020-03-20 19:33:05,894 : INFO : Loading a fresh vocabulary
2020-03-20 19:33:06,419 : INFO : effective_min_count=20 retains 21623 unique words (26% of original 81541, drops 59918)
2020-03-20 19:33:06,420 : INFO : effective_min_count=20 leaves 29234285 word corpus (99% of original 29476562, drops 242277)
```

Para testar el rendimiento del modelo le pediremos que obtenga términos cercanos a algunos términos clínicos para comprobar si devuelve resultados coherentes con lo esperado. Primero pedimos hasta 10 términos similares a "tumoration" obteniendo algunos términos que efectivamente podrían considerarse sinónimos como "tumor", "tumores", o "cáncer" junto a otros para los que el modelo extrapola cierto grado de similitud semántica, en este caso "metástasis", "glioma", "neoplasia", "lesión" y "melanoma". Estos resultados son coherentes y se acercan bastante a los que podría esperarse de un diccionario terminológico confeccionado manualmente.

```
w1 = "tumoration"
model.wv.most_similar (positive=w1)
```

```
2020-03-20 19:42:47,923 : INFO : precomputing L2-norms of word weight vectors
/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of the second argument
  if np.issubdtype(vec.dtype, np.int):
[('tumour', 0.6785714626312256),
 ('tumors', 0.4661846160888672),
 ('cancer', 0.42841821908950806),
 ('metastasis', 0.3352215588092804),
 ('glioma', 0.3339627683162689),
 ('neoplastic', 0.3214406967163086),
 ('lesion', 0.31742921471595764),
 ('melanoma', 0.3150254487991333),
 ('tumours', 0.30131208896636963),
 ('polyp', 0.2982310652732849)]
```

A continuación pedimos al modelo que nos de 6 términos similares al oncogen "kit" y obtenemos las salidas "pdgfra", "braf", "egfr" con similitud mayor al 30%, lo cual es coherente siendo todos ellos oncogenes estrechamente relacionados con las tirosina-quinasas y los términos "kdr" (abreviatura de kinase dependent receptor, o receptor dependiente de la quinasa en español), "imatinib" un fármaco oncológico que tiene como diana terapéutica c-kit y "kitssr" (abreviatura de kit signal receptor) con una similitud alrededor del 25%. Todos estos resultados parecen coherentes y validar el modelo.

```
# look up top 6 terms similar to 'kit'
w1 = ["kit"]
model.wv.most_similar (positive=w1,topn=6)
```

```
/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of the second argument
  if np.issubdtype(vec.dtype, np.int):
[('pdgfra', 0.37065818905830383),
 ('braf', 0.35995805263519287),
 ('egfr', 0.3328053057193756),
 ('kdr', 0.26850762963294983),
 ('imatinib', 0.25712716579437256),
 ('kits', 0.2557176947593689)]
```

Por último, pediremos al modelo que nos devuelva fármacos con principio activo similar a "imatinib". Esta prueba es importante ya que es la primera vez que pondremos a prueba si el modelo es capaz de recoger la semántica que relaciona un fármaco con una patología.

```
# look up top 6 drugs similar to "imatinib"
w1 = ["imatinib"]
model.wv.most_similar (positive=w1,topn=6)
```

```
/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of the second argument
  if np.issubdtype(vec.dtype, np.int):
[('drug', 0.4683585464954376),
 ('crizotinib', 0.4495617747306824),
 ('tki', 0.44542059302330017),
 ('ibrutinib', 0.43829792737960815),
 ('gefitinib', 0.4154832661151886),
 ('dasatinib', 0.3916347026824951)]
```

Nuevamente obtenemos resultados coherentes del modelo que nos indica que “imatinib” es una droga y guarda alta similitud con “crizotinib”, “ibrutinib”, “gefitinib” y “desatinib” todos ellos fármacos inhibidores de vías de señalización dependientes de tirosina-quinasa y con espectros terapéuticos similares.

```
# get everything related to w1 but not near w2
w1 = ["cancer",'kit','vegf']
w2 = ['breast']
model.wv.most_similar (positive=w1,negative=w2,topn=10)
```

```
/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of the second argument
  if np.issubdtype(vec.dtype, np.int):
[('vegfr', 0.32750403881073),
 ('cancers', 0.31061044335365295),
 ('kdr', 0.27498650550842285),
 ('pdgfra', 0.26986145973205566),
 ('carcinoma', 0.2614326775074005),
 ('carcinogenesis', 0.259184330701828),
 ('carcinomas', 0.25440338253974915),
 ('adenocarcinoma', 0.2533445358276367),
 ('egfr', 0.2477392703294754),
 ('erk', 0.2406499683856964)]
```

CAPÍTULO IV

PROPUESTA E IMPLEMENTACIÓN.

4. Propuesta de implementación

Hemos desarrollado un modelo neuronal profundo híbrido para la clasificación automática de textos biomédicos. El nivel superior de nuestra arquitectura consta de tres etapas: convolucional, recursiva y densa. La Figura 12 muestra la interfaz detallada de nuestra red donde puede ver las tres etapas descritas y sus diferentes capas. Como entrada, la red recibe un documento clínico, que pasa a través de una serie de capas convolucionales, cada una entrenada en un modelo Fasttext (**Yonghui, et al., 2016**) especializado en un tipo de entrada. Varios estudios recientes (ver capítulos anteriores) demuestran que el éxito del aprendizaje profundo para la clasificación de textos depende en gran medida de la efectividad de las incrustaciones de palabras. Usamos el modelo Stanford Glove (**Pennington et al. 2014**) entrenado para inglés americano y dos modelos FastText de 500 dimensiones entrenados en un corpus seleccionado personalizado, esa es la razón por la que aplicamos ventanas de diferente longitud y varias matrices de peso en operaciones convolucionales. La Tabla 2 muestra la pila de capas y los parámetros de nuestro modelo y consta de tres columnas: la primera indica el tipo de capa (entrada, embedding, GRU, concatenación, atención, submuestreo o densidad), la segunda se refiere al tamaño de la capa tensor de salida, y el último indica la capa inmediatamente anterior, es decir, la que está directamente conectada a la capa. Tenga en cuenta que la forma correcta de leer la tabla es de arriba a abajo para que las capas superiores correspondan a las entradas de datos. La arquitectura de aprendizaje profundo de nuestro modelo aprovecha las características locales codificadas extraídas del modelo CNN y las dependencias a largo plazo capturadas por el modelo RNN que implementa mecanismos de atención de contexto (contexto atencional).

Tabla 2. Estructura de capas y parametrización del modelo.

Layer (type)	Output Shape	Connected to
input_1 (InputLayer)	(None, 3000)	
input_2 (InputLayer)	(None, 3000)	
embedding_1 (Embedding)	(None, 3000, 128)	input_1[0][0]
embedding_2 (Embedding)	(None, 3000, 128)	input_2[0][0]
input_3 (InputLayer)	(None, 401)	
input_4 (InputLayer)	(None, 341)	
bidirectional_1 (Bidirectional)	(None, 3000, 392)	embedding_1[0][0]
bidirectional_2 (Bidirectional)	(None, 3000, 392)	embedding_2[0][0]
concatenate_1 (Concatenate)	(None, 742)	input_3[0][0] input_4[0][0]
attention_with_context_1 (Attent	(None, 392)	bidirectional_1[0][0]
attention_with_context_2 (Attent	(None, 392)	bidirectional_2[0][0]
dense_1 (Dense)	(None, 32)	concatenate_1[0][0]
concatenate_2 (Concatenate)	(None, 816)	attention_with_context_1[0][0] attention_with_context_2[0][0] dense_1[0][0]
dense_2 (Dense)	(None, 9)	concatenate_2[0][0]

Total params: 3,664,425

Nuestra propuesta de modelo integra cuatro conceptos principales que explicaremos en detalle en las siguientes subsecciones.

4.1 Etapa Convolucional multicanal

Utilizamos tres capas de embedding diferentes en una arquitectura de incrustación multicanal. En lugar de concatenar tres embedding parciales de cada token en un vector de palabras, mantenemos tres canales de incrustación independientes para ellos. Los canales para las relaciones en la palabra vector son incrustaciones idénticas. El uso de varias embedding es perfectamente posible y puede mejorar considerablemente la calidad de las convoluciones (véase figura 9). Como resultado, los términos se incrustan en R^{nxdxc}, donde n es el número de todos los tokens y las relaciones de dependencia entre ellos, d es el número de dimensión de incrustaciones y $c = 3$ es el número de canales de incrustación.

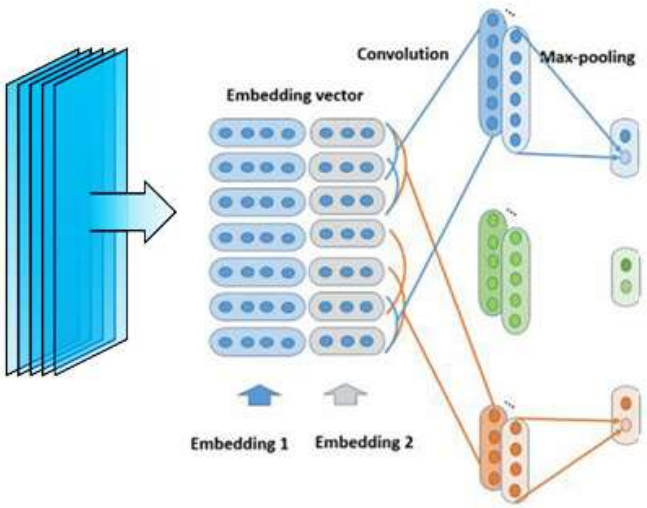

Figura 9. Fragmento de red CNN haciendo uso de varios embedding.

Como se puede ver en la Figura 9, en nuestro modelo las capas de embedding se procesan de forma independiente (cada capa ha sido previamente entrenada previamente por separado) y los filtros sucesivos se aplican una vez por canal como parte de la entrada de la etapa convolucional. Para calcular los mapas de características de CNN seguimos el esquema de **Kim et al., 2014**. El filtro f_i de cada CNN se desliza a lo largo de cada canal de incrustación (c) de forma independiente, creando un mapa de características correspondiente $f_{m\_i\_c}$. Luego se aplica un operador de agrupación máxima en los mapas de características creados en todos los canales (tres en nuestro caso) para crear un valor de característica para el filtro f_i. En el modelo CNN, se aplican diferentes matrices de peso para generar múltiples mapas de características. Como ejemplo, la figura 10 muestra los 100 términos más relevantes para uno de los grupos de estudio. Esto da una idea de la riqueza semántica que queremos mapear. Después de las operaciones de convolución y agrupación, el mapa de características codificadas se utilizará como entrada del modelo RNN. Las dependencias a largo plazo aprendidas por RNN podrían ser como una representación a nivel de oración. La representación a nivel de oración se lleva a la red totalmente conectada, y la salida softmax genera los resultados de la clasificación.

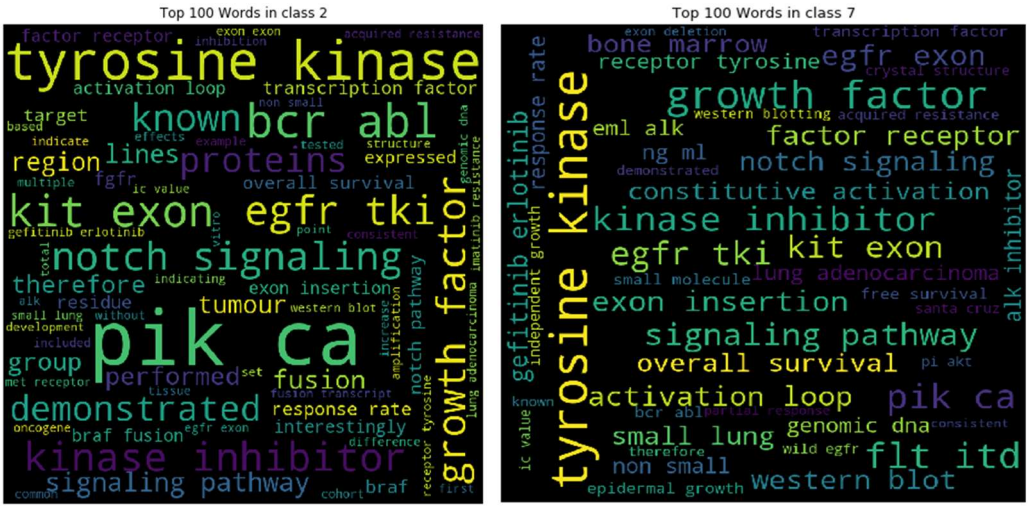

Figura 10. Algunos de los términos más relevantes para dos de los mayores clústeres.

4.2 Entrenamiento de modelos Fasttext a medida.

Los métodos tradicionales de Word embedding tienen algunas limitaciones importantes, que incluyen:

(i) La complejidad del método aumenta con el tamaño del vocabulario, que requiere técnicas de muestreo avanzadas.

(ii) Necesitan un conocimiento preciso del corpus al que se enfoca el método y obtener previamente un muestreo suficientemente representativo y completo.

(iii) Los métodos de inclusión tradicionales aprenden un vector de inserción para cada palabra de la misma palabra en dos contextos diferentes. Por ejemplo, el término "corazón" en "Te llevo en mi corazón" y "Tengo problemas de corazón") no tienen un vector de inserción diferente, aunque deberían tenerlo porque el contexto semántico es totalmente diferente.

(iv) Las representaciones de cada término deben ser formadas independientemente y cada palabra nueva se agrega como un elemento adicional y se antepone a un modelo específico. Esto significa que estos modelos son estáticos y poco efectivos si ocurren cambios en el uso o la gramática del lenguaje o cuando aparecen términos nuevos o desconocidos.

El modelo FastText intenta resolver las limitaciones de métodos como Word2vec y GloVe. Específicamente, fue diseñado para manejar nuevos términos fuera del vocabulario (OOV) al extender el modelo de Word2vec con información interna de subpalabras, en forma de n-gramos de caracteres (por ejemplo, secuencias de caracteres adyacentes). El método construye una representación vectorial para una palabra basada en la composición de estos componentes de subpalabras, que permiten que el modelo represente mejor la morfología y la similitud léxica de las palabras. Internamente, FastText utiliza modelos de mezcla de Gauss internamente (GMM) para que cada palabra se represente como un GMM de k componentes, que representan k diferentes sentidos de una palabra. Esta representación puede capturar la estructura de subpalabras

y detectar diferentes usos semánticos y proporcionar una mejor representación de palabras raras o invisibles.

Para un término *w*, el vector medio que representa el grupo se calcula como la media de los vectores de subpalabras y su entrada de diccionario como se expresa en la ecuación 1, donde z_g es el vector parcial *g* y NG_w es el conjunto de vectores vecinos.

$$\mu_w = \frac{1}{|NG_w|+1} \cdot \left(v_w + \sum_{g \in NG_w} z_g \right) \qquad (1)$$

Este método es más eficiente y rápido, lo que permite un mejor aprendizaje y capacitación de un corpus más grande. Además, permite compartir las representaciones entre palabras y hacer representaciones más consistentes para las palabras que ocurren con poca frecuencia y que otros modelos de inclusión de palabras descartarían como ruido.

4.3 Resolución distante por una red bidireccional-RNN.

Las características generadas por las operaciones de convolución y agrupación pueden considerarse características de alto nivel. Dado que las redes neuronales recurrentes (RNN) pueden procesar entradas secuenciales y comprender dependencias a largo plazo, utilizamos estas funciones como entradas para la etapa recursiva. Para combinar palabras y su contexto, primero usamos una red RNN bidireccional para procesar palabras de la siguiente manera. Una palabra en una oración se puede estudiar hacia adelante y hacia atrás dos veces en una estructura de dos vías, de modo que se puedan obtener más características de representación de palabras, evitando el problema de que algunos gradientes desaparezcan en oraciones largas. Las memorias LSTM están diseñadas para resolver el problema de desaparición de gradiente RNN, especialmente el aprendizaje de frases largas y combina las puertas de olvido y entrada en una única puerta de actualización. El modelo resultante es más simple que los modelos LSTM estándar y se ha vuelto cada vez más popular. El GRU también puede considerarse como una variación más simple de la red LSTM y, en algunos casos, ambos producen resultados igualmente excelentes. Aunque existe una similitud significativa en la estructura diseñada con el fin de resolver problemas de gradiente de fuga, existen varias diferencias. Aprovechando su celda más simple, GRU puede obtener un rendimiento cercano al LSTM que necesita menos tiempo.

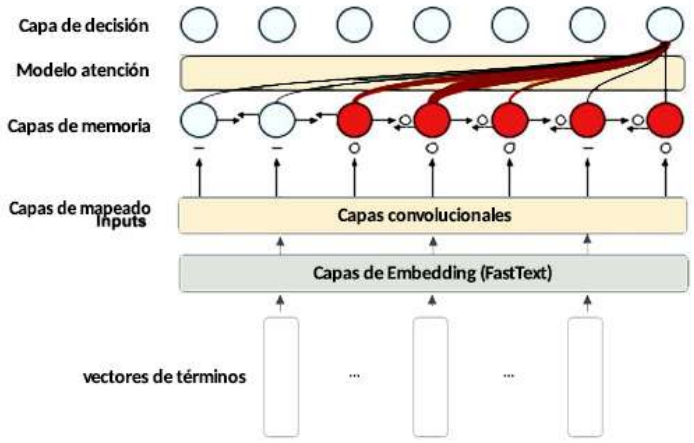

Figura 11. El mecanismo de atención modula el funcionamiento de la capa recursiva. Las celdas rojas denotan prioridades de atención.

4.4 Implementando un mecanismo de atención simple.

En cualquier documento técnico, pero aún más en un documento médico, su clasificación se construye a partir de información parcial basada en los elementos más básicos. El contexto local de un término influye en su importancia en el párrafo y el contexto global se basa en la utilidad de cada uno de estos términos en la representación completa del texto. Por esta razón, no solo es importante prestar atención a la apariencia de la terminología individual sino también a ciertas secuencias o entidades más complejas cuyas ocurrencias pueden afectar la clasificación. Para nuestro estudio estamos adoptando mecanismo de atención de (Yang et al., 2016) porque permite detectar dependencias directas entre los estados del modelo en diferentes momentos. Los vectores de peso producidos por la secuencia de estado oculto de cada etapa se ingresan en una función de aprendizaje que produce un vector de probabilidad, que a su vez se utiliza para encontrar el promedio ponderado de los estados de salida ocultos. La intuición detrás del modelo es que las diferentes palabras y oraciones en un documento son mutuamente informativas. La representación de documentos se construye a partir de cada oración, y cada representación de oración se construye a partir de palabras. Como se aprecia en la figura 11, el mecanismo de atención funciona marcando prioridades de atención a la salida de cada etapa. Cada oración en un documento puede considerarse como una secuencia ordenada de palabras que se alimenta en cada capa LSTM, y nuevamente se sigue el mismo procedimiento para las oraciones con atención en la parte superior.

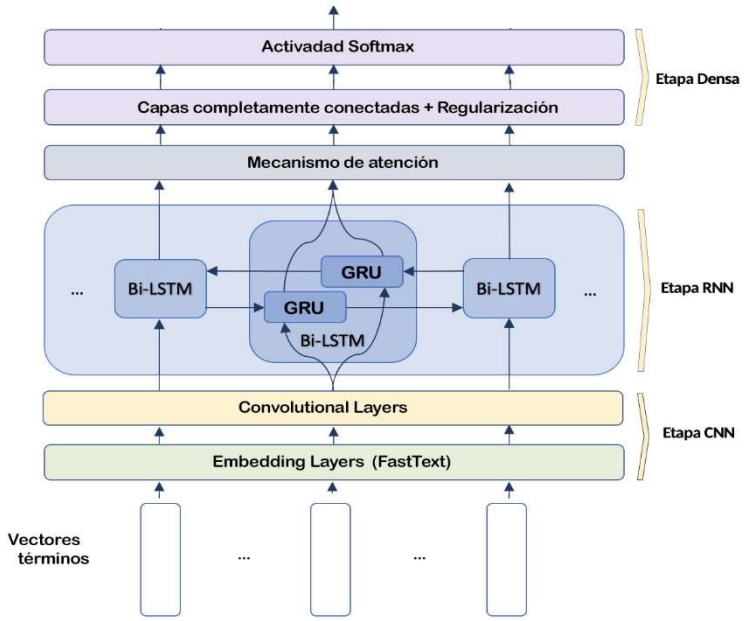

Figura 12. Estructura completa de nuestra propuesta de arquitectura completa.

4.5 Arquitectura final

Nuestra arquitectura combina todos los modelos anteriores para crear un modelo híbrido que consta de tres etapas: una etapa convolucional para mapear las características locales del grano grueso; Una etapa recursiva que se centró en detectar dependencias de larga distancia; y una etapa densa, que a su vez está formada por una capa de regularización y dos capas completamente conectadas para modelar las interrelaciones entre las diferentes características.

Nuestra arquitectura consta de cinco tipos de capas que se dividen en las tres etapas ya mencionadas: convolucional, recursiva y de salida (ver figura 12):

(1) Capa de entrada: el documento normalizado se divide en oraciones que serán la entrada para el modelo.

(2) Capas convolucionales: consiste en varias capas de incrustación que asignan cada palabra a un vector real de baja dimensión.

(3) Capas recursivas: utilizamos unidades de memoria GRU para obtener características de alto nivel.

(4) Capa de atención: produce un vector de peso y combina características de nivel de palabra de cada paso de tiempo en un vector de característica de nivel de oración, multiplicando el vector de peso.

(5) Capas de salida: las características generadas desde la etapa recursiva se pasan a una capa softmax totalmente conectada cuyas ponderaciones generan las características de codificación de sus interrelaciones de forma similar a como lo hace una red de alimentación directa. Además, se incluyen algunas capas de regulación. La salida es la distribución de probabilidad en todas las categorías utilizando una función estándar softmax.

El detalle de estas cinco etapas y su conexión se pueden ver en detalle en la tabla 2, mientras que el código fuente de Python se puede encontrar en el repositorio del proyecto (ver el anexo 1 para más detalles).

La primera etapa consta de varias capas convolucionales, cada una de ellas entrenada en un modelo Fasttext sobre diferentes tipos de cuerpos específicos. FastText es una extensión bien conocida del modelo Word2Vec para extraer vectores de palabras. Esta etapa toma los pesos de las diferentes incorporaciones de palabras como entrada y aplica ventanas de diferentes longitudes y varias matrices de pesos para generar varios mapas de características. La segunda etapa consiste en dos capas Bilaterales LSTM compuestas de varias unidades recurrentes cerradas (GRU) para detectar relaciones de distancia relevantes respaldadas por un mecanismo de atención que mantiene enfocado el modelo para elegir la mejor convolución en cada momento, creando pares y combinaciones de atributos distantes. Más tarde, se aplican las capas de convolución y agrupación, y los mapas de características codificados son ponderados por el mecanismo de atención. Las dependencias a largo plazo aprendidas en esta etapa pueden verse como la representación a nivel de párrafo. Finalmente, la etapa de salida está formada por una capa de olvido (dropout) y dos capas completamente conectadas (dos capas densas y una capa de regularización adicional) para codificar las interrelaciones entre las diferentes características de manera similar a como lo hace una red neuronal convencional. Los hiperparámetros del modelo se determinan empíricamente, lo más importante como se especifica en la tabla 3. La tabla muestra dos columnas, la primera es el nombre del parámetro de red, mientras que la segunda es el valor del mismo conjunto durante el entrenamiento.

Nuestro algoritmo de aprendizaje profundo difiere de otros métodos existentes en la aplicación de ventanas de diferentes longitudes y varias matrices de peso en la operación convolucional. Además, la agrupación máxima opera en características adyacentes y los mapas de características generados en nuestra etapa CNN retienen la información secuencial en el contexto de la oración. La arquitectura de aprendizaje profundo de nuestro modelo aprovecha las características codificadas locales extraídas de las capas CNN y las dependencias a largo plazo capturadas por la etapa RNN. También realizamos un pretratamiento dirigido a corregir el desequilibrio de clases en el conjunto de datos de entrada. En lugar de utilizar modelos de *embedding* previamente entrenados, que están muy limitados para usos médicos, creamos nuestro propio corpus a partir

de la terminología extraída de diferentes glosarios y diccionarios médicos (enfermedades, síntomas, ingredientes activos ...), y nos aseguramos de usar filtros de distintas formas y tamaños y capas convolucionales para mapear y extraer tanta información contextual como sea posible.

Table 3. Parámetros utilizados durante los entrenamientos.

Parámetro	Valor
Filters	64
Kernel	3
Embedding Dimension	500
Epochs	50, 75, 100
Activation Function	Sigmoid
Batch Size	256
Word Embedding	GloVe and FastText (custom)
Pool Size	2
Dropouts	0.25:ConvNet, 0.20:Bi-LSTM
Optimizer	Adam (lr=0.001)

El enfoque de los datos de texto que utilizamos fue doble, nuestros objetivos fueron:

i) Usar diferentes glosarios y diccionarios clínicos y nuestro conocimiento previo de la literatura científica para crear palabras clave inteligentes y filtrar datos de una manera significativa.

ii) Explorar diferentes métodos de inclusión para Procesamiento de lenguaje natural (Word2vec1, Glove2, Fasttext) para traducir el texto en funciones numéricas con las que nuestro clasificador puede trabajar de manera eficiente. Como estábamos tratando con grandes bloques de texto especializado, nuestro primer objetivo era extraer solo las secciones relevantes del texto (por ejemplo, incluir resultados, eliminar métodos y materiales). También seleccionamos oraciones que contienen palabras que podrían contener otras palabras que ayudarían en la clasificación, como "objetivo", "metástasis", "tumor", etc.

En nuestra arquitectura, utilizamos la etapa convolucional para mapear características locales en la secuencia mientras se usa la capa recursiva enfocada en detectar dependencias de larga distancia. La etapa RNN a su vez consta de dos capas Bi-LSTM formadas por varias celdas GRU (véase figura 10). Nuestra arquitectura está inspirada en el modelo de red propuesto por (**Krause et al. 2016**), pero integrando una capa recursiva con elementos de memoria a largo plazo (GRU) para ayudar a capturar la información contextual de cada oración a nivel de todo el documento y múltiples filtros en la capa convolucional de la misma manera que en (**Yin et al., 2016**) a lidiar con características de granularidad diferente y ganar mayor representatividad. Además, nuestro modelo tiene múltiples capas de no linealidad que nos permiten aprender representaciones más complejas. Otras mejoras menores del modelo incluyen la sustitución del mecanismo de agrupación máxima con una de agrupación máxima global, agregando capas de abandono para evitar el sobreajuste (este mecanismo se usa a menudo en conjuntos de datos pequeños),

reemplazando la convolución de capas 1d por convoluciones separadas y la eliminación de Regulador L2 en capas de convolución. Además, nuestro modelo también se basa en el modelo BLSTM propuesto por (Zhang et al., 2016), RNN empleó un bidireccional para aprender patrones de relación. El ajuste de los diferentes hiperparámetros se realizó empíricamente mediante pruebas y experimentación (ver detalles de los valores utilizados para los hiperparámetros de la red durante el entrenamiento en la tabla 3). Además, nuestro modelo tiene múltiples capas de no linealidad que permiten aprender representaciones más complejas.

Entre otras mejoras menores acometidas al modelo se encuentran el reemplazo del mecanismo de *max pooling* por uno de *global max pooling*, el añadir un capa de olvido aleatorio para evitar sobreajuste (este mecanismo suele utilizarse con frecuencia en dataset pequeños), el reemplazo de las capas de convolución 1d por convoluciones separadas, la eliminación del regularizador L2 en capas de convolución; y las dos que creemos más importantes por su afectación al modelo que son la agregación de una capa full-connect + dropout después de cada capa de maxpooling y la no inclusión de una capa de aplanado + dropout después de cada concatenación. El ajuste de los distintos hiperparámetros se realizó de forma empírica mediante experimentación.

CÓDIGO FUENTE EN KERAS PARA LA ARQUITECTURA

```
from keras.models import Sequential, Model
from keras.layers import Dense, Embedding, LSTM, GRU, Bidirectional, Merge,
Input, concatenate
from keras.layers.merge import Concatenate
from keras.utils.np_utils import to_categorical
from keras.callbacks import ModelCheckpoint
from keras.models import load_model
from keras.optimizers import Adam
# Build out our simple LSTM
embed_dim = 300
lstm_out = 196

# Model saving callback
ckpt_callback = ModelCheckpoint('keras_model',
                                monitor='val_loss',
                                verbose=1,
                                save_best_only=True,
                                mode='auto')


input_sequence_begin = Input(shape=(train_set_input_begin.shape[1],))
input_sequence_end = Input(shape=(train_set_input_end.shape[1],))
input_gene = Input(shape=(train_set_input_gene.shape[1],))
input_variant = Input(shape=(train_set_input_variation.shape[1],))

merged = concatenate([input_gene, input_variant])
dense = Dense(32, activation='sigmoid')(merged)

#Load word embeddings
embeddings_index = dict()
```

```
f = open("/content/drive/My Drive/datasets/model.txt", encoding="utf8")
for line in f:
  values = line.split()
  word = values[0]
  coefs = np.asarray(values[1:], dtype='float32')
  embeddings_index[word] = coefs
f.close()

#Create a weight matrix
embedding_matrix = np.zeros((VOCABULARY_SIZE, 100))
for word, index in tokenizer.word_index.items():
  if index > VOCABULARY_SIZE - 1:
    break
  else:
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
      embedding_matrix[index] = embedding_vector
print(embedding_matrix)

embeds_begin = Embedding(VOCABULARY_SIZE, embed_dim, input_length = SEQUENCE
_LENGTH, weights=[embedding_matrix], trainable=True)(input_sequence_begin)
embeds_out_begin = Bidirectional(GRU(lstm_out, recurrent_dropout=0.2, dropou
t=0.2, return_sequences=True))(embeds_begin)

attention_begin = AttentionWithContext()(embeds_out_begin)

embeds_end = Embedding(VOCABULARY_SIZE, embed_dim, input_length = SEQUENCE_L
ENGTH)(input_sequence_end)
embeds_out_end = Bidirectional(GRU(lstm_out, recurrent_dropout=0.2, dropout=
0.2, return_sequences=True))(embeds_end)
attention_end = AttentionWithContext()(embeds_out_end)

merged2 = concatenate([attention_begin, attention_end, dense])
dense2 = Dense(9,activation='softmax')(merged2)

model = Model(inputs=[input_sequence_begin, input_sequence_end, input_gene,
input_variant], outputs=dense2)
model.compile(loss = 'categorical_crossentropy', optimizer='adam')
print(model.summary())

class AttentionWithContext(Layer):
    """
    Attention operation, with a context/query vector, for temporal data.

Follows the work of Yang et al. [https://www.cs.cmu.edu/~diyiy/docs/naacl16.
pdf]
    "Hierarchical Attention Networks for Document Classification"
    by using a context vector to assist the attention
```

```
    def __init__(self,
                 W_regularizer=None, u_regularizer=None, b_regularizer=None,
                 W_constraint=None, u_constraint=None, b_constraint=None,
                 bias=True, **kwargs):

        self.supports_masking = True
        self.init = initializers.get('glorot_uniform')

        self.W_regularizer = regularizers.get(W_regularizer)
        self.u_regularizer = regularizers.get(u_regularizer)
        self.b_regularizer = regularizers.get(b_regularizer)

        self.W_constraint = constraints.get(W_constraint)
        self.u_constraint = constraints.get(u_constraint)
        self.b_constraint = constraints.get(b_constraint)

        self.bias = bias
        super(AttentionWithContext, self).__init__(**kwargs)

    def build(self, input_shape):
        assert len(input_shape) == 3

        self.W = self.add_weight((input_shape[-1], input_shape[-1],),
                                 initializer=self.init,
                                 name='{}_W'.format(self.name),
                                 regularizer=self.W_regularizer,
                                 constraint=self.W_constraint)
        if self.bias:
            self.b = self.add_weight((input_shape[-1],),
                                     initializer='zero',
                                     name='{}_b'.format(self.name),
                                     regularizer=self.b_regularizer,
                                     constraint=self.b_constraint)

        self.u = self.add_weight((input_shape[-1],),
                                 initializer=self.init,
                                 name='{}_u'.format(self.name),
                                 regularizer=self.u_regularizer,
                                 constraint=self.u_constraint)

        super(AttentionWithContext, self).build(input_shape)

    def compute_mask(self, input, input_mask=None):
        # do not pass the mask to the next layers
        return None

    def call(self, x, mask=None):
        uit = dot_product(x, self.W)
```

```
        if self.bias:
            uit += self.b

        uit = K.tanh(uit)
        ait = dot_product(uit, self.u)

        a = K.exp(ait)

        # apply mask after the exp. will be re-normalized next
        if mask is not None:
            # Cast the mask to floatX to avoid float64 upcasting in theano
            a *= K.cast(mask, K.floatx())

        # in some cases especially in the early stages of training the sum m
ay be almost zero
        # and this results in NaN's. A workaround is to add a very small pos
itive number ε to the sum.
        # a /= K.cast(K.sum(a, axis=1, keepdims=True), K.floatx())
        a /= K.cast(K.sum(a, axis=1, keepdims=True) + K.epsilon(), K.floatx(
))

        a = K.expand_dims(a)
        weighted_input = x * a
        return K.sum(weighted_input, axis=1)

    def compute_output_shape(self, input_shape):
        return input_shape[0], input_shape[-1]
```

Capítulo V

Experimentación.

5. Experimentación

Como primer paso a la experimentación configuramos el entorno Python para trabajar con Keras y la TenserFlow API (ambas herramientas ya fueron descritas en las secciones anteriores) e instalamos las dependencias y paqueterías que serán necesarias para los distintos modelos:

Configuración del entorno de trabajo

```
Descargar e instalar anaconda3 en "~/Programs/anaconda3"

Crear un entorno virtual usando "cd ~/Programs/anaconda3 && mkdir e
nvs and cd envs && ../bin/conda create -p ~/Programs/anaconda3/envs
/dsotc-c3 python=3.6 anaconda."

Activar el espacio de trabajo "/home/user/Programs/anaconda3/envs/d
sotc-c3/bin/activate dsotc-c3"

Instalar pip mediante "~/Programs/anaconda3/envs/dsotc-c3/bin/pip u
sing conda install pip"

Instalación de las librerías de keras y tensorflow:
"pip install --upgrade"
"pip install -U pip keras tensorflow"
"pip install -U pip tensorflow-gpu"
"pip install --upgrade scipy"

Ejecutar el comando "pip install -r requirements.txt" para instalar
todas las dependencias del proyecto

Instalación de jupyter IDE:
"pip install jupyter"
"pip install --use-wheel --no-index --find-links=TF-master\numpymkl
.whl"
"jupyter notebook"

Activar las extensiones de jupyter ejecutando "jupyter extensions j
upyter nbextensions_configurator enable --user"

Activar las opciones de configuración mediante "jupyter contrib nbe
xtension install --user"

Las versiones de referencia de la paquetería utilizados fueron:

Python 3.6.2, Keras 2.0.8, tensorflow 1.12.0, tensorflow-gpu 1.3.0,
tensorflow-tensorboard 0.1.8.
```

Inicialización de librerías de terceros

Utilizaremos algunas otras librerías de terceros bajo licencia GPL, entre otras:

- Anaconda como gestor de paquetes

- Nltk y BeautifulSoup para el tratamiento de textos

- NumPy, SciPy y Matplotlib para el manejo de datos numéricos y tensores

- Sklearn como marco de trabajo para el desarrollo de los modelos y otras herramientas auxiliares.

```
# Core
import os
import sys
import re
import collections
import itertools
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf

# preproceso
import re, string, unicodedata
import nltk
#import contractions
import inflect

# metricas
from sklearn.preprocessing import LabelBinarizer, LabelEncoder
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier

# parseo
from bs4 import BeautifulSoup
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import LancasterStemmer, WordNetLemmatizer

# sklearn others
from sklearn.preprocessing import LabelBinarizer, LabelEncoder
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import TruncatedSVD
from sklearn.base import BaseEstimator, TransformerMixin

# utils
import gc
import random
import smart_open
import h5py
import csv
import gensim
```

```
import datetime as dt
from tqdm import tqdm_notebook as tqdm
import matplotlib.pyplot as plt
import seaborn as sns
import hypertools as hyp
```

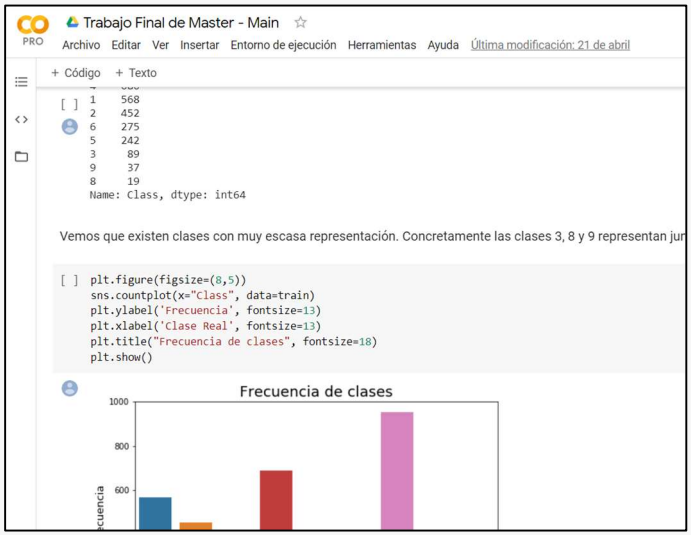

Figura 13. Entorno de trabajo Python - Keras

5.1 Pruebas y evaluación

Para hacer la recogida de datos uniforme y poder dar soporte estadístico a nuestro estudio definimos algunas funciones para la recogida de traza durante la ejecución.

```
def show_scores(model, h, X_train, Y_train, X_test, Y_test):
    loss, acc = model.evaluate(X_train, Y_train, verbose=0)
    print ("Training: accuracy   = %.6f loss = %.6f" % (acc, loss))
    loss, acc = model.evaluate(X_test, Y_test, verbose=0)
    print ("Validation: accuracy = %.6f loss = %.6f" % (acc, loss))
    if 'val_acc' in h.history:
        print ("Over fitting score   = %.6f" % over_fitting_score(h
))
        print ("Under fitting score  = %.6f" % under_fitting_score(
h))
    print ("Params count:", model.count_params())
    print ("stop epoch =", max(h.epoch))
    print ("batch_size =", h.params['batch_size'])
    #view_acc(h)
    id = model.name[-1]
    plt.savefig(model.name + '_acc_graph.png')
    plt.show()
    view_loss(h)
    plt.savefig(model.name + '_loss_graph.png')
    plt.show()

def view_acc(h):
    # Accuracy history graph
    plt.plot(h.history['acc'])
    if 'val_acc' in h.history:
```

```
        plt.plot(h.history['val_acc'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    leg = plt.legend(['train', 'validation'], loc='best')
    plt.setp(leg.get_lines(), linewidth=3.0)

def view_loss(h):
    # Loss history graph
    plt.plot(h.history['loss'])
    if 'val_loss' in h.history:
        plt.plot(h.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    leg = plt.legend(['train', 'validation'], loc='best')
    plt.setp(leg.get_lines(), linewidth=3.0)

def over_fitting_score(h):
    gap = []
    n = len(h.epoch)
    for i in h.epoch:
        acc = h.history['acc'][i]
        val_acc = h.history['val_acc'][i]
        # late gaps get higher weight ..
        gap.append( i * abs(acc-val_acc))
    ofs = sum(gap) / (n * (n-1) / 2)
    return ofs

def under_fitting_score(h):
    gap = []
    for i in h.epoch:
        acc = h.history['acc'][i]
        val_acc = h.history['val_acc'][i]
        gap.append(abs(acc-val_acc))
    gap = np.array(gap)
    return gap.mean()

def find_best_epoch(h, thresh=0.02):
    epochs = []
    for i in h.epoch:
        acc = h.history['acc'][i]
        val_acc = h.history['val_acc'][i]
        if abs(acc-val_acc) <= thresh:
            epochs.append(i)

    if not epochs:
        print ("No result")
        return None
    max_e = -1
    max_val_acc = -1
    for i in epochs:
        if h.history['val_acc'][i] > max_val_acc:
            max_e = i
            max_val_acc = h.history['val_acc'][i]
    print ("best epoch = %d ; best acc = %.6f ; best val_acc = %.6f
" % (max_e, h.history['acc'][max_e], h.history['val_acc']))
    return max_e
```

5.2 Modelos de referencia

Como modelos de referencia implementamos cuatro arquitecturas distintas extraídas desde el estado-del-arte:

MODELO REF1 (Ensemble CNN + RNN) propuesto en Ensemble Application of Convolutional and Recurrent Neural Networks for Multi-label Text Categorization. (Chen et al., 2017).

CNN

```
text_input_layer = Input(shape=(MAX_TEXT_LEN,), dtype='int32')
doc_embedding   = Embedding(vocab_size, WORD_EMB_SIZE,
input_length=MAX_TEXT_LEN, trainable=True)(text_input_layer)
convs = []
filter_sizes = [10, 20, 30, 40, 50]

for filter_size in filter_sizes:

    l_conv = Conv1D(filters=256, kernel_size=filter_size, padding='
valid', activation='relu')(doc_embedding)
    l_pool = MaxPool1D(filter_size)(l_conv)
    convs.append(l_pool)

l_merge = Concatenate(axis=1)(convs)
l_flat = Flatten()(l_merge)
cnn_dense = Dense(128, activation='relu')(l_flat)
```

RNN

```
rnn_layer = LSTM(128, return_sequences=False, stateful=False)(doc_em
bedding, initial_state=[cnn_dense, cnn_dense])

output_layer = Dense(9, activation='softmax')(rnn_layer)

model_1 = Model(inputs=[text_input_layer], outputs=[output_layer])
sgd = SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True, clipnor
m=0.5)

adamo = Adam(lr=0.003, beta_1=0.92, beta_2=0.999, epsilon=1e-08, cli
pnorm=1.)

my_metrics = [categorical_accuracy, top_k_categorical_accuracy ]

model_1.compile(loss='categorical_crossentropy', optimizer=adamo, me
trics=my_metrics)
```

ARQUITECTURA

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 861)	0	
embedding_1 (Embedding)	(None, 861, 50)	43050	input_1[0][0]
conv1d_1 (Conv1D)	(None, 852, 256)	128256	embedding_1[0][0]
conv1d_2 (Conv1D)	(None, 842, 256)	256256	embedding_1[0][0]
conv1d_3 (Conv1D)	(None, 832, 256)	384256	embedding_1[0][0]
conv1d_4 (Conv1D)	(None, 822, 256)	512256	embedding_1[0][0]
conv1d_5 (Conv1D)	(None, 812, 256)	640256	embedding_1[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 85, 256)	0	conv1d_1[0][0]
max_pooling1d_2 (MaxPooling1D)	(None, 42, 256)	0	conv1d_2[0][0]
max_pooling1d_3 (MaxPooling1D)	(None, 27, 256)	0	conv1d_3[0][0]
max_pooling1d_4 (MaxPooling1D)	(None, 20, 256)	0	conv1d_4[0][0]
max_pooling1d_5 (MaxPooling1D)	(None, 16, 256)	0	conv1d_5[0][0]
concatenate_1 (Concatenate)	(None, 190, 256)	0	max_pooling1d_1[0][0] max_pooling1d_2[0][0] max_pooling1d_3[0][0] max_pooling1d_4[0][0] max_pooling1d_5[0][0]
flatten_1 (Flatten)	(None, 48640)	0	concatenate_1[0][0]
dense_1 (Dense)	(None, 128)	6226048	flatten_1[0][0]
lstm_1 (LSTM)	(None, 128)	91648	embedding_1[0][0] dense_1[0][0] dense_1[0][0]
dense_2 (Dense)	(None, 9)	1161	lstm_1[0][0]

Total params: 8,283,187
Trainable params: 8,283,187
Non-trainable params: 0

MODELO REF2 (Combination CNN + RNN) propuesto por Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts (Wang et al., 2017).

CNN

```
text_input_layer = Input(shape=(MAX_SENT_LEN,), dtype='int32')
doc_embedding   = Embedding(vocab_size, WORD_EMB_SIZE, #weights=[trained_embeddings], input_length=MAX_SENT_LEN, trainable=True)(text_input_layer)
convs = []
filter_sizes = [3, 4, 5]
for filter_size in filter_sizes:
    l_conv = Conv1D(filters=64, kernel_size=filter_size, padding='valid', activation='relu')(doc_embedding)
```

```
    l_pool = MaxPool1D(filter_size)(l_conv)
    convs.append(l_pool)
cnn_feature_maps = Concatenate(axis=1)(convs)
```

LSTM

```
sentence_encoder = LSTM(64,return_sequences=False)(cnn_feature_maps
)
fc_layer =Dense(128, activation="relu")(sentence_encoder)
output_layer = Dense(9,activation="softmax")(fc_layer)

model_2 = Model(inputs=[text_input_layer], outputs=[output_layer])

sgd = SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True, clipno
rm=0.5)

adamo = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, cli
pnorm=1.)

my_metrics = [categorical_accuracy, top_k_categorical_accuracy ]

model_2.compile(loss='categorical_crossentropy', optimizer=adamo, m
etrics=my_metrics)
```

ARQUITECTURA

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	(None, 861)	0	
embedding_2 (Embedding)	(None, 861, 50)	43050	input_2[0][0]
conv1d_6 (Conv1D)	(None, 859, 64)	9664	embedding_2[0][0]
conv1d_7 (Conv1D)	(None, 858, 64)	12864	embedding_2[0][0]
conv1d_8 (Conv1D)	(None, 857, 64)	16064	embedding_2[0][0]
max_pooling1d_6 (MaxPooling1D)	(None, 286, 64)	0	conv1d_6[0][0]
max_pooling1d_7 (MaxPooling1D)	(None, 214, 64)	0	conv1d_7[0][0]
max_pooling1d_8 (MaxPooling1D)	(None, 171, 64)	0	conv1d_8[0][0]
concatenate_2 (Concatenate)	(None, 671, 64)	0	max_pooling1d_6[0][0] max_pooling1d_7[0][0] max_pooling1d_8[0][0]
lstm_2 (LSTM)	(None, 64)	33024	concatenate_2[0][0]
dense_3 (Dense)	(None, 128)	8320	lstm_2[0][0]
dense_4 (Dense)	(None, 9)	1161	dense_3[0][0]

Total params: 124,147
Trainable params: 124,147
Non-trainable params: 0

MODELO REF3 (CNN Multi-channel w/variable size) propuesto por Multichannel variable-size convolution for sentence classification (Yin et al., 2016).

CNN

```
text_embedding1 = Embedding(vocab_size, WORD_EMB_SIZE, input_length
=MAX_SENT_LEN,
trainable=True)(text_seq_input)
text_embedding2 = Embedding(vocab_size, WORD_EMB_SIZE, input_length
=MAX_SENT_LEN,
trainable=True)(text_seq_input)

k_top = 4
filter_sizes = [3,5]

layer_1 = []

for text_embedding in [text_embedding1, text_embedding2]:
    conv_pools = []

    for filter_size in filter_sizes:
        l_zero = ZeroPadding1D((filter_size-1,filter_size-1))(text_
embedding)
        l_conv = Conv1D(filters=128, kernel_size=filter_size, paddi
ng='same', activation='tanh')(l_zero)
        l_pool = KMaxPooling(k=30, axis=1)(l_conv)
        conv_pools.append((filter_size,l_pool))
    layer_1.append(conv_pools)

last_layer = []

for layer in layer_1: # no of embeddings used

    for (filter_size, input_feature_maps) in layer:

        l_zero = ZeroPadding1D((filter_size-1,filter_size-1))(input
_feature_maps)
        l_conv = Conv1D(filters=128, kernel_size=filter_size, paddi
ng='same', activation='tanh')(l_zero)
        l_pool = KMaxPooling(k=k_top, axis=1)(l_conv)
        last_layer.append(l_pool)

l_merge = Concatenate(axis=1)(last_layer)
l_flat = Flatten()(l_merge)
l_dense = Dense(128, activation='relu')(l_flat)
l_out = Dense(9, activation='softmax')(l_dense)

model_3 = Model(inputs=[text_seq_input], outputs=l_out)

sgd = SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True, clipno
rm=0.5)
adamo = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, cli
pnorm=1.)

my_metrics = [categorical_accuracy, top_k_categorical_accuracy ]
model_3.compile(loss='categorical_crossentropy', optimizer=adamo, m
etrics=my_metrics)
```

ARQUITECTURA

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	(None, 861)	0	
embedding_3 (Embedding)	(None, 861, 50)	43050	input_3[0][0]
embedding_4 (Embedding)	(None, 861, 50)	43050	input_3[0][0]
zero_padding1d_1 (ZeroPadding1D	(None, 865, 50)	0	embedding_3[0][0]
zero_padding1d_2 (ZeroPadding1D	(None, 869, 50)	0	embedding_3[0][0]
zero_padding1d_3 (ZeroPadding1D	(None, 865, 50)	0	embedding_4[0][0]
zero_padding1d_4 (ZeroPadding1D	(None, 869, 50)	0	embedding_4[0][0]
conv1d_9 (Conv1D)	(None, 865, 128)	19328	zero_padding1d_1[0][0]
conv1d_10 (Conv1D)	(None, 869, 128)	32128	zero_padding1d_2[0][0]
conv1d_11 (Conv1D)	(None, 865, 128)	19328	zero_padding1d_3[0][0]
conv1d_12 (Conv1D)	(None, 869, 128)	32128	zero_padding1d_4[0][0]
k_max_pooling_1 (KMaxPooling)	(None, 30, 128)	0	conv1d_9[0][0]
k_max_pooling_2 (KMaxPooling)	(None, 30, 128)	0	conv1d_10[0][0]
k_max_pooling_3 (KMaxPooling)	(None, 30, 128)	0	conv1d_11[0][0]
k_max_pooling_4 (KMaxPooling)	(None, 30, 128)	0	conv1d_12[0][0]
zero_padding1d_5 (ZeroPadding1D	(None, 34, 128)	0	k_max_pooling_1[0][0]
zero_padding1d_6 (ZeroPadding1D	(None, 38, 128)	0	k_max_pooling_2[0][0]
zero_padding1d_7 (ZeroPadding1D	(None, 34, 128)	0	k_max_pooling_3[0][0]
zero_padding1d_8 (ZeroPadding1D	(None, 38, 128)	0	k_max_pooling_4[0][0]
conv1d_13 (Conv1D)	(None, 34, 128)	49280	zero_padding1d_5[0][0]
conv1d_14 (Conv1D)	(None, 38, 128)	82048	zero_padding1d_6[0][0]
conv1d_15 (Conv1D)	(None, 34, 128)	49280	zero_padding1d_7[0][0]
conv1d_16 (Conv1D)	(None, 38, 128)	82048	zero_padding1d_8[0][0]
k_max_pooling_5 (KMaxPooling)	(None, 4, 128)	0	conv1d_13[0][0]
k_max_pooling_6 (KMaxPooling)	(None, 4, 128)	0	conv1d_14[0][0]
k_max_pooling_7 (KMaxPooling)	(None, 4, 128)	0	conv1d_15[0][0]
k_max_pooling_8 (KMaxPooling)	(None, 4, 128)	0	conv1d_16[0][0]
concatenate_3 (Concatenate)	(None, 16, 128)	0	k_max_pooling_5[0][0] k_max_pooling_6[0][0] k_max_pooling_7[0][0] k_max_pooling_8[0][0]
flatten_2 (Flatten)	(None, 2048)	0	concatenate_3[0][0]
dense_5 (Dense)	(None, 128)	262272	flatten_2[0][0]
dense_6 (Dense)	(None, 9)	1161	dense_5[0][0]

Total params: 715,101
Trainable params: 715,101
Non-trainable params: 0

MODELO REF4 (Multiplicative LSTM) propuesto por Multiplicative LSTM for sequence modelling (Krause et al., 2017)

CNN

```
text_input_layer = Input(shape=(MAX_SENT_LEN,), dtype='int32')
doc_embedding   = Embedding(vocab_size, WORD_EMB_SIZE,
input_length=MAX_SENT_LEN, trainable=True)(text_input_layer)
convs = []
```

```
filter_sizes = [16, 32, 64, 128]

for filter_size in filter_sizes:
    l_conv = Conv1D(filters=128, kernel_size=filter_size, padding='
valid', activation='relu')(doc_embedding)
    l_pool = MaxPool1D(filter_size)(l_conv)
    convs.append(l_pool)
l_merge = Concatenate(axis=1)(convs)
l_flat = Flatten()(l_merge)
cnn_dense = Dense(128, activation='relu')(l_flat)
```

RNN

```
rnn_layer = LSTM(128, return_sequences=False, stateful=False)(doc_em
bedding, initial_state=[cnn_dense, cnn_dense])
output_layer = Dense(9, activation='softmax')(rnn_layer)
model_5 = Model(inputs=[text_input_layer], outputs=[output_layer])
sgd = SGD(lr=0.001, decay=1e-4, momentum=0.9, nesterov=True, clipnor
m=0.5)
adamo = Adam(lr=0.003, beta_1=0.9, beta_2=0.999, epsilon=1e-08, clip
norm=1.)
my_metrics = [categorical_accuracy, top_k_categorical_accuracy ]
model_5.compile(loss='categorical_crossentropy',     optimizer=adamo,
metrics=my_metrics)
```

ARQUITECTURA

.

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	(None, 861)	0	
embedding_5 (Embedding)	(None, 861, 64)	55104	input_4[0][0]
conv1d_17 (Conv1D)	(None, 846, 128)	131200	embedding_5[0][0]
conv1d_18 (Conv1D)	(None, 830, 128)	262272	embedding_5[0][0]
conv1d_19 (Conv1D)	(None, 798, 128)	524416	embedding_5[0][0]
conv1d_20 (Conv1D)	(None, 734, 128)	1048704	embedding_5[0][0]

max_pooling1d_9 (MaxPooling1D)	(None, 52, 128)	0	conv1d_17[0][0]
max_pooling1d_10 (MaxPooling1D)	(None, 25, 128)	0	conv1d_18[0][0]
max_pooling1d_11 (MaxPooling1D)	(None, 12, 128)	0	conv1d_19[0][0]
max_pooling1d_12 (MaxPooling1D)	(None, 5, 128)	0	conv1d_20[0][0]
concatenate_4 (Concatenate)	(None, 94, 128)	0	max_pooling1d_9[0][0] max_pooling1d_10[0][0] max_pooling1d_11[0][0] max_pooling1d_12[0][0]
flatten_3 (Flatten)	(None, 12032)	0	concatenate_4[0][0]
dense_7 (Dense)	(None, 128)	1540224	flatten_3[0][0]
lstm_3 (LSTM)	(None, 128)	98816	embedding_5[0][0] dense_7[0][0] dense_7[0][0]
dense_8 (Dense)	(None, 9)	1161	lstm_3[0][0]

Total params: 3,661,897
Trainable params: 3,661,897
Non-trainable params: 0

5.3 Resultados de los distintos modelos y comparativa

En esta sección se presentan los resultados de las 4 arquitecturas testadas. Todas las pruebas se realizaron sobre el mismo dataset de origen y en las mismas condiciones experimentales (hardware y parametrización) lo que nos permite realizar comparativas cruzadas de rendimiento. Una vez reentrenados los modelos, comparamos sus resultados utilizando tres métricas de evaluación sobre el conjunto de datos reservados para test. Estas son el *accuracy* o tasa de acierto, el **categorical_accuracy** o tasa de acierto por clase y la función *loss* como indicador del error y la convergencia de la red. Cuando las clases no están completamente balanceadas conviene emplear otras medidas de evaluación que tienen en cuenta la distribución del error entre clases. Los trabajos originales utilizaban **avg-f-score** (media armónica de la capacidad predictora por clase) entre PPV (*Positive Predictive Value*) y TPR (*True Positive Rate*).

Al promediar los valores macro de accuracy, le damos el mismo peso a cada clase. Para compensar este efecto se decide tomar la medida **weighted-average-accuracy** o precisión promedia ponderada donde el valor de accuracy es corregido con el peso que suponen las muestras de esa clase sobre el total.

La adecuación de estas métricas sobrepasan el alcance de este proyecto y deberán ser evaluadas en caso de que sea necesario reevaluar los resultados pero el lector puede encontrar un extenso análisis de la adecuación de algunas de estas métricas a los problemas de clasificación puede ser consultada en (**Morales & Elgueta, 2017**) y (**Salmeron-Majadas et al., 2018**). El número de épocas se fija a 100 para mantener los tiempos de cómputo en niveles aceptables. El resto de hiperparámetros como las funciones de pérdida, el optimizador utilizado, el tamaño del paso y la tasa de aprendizaje se fijan empíricamente y se mantienen inalterados en todas las pruebas para no influir en los resultados.

Resultados del experimento con el Modelo 1 (Ensemble CNN + RNN)

Este modelo consigue resultados bastante pobres, obteniendo valores de precisión por categoría inferiores al 50%. Si bien no son valores tan malos teniendo en cuenta el gran número de categorías con las que trabajamos está muy lejos de los resultados esperados.

Si observamos con detenimiento la figura 14 podemos observar su convergencia irregular y su tendencia al fracaso (área marcada en rojo) incluso en épocas avanzadas del aprendizaje.

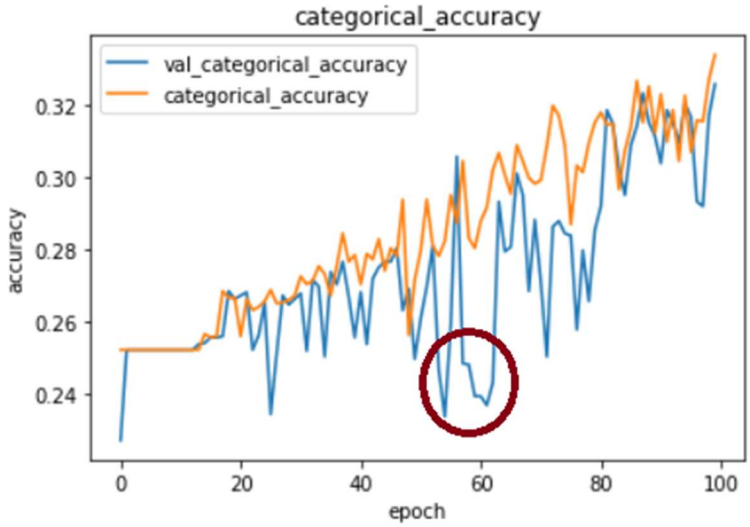

Figura 14. Representación de la precisión alcanzada por el modelo 1.

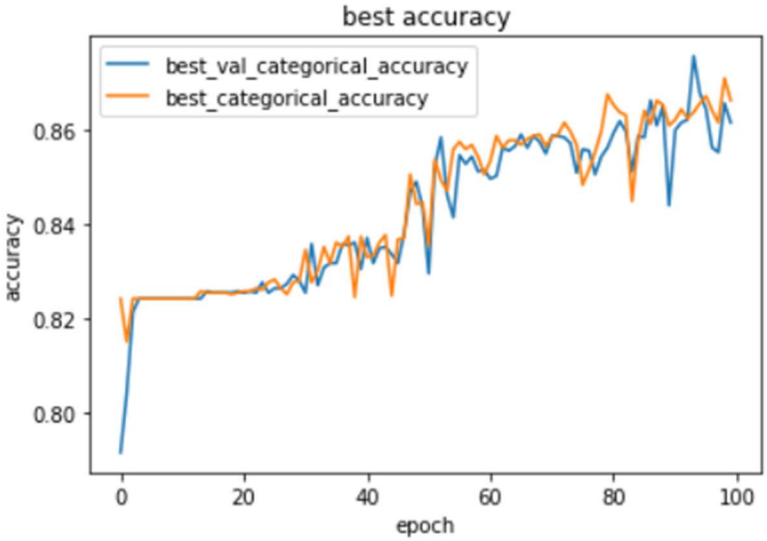

Figura 15. Representación del mejor valor promedio por categoría para el modelo 1.

Las impresiones iniciales se confirman cuando observamos la distribución del mejor valor para la precisión a lo largo del entrenamiento (figura 15) donde puede observarse que se obtuvieron buenos datos de precisión para algunas categorías mientras que el resultado fue muy pobre para otras, presumiblemente aquellas más escasamente representadas.

Resultados del experimento con el modelo 2 (Combination CNN + RNN).

En este modelo obtenemos resultados mucho mejores que en el anterior y también un rendimiento más estable. Podemos ver como la capacidad predictora crece rápidamente hasta niveles del 30% y a partir de ahí mantiene un crecimiento sostenido (ver figura 17) hasta situarse en promedio en torno al 60% de predicciones correctas.

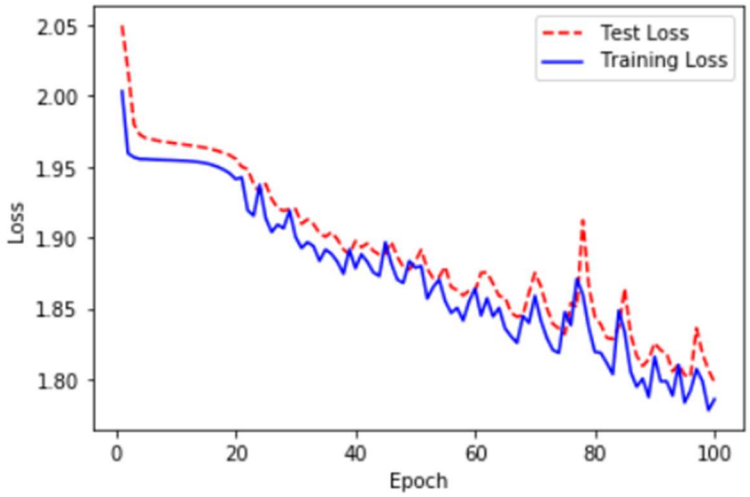

Figura 16. Evolución del error a lo largo de las distintas épocas para el modelo 1.

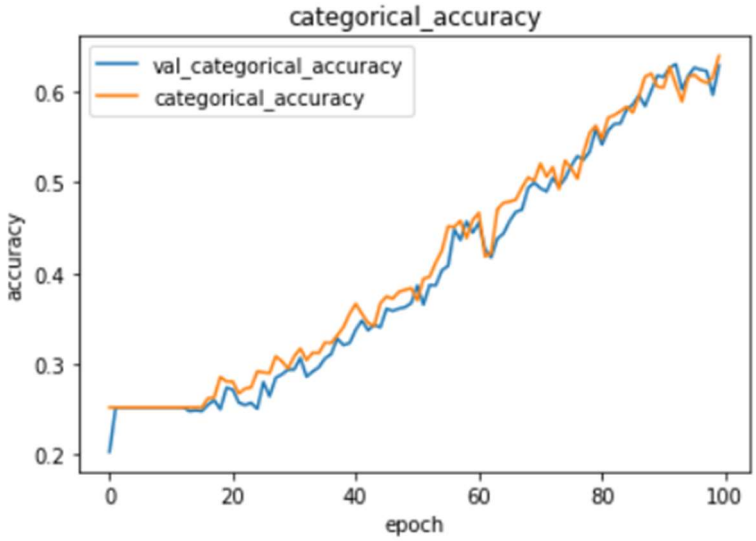

Figura 17. Representación de la precisión alcanzada por el modelo 2.

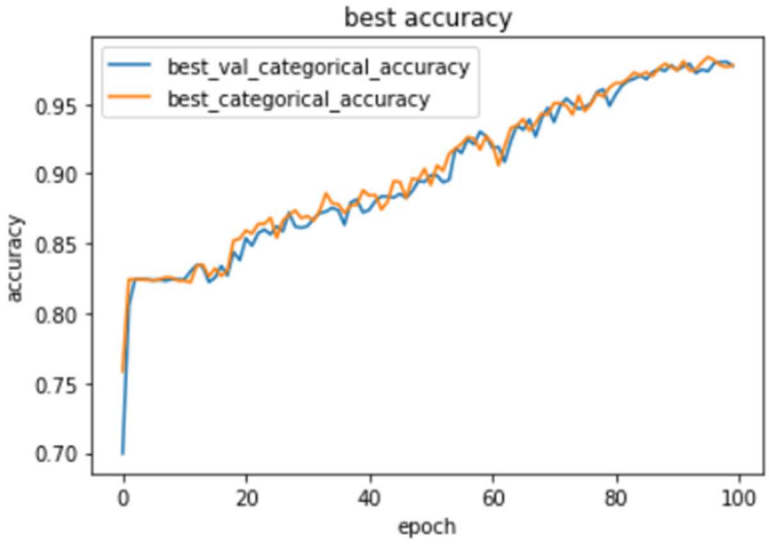

Figura 18. Representación del mejor valor promedio por categoría para el modelo 2.

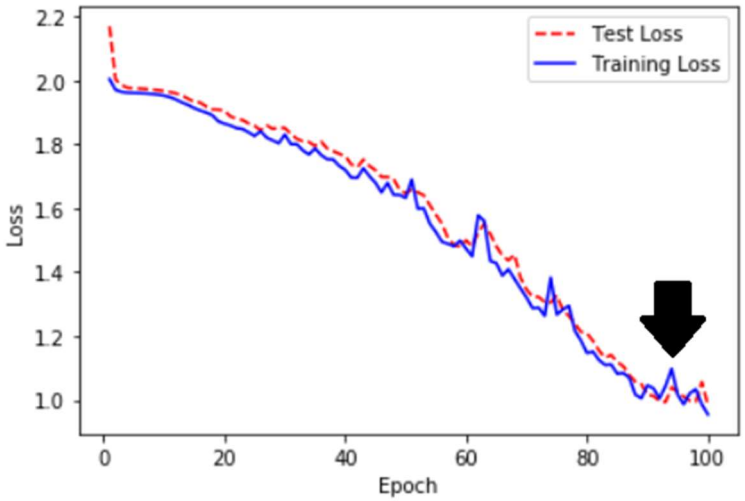

Figura 19. Evolución del error a lo largo de las distintas épocas para el modelo 2.

Si discriminamos únicamente los resultados para las clases mayoritarias vemos como estas rápidamente son aprendidas por la red obteniendo predicciones que se acercan incluso al 100% de clasificaciones correctas (ver figura 18). La diferencia con las clases menos representadas nos indica de nuevo una incapacidad por parte de la red para obtener información suficientemente representativa de aquellas clases con menor número de ejemplos.

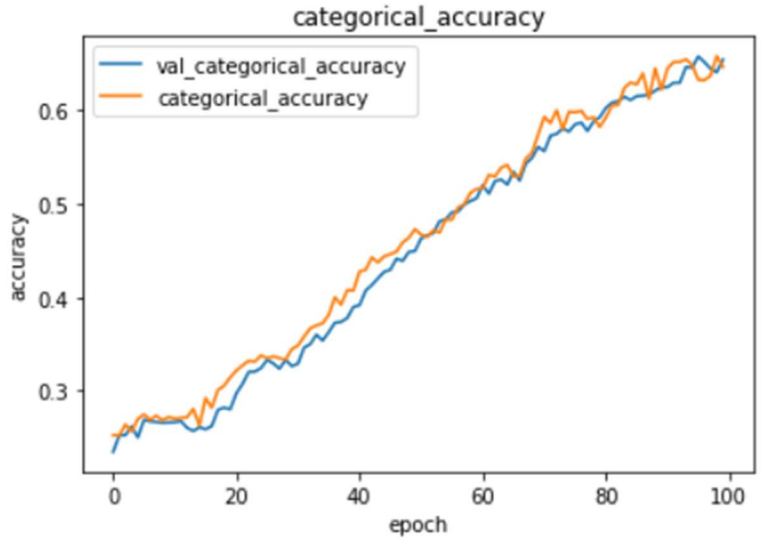

Figura 20. Representación de la precisión alcanzada por el modelo 3.

La curva de aprendizaje se mantiene sostenida en el tiempo y no muestra signos de sobreentrenamiento situándose en algunas ocasiones a partir de la época 90 mejores valores para el grupo de testing (ver figura 20).

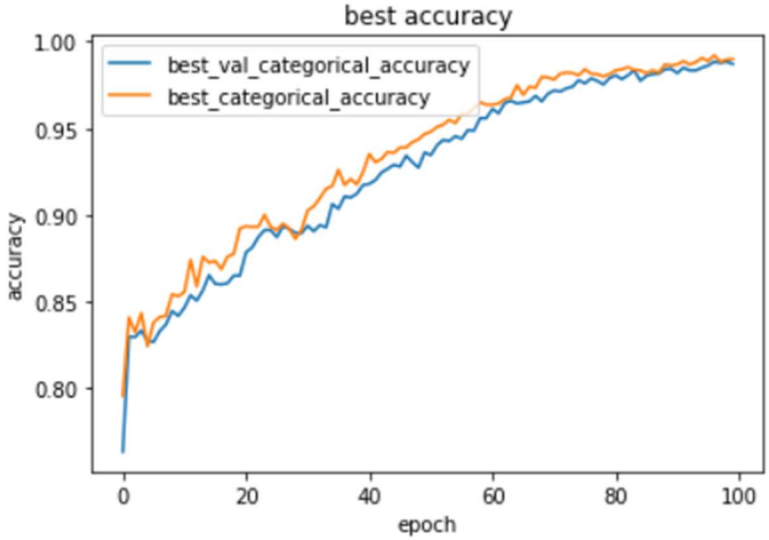

Figura 21. Representación del mejor valor promedio por categoría para el modelo 3.

Resultados del experimento con el modelo 3 (CNN multi-channel w/variable size)

Los resultados en cuanto a precisión del modelo 3 son muy parecidos a los del modelo anterior mostrando resultados irregulares según la clase siendo de en torno a un 60% en precisión promedio y superiores al 95% en aquellas clases mejor conocidas o de más fácil diferenciación (véase figuras 20 y 21).

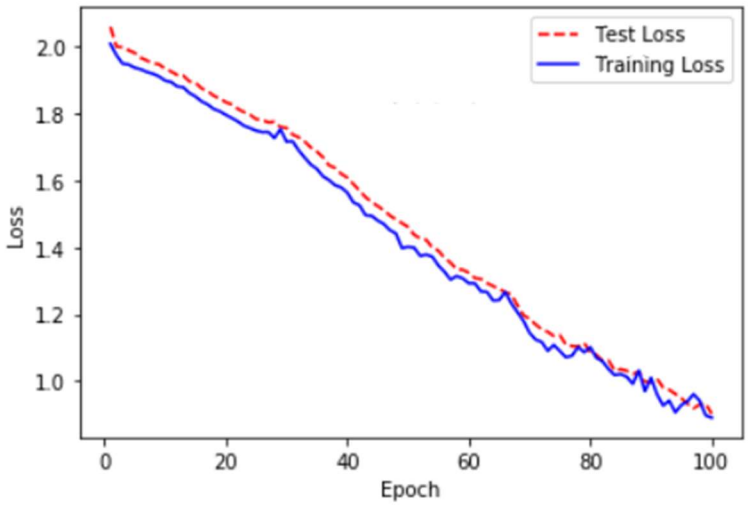

Figura 22. Evolución del error a lo largo de las distintas épocas para el modelo 3.

De nuevo una rápida convergencia y tasas de reconocimiento muy altas en las clases mayoritarias. Resulta llamativa la rapidez de convergencia de estas que ya en épocas muy tempranas presentaba una precisión en el reconocimiento por encima de un 90%, una tasa muy alta si tenemos en cuenta el gran número de atributos de clase.

La curva de aprendizaje en este caso el más suave y progresiva en las primeras iteraciones, pero su evolución es mucho más estable y predecible teniendo una progresión cuasi-lineal con pocas oscilaciones (ver figura 22).

Resultados del experimento con el modelo 4 (Multiplicative LSTM)

En nuestras pruebas con el modelo la *multiplicative* RNN tuvo resultados discretos, aunque si podemos indicar que el modelo mantuvo una progresión de aprendizaje correcta incluso después de recuperarse de varios momentos de fracaso (señalados en rojo en la figura 23) lo que nos hace pensar que el modelo es lo bastante flexible como para aceptar cierto nivel de sorpresa en la entrada y aprender de esta nueva diversidad. Aunque todos los modelos fueron testados en igualdad de condiciones por razones de equidad, parece razonable pensar que el rendimiento del modelo podría haber mejorado de haber tenido una parametrización más cuidada. La situación comentada se repite de nuevo para las clases mayoritarias (ver figura 24) siendo curioso que la capacidad predictiva para estas se desplome también entorno a la época 25 a la vez que lo hace la media por categorías.

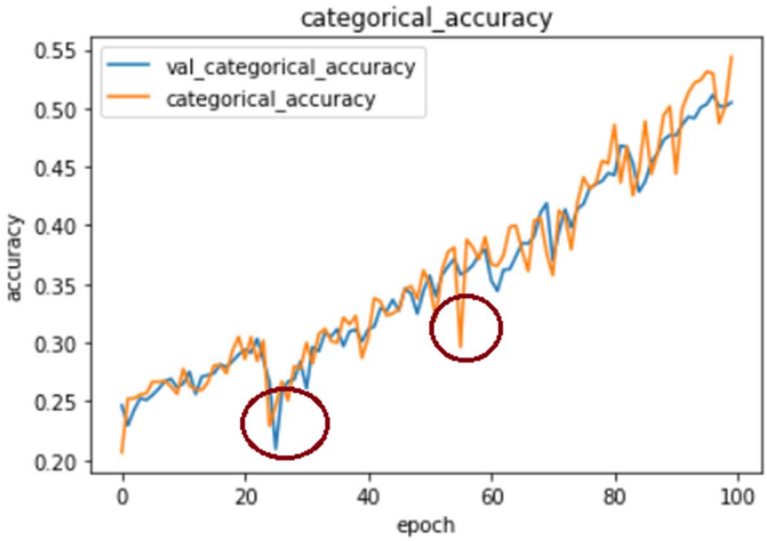

Figura 23. Representación de la precisión alcanzada por el modelo 4.

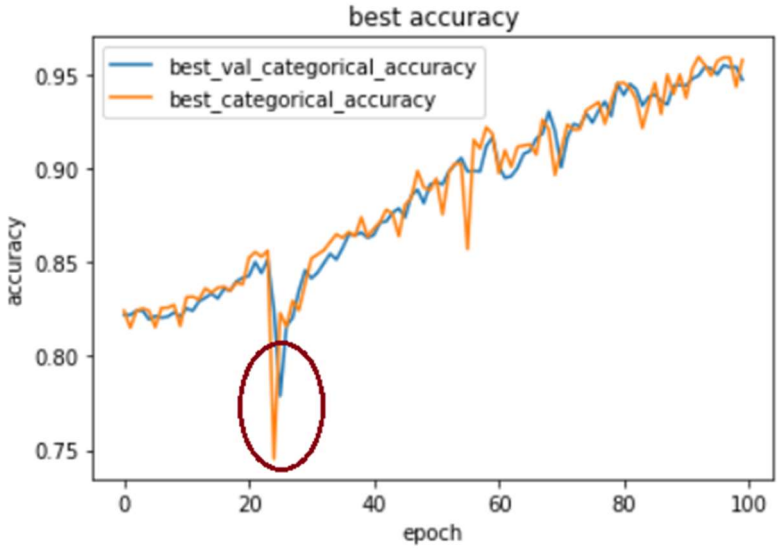

Figura 24. Representación del mejor valor promedio por categoría para el modelo 4.

En la gráfica de errores, nuevamente encontramos fuertes oscilaciones que aun así permiten converger al modelo de forma clara si bien este no es tan estable como en los modelos 2 y 3. Destacable las fluctuaciones alrededor de la época 80, muy avanzado en el entrenamiento del algoritmo donde cabría esperar oscilaciones más tenues conforme el método converge (figura 23 en rojo).

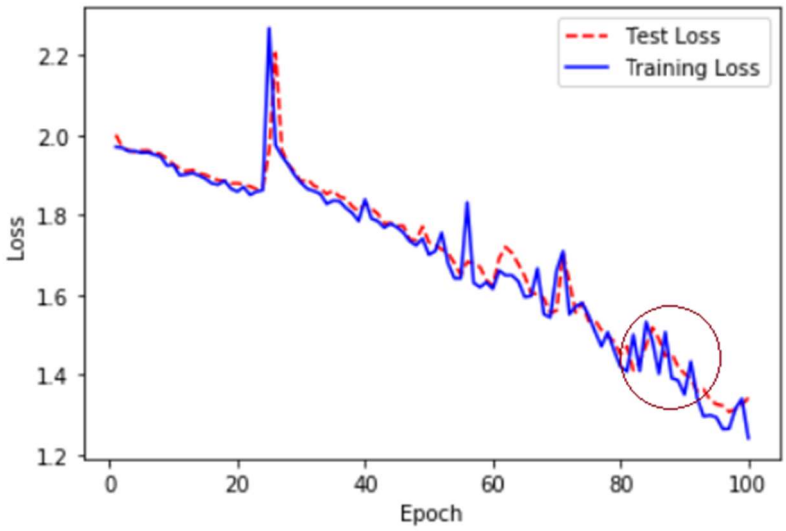

Figura 25. Evolución del error a lo largo de las distintas épocas para el modelo 4.

A continuación, exponemos los resultados obtenidos mediante nuestro modelo. Aunque la precisión del modelo parece aumentar de forma más pausada podemos apreciar como esta aumenta de forma sostenida, con pocas oscilaciones y sin presentar fracasos ni retrocesos en el aprendizaje hasta situarse por encima de los cuatro modelos anteriores en torno a la época 90 (ver figura 25).

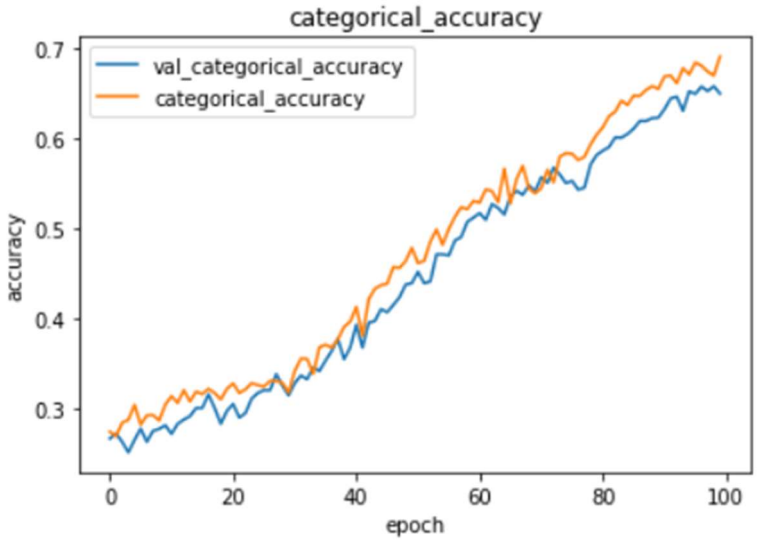

Figura 26. Representación de la precisión alcanzada para el modelo propuesto.

La precisión alcanzada en las clases predominantes alcanza valores muy altos a las pocas iteraciones y como los modelos anteriores esta crece rápidamente hasta alcanzar tasas predictivas cercanas al 100% dado que están extensamente representadas en el conjunto de datos.

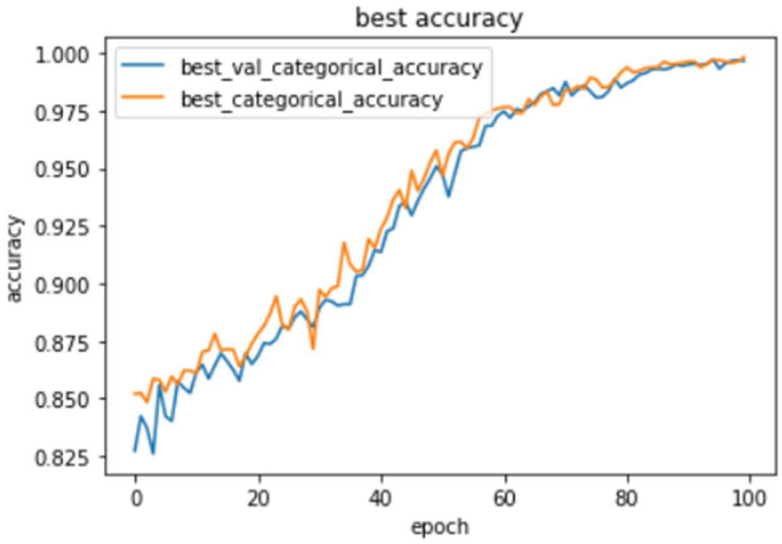

Figura 27. Representación del mejor valor promedio por categoría para el modelo propuesto.

En cuanto a la curva de aprendizaje podemos observar un buen comportamiento con un descenso predecible y estable de la tasa de error conforme se presentan las entradas a la red (ver figura 27).

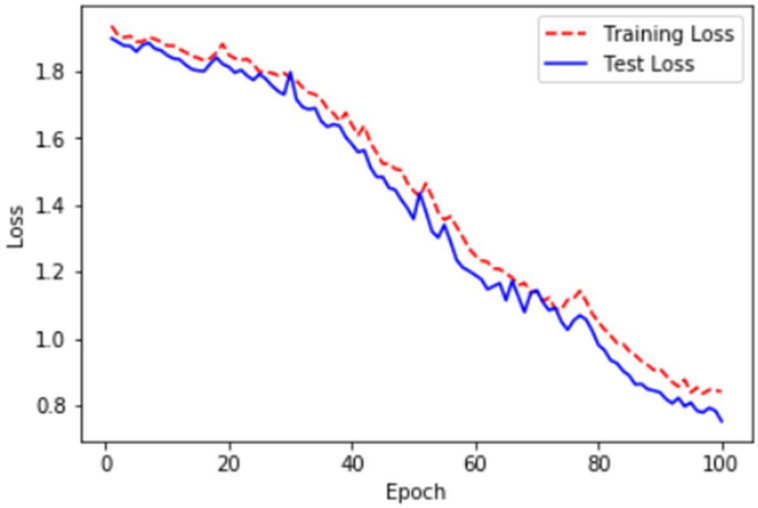

Figura 28. Evolución del error a lo largo de las distintas épocas para el modelo propuesto.

De los datos anteriores parece concluirse que el comportamiento de los cuatro modelos de referencia fue peor en las pruebas realizadas, alcanzando valores de precisión para la clase inferiores comportándose peor ante la introducción de nuevos patrones y generando curvas de aprendizaje más irregulares. Estos resultados se discutirán en los dos próximos capítulos, así como la importancia de estos en futuras líneas experimentales.

Capítulo VI

Resultados.

6. Análisis de los resultados obtenidos

Para entrenar el modelo, omitimos el 10% del conjunto de datos (alrededor de 350 registros médicos), que se utilizaron para la validación. La Figura 29 muestra los resultados de las diferentes fases de capacitación y validación de nuestro modelo en comparación con los tres modelos más conocidos disponibles (ver capítulo 5). Limitamos la CNN a un máximo de 100 épocas con un tamaño de lote de 128 Ambos valores se establecen experimentalmente. Establecer el primero nos permite comparar los diferentes algoritmos bajo las mismas condiciones de ejecución. Mientras que el segundo valor tiene una influencia meramente práctica, ya que su elección afecta los tiempos de entrenamiento.

Encontramos que en nuestro modelo la precisión conseguida cuando se alcanza el criterio de detención supera el 68.0%. En la época 1 se obtiene la precisión mínima de entrenamiento del 21%. En la época 50, la precisión máxima de entrenamiento es superior al 50% y en la época 100, la precisión máxima de validación es del 64,5%. La tasa de aprendizaje (*training loss*) para este caso es de aproximadamente 0.870. Es notable que nuestro modelo siempre se mantenga arriba, sea más estable y logre mejores objetivos de precisión por clase (cerca de 5-10% dependiendo de si se consideran los resultados promedio o solo el mejor resultado) que los modelos de referencia mencionados anteriormente (ver detalles en la tabla 4). La Figura 30 muestra la matriz de confusión obtenida durante la fase de validación de nuestro modelo. Además, con respecto a los tiempos de entrenamiento, nuestro modelo tarda 1 hora en la fase de preentrenamiento, mucho menos que las 4 horas consumidas en el entrenamiento del modelo MV-CNN y algo por encima del tiempo consumido por CNN-GRU (véase figura 31).

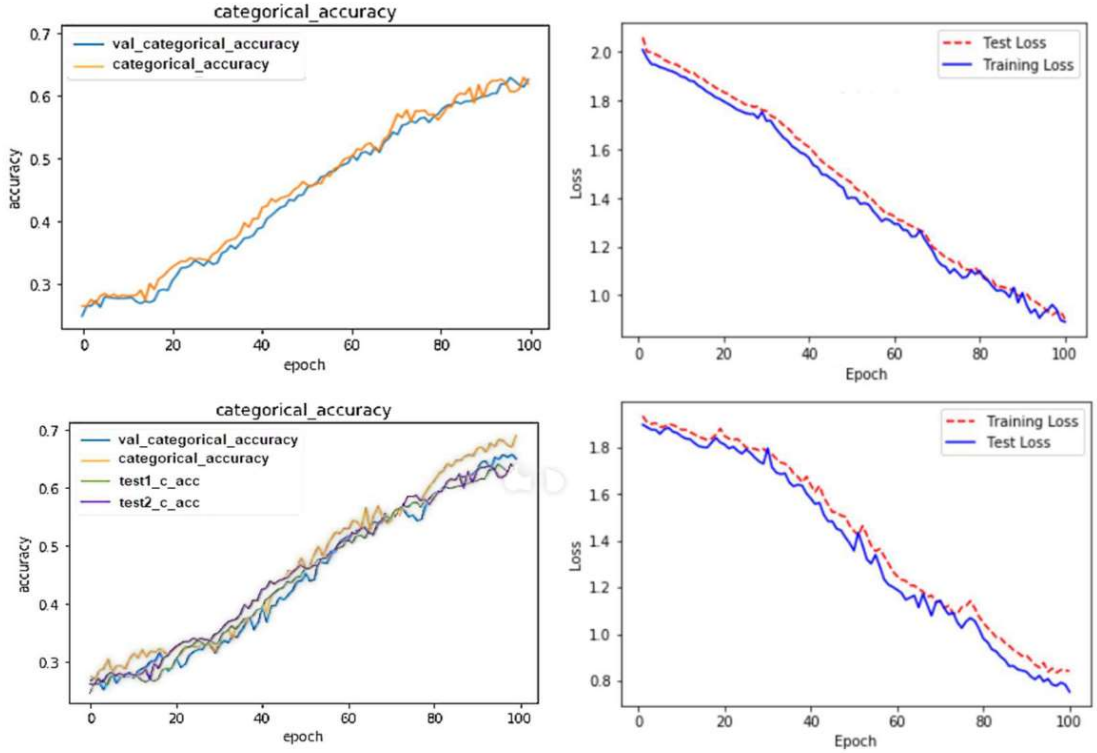

Figura 29. Arriba. Precisión y distribución de pérdidas por época para el mejor modelo de referencia sobre el conjunto de datos (Yin et al., 2017). Abajo. Precisión (fases de training, validación y prueba) y distribución de pérdidas por época para nuestro modelo.

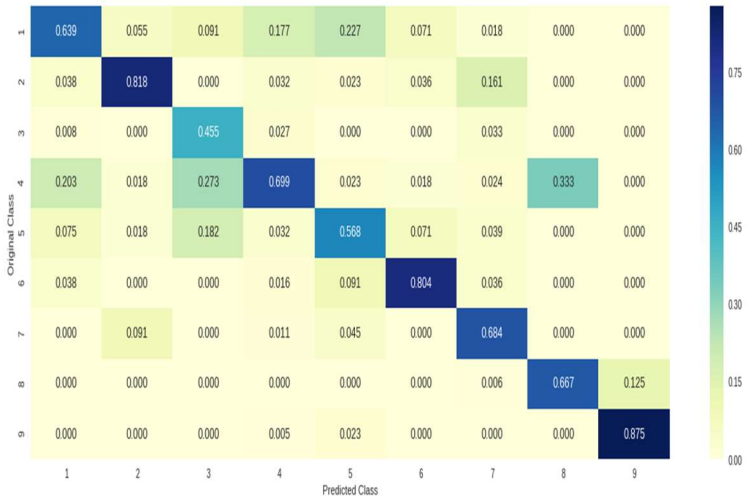

Figura 30. Matriz de confusión obtenida durante la fase de test de nuestro modelo.

Descubrimos que usar el modelo Fasttext entrenado en subconjuntos del texto era una buena solución. Las interdependencias entre los términos se construyeron a partir de diferentes diccionarios de terminología clínica, lo que nos permitió lograr una mayor riqueza semántica (los resultados exactos se pueden cotejar en la tabla 4). Los pesos de nuestro modelo semántico se actualizaron durante el entrenamiento, pero los pesos se inicializaron con el modelo Fasttext aprendido. De esta manera, se permite la transferencia de la semántica distributiva aprendida, pero esto no es limitante, por lo que nuestras circunvoluciones imitan el modelo de cooperación multicanal y los filtros de tamaño variable de Kim et al. pero, en nuestro caso, el multicanal no se basó en una versión única de la incrustación previamente entrenada para obtener representaciones más adecuadas.

Table 4. Resultados de nuestra arquitectura híbrida frente a los distintos modelos de referencia.

Algorithm	Final Validation Loss	Mean Validation Accuracy
Glove (general eng[9].) + XGBoost	1.56	37%
FastText (adhoc) + XGBoost	1.37	49%
CNN Var.Conv. (Yin et al., 2016)	1.20	54%
CNN + RNN (Wang at al., 2017)	1.58	42%
CNN + LSTM (Chen et al., 2017)	1.07	57%
Multiplicative LSTM (Krause et al., 2017)	1.12	60%
Our model	**0.83**	**69%**

Creemos que el rendimiento mejorado de nuestro algoritmo se beneficia principalmente de los siguientes aspectos:

(1) Los vectores distributivos de alta dimensión dotan a palabras similares de orientación semántica de gran similitud. Las embedding multicanal pueden resolver mejor la escasez semántica de textos cortos en comparación con la representación única.

[9] https://code.google.com/archive/p/word2vec/

(2) Al igual que ocurre en rotación e invariancia de escala de imágenes en CNN, una CNN puede aprender las características locales de palabras o frases en diferentes lugares de textos.

(3) La RNN puede trabajar de forma colaborativa a la red convolucional, la RNN toma palabras en una oración en orden secuencial y puede aprender las dependencias a largo plazo de los textos en lugar de las características locales.

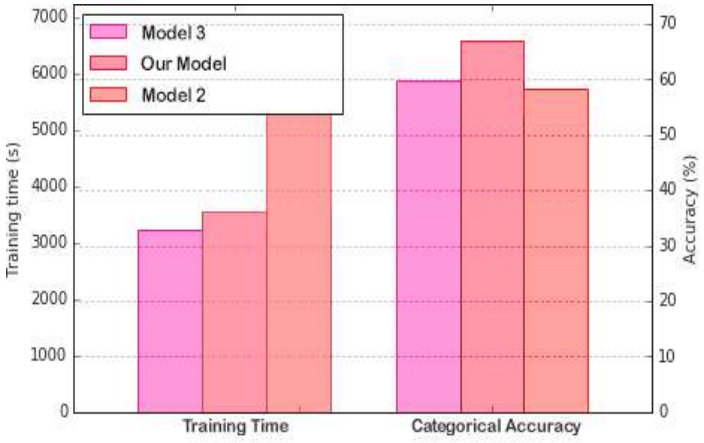

Figura 31. Tiempos de entrenamiento de nuestro modelo contra los dos mejores modelos de referencia medidos en segundos.

Estos puntos se desarrollarán en detalle en la siguiente sección y en conclusiones.

Capítulo VII

Discusión.

7. Discusión.

En la sección anterior hemos visto cómo nuestros experimentos en el conjunto de validación ofrecen resultados prometedores. Los resultados de cada experimento se obtuvieron mediante una validación cruzada 10-fold, donde los ejemplos se dividieron aleatoriamente en los 10 grupos. Los resultados experimentales muestran que nuestro modelo alcanza una precisión promedio cercana al 70%.

Del análisis de la figura 30 también podemos concluir que, aunque las clases 3, 4, 8 y 9 estaban insuficientemente representadas, nuestro modelo obtuvo resultados aceptables para estas clases y más de la mitad de las entradas de datos para la clase 9 se predijeron correctamente, en Comparación con los otros modelos. Como puede ver, todavía hay problemas con algunas de las clases que son demasiado similares o muy pocas en el conjunto de datos de referencia. Aun así, los resultados son los esperados, con la mayoría de las clasificaciones acumuladas alrededor de la diagonal principal.

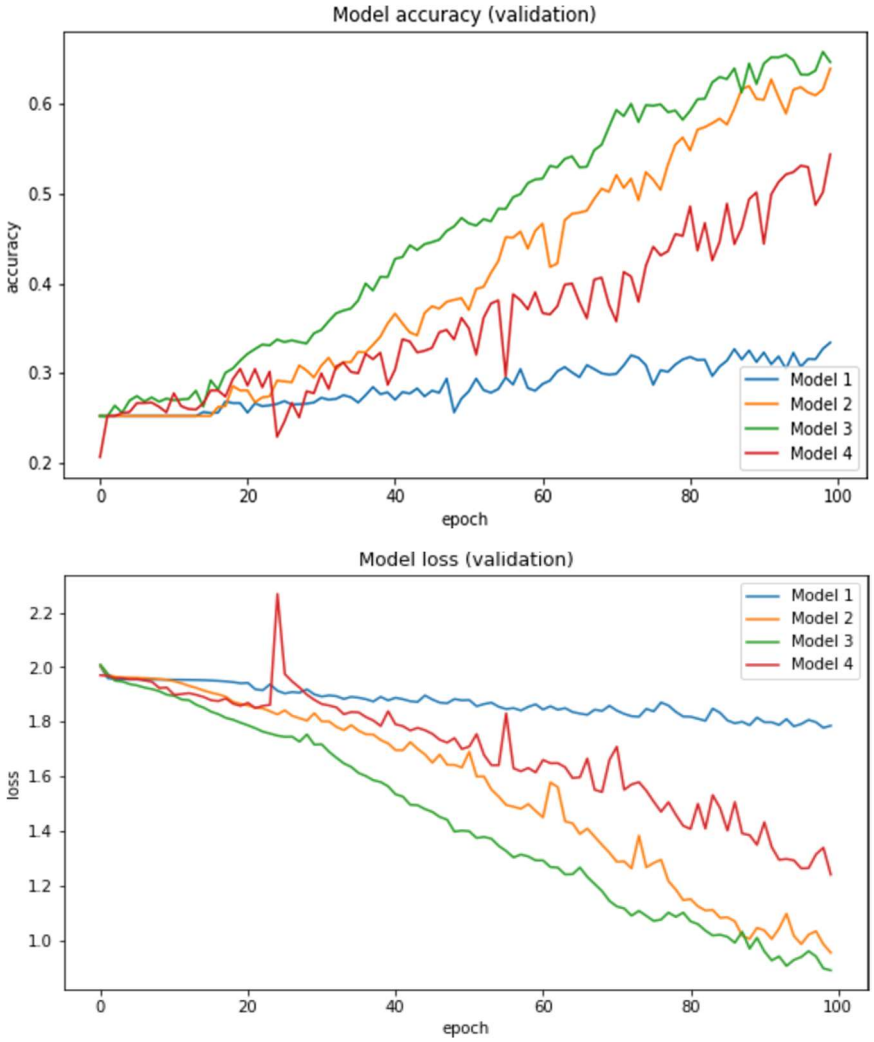

Figura 32. Precisión (Arriba) y tasa de aprendizaje (Abajo) para nuestros modelos de referencia: Chen et al. 2017 (azul), Wang et al., 2017 (naranja), Yin et al., 2016 (verde), Krause et al., 2017 (rojo).

Tomamos los modelos de Chen et al. 2017 (figura 32. azul), Krause et al. 2017 (figura 32. rojo), Wang et al. 2017 (figura 32. naranja) y Yin et al. 2016 (figura 32. verde) como referencia de comparación debido a su valor, respaldado por su impacto en la literatura disponible y la novedad de su mecanismo y no solo en sus respectivos resultados, simplemente porque muchos de ellos nunca han sido probados en tal conjunto de datos tan complejo. Cuando se compara con estos modelos de referencia, nuestro estudio muestra que Chen et al. no logra capturar características distintivas dentro de los textos, obteniendo un bajo rendimiento que no supera el 30% de éxito en ninguna clase. Como en nuestro modelo, el propósito de Chen también se centró en proporcionar una sinergia entre los dos tipos de redes utilizadas (la red convolucional y la RNN), de modo que se capturara la semántica global y local. Sin embargo, de los resultados con el conjunto de datos MSKCC se deduce que el método de Chen no pudo abordar un conjunto de datos tan grande y, por lo tanto, está lejos de lo que se puede lograr utilizando métodos generales de embedding genéricos. Estos resultados nos llevan a pensar que este modelo no logra obtener la semántica global involucrada o su ausencia de un mecanismo de atención multinivel hace que la información contextual recopilada sea insuficiente. Serían necesarias más pruebas, pero es posible que solo un enfoque híbrido pueda ser capaz de manejar textos de la complejidad de nuestro dataset al requerir la implementación de mecanismos de persistencia para identificar dependencias de texto remotas.

Los modelos propuestos por Wang et al. y Krause et al. obtienen resultados moderados, aunque muestran una tendencia a la inestabilidad con algunos casos de fallas durante las pruebas de entrenamiento y validación, donde el rendimiento cae repentinamente. Ambos modelos utilizan una arquitectura híbrida CNN y RNN con cierta similitud, ya que en ambos casos el propósito es capturar características locales de CNN de grano grueso y dependencias de larga distancia utilizando la red RNN. Nuestro modelo evitó este problema utilizando modelos semánticos específicamente entrenados para esto. El modelo de Krause et al. sin embargo, muestra un comportamiento mucho mejor durante la validación, mientras que el modelo de Wang et al. oscila dentro de un rango de 10-15%, como se puede ver en la Figura 29. Sin embargo, ambos modelos son demasiado inestables. En nuestro modelo, agregar un mecanismo de atención para guiar la búsqueda contribuye a obtener un aprendizaje más estable y un mejor comportamiento hacia las clases minoritarias.

En nuestro caso, la inclusión de un mecanismo de atención produce un mejor rendimiento, unos menos requerimientos de memoria RAM y menores requisitos de CPU al permitir que el modelo se centre en aquellas relaciones que son las más importantes para la clasificación. Como se discutió anteriormente, el modelo MV-CNN de Yin et al. obtiene el mejor desempeño de los 4 modelos de referencia debido al uso de diferentes incrustaciones de palabras pre-entrenadas. Yin y col. combina diferentes versiones de Word embedding, extrae características de frases de granularidad diferente con convoluciones de tamaño variable e incluye una arquitectura LSTM multiplicativa, que se caracteriza por tener diferentes funciones de transición recurrentes para cada entrada posible, por lo que hace un buen uso del uso de las incrustaciones múltiples. Todo esto da lugar a un aprendizaje más rápido y mejores tasas de rendimiento de alrededor del 65% durante las pruebas de validación. Pero cuando comparamos el modelo MV-CNN (modelo de referencia más prometedor) con nuestros resultados en el conjunto de datos MSKCC (línea verde en la figura 32), encontramos que nuestro modelo muestra un mejor comportamiento dinámico; manteniéndose siempre arriba, más estable y aun así obteniendo mejores objetivos de precisión por clase. A partir de estos resultados, argumentamos que nuestra propuesta ofrece cinco beneficios prácticos clave que son una contribución al estado del arte:

Captura mejor información contextual. Modelos como Word2vec o Glove son independientes del contexto. Esto significa que estos modelos generan solo un vector de embedding (incrustación) para cada término, combinando todos los diferentes significados de la palabra en un vector, independientemente de dónde se ubicaron las palabras en una oración y los diferentes significados que puedan tener. Por ejemplo, si después de entrenar un modelo word2vec en un corpus (entrenamiento no supervisado) obtenemos una representación vectorial para el término "enfermedad cardíaca", y hay una oración como "el paciente tiene antecedentes de enfermedad cardíaca inflamatoria, aunque no hay enfermedad cardíaca obstructiva, "donde el término" enfermedad cardíaca "tiene diferentes significados dependiendo del contexto de la oración, estos modelos simplemente los colapsan en un solo vector. De esta manera, la semántica completa del término no se capturará y habrá información faltante al aplanar el término. Sin embargo, nuestro modelo puede generar diferentes vectores incrustados de palabras que capturan el contexto de una palabra y su posición en una oración. Por ejemplo, para el mismo ejemplo anterior, la red generaría diferentes incrustaciones para los dos usos de ese término. El primer uso (enfermedad cardíaca inflamatoria) estaría más cerca de palabras como miocarditis o insuficiencia, etc., mientras que el segundo uso (enfermedad cardíaca obstructiva) estaría más cerca de palabras como lípidos, colesterol o infarto de miocardio, etc.

Se considera el orden de aparición. Los enfoques tradicionales de NPL no tienen en cuenta el orden de las palabras en su capacitación. Nuestro modelo, basado en una red recurrente con unidades de memoria, por la naturaleza misma de cómo funciona una red LSTM, tiene en cuenta el orden de las palabras. Además, una de las cualidades de implementar un mecanismo de atención es que estos pueden actuar como codificadores posicionales para ponderar una secuencia ordenada sobre otras.

Captura mejor los diferentes significados del vocabulario específico. Por las mismas razones dadas en el punto anterior, nuestro modelo es más preciso al capturar usos de vocabulario o terminología específicos que adquieren un significado diferente en el campo médico. Por ejemplo, en el texto "Los tuberculomas son opacidades nodulares persistentes, que pueden ser redondas u ovaladas, ubicadas con mayor frecuencia en los lóbulos superiores; la mayoría son regulares y están bien delimitados, aunque hasta el 25% de los casos pueden tener contornos difusos". Podemos ver varios usos de este tipo de vocabulario (opacidades, nodular, regular, difuso, etc.) que no se usan en lenguaje coloquial o varían su significado para este tipo de contexto.

Mayor precisión con menos entrenamiento supervisado. Los algoritmos tradicionales de PLN utilizan una matriz de recuento de coincidencias para crear vectores de incrustación. Cada fila de la matriz representa una palabra, mientras que cada columna representa los contextos en los que pueden aparecer las palabras. Los valores en la matriz representan la frecuencia con la que aparece una palabra en cada contexto. Luego, se aplica la reducción de dimensionalidad a esta matriz para crear la matriz de embedding resultante (cada fila será el vector de incrustación de una palabra). Nuestro modelo, aunque utiliza en una primera etapa un modelo Fasttext (también basado en incrustaciones de palabras), aplica un modelo de lenguaje LSTM bidireccional profundo para aprender qué términos son predictivos y su contexto. La arquitectura Bi-LSTM permite al modelo aprender aspectos más complejos y / o dependientes del contexto de los significados de las palabras en las capas superiores junto con aspectos de sintaxis en las capas inferiores. Esto da como resultado una mejor extracción semántica de los textos, lo que proporciona un mejor rendimiento con un uso más reducido de la CPU (los tiempos de entrenamiento detallados se pueden comparar en las figuras 31 y 32).

Características dinámicas y tolerancia a términos desconocidos. Nuestro modelo utiliza hasta 3 matrices de incrustación diferentes que también admiten reentrenamiento en línea. Esto significa que el modelo no solo tiene más libertad a la hora de adaptarse a las diferentes granularidades del

texto, sino que, al ser reentrenable, la red reacciona mejor a términos desconocidos y les permite incorporarse cuando aumenta su peso relativo.

Capítulo VIII

Conclusiones del estudio.

8. Contribuciones de la investigación y líneas futuras

En este trabajo hemos propuesto e implementado una arquitectura híbrida para la clasificación de textos biomédicos no estructurados. En primer lugar, basamos nuestro trabajo en explorar el contexto del problema y las características especialmente complejas de este tipo de textos. El conjunto de datos MSKCC ha servido como modelo de referencia en la literatura para comparar diferentes soluciones a ese problema. Es un conjunto de datos complejo por muchas razones, y en particular por la longitud de sus textos, que en nuestro conjunto de datos supera las 2000 palabras en muchos casos (véase como en la figura 6 el pico de datos se alcanza alrededor de las 750 palabras, pero existe una larga cola que llega hasta 4000 palabras en algunos casos). También su parecido entre clases al no haber una distinción temática clara supuso un desafío (véase figura 8). Es por eso qué era tan importante unir en una sola arquitectura la capacidad de obtener una gran capacidad de extracción contextual y semántica, la capacidad de explorar características de granularidad diferente y una suficiente capacidad espacial y temporal, que permiten que el método sea sensible al contexto local a la vez que posee capacidades para memorizar secuencias o apariciones distantes en el tiempo. Como hemos descrito en este documento, nuestra arquitectura combina características de otras propuestas que han sido reconocidas por su valor en áreas como la inclusión de palabras, redes de convolución multicanal, redes LSTM (especialmente las unidades de memoria GRU) y mecanismos de atención. La eficiencia de los diferentes métodos de referencia se comparó con la arquitectura propuesta y sus actuaciones se discutieron en este documento. Del análisis de los resultados obtenidos durante nuestra experimentación se deduce que nuestro modelo muestra un comportamiento sólido y estable. Los resultados experimentales obtenidos del conjunto de datos de referencia (MSKCC) demuestran que nuestro modelo supera a otros modelos de referencia recientes y de última generación. Nuestra propuesta de arquitectura es más precisa que todos los modelos de referencia probados y requiere menores tiempos de cálculo y requisitos de memoria. A partir de estos resultados, sostenemos que nuestra propuesta unifica varias de las características de estos modelos que hacen que este nuevo enfoque sea interesante desde un punto de vista práctico, lo que proporciona una contribución al estado del arte.

Objetivos alcanzados

Con este trabajo nos propusimos testar la aplicación de redes convolucionales a la clasificación de documentos médicos a partir del tratamiento en bruto de historias en lenguaje natural. Para llevar a cabo este punto se implementaron distintas arquitecturas descritas en el estado del arte que cuyo rendimiento fue estudiado dentro de un entorno estandarizado de pruebas. Sobre este trabajo se llevan a cabo una serie de mejoras para lo cual proponemos una arquitectura hibrida ligera con una etapa convolucional (CNN) como etapa predictora y una etapa recursiva de tipo LSTM encargada de aportar enriquecimiento contextual a la clasificación. Como objetivo principal del estudio, nos proponemos conseguir resultados de precisión en weighted-average-accuracy comparables a los últimos algoritmos del estado del arte, pero con una mucho menor carga computacional gracias a un mayor esfuerzo en preprocesamiento de los datos y a la no

utilización de vectores cuantificación preentrenados. Este objetivo se ha alcanzado, demostrando que es posible mejorar la precisión del modelo sin disparar la carga computacional.

También se alcanzaron los objetivos funcionales F1 a F5 y metodológicos M1 a M5 descritos en el capítulo 1 y establecidos como objetivos secundarios del estudio.

Como resultado del proyecto se incluye también el desarrollo de dos demostradores software, cuya implementación permite ofrecer a toda la comunidad científica la posibilidad de probar el sistema y cuya documentación y disponibilidad están abiertos para seguir avanzando en esta área.

Los principales resultados de este estudio se reflejan en un artículo científico (adjunto a la memoria) para su admisión en una revista de prestigio "**Artificial Intelligence in Medicine**" (JCR Q2 *Artificial Intelligence*, Q1 *Medical Informatics*, Q1 *Engineering, Biomedical*), con la finalidad de difundir a la comunidad internacional las aportaciones de nuestra investigación.

Hipótesis validadas

Los resultados del trabajo permiten validar nuestras hipótesis iniciales y afirmar que las propiedades intrínsecas de las redes neuronales para abordar problemas con un alto factor de escalado y las capacidades de las redes convolucionales híbridas para tratar con información desestructurada y no anotada, y extraer interrelaciones; como ya apuntaban los resultados prometedores de los primeros trabajos en el área (**Ö.Uzuner & DuVall, 2011**; **Rebholz-Schuhmann et al., 2013**; **Yoon et al., 2014**; **Kim et al., 2014**; **Mikolov et al., 2016**) hace a los algoritmos convolucionales adecuados como herramienta de trabajo para el procesamiento de textos clínicos mostrándose capaces de extraer la carga semántica imprescindible que permita guiar la exploración entre documentos que guardan cierto grado de relación.

Parece razonable afirmar que los buenos resultados obtenidos en nuestros experimentos se deben al manejo específico que se hace de la terminología clínica y a que no hacemos uso de mapas semánticos generalistas que, como establecía nuestra hipótesis inicial, no parecen ser adecuados para el procesamiento automático de lenguaje médico y su uso redunda en peores tasas de sensibilidad del método y de F1-score en su conjunto. Por último, nuestros resultados confirman que el desbalanceo de las clases afecta negativamente a las capacidades clasificadoras capaces de ser obtenidas por el modelo entrenado creando oscilaciones y haciendo que el aprendizaje sea inestable y propenso a fallas. Por tanto, aplicar medidas correctivas para limitar el desbalanceo del conjunto de entrenamiento repercute efectivamente en un mejor rendimiento del modelo.

Puntos abiertos

El campo de estudio donde se engloba este trabajo es aún un área de investigación en desarrollo con un gran potencial debido a las repercusiones que tendrá en la forma en que se organizan y trabajan los profesionales médicos en todo el mundo. El obtener buenos algoritmos de procesamiento y clasificación de documentación clínica redunda en enormes beneficios sociales y será un área activa de investigación durante los próximos años. Debido a la finalidad y al alcance de este trabajo muchos aspectos no fueron abordados o se trataron de forma escueta. El posible escalado del trabajo desarrollado para abordar otros tipos de documentación clínica, el desarrollo de una herramienta en producción para poner a prueba el modelo en con usuarios reales, o la generación de recomendaciones y contenido clínico basándonos en la preclasificación del documento son algunos aspectos que podrían constituir una tesina de máster en sí mismos. Son muchas las posibilidades de ampliación de este trabajo también enfocadas al aprendizaje asistido y al aprendizaje colaborativo. Por ejemplo, un estudiante de medicina puede aprovechar las

ventajas de un recomendador clínico para aprender a identificar patrones y patologías o para ayudarle en el diagnóstico asistido y a la vez, profesionales experimentados pueden encontrar en la herramienta una forma de volcar y compartir sus conocimientos.

En cuanto a la arquitectura propuesta, podría ser muy interesante testar la misma sustituyendo las capas LSTM por la arquitectura LSTM multiplicativa propuesta por (**Krause et al.,2016**) dado el rendimiento tan bueno alcanzado por esta. Este enfoque contempla dotar de mayor flexibilidad a los elementos LSTM de forma parecida a las RNN generativas que combinan unidades LSTM con pesos ocultos factorizados multiplicativos RNN, lo que permite transiciones flexibles dependientes de entrada que son más fáciles de controlar debido a las unidades de compuerta de LSTM. El LSTM multiplicativo parece ser más apropiado cuando puede aprender parámetros específicamente para cada entrada posible en un paso de tiempo dado. Por lo tanto, su aplicación podría ser sumamente interesante y compatible con las propuestas de este trabajo.

Estas líneas pendientes pretenden ser la base del posterior desarrollo de una tesis doctoral futura centrada en estas oportunidades y que constituye un campo de investigación aún abierto y con posibilidades de ampliación futuras dentro del área del guiado del aprendizaje médico y del manejo experto de fuentes desestructuradas de información en biomedicina con la asistencia de sistemas inteligentes.

Capítulo IX

Bibliografía y anexos.

9. Bibliografía

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Kudlur, M. (2016). Tensorflow: a system for large-scale machine learning. OSDI, Vol. 16, pp. 265-283.

Astels, D. (2003). Test driven development: A practical guide. Prentice Hall Professional Technical Reference.

Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications (Vol. 27, No. 2, pp. 94-105). ACM.

Aguilar Nieto, C. A., & Soriano Taboada, D. O. (2017). Comparar modelos diseñados mediante redes neuronales para reconocimiento de patrones en imágenes médicas y detección de cáncer de mamas.

Allen Jr, Bibb, et al. "A road map for translational research on artificial intelligence in medical imaging: from the 2018 National Institutes of Health/RSNA/ACR/The Academy Workshop." Journal of the American College of Radiology 16.9 (2019): 1179-1189.

Baud, R. H., Rassinoux, A. M., & Scherrer, J. R. (1992). Natural language processing and semantical representation of medical texts. Methods of information in medicine, 31(02), 117-125.

Bermejo Peláez, D., San José Estépar, R., & Ledesma Carbayo, M. J. (2016). Detección y clasificación de enfisema pulmonar en imágenes de TAC mediante Redes Neuronales Convolucionales Multiescala.

Bespalov, Dmitriy, et al. "Sentiment classification based on supervised latent n-gram analysis." Proceedings of the 20th ACM international conference on Information and knowledge management, 2011.

Bhandare, A., Bhide, M., Gokhale, P., & Chandavarkar, R. (2016). Applications of convolutional neural networks. International Journal of Computer Science and Information Technologies, 7(5), 2206-2215.

Bodenreider, O. (2004). The unified medical language system (UMLS): integrating biomedical terminology. Nucleic acids research, 32(suppl_1), D267-D270.

Carrell, D. S., Halgrim, S., Tran, D. T., Buist, D. S., Chubak, J., Chapman, W. W., & Savova, G. (2014). Using natural language processing to improve efficiency of manual chart abstraction in research: the case of breast cancer recurrence. American journal of epidemiology, 179(6), 749-758.

Chen, G., Ye, D., Xing, Z., Chen, J., & Cambria, E. (2017, May). Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In 2017 International Joint Conference on Neural Networks (IJCNN) (pp. 2377-2383). IEEE.

Chen, X., et al. "Adversarial deep averaging networks for cross-lingual sentiment classification." Transactions of the Association for Computational Linguistics 6 (2018): 557-570.

Choi, Jaekeol, and Sang-Woong Lee. "Improving FastText with inverse document frequency of subwords." Pattern Recognition Letters (2020).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of machine learning research, 12(Aug), 2493-2537.

Croskerry, P. (2003). Cognitive forcing strategies in clinical decision making. Annals of emergency medicine, 41(1), 110-120.

Danger, R., Segura-Bedmar, I., Martínez, P., & Rosso, P. (2010). A comparison of machine learning techniques for detection of drug target articles. Journal of biomedical informatics, 43(6), 902-913.

Darren Yates, Md Zahidul Islam, Junbin Gao (2018). SPAARC: A Fast Decision Tree Algorithm (AusDM 2018). Presented at the 2018 Australasian Data Mining conference (AusDM 2018) at Bathurst, NSW.

Dolin, R. H., Alschuler, L., Boyer, S., Beebe, C., Behlen, F. M., Biron, P. V., & Shabo, A. (2006). HL7 clinical document architecture, release 2. Journal of the American Medical Informatics Association, 13(1), 30-39.

Donnelly, K. (2006). SNOMED-CT: The advanced terminology and coding system for eHealth. Studies in health technology and informatics, 121, 279.

Doval, Y., Gómez Rodríguez, C., & Vilares Ferro, J. (2016). Segmentación de palabras en español mediante modelos del lenguaje basados en redes neuronales.

Friedman, C., & Hripcsak, G. (1999). Natural language processing and its future in medicine. Acad Med, 74(8), 890-5.

Ford, E., Carroll, J. A., Smith, H. E., Scott, D., & Cassell, J. A. (2016). Extracting information from the text of electronic medical records to improve case detection: a systematic review. Journal of the American Medical Informatics Association, 23(5), 1007-1015.

Gobeill, J., Gaudinat, A., & Ruch, P. (2015). Exploiting incoming and outgoing citations for improving Information Retrieval in the TREC 2015 Clinical Decision Support Track. In TREC.

González, O. L., Trinidad, J. F. M., Borroto, M. G., & Erro, L. E. (2014). Clasificadores Supervisados basados en Patrones Emergentes para Bases de Datos con Clases Desbalanceadas. Reporte Técnico No. CCC-14–004, Coordinación de Ciencias Computacionales, INAOE.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y. & Larochelle, H. (2017). Brain tumor segmentation with deep neural networks. Medical image analysis, 35, 18-31.

Herrera Alonso, A. (2015). Detección de texto utilizando Redes Neuronales Convolucionales (Bachelor's thesis, Universitat Politècnica de Catalunya).

Heusser, A. C., Ziman, K., Owen, L. L., & Manning, J. R. (2017). HyperTools: A Python toolbox for visualizing and manipulating high-dimensional data. arXiv preprint arXiv:1701.08290.

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. Computer, 34(9), 120-127.

Hinneburg, A., & Gabriel, H. H. (2007, September). Denclue 2.0: Fast clustering based on kernel density estimation. In International symposium on intelligent data analysis (pp. 70-80). Springer, Berlin, Heidelberg.

Huang, Yan, et al. "A comparative analytic study on the gaussian mixture and context dependent deep neural network hidden markov models." Fifteenth Annual Conference of the International Speech Communication Association. 2014.

Hughes, M., Li, I., Kotoulas, S., & Suzumura, T. (2017). Medical text classification using convolutional neural networks. Stud Health Technol Inform, 235, 246-250.

Hurtado, C., Romario, E., Arias, M., & Jhon, E. (2017). Clasificación multicategórica de mamografías mediante el uso de redes neuronales convolucionales.

Janzen, D., & Saiedian, H. (2005). Test-driven development concepts, taxonomy, and future direction. Computer, 38(9), 43-50.

Jiang M., Chen Y, Liu M., Rosenbloom S.T., Mani S., Denny J.C. (2011), A study of machine-learning-based approaches to extract clinical entities and their assertions from discharge summaries, Journal of the American Medical Informatics Association, Volume 18, Issue 5, 1, Pages 601–606.

Jianqiang, Zhao, Gui Xiaolin, and Zhang Xuejun. "Deep convolution neural networks for twitter sentiment analysis." IEEE Access 6 (2018): 23253-23260.

Kalmegh, S. (2015). Analysis of WEKA data mining algorithm REPTree, Simple CART and RandomTree for classification of Indian news. Int. J. Innov. Sci. Eng. Technol, 2(2), 438-446.

Karimi, S., Dai, X., Hassanzadeh, H., & Nguyen, A. (2017, August). Automatic diagnosis coding of radiology reports: a comparison of deep learning and conventional classification methods. In BioNLP 2017 (pp. 328-332).

Karystianis, G., Adily, A., Schofield, P., Knight, L., Galdon, C., Greenberg, D., ... & Butler, T. (2018). Automatic extraction of mental health disorders from domestic violence police narratives: text mining study. Journal of medical internet research, 20(9).

Kim, Y. (2014). Convolutional neural networks for sentence classification. Conference on Empirical Methods in Natural Language Processing (EMNLP'14).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Krause, B., Lu, L., Murray, I., & Renals, S. (2016). Multiplicative LSTM for sequence modelling. arXiv preprint arXiv:1609.07959.

Kukovacec, M., Kukurin, T., & Vernier, M. Personalized Medicine: Redefining Cancer Treatment. Text Analysis and Retrieval 2018 Course Project Reports, 69.

Kursa, M. B., Jankowski, A., & Rudnicki, W. R. (2010). Boruta–a system for feature selection. Fundamenta Informaticae, 101(4), 271-285.

Lanzarini, L. C., & Leguizamón, G. (2013). Optimización multiobjetivo: aplicaciones a problemas del mundo real. Buenos Aires, Argentina, Universidad Nacional de la Plata, 66-90.

Lanzarini, L. C., Hasperué, W., Estrebou, C. A., Ronchetti, F., Villa Monte, A., Aquino, G. O., ... & Luna, C. (2017, August). Minería de datos y big data: aplicaciones en señales y textos. In XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).

Lee, K., Lee, S., Park, S., Kim, S., Kim, S., Choi, K., ... & Kang, J. (2016). BRONCO: Biomedical entity Relation ONcology COrpus for extracting gene-variant-disease-drug relations. Database, 2016.

Lingren, T., Chen, P., Bochenek, J., Doshi-Velez, F., Manning-Courtney, P., Bickel, J., ... & Barbaresi, W. (2016). Electronic health record-based algorithm to identify patients with autism spectrum disorder. PloS one, 11(7).

Lisitsa, Daria, and Anton A. Zhilenkov. "Comparative analysis of the classical and nonclassical artificial neural networks." 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). IEEE, 2017.

Lizcano, J. A. M. (2015). Sistemas de ayuda al diagnóstico y reconocimiento de microcalcificaciones en mamografía mediante descriptores de escala y redes jerárquicas (Doctoral dissertation, PhD thesis, Escuela técnica superior de Ingenieros de Telecomunicación).

Martin, R. C. (2005). Agile software development: principles, patterns, and practices. Computing Reviews, 46(2), 91.

Mehta, S., Mercan, E., Bartlett, J., Weaver, D., Elmore, J. G., & Shapiro, L. (2018, September). Y-Net: joint segmentation and classification for diagnosis of breast biopsy images. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 893-901). Springer, Cham.

Morales, E., & Jorge, A. (2017). Comparación de rendimiento de técnicas de aprendizaje automático para análisis de afecto sobre textos en español.

Murff, H. J., FitzHenry, F., Matheny, M. E., Gentry, N., Kotter, K. L., Crimin, K., ... & Speroff, T. (2011). Automated identification of postoperative complications within an electronic medical record using natural language processing. Jama, 306(8), 848-855.

Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. Journal of the American Medical Informatics Association, 18(5), 544-551.

Parsons, L., Haque, E., & Liu, H. (2004). Subspace clustering for high dimensional data: a review. Acm Sigkdd Explorations Newsletter, 6(1), 90-105.

Pennington, J., Socher, R., & Manning, C. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Rebholz-Schuhmann, D., Clematide, S., Rinaldi, F., Kafkas, S., van Mulligen, E. M., Bui, C., & Jimeno-Yepes, A. (2013, September). Entity recognition in parallel multi-lingual biomedical corpora: The CLEF-ER laboratory overview. In International Conference of the Cross-Language Evaluation Forum for European Languages (pp. 353-367). Springer, Berlin, Heidelberg.

Rios, A., & Kavuluru, R. (2015, September). Convolutional neural networks for biomedical text classification: application in indexing biomedical articles. In Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics (pp. 258-267). ACM.

Rodriguez-Castello, D. (2017). Extracción de cráneo en imágenes de resonancia magnética del cerebro utilizando una red neuronal convolucional 3D (Bachelor's thesis, Universitat Politècnica de Catalunya).

Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.

Roth, H. R., Oda, H., Zhou, X., Shimizu, N., Yang, Y., Hayashi, Y., & Mori, K. (2018). An application of cascaded 3D fully convolutional networks for medical image segmentation. Computerized Medical Imaging and Graphics, 66, 90-99.

Runshisky A, Uzuner O. (2013). Annotating temporal information in clinical narratives. Journal of American Medical Informatics Association, 46, 5-12.

Sánchez-Atienza, E. (2017). Detección de personas mediante redes convolucionales, Tesis de Licenciatura, Universitat Politècnica de Catalunya.

Savova, G. K., Olson, J. E., Murphy, S. P., Cafourek, V. L., Couch, F. J., Goetz, M. P., ... & Weinshilboum, R. M. (2011). Automated discovery of drug treatment patterns for endocrine therapy of breast cancer within an electronic medical record. Journal of the American Medical Informatics Association, 19(e1), e83-e89.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural networks, 61, 85-117.

Sereno-Rodriguez, P. (2017). Reconocimiento de expresiones faciales mediante el uso de redes neuronales convolucionales, Tesis de Licenciatura, Universitat Politècnica de Catalunya.

Serrano, E. R. P., & Hernández, M. G. G. (2015). Diseño de un sistema de reconocimiento automático de palabras Contenidas en documentos históricos. Jóvenes científicos, 1(2), 1607-1612.

Silva, G. L. (2017). Diagnóstico de nódulos pulmonares em imágenes de tomografía computadorizada usando redes neurais convolucionais evolutivas.

Small, S. G., & Medsker, L. (2014). Review of information extraction technologies and applications. Neural computing and applications, 25(3-4), 533-548.

Spackman, K. A., Campbell, K. E., & Côté, R. A. (1997). SNOMED RT: a reference terminology for health care. In Proceedings of the AMIA annual fall symposium (p. 640). American Medical Informatics Association.

Sundararajan, V., Henderson, T., Perry, C., Muggivan, A., Quan, H., & Ghali, W. A. (2004). New ICD-10 version of the Charlson comorbidity index predicted in-hospital mortality. Journal of clinical epidemiology, 57(12), 1288-1294.

Strauss, J. A., Chao, C. R., Kwan, M. L., Ahmed, S. A., Schottinger, J. E., & Quinn, V. P. (2012). Identifying primary and recurrent cancers using a SAS-based natural language processing algorithm. Journal of the American Medical Informatics Association, 20(2), 349-355.

Tierney, W. (2001) Improving clinical decisions and outcomes with information: a review. International journal of medical informatics vol. 62, no 1, p. 1-9.

Tyrer, P., Crawford, M., Mulder, R., Blashfield, R., Farnam, A., Fossati, A., ... & Swales, M. (2011). The rationale for the reclassification of personality disorder in the 11th revision of the International Classification of Diseases (ICD-11). Personality and Mental Health, 5(4), 246-259.

Uysal, Alper Kursat, and Yi Lu Murphey. "Sentiment classification: Feature selection-based approaches versus deep learning." 2017 IEEE International Conference on Computer and Information Technology (CIT). IEEE, 2017.

Uzuner, Ö., Bodnari A, Shen S, Forbush T, Pestian J, South BR. (2012). Evaluating the state of the art in coreference resolution for electronic medical records. Journal of American Medical Informatics Association. 8(5) ,786-91.

Uzuner, Ö., Goldstein, I., Luo, Y., & Kohane, I. (2008). Identifying patient smoking status from medical discharge records. Journal of the American Medical Informatics Association, 15(1), 14-24.

Uzuner, Ö., South B., Shen S., DuVall S. (2011). I2B2/VA Challenge on Concepts, Assertions, and Relations in Clinical Text. Journal of American Medical Informatics Association.,18:552-556.

Van Der Maaten, L., & Weinberger, K. (2012, September). Stochastic triplet embedding. In 2012 IEEE International Workshop on Machine Learning for Signal Processing (pp. 1-6). IEEE.

Wang, G., Li, W., Ourselin, S., & Vercauteren, T. (2017, September). Automatic brain tumor segmentation using cascaded anisotropic convolutional neural networks. In International MICCAI Brainlesion Workshop (pp. 178-190). Springer, Cham.

Wang, X., Jiang, W., & Luo, Z. (2016, December). Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers (pp. 2428-2437).

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. In Advances in Neural Information Processing Systems (pp. 4148-4158).

Wu, D. T., Hanauer, D. A., Mei, Q., Clark, P. M., An, L. C., Lei, J., ... & Zheng, K. (2013). Applying multiple methods to assess the readability of a large corpus of medical documents. Studies in health technology and informatics, 192, 647.

Yin, W., & Schütze, H. (2016). Multichannel variable-size convolution for sentence classification. arXiv preprint arXiv:1603.04513.

Zaccone, G. (2016). Getting Started with TensorFlow. Packt Publishing Ltd.

Zea, C., & Linet, J. (2017). Reconocimiento de entidades nombradas para el idioma español utilizando Conditional Random Fields con características no supervisadas.

Zhang, Y., Marshall, I., & Wallace, B. C. (2016, November). Rationale-augmented convolutional neural networks for text classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing (Vol. 2016, p. 795). NIH Public Access.

Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

Zhou, Shusen, Qingcai Chen, and Xiaolong Wang. "Active deep learning method for semi-supervised sentiment classification." Neurocomputing 120 (2013): 536-546.

ANEXO 1. Código fuente y repositorio del proyecto.

El código fuente completo utilizado para producir este documento y los resultados en CSV (datos tabulados) puede ser descargado incluyendo comentarios e instrucciones para la reproducibilidad desde el repositorio del público proyecto[10] (véase figura 33).

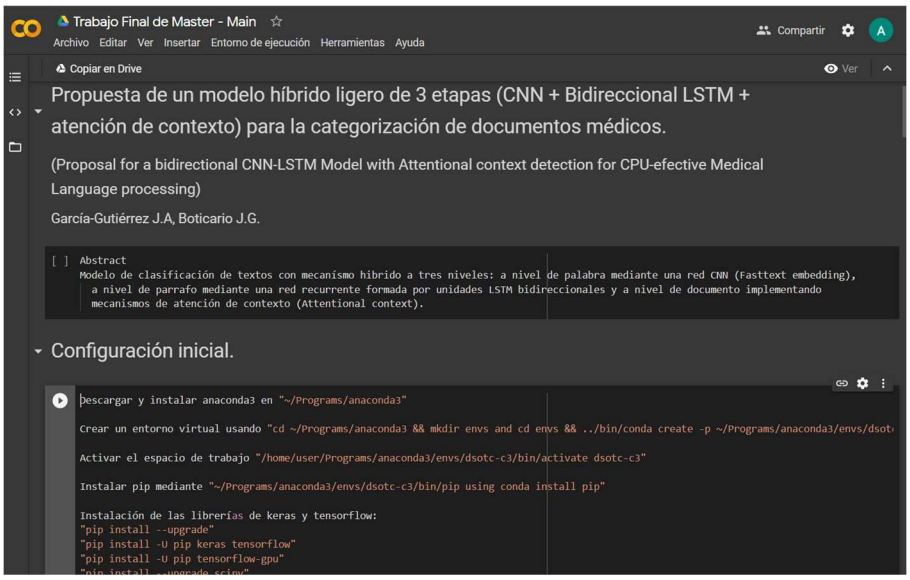

Figura 33. Repositorio público del proyecto, disponible desde **https://cutt.ly/RuSvLAv**

Descargar una copia local del repositorio

Python Package Installer (PIP) es una herramienta que permite instalar, desinstalar y actualizar paquetes Python. Utilizaremos ***pip*** para listar todas las dependencias del proyecto y hacer una instalación rápida de requerimientos.

```
Descargar e instalar anaconda3 en "~/Programs/anaconda3"

Crear un entorno virtual usando "cd ~/Programs/anaconda3 && mkdir
 envs and cd envs && ../bin/conda create -
p ~/Programs/anaconda3/envs/dsotc-c3 python=3.6 anaconda."

Activar el espacio de trabajo "/home/user/Programs/anaconda3/envs
/dsotc-c3/bin/activate dsotc-c3"
```

[10] **https://colab.research.google.com/drive/1rKaQl3SD32zXFvr58e5m4ThSjmgcPGyy**

```
Instalar pip mediante "~/Programs/anaconda3/envs/dsotc-
c3/bin/pip using conda install pip"

Instalación de las librerías de keras y tensorflow:
"pip install --upgrade"
"pip install -U pip keras tensorflow"
"pip install -U pip tensorflow-gpu"
"pip install --upgrade scipy"

Ejecutar el comando "pip install -
r requirements.txt" para instalar todas las dependencias del proy
ecto

Instalación de jupyter IDE:
"pip install jupyter"
"pip install --use-wheel --no-index --find-links=TF-
master\numpymkl.whl"
"jupyter notebook"

Activar las extensiones de jupyter ejecutando "jupyter extensions
 jupyter nbextensions_configurator enable --user"

Activar las opciones de configuración mediante "jupyter contrib n
bextension install --user"

Las versiones de referencia de la paquetería utilizados fueron:

Python 3.6.2, Keras 2.0.8, tensorflow 1.12.0, tensorflow-
gpu 1.3.0, tensorflow-tensorboard 0.1.8.

------------------------------------------------------------
Ahora estamos listos para ejecutar el proyecto garantizando
la reproducibilidad de la investigación.
------------------------------------------------------------
```

Obtener una copia completa del dataset

Es posible descargar una copia de los distintos diccionarios clínicos y tablas empleadas durante el entrenamiento desde https://drive.google.com/drive/folders/1917XEkFeygfqVoNFjqnbSui8Sh3Wl1NN. Si es necesario, se proporcionarán más instrucciones para descargar el conjunto de datos preprocesado completo.

ANEXO 2. Manual de usuario.

Se proporcionan dos demostradores que muestran la funcionalidad básica del modelo clasificador entrenado. Estos demostradores son totalmente en línea por lo que el usuario no necesita de ninguna descarga para poder ejecutarlos y funcionan mediante *Tensorflow.js API*[11].

Demostrador 1

La primera demo puede ser accedida desde **http://medaid.jvmhost.net/Medaid_nb/** y se corresponde con el modelo más simple donde introducimos una serie de campos que se corresponden con un informe de ingreso clásico (antecedentes, exploración, tratamiento actual, síntomas..) y obtendremos un desglose de la posibilidad de pertenencia del enfermo a los diferentes clústeres previamente creados. (véase figura 34).

Figura 34. Captura de pantalla de la aplicación de demostración 1.

También obtendremos una relación de documentos cercanos basándonos en los porcentajes obtenidos para la pertenencia a cada clúster.

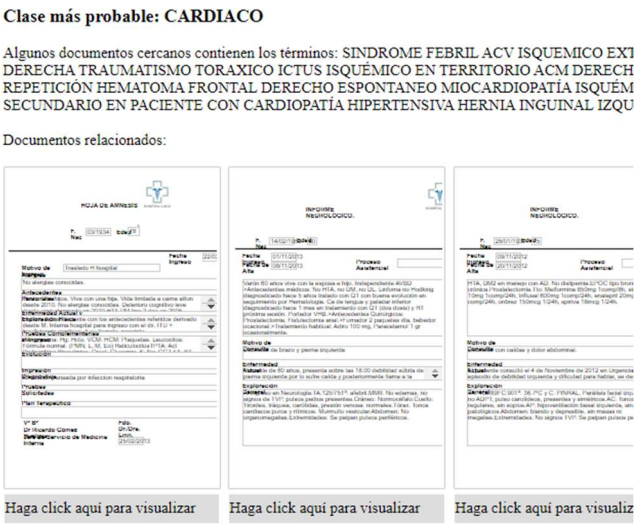

Figura 35. Algunos documentos relacionados obtenidos como resultado a la consulta.

[11] **https://www.tensorflow.org/js**

Demostrador 2

La segunda demo puede ser accedida desde **http://medaid.jvmhost.net/Medaid_model_2/** y nos permite clasificar un estudio oncológico en cada uno de los 9 clústeres que contempla el dataset MSKCC. El la figura 36 podemos ver la salida generada por el modelo donde se nos visualiza también los términos más usuales dentro del clúster asignado y la firma correspondiente del documento.

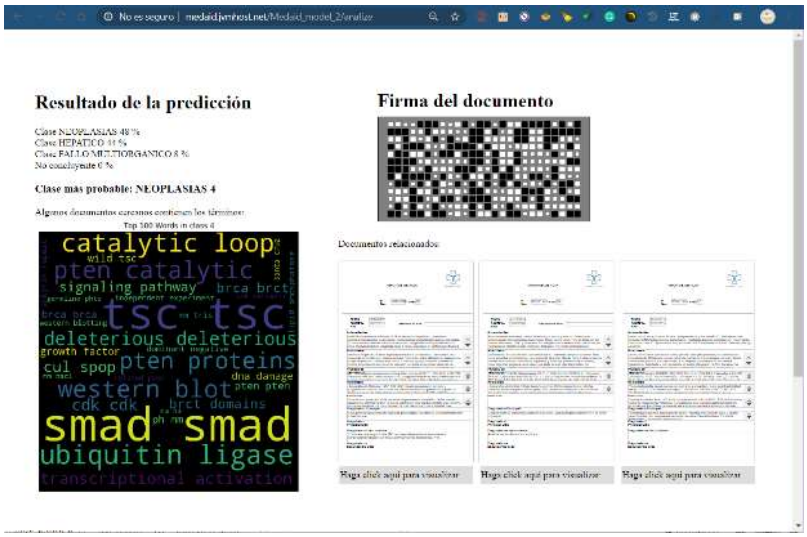

Figura 36. Captura de pantalla de la aplicación de demostración 2.

Visor de informes y acceso a tablas y repositorios.

Adicionalmente se hace público un visor de informes clínicos que nos permite la visualización más sencilla de las entradas del conjunto de datos, así como de los diferentes diccionarios y glosarios clínicos. Esta herramienta está incluida en ambos demostradores y puede ser accedida desde **http://medaid.jvmhost.net/Medaid_model_2/admon.html**

Figura 37. Visor de historias clínicas integrado

El mejor poema del mundo está escrito en una botella, que alguien arrojó al mar; y tal como dejó la mano, trascendió lo humano, y se echó a navegar. El mensaje que nunca espera ser encontrado no tiene dueño, es esperanza, no conoce tiempo ni añoranza. Nunca se sabrá descubierto, y sin embargo, tiene un destino preciso y cierto: superar las olas y vencer al viento; ser testigo mudo de que algo del alma que queda en las palabras, será entregado de su sello abierto; una lágrima, un destello, una mirada, y tal vez, en el fondo, una sonrisa largo tiempo planeada.

"De vivir naufragando", José A. García Gutiérrez, 2020.

This page was intentionally left blank