PONTIFICIA UNIVERSIDAD JAVERIANA - CALI

FACULTAD DE INGENIERÍA Y CIENCIAS

DEPARTAMENTO DE ELECTRÓNICA Y CIENCIAS DE LA COMPUTACIÓN

Generador de resúmenes para noticias sobre economía

Natalia Guerrero Caicedo

Director:

Jesús David Rivero Ortega

Santiago de Cali, Septiembre 30, 2024

Resumen

Se diseñó y desarrolló un generador de resúmenes extractivos enfocado en noticias económicas utilizando técnicas de aprendizaje automático. Se exploraron diversas tecnologías relacionadas con el ámbito de la generación de resúmenes, de las cuales se escogieron las dos más pertinentes. Además, se discutieron las limitaciones técnicas asociadas a esta tarea y las estrategias posibles para mitigar estos desafíos.

Palabras clave: Aprendizaje profundo, Resúmenes extractivos, Codificador-Decodificador, Redes Neuronales Recurrentes, Aprendizaje automático

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Descripción del problema

1.1. Planteamiento del problema

Las noticias son un medio de comunicación fundamental para informar a las personas sobre lo que está aconteciendo alrededor de ellas. Estas pueden verse beneficiadas o perjudicadas según cómo la noticia afecta el contexto que estén viviendo. En Colombia hay aproximadamente 95 periódicos que diariamente publican contenido relacionado con diversos temas [1], esto indica una amplia disponibilidad de información para los lectores. Sin embargo, también significa que el tiempo de lectura se incrementa considerablemente para aquellos que desean mantenerse informados sobre múltiples asuntos.

En cuanto al periodismo económico, este consiste en informar sobre eventos relacionados con la economía, incluyendo temas sobre finanzas, banco o el mercado bursátil [2]. La economía está presente en muchas acciones de la vida diaria, disponer de una visión rápida de la información más relevante sobre estos temas, sin tener que leer artículos completos, resulta fundamental para mantenerse informado de manera eficiente.

Por ejemplo, en las noticias que contienen valores importantes, como incrementos o porcentajes, a menudo se encuentra una cantidad considerable de información introductoria. Esta puede, por un lado, dificultar la comprensión rápida del objetivo principal de la noticia, y por otro lado, ayudar a interpretar mejor los datos en su contexto. Por lo tanto, la decisión de omitir o no estos datos depende únicamente de las preferencias del lector.

Por esta razón, para este trabajo se plantea buscar una manera en que las noticias puedan leerse en menor tiempo al disminuir su longitud sin afectar la coherencia y la finalidad de la noticia. Esto con la intención de ofrecer una alternativa para aquellos lectores que no pretenden leer la noticia a profundidad.

Actualmente, en internet existen herramientas que permiten hacer resúmenes extractivos como QuillBoot y Resoomer, los cuales toman fragmentos exactos del texto que son considerados relevantes usando procesamiento del lenguaje natural para localizar información crítica y lo hacen para diversos tipos de escritos, como noticias, documentos científicos, libros, presentaciones de artistas y sus obras, e incluso párrafos confusos.

Aunque nuestra propuesta es similar a las dos anteriores herramientas, la diferencia está en que estas ofrecen resumir diferentes tipos de documentos, y adicionalmente brindan otras opciones como corrector gramatical, verificador de plagio, generador de citas, entre otros. Por eso, la propuesta que se presenta en este documento es crear por medio de aplicaciones de la rama del procesamiento de lenguaje natural, un generador de resúmenes únicamente para en el nicho de noticias económicas, como una alternativa para aquellas personas que solo quieren leer el resúmen de alguna noticia económica.

1.1.1. Formulación

¿Cómo desarrollar un sistema para la generación automática de resúmenes de noticias económicas usando la técnica de aprendizaje profundo?

1.1.2. Sistematización

P1 ¿Cómo obtener y preprocesar el conjunto de datos a utilizar para la construcción de los modelos que permiten generar los resúmenes?

P2 ¿Qué experimentos hacer para construir dos modelos que permitan la generación automática usando el enfoque de aprendizaje profundo?

P3 ¿Cómo evaluar el desempeño de los dos modelos desarrollados para poder compararlos?

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar un sistema para la generación automática de resúmenes de noticias económicas usando la técnica de aprendizaje profundo.

1.2.2. Objetivos Específicos

- Seleccionar y preprocesar noticias económicas de diferentes portales de noticias
- Desarrollar dos modelos de generación automática de resúmenes usando el enfoque de aprendizaje profundo
- Evaluar el desempeño de cada modelo haciendo uso de la métrica ROUGE (Recall-Oriented Understudy for Gisting Evaluation) y comparar

1.3. Justificación

Las noticias económicas se publican a diario y son esenciales tanto a nivel cultural como financiero. Proporcionan una visión general del contexto económico, influyendo en las decisiones financieras y afectando diversos aspectos de la vida diaria.

Dado que las noticias sobre economía se difunden diariamente y diferentes portales publican sobre estas, se crea una sobreinformación que supera la capacidad limitada de procesamiento cognitivo del lector. Esto ocasiona que se pierda el mensaje principal de la noticia si el usuario final o receptor del mensaje no cuenta con tiempo suficiente para leerla nuevamente. Por otro lado, se estableció que el resumen automático de texto es muy eficaz en las tareas de evaluación de relevancia de los artículos de noticias, dentro de los beneficios se encuentra que los resúmenes tan cortos como el 17 % de la longitud del texto original aceleraron la toma de decisiones en casi un factor de 2, es decir, se duplicó, sin una degradación estadísticamente significativa de la precisión del resumen [3].

Por lo anterior, este proyecto consiste en el desarrollo de un sistema que resuma noticias económicas, para que puedan leerse en menor tiempo sin afectar la coherencia y la finalidad de la noticia. Esto sería una alternativa útil para aquellas personas que no quieren leer la noticia a profundidad.

Nuestra propuesta no requiere de muchos recursos. Se cuenta con un amplio conjunto de datos sobre noticias económicas obtenido de internet, y existen librerías que incluyen técnicas de generación automática de resúmenes, las cuales pueden ser utilizadas en el desarrollo del presente proyecto.

1.4. Delimitaciones y Alcances

- La técnica a explorar estará enfocada en aprendizaje profundo
- El sistema únicamente se basará en generar resúmenes de noticias económicas en inglés
- No se implementará una interfaz gráfica
- Se usarán librerías de Python para desarrollar el sistema
- Habrán limitaciones en algunos métodos debido al consumo de recursos computacionales
- Se hará un análisis comparativo entre los dos modelos para identificar cuál es el que tiene un mejor rendimiento para generar resúmenes de noticias económicas

1.4.1. Entregables

- Un programa cuya entrada es una noticia económica, y la salida es un resumen de la misma
- Un documento que contenga la descripción del proceso realizado para la construcción de los modelos y el análisis comparativo de los resultados obtenidos con ellos

Capítulo 2

Marco teórico

2.1. Generación automática de resúmenes

El tiempo es un recurso muy preciado y a la hora de leer diversos tipos de documentos se puede ver afectado en aquellas personas que no desean leerlos a profundidad. Por lo que, al no tener una alternativa diferente a leerla por completo, se aumentaría el tiempo de lectura y en ocasiones se podría limitar la visibilidad de elementos relevantes debido a la cantidad de información. Una solución para esto sería enfocarse únicamente en el contenido principal del documento, donde la generación automática de resúmenes es una herramienta propicia para esta tarea.

La tarea de hacer resúmenes es compleja y varía significativamente dependiendo de la persona encargada de realizarla, su nivel de comprensión del texto original y el criterio empleado para evaluar la relevancia de las oraciones. Además de su complejidad, esta labor requiere una considerable inversión de tiempo, especialmente al resumir grandes volúmenes de documentos. Por consiguiente, la asignación manual de esta tarea se vuelve laboriosa y poco práctica en términos de eficiencia.

La generación automática de resúmenes ayuda a solucionar este problema mediante el uso de recursos computacionales, lo que reduce tanto la carga humana como el tiempo requerido para realizar el resumen. Esto permite que los documentos puedan leerse en menor tiempo sin comprometer la claridad y la precisión del contenido. Este enfoque no solo mejora la eficiencia del proceso,

sino que también facilita el acceso rápido a información relevante.

Según [4] existen diferentes tipos de resúmenes que pueden ser obtenidos a través de la generación automática de resúmenes, se explican a continuación:

- Según la cantidad de documentos que se consideren para generarlo, el resumen puede ser de un solo documento y de múltiples documentos.
- Con base al contenido de un resumen se puede clasificar como, resumen indicativo cuando se establece una idea principal de todo el documento sin explicar su contenido, y resumen informativo cuando se describe con cierto nivel de detalle el contenido del documento.
- Con respecto al idioma del documento, pueden ser monolenguaje o multilenguaje.
- De acuerdo con el género del documento, pueden ser artículos científicos, noticias, blogs, entre otros.
- Dependiendo de la naturaleza del resumen, pueden ser resúmenes extractivos y resúmenes abstractivos. A pesar de que se pueden encontrar otros tipos de resúmenes, como los híbridos, suelen ser poco usados o resultar de la combinación de los dos principales [5].

Considerando estas características y el alcance del presente proyecto, es importante señalar que este proyecto se enfoca en la generación automática de resúmenes extractivos a partir de un solo documento, centrado en el género de noticias económicas en inglés.

2.1.1. Resumen extractivo

Un resumen extractivo se crea concatenando varias oraciones tomadas exactamente como aparecen en el texto original [5]. Los métodos de extracción se caracterizan por seleccionar unidades de texto sobresalientes o importantes, por medio de la relevancia léxica de la unidad de texto o haciendo coincidir patrones de frases. Entre los métodos para generar resúmenes extractivos se encuentran: métodos estadísticos, basados en técnicas de reducción algebraica, basados en técnicas de aprendizaje de máquina, basados en conectividad

de texto, basados en grafos, basados en modelos evolutivos, entre otros [6].

Una de las desventajas de emplear técnicas extractivas es que los resúmenes pueden presentar problemas de consistencia y coherencia, dado que son formados a partir de la reutilización de porciones del texto original [7]. Esto puede resultar en transiciones abruptas y una falta de fluidez en el resumen final, pero se hace lo posible por ordenar las oraciones en el orden correcto según el texto original [8]. Sin embargo, las técnicas extractivas son muy utilizadas debido a su sencillez computacional, y su rápida implementación en comparación con los métodos abstractivos, pues no requieren de la generación de nuevo texto.

2.2. Aprendizaje automático

El aprendizaje se define como la capacidad que tiene un agente de evidenciar cambios en el sistema que son de naturaleza adaptativa, los cuales le ayudan a mejorarse a sí mismo para realizar las mismas tareas de manera más eficiente y efectiva la próxima vez [9]. El diseño de un agente se ve afectado principalmente por los siguientes puntos:

- La representación usada para manejar los datos.
- Las retroalimentaciones disponibles para ser usadas en el aprendizaje.

El tipo de retroalimentación usado suele ser uno de los factores más importantes para identificar las tareas de aprendizaje que enfrentan los agentes. La retroalimentación indica tres formas de aprendizaje:

- Aprendizaje supervisado: el agente aprende a partir de observaciones de ejemplos de sus entradas y salidas.
- Aprendizaje no supervisado: se reciben datos sin etiquetar y procesar, por lo que el agente debe encontrar regularidades sin un conocimiento explícito sobre cómo trabajar con ellos.
- Aprendizaje por refuerzo: el agente aprende a partir del refuerzo, mediante interacciones de prueba y error en un entorno dinámico.

2.2.1. Aprendizaje supervisado

El aprendizaje supervisado es un enfoque del aprendizaje automático que emplea un conjunto de datos etiquetados como datos de entrenamiento [10], que incluye tanto parámetros de entrada como sus correspondientes salidas. Esto se hace con la intención de instruir a los modelos a producir el resultado esperado.

Dado un conjunto de datos N de la forma $\{(x_i, y_i), ..., (x_n, y_n)\}$ donde x_i representa un dato de entrada y y_i representa un dato de salida. Para cada punto en la muestra de entrenamiento (x_i), se aplica la función $y = f(x)$ para obtener su correspondiente etiqueta y. La finalidad es encontrar una hipótesis h que aproxime el valor real de f. Desde un punto de vista conceptual, el aprendizaje es difícil puesto que no es fácil decidir si cualquier h es una buena aproximación de f [9]. Una hipótesis h se considera buena cuando predice correctamente el valor de y en los datos de entrenamiento y es capaz de generalizar a datos nuevos.

Existen dos tipos de problemas en los cuales se puede clasificar el aprendizaje supervisado: clasificación y regresión. La clasificación se utiliza para asignar con precisión los datos a categorías específicas. En este caso la salida y es un valor discreto, el cual puede tomar solo un valor determinado entre el conjunto de todos los valores posibles. Por otro lado, la regresión se emplea para encontrar correlaciones entre variables dependientes e independientes, comúnmente para realizar proyecciones. La salida y es un valor continuo, que puede tomar cualquier valor dentro de un intervalo especificado de valores asumidos por la variable.

2.2.2. Aprendizaje no supervisado

Este tipo de aprendizaje se caracteriza por recibir datos de entrenamiento sin etiquetar, en donde los modelos crean respuestas por sí mismos, descubriendo patrones y relaciones entre los datos de entrada. Algunos de los problemas en los que se puede clasificar el aprendizaje no supervisado son los siguientes:

- Método de agrupamiento (Clustering): su finalidad es agrupar los objetos

en función de similitudes o diferencias que se encuentran al explorar el conjunto de datos de entrada [11]. Esto se hace a partir de patrones que se encuentran en los datos no categorizados. El algoritmo k-means es un ejemplo de los principales algoritmos basados en Clustering.

- Método de asociación (Association Rule Learning): su objetivo es descubrir asociaciones entre variables de un conjunto de datos. Se centra en cómo se vinculan variables particulares por determinadas características, con el fin de encontrar objetos relacionados entre sí, sin ser iguales. El algoritmo Apriori es uno de los más utilizados dentro del método de asociación para identificar los conjuntos de elementos que están relacionados [12].

2.2.3. Aprendizaje por refuerzo

De acuerdo con [13], el aprendizaje por refuerzo no depende de datos estáticos para su aprendizaje. En cambio, utiliza datos provenientes de la interacción entre un agente con el entorno, la cual se puede ver de la siguiente forma:

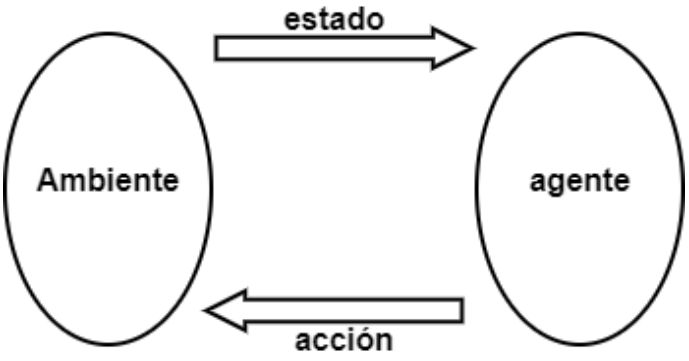

Figura 2.1: Esquema que representa la interacción entre un agente con el entorno [14].

En el modelo presentado en la Figura 2.1, el agente examina el estado del entorno y ejecuta una acción, después el entorno responde a esta acción produciendo un nuevo estado. Este aprendizaje se centra en utilizar un enfoque de recompensas y penalizaciones para el procesamiento de los datos, el cual permite que se aprenda de la retroalimentación recibida de cada acción para encontrar mejores estrategias y así alcanzar los objetivos.

2.2.4. Técnicas de aprendizaje automático

Se detallan algunas técnicas de aprendizaje automático aplicadas a la generación automática de resúmenes.

Naïve Bayes

Naïve Bayes es un algoritmo supervisado de clasificación utilizado en diversas aplicaciones, destacándose en el procesamiento de lenguaje natural (PLN). Se basa en el teorema de Bayes, que calcula la probabilidad de una hipótesis dados unos datos y ciertos conocimientos previos. Este teorema asume que el efecto de una característica particular en una clase es independiente de otras características dentro del mismo.

El modelo de un clasificador probabilístico Naïve Bayes tiene como objetivo determinar la probabilidad de ocurrencia de una serie de características X representadas como $X = (x_1, x_2, x_3, \ldots, x_n)$ en un conjunto de clases. Para cada clase se calcula la probabilidad condicional $P(y_i|x_1, x_2, x_3, \ldots, x_n)$ de la siguiente forma [15]:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{2.1}$$

$$P(y|x_1, x_2, x_3, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)} \tag{2.2}$$

Es importante mencionar que en el algoritmo de Naive Bayes, se debe calcular el $P(y_i)$ de cada clase en la etapa de entrenamiento del modelo y las probabilidades condicionales de todas las características x_i [15]. Dado que el denominador permanece constante para una entrada determinada, podemos eliminarlo e inyectar proporcionalidad ($\propto$):

$$P(y|x_1, x_2, x_3, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.3}$$

Finalmente, para realizar la clasificación, se necesita encontrar la máxima probabilidad de la clase y, en la cual se puede clasificar una determinada característica. La ecuación se ve de la siguiente forma:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.4}$$

Este algoritmo ha sido utilizado para múltiples propósitos, pero funciona particularmente bien con problemas de procesamiento de lenguaje natural (PLN). En el área de generación de resúmenes, se ha destacado su uso en [16], donde se utilizó Naïve Bayes para clasificar las oraciones del documento y agregarlas al resumen según la probabilidad de la oración medida por características como la longitud de las oraciones, la frecuencia de palabras, y las palabras del título.

Máquinas de Vectores de Soporte (MVS)

Las Máquinas de Vectores de Soporte fueron introducidas por V. Vapnik [17]. Son un algoritmo de aprendizaje supervisado para problemas de clasificación binaria, que busca encontrar el hiperplano que mejor separe los datos en diferentes clases. Un hiperplano en un espacio n-dimensional (donde n es el número de características) es una superficie plana que divide el espacio de datos en dos partes distintas.

Esto se puede representar mediante la ecuación 2.5, donde el conjunto de datos de entrenamiento esta dado por [18]:

$$(x_1 y_1), ..., (x_u y_u), x_j \in R^n, y_j \in \{+1, -1\} \tag{2.5}$$

Donde x_j es un vector de características, y y_j es una etiqueta que puede ser positiva (+1) o negativa (-1). El hiperplano que separa los positivos y negativos está definido por:

$$w \cdot x + b = 0, w \in R^n, b \in R \tag{2.6}$$

Donde " $\cdot$ " representa el producto interno.

En la Figura 2.2 se muestra un ejemplo con datos linealmente separables. La MVS determina el hiperplano óptimo de clasificación de dos clases maximizando el margen, que es la distancia entre las clases positivas y las clases negativas.

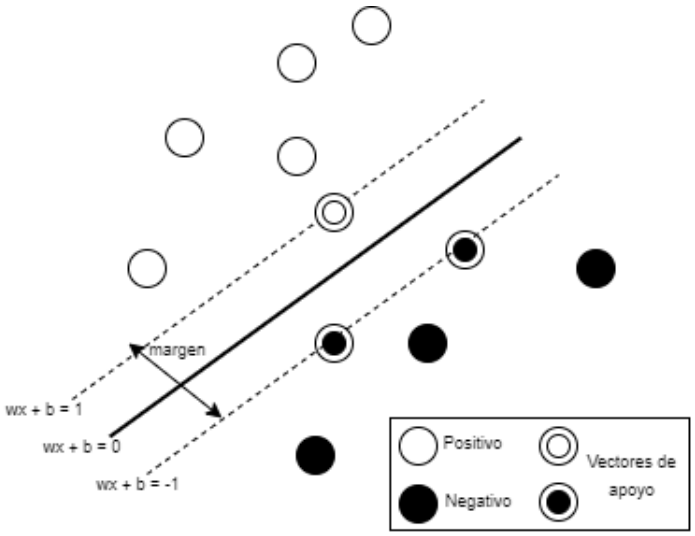

Figura 2.2: MVS con datos linealmente separables [18].

Sin embargo, no siempre es posible separar los datos de manera lineal. Para solucionar esto se introducen las variables de holgura ξ_j para cada x_j. Estas variables permiten que algunos puntos de datos estén dentro del margen o incluso en el lado incorrecto del hiperplano, proporcionando cierta flexibilidad en el modelo. Además, se utilizan funciones kernel para transformar los datos a un espacio donde puedan ser separados linealmente.

Redes Neuronales Artificiales

Una red neuronal es un método de la inteligencia artificial, que se asemeja en estructura y funcionamiento a las neuronas biológicas para procesar información de entrada y así generar una salida específica de acuerdo con el entorno del problema a resolver.

Las redes neuronales están compuestas por nodos o unidades interconectados a través de enlaces directos. En la figura 2.3 se puede observar un ejemplo de una red neuronal totalmente conectada, en donde los datos ingresan por medio de la “capa de entrada”, luego atraviesan la “capa oculta” (puede constar de varias capas), y salen por la “capa de salida” [19].

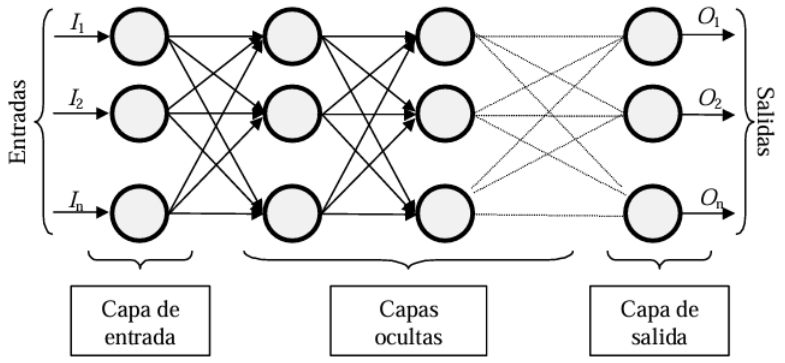

Figura 2.3: Ejemplo de una red neuronal totalmente conectada [19].

Conforme a lo expuesto por Stuart Russell y Peter Norving en [9], estos enlaces parten de la unidad i a la unidad j, y permiten propagar la activación a_i. A su vez, tienen un peso asociado $W_{i,j}$ que determina la importancia y alcance de la conexión. Por lo que cada neurona procesa múltiples valores de entrada como si fueran uno solo (llamada entrada global), cada unidad j debe calcular primero una suma ponderada de sus entradas. La siguiente ecuación, denominada función de entrada, describe lo anteriormente dicho:

$$entrada_j = \sum_{i=0}^{n} W_{i,j} a_j \tag{2.7}$$

Luego, se aplica una función de activación g a la suma anterior para derivar la salida:

$$a_i = g(entrada_j) = g(\sum_{i=0}^{n} W_{i,j} a_j) \tag{2.8}$$

La función de activación tiene varios propósitos. Entre ellos están normalizar las salidas a rangos específicos. Esta función transforma la entrada global en un valor (estado) de activación. La unidad es “activa” (cerca de +1) cuando se den las entradas “correctas”, e “inactiva” (cerca de 0) cuando se den las entradas incorrectas. Por otro lado, las funciones de activación deben ser no lineales, pues de esta forma permiten que la red aprenda y modele relaciones complejas entre entradas y salidas, de lo contrario, se reducirían a una simple función lineal [9].

La distribución de neuronas dentro de una red se realiza formando nive-

les o capas, con una cantidad específica de ellas. Existen las redes monocapa, donde se establecen conexiones directas entre las neuronas que hacen parte de la única capa que forma la red. Por lo general todas las neuronas de una sola capa utilizan conexiones hacia adelante, en la que reciben señales de entrada desde otra capa anterior, y envían señales de salida a una capa posterior. Además, se encuentran las redes multicapa. Estas se crean a partir de un conjunto de neuronas agrupadas en distintos niveles o capas, que, además de hacer uso de las conexiones hacia adelante, tienen la posibilidad de realizarlas hacia atrás, donde conectan la salida de las neuronas de capas posteriores a la entrada de capas anteriores. Esto se emplea para llevar a cabo un proceso denominado retropropagación, en el que los valores de los pesos de la red se van ajustando por medio de la diferencia entre la salida obtenida por la red y la salida deseada, con la intención de mejorar el rendimiento de esta.

Para abordar problemas como el procesamiento de imágenes y procesamiento del lenguaje natural, se emplean redes neuronales de aprendizaje profundo o Deep Learning. Algunos ejemplos de estas son: las **Máquinas de Boltzmann restringidas**, las **Redes Neuronales Recurrentes** y las **arquitecturas Codificador-Decodificador**.

Máquinas de Boltzmann restringidas (MBR)

Es un tipo de red neuronal basada en la Máquina Boltzmann, inventada en 1985 por Geoffrey Hinton. En la Figura 2.4 se muestra un ejemplo de una máquina de Botlzmann, donde se observan las capas de entrada v_i, denominadas capas visibles, y las capas ocultas h_i. En este tipo de red neuronal, se tienen múltiples neuronas en cada capa, que no solo están conectadas bidireccionalmente a neuronas de otras capas, sino también a neuronas dentro de la misma capa.

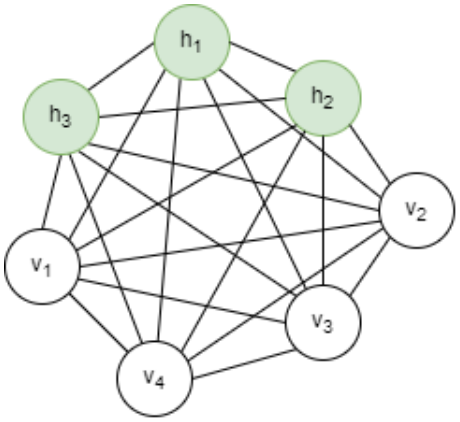

Figura 2.4: Representación gráfica de una máquina de Boltzmann [20].

Una máquina de Boltzmann restringida (MBR) es una red neuronal con distribuciones de probabilidad aleatorias, que se diferencia de la máquina de Boltzmann por sus conexiones restringidas. Como se puede ver en la Figura 2.5, en este tipo de red no hay nodos del mismo grupo conectados entre sí, ni en capa visible ni en la capa oculta.

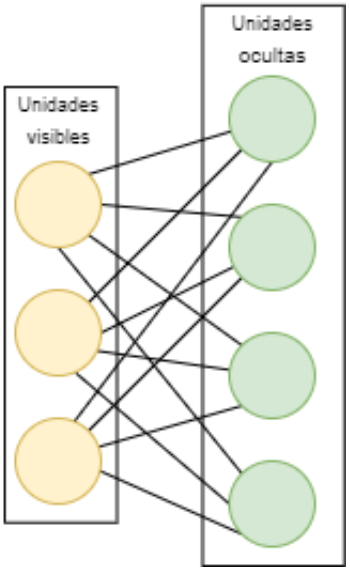

Figura 2.5: Representación gráfica de una Máquina de Boltzmann restringida [20].

Las redes MBR son modelos basados en energía, donde la interacción entre unidades visibles y ocultas se describe en términos de una función energética, la cual describe la probabilidad de cierto estado [21] y se puede representar con la siguiente ecuación:

$$E(v,h) = -\sum_{i \in visible} a_i * v_i - \sum_{j \in hidden} b_j * h_j - \sum_{i,j} v_i * h_j * w_{i,j} \tag{2.9}$$

E es el modelo basado en energía para la capa oculta h y la capa visible v, donde v_i son los elementos del vector de entrada, h_i corresponde a los elementos del vector oculto, a_i y b_j son los términos de bias para las neuronas visibles y ocultas respectivamente, y $w_{i,j}$ son los pesos entre la neurona i en la capa de entrada y la neurona j en la capa oculta. En una red MBR, se busca minimizar la energía de las configuraciones que representan los datos de entrenamiento, para lograr aumentar la probabilidad de estas configuraciones, haciendo que el modelo aprenda las distribuciones de los datos de entrenamiento.

Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes (RNN) son un tipo de arquitectura de red neuronal empleada en diversas tareas de aprendizaje automático que implican el modelado de secuencias para detectar patrones, donde tanto la longitud de los datos de entrada como la de los datos de salida pueden variar [22]. Este tipo de red es capaz de manejar datos secuenciales debido a la presencia de uno o más bucles de retroalimentación, conocidos como *tiempo* en este contexto. Estos bucles de retroalimentación son ciclos recurrentes a lo largo del tiempo o secuencia, permiten que la información se procese y recuerde a lo largo de varios pasos de tiempo.

De acuerdo con lo descrito en [23], la RNN consta de tres capas: entrada, oculta recurrente y salida. La capa de entrada contiene N unidades. Los datos que ingresan a esta capa son una secuencia de vectores a lo largo del tiempo t, como por ejemplo $\{..., x_{t-1}, x_t, x_{t+1}, ...\}$, donde $x_t = (x_1, x_2, ..., x_N)$.

Por otro lado, la capa oculta tiene M unidades ocultas $h_t = (h_1, h_2, ..., h_M)$, las cuales están conectadas entre sí por medio del tiempo con conexiones recurrentes definidas por una matriz de pesos W_{HH}. A su vez, tiene conexiones con las unidades de entrada por medio de W_{IH}.

Con base en lo anterior, se tiene la ecuación 2.10 , que describe el cálculo de un nuevo estado oculto h_t en el tiempo t mediante una función de activación:

$$h_t = \phi(W_{IH}x_t + W_{HH}h_{t-1} + b_h) \tag{2.10}$$

Donde ϕ es una función de activación, como la tangente hiperbólica o la función sigmoide, $W_{IH}x_t$ representa la entrada actual multiplicada por la matriz de pesos de entrada a oculta, $W_{HH}h_{t-1}$ representa la multiplicación entre el estado oculto anterior por la matriz de pesos recurrentes, y b_h es un vector de sesgo (bias) de la capa oculta, que permite aprender y representar relaciones complejas de datos.

En esta ecuación, se toma una entrada de la secuencia x_t y se combina con el estado oculto anterior h_{t-1}, multiplicados por las respectivas matrices de pesos y sumados con el vector de sesgo b_h de la capa oculta.

La capa de salida se encuentra conectada a las unidades ocultas por una matriz de pesos W_{HO}. Cada unidad de la capa de salida, representada por $y_t = (y_1, y_2, ..., y_P)$, produce un resultado y_t en el tiempo t que se calcula como:

$$y_t = \phi(W_{HO}h_t + b_o) \tag{2.11}$$

Una representación gráfica sobre el funcionamiento de una red neuronal recurrente se puede observar en la Figura 2.6.

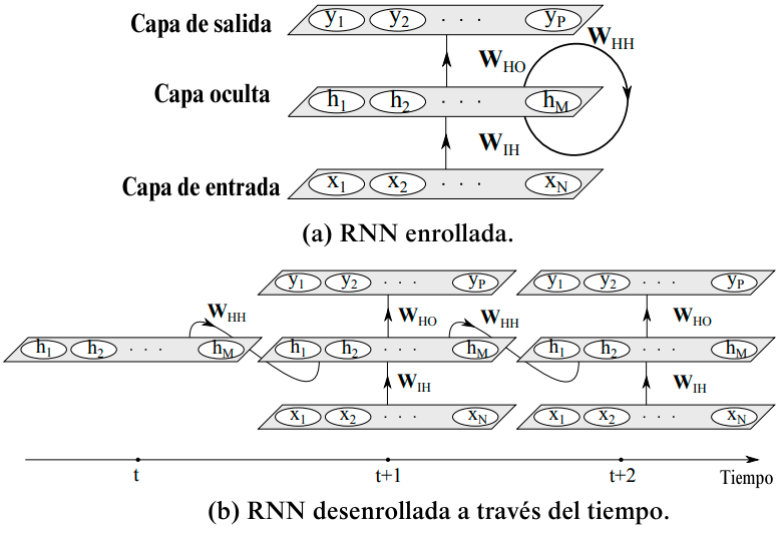

Figura 2.6: Una red neuronal recurrente y su estructura desenrollada a través del tiempo t [23].

En la figura anterior dentro de la parte a, se pueden observar las capas que componen una RNN, incluyendo la recurrencia de la capa oculta. En la parte b se encuentra esta red desenrollada, puesto que las unidades de la capa oculta están conectadas entre sí a través del tiempo con conexiones recurrentes.

Es importante mencionar que, una de las principales limitaciones de las redes neuronales recurrentes es su dificultad para aprender dependencias a largo plazo [22], es decir, la capacidad del modelo para recordar y utilizar información en secuencias largas se encuentra restringida. Para abordar esto, se desarrollaron variantes de RNN como: **Long Short-Term Memory (LSTM)** y **Gated Recurrent Unit (GRU)**, que se describen a continuación.

LSTM - Long Short Term Memory

Hochreiter y Schmidhuber, propusieron en 1997 las redes LSTM, caracterizadas por ser eficaces para retener y utilizar información en secuencias más largas [24]. Para lograr esto, introducen una estructura que está formada por celdas de memoria y compuertas (de entrada, de olvido y de salida) para controlar el flujo de información.

Conforme a lo expuesto en [22], cada una de las partes nombradas anteriormente cumplen una función específica. La celda de memoria c_t almacena la información contenida de una unidad LSTM, la compuerta de entrada i_t decide cómo actualizar el estado interno basado en la entrada actual y el estado interno anterior, o sea, cuánto contenido nuevo se debe memorizar, la compuerta de olvido f_t determina qué parte del estado interno anterior debe olvidarse, y la compuerta de salida o_t controla cuánto del estado interno afecta al sistema.

Ahora bien, el contenido de la celda de memoria c_t^j , donde j representa la j-ésima unidad LSTM en el tiempo t, se actualiza de acuerdo con la siguiente ecuación:

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j \tag{2.12}$$

Donde

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1}) \tag{2.13}$$

En esta ecuación, c_t^j se calcula a través de la suma ponderada del nuevo contenido $\tilde{c}_t$ y el contenido anterior de la celda c_{t-1}, modulada por la compuerta de entrada i_t y la compuerta de olvido f_t, respectivamente.

Por otro lado, las compuertas de entrada, olvido y salida se calculan a partir de los estados ocultos anteriores h_{t-1} y la entrada actual x_t, formando las siguientes expresiones [22]:

$$i_t = \phi(W_i x_t + U_i h_{t-1}) \tag{2.14}$$

$$f_t = \phi(W_f x_t + U_f h_{t-1}) \tag{2.15}$$

$$o_t = \phi(W_o x_t + U_o h_{t-1}) \tag{2.16}$$

Después de que el contenido de la memoria se actualiza considerando las señales de las diferentes compuertas, se puede formar la siguiente ecuación:

$$h_t^j = o_t^j tanh(c_t^j) \tag{2.17}$$

La ecuación 2.17 muestra que el estado oculto h_t^j se calcula aplicando la función tanh al contenido de la memoria c_t^j, y luego ajustando ese valor con o_t^j. Este ajuste permite que la LSTM controle cuánto del contenido de la memoria se trasfiere al estado oculto, dependiendo del valor de la compuerta de salida.

Sintetizando lo anteriormente expuesto, una red neuronal recurrente con unidades LSTM, modula el contenido de la memoria a través de diferentes compuertas, asegurando que se olvide, memorice, y exponga de manera adaptativa el contenido de la memoria [22]. Esto le permite a la red manejar dependencias a largo plazo de manera efectiva, facilitando el aprendizaje de patrones complejos y mejorando la capacidad de predicción en secuencias largas.

GRU - Gated Recurrent Unit

En el 2014, Cho et Al. Propusieron una variante de RNN que también aborda el problema de memoria a corto plazo, pero con una estructura más simple en comparación con la LSTM. Según [24], una unidad GRU consta de tres

componentes principales: la compuerta de actualización, que combina la compuerta de entrada y la compuerta de olvido de una LSTM, la compuerta de reinicio y el contenido de la memoria actual.

La compuerta de actualización z_t decide cuánta información debe mantenerse y combinarse con la entrada actual en un paso de tiempo t específico. Se calcula a partir los estados ocultos anteriores h_{t-1} y la entrada actual x_t, cada una multiplicada por una matriz de pesos. En la siguiente ecuación se muestra lo anterior:

$$z_t = \phi(W_z x_t + U_z h_{t-1}) \tag{2.18}$$

La compuerta de reinicio r_t determina qué cantidad de información pasada debe olvidarse. Se calcula de manera similar a la compuerta anterior:

$$r_t = \phi(W_r x_t + U_r h_{t-1}) \tag{2.19}$$

La principal diferencia con las LSTM es que las GRU muestran todo el contenido de su memoria en cada paso de tiempo, y deben equilibrar la información anterior con la nueva para evitar que la memoria se sature o se pierdan datos importantes.

Retomando lo anterior, el contenido de la memoria actual $\tilde{h}_t$ se calcula en función de la compuerta de reinicio r_t y la concatenación del estado oculto anterior h_{t-1} y la entrada actual x_t. Esto se pasa a través de la función de activación tanh, como se observa en la siguiente ecuación:

$$\tilde{h}_t = tanh(W x_t + r_t \odot U h_{t-1}) \tag{2.20}$$

Por otro lado, el estado final de la memoria h_t se calcula mediante una combinación del estado oculto anterior h^j_{t-1} y la activación candidata $\tilde{h}_t$. La activación candidata es el valor propuesto para el nuevo estado oculto. La puerta de actualización z_t regula el equilibrio entre el estado oculto anterior y la activación candidata. A continuación, se presenta la ecuación que modela el estado final de memoria:

$$h^j_t = (1 - z^j_t) h^j_{t-1} + z^j_t \tilde{h}^j_t \tag{2.21}$$

Resumiendo lo anterior, las compuertas de actualización y reinicio de una unidad GRU equilibran la incorporación de nueva información y deciden qué información retener y olvidar en cada paso del tiempo.

Sin embargo, aunque las LSTM y GRU han mejorado significativamente la capacidad de las redes neuronales recurrentes para manejar dependencias a largo plazo, pueden tener dificultades cuando se trata de comprender estructuras de entrada complejas, como secuencias de diferentes longitudes [24]. Para abordar esto, se introdujeron variaciones en las RNN, siendo la arquitectura codificador-decodificador una de las más utilizadas en el campo del procesamiento de lenguaje natural.

Codificador-Decodificador

La arquitectura codificador-decodificador (encoder-decoder), comúnmente conocida como sequence-to-sequence, introduce un enfoque distinto para transformar secuencias al combinar dos redes recurrentes: un codificador y un decodificador. De acuerdo con [25], la idea central de esta arquitectura radica en utilizar un codificador para tomar la secuencia de entrada y crear una representación contextualizada de la misma. Posteriormente, esta representación se utiliza en el decodificador para generar una secuencia de salida específica.

Las redes codificador-decodificador estan compuestas por tres partes: un codificador, un vector de contexto c, y un decodificador. En la figura 2.7 se pueden observar una arquitectura básica de esta red.

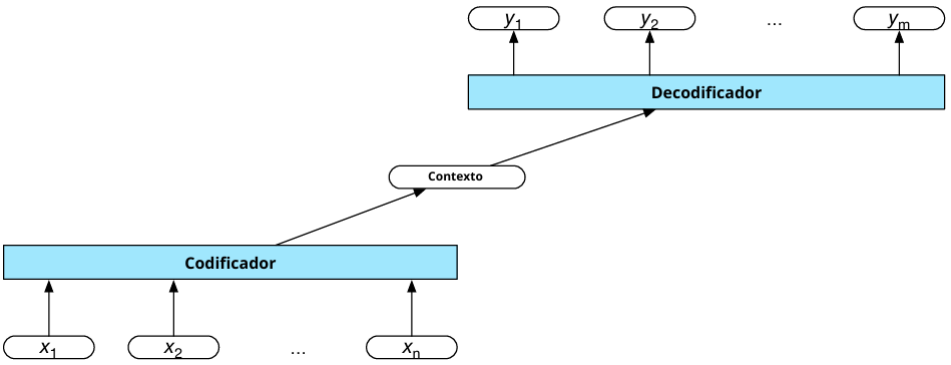

Figura 2.7: Arquitectura de una red codificador-decodificador [25].

El codificador puede utilizar diferentes tipos de redes para procesar las secuencias de entrada, como RNN, LSTM, GRU, entre otras. En este proceso, toma la secuencia de entrada $x_1, x_2, ..., x_n$ y a medida que procesa cada elemento, genera una secuencia de representaciones contextualizadas $h_1, h_2, ..., h_n$. Estas representaciones encapsulan la información relevante de los elementos de entrada en un formato que se utiliza para crear el vector de contexto.

Por otro lado, el vector de contexto c se define como una función de todos los estados ocultos producidos por el codificador, es decir, $c = f(h_1, h_2, ..., h_n)$. Trasmite la información esencial de la secuencia de entrada al decodificador, pues debido a que la cantidad de estados ocultos varía con la longitud de la secuencia de entrada, resulta difícil utilizarlos directamente como contexto para el decodificador.

De esta manera, el decodificador toma el vector de contexto c como entrada inicial. A partir de este, genera una secuencia de estados ocultos $h'_1, h'_2, ..., h'_m$ que se utilizan para crear la secuencia de salida $y_1, y_2, ..., y_m$ por medio de un proceso de generación autorregresiva. Es decir, cada elemento se produce uno a la vez, basándose en los elementos generados anteriormente. A continuación, se presentan las expresiones que modelan el proceso del decodificador:

$$h_0^d = c \tag{2.22}$$

$$h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d) \tag{2.23}$$

$$z_t = f(h_t^d) \tag{2.24}$$

$$y_t = softmax(z_t) \tag{2.25}$$

Donde h_0^d corresponde al estado oculto inicial del decodificador, h_t^d representa el estado oculto del decodificador en el paso del tiempo t. Este se calcula utilizando una función g que toma como entrada la salida estimada en el paso anterior $\hat{y}_{t-1}$, y el estado oculto anterior del decodificador. La función g puede ser una función de activación o una unidad recurrente como LSTM o GRU. A partir del estado oculto h_t^d, se calcula una posible puntuación z_t para cada

posible elemento en la secuencia de salida. Esta puntuación se transforma mediante una función softmax para convertirla en probabilidades normalizadas.

En síntesis, la arquitectura codificador-decodificador no solo aborda dependencias a largo plazo, sino que también cuentan con una estructura formada por dos redes neuronales recurrentes diseñadas para mantener y actualizar información a lo largo de la secuencia. Además, estas redes tienen la capacidad de manejar secuencias de entrada y salida de longitudes variables.

2.3. Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural (PLN) es una técnica del aprendizaje automático, que permite interpretar, manipular y comprender el lenguaje humano. Las palabras son los elementos fundamentales del lenguaje, y por tal razón, constituyen la base sobre la que se construyen todos los procesos de PLN.

A continuación, se presentan algunas técnicas utilizadas en el procesamiento del lenguaje natural para el preprocesamiento inicial de los datos y su posterior transformación numérica.

2.3.1. Tokenización

Es un proceso en el que se divide el texto en unidades más pequeñas, conocidas como tokens. Estos pueden ser caracteres, palabras, oraciones, entre otros, dependiendo de la estrategia de tokenización elegida y el objetivo esperado [26].
Por ejemplo, al tokenizar por palabras, se puede identificar la frecuencia de aparición de ciertas palabras. Por otro lado, al tokenizar por oraciones, es posible analizar cómo las palabras se relacionan entre sí dentro de un contexto.
La elección de la estrategia de tokenización depende de la tarea a realizar y del tipo de datos a manejar. Es un paso fundamental dentro del procesamiento del lenguaje natural, puesto que prepara el texto para su posterior manipulación.

2.3.2. StopWords

En ciertas tareas del procesamiento del lenguaje natural, algunas palabras son tan comunes que aportan poca información relevante. Estas palabras, conocidas como palabras vacías o stopwords, son un conjunto de términos de uso común en cualquier idioma [27]. Por ello, es importante identificarlas y removerlas para evitar sesgos en el modelamiento.

2.3.3. Word Embedding

Dentro de la comunicación se pueden producir diversas oraciones con un número finito de palabras. Estas pueden tener significados diferentes según el contexto en el que se encuentren [28]. Desde un punto de vista computacional, no se puede comprender el contexto de las oraciones de la misma manera que se hace en el lenguaje natural. Por ello, en el área de procesamiento del lenguaje natural, se han desarrollado diferentes métodos para representar numéricamente el significado de las palabras.

Uno de los métodos más utilizados es la representación semántica de palabras a través de vectores (*Vector Semantics*), el cual aprende representaciones del significado de las palabras a partir de sus distribuciones en los textos. La idea central de este método es que palabras que aparecen en contextos similares tienen significados similares. Estas palabras se representan como vectores llamados **embeddings**, los cuales se ubican en un espacio vectorial [29]. Como se puede observar en la figura 2.8, las palabras que están vinculadas semántica o sintácticamente están más cerca en el espacio vectorial que aquellas que no están relacionadas.

Figura 2.8: Proyección de embeddings para algunas palabras y frases en inglés con significados similares [29].

Algunos modelos que generan representaciones de palabras utilizando el tipo de vectores nombrado anteriormente son GloVe y Word2Vec. Estos modelos emplean grandes conjuntos de texto para capturar relaciones semánticas de palabras basadas en el contexto de cada una de ellas.

2.4. Métricas para la evaluación

Para evaluar los modelos a través de diversas métricas, primero se debe utilizar la matriz de confusión. Esta matriz permite visualizar los resultados del modelo comparándolos con los valores reales de un conjunto de datos. Es una tabla que desglosa el número de instancias de verdad fundamental (*ground truth*) de una clase específica frente al número de instancias de clase previstas.

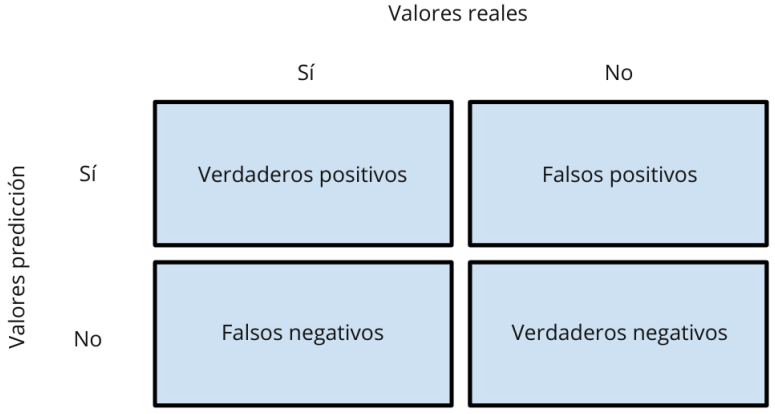

Figura 2.9: Matriz de confusión [30].

En la Figura 2.9, se puede observar que las filas representan los valores predichos y las columnas representan los valores reales (positivo o negativo) de una clase determinada. A continuación, se desglosan cada uno de sus elementos:

- True Positive (TP): indica la cantidad de valores predichos que concuerda con los valores reales en la clase positiva.
- True Negative (TN): corresponde a la cantidad de valores predichos que coinciden con los valores reales en la clase negativa.
- False Positives (FP): expresa la cantidad de valores predichos que fueron clasificados como positivos cuando en realidad eran negativos. Se conoce como un error de tipo 1.
- False Negative (FN): muestra la cantidad de valores predichos que fueron clasificados como negativos cuando en realidad eran positivos. Se conoce como un error de tipo 2.

La matriz de confusión es útil para calcular otras métricas de evaluación del modelo, como *Precision*, *Recall*, y *F-score* [31].

Precision

Esta métrica mide la frecuencia con la que un modelo predice correctamente la clase positiva. Un valor alto de Precision indica menos errores de tipo 1. Se define como:

$$Precision = \frac{TP}{TP + FP} \tag{2.26}$$

Recall

Esta métrica mide la frecuencia con la que un modelo identifica correctamente instancias positivas (True Positives) de todas las muestras positivas reales en el conjunto de datos. Un valor alto de Recall indica menos errores de tipo 2. Se define como:

$$Recall = \frac{TP}{TP + FN} \tag{2.27}$$

F-score

Esta métrica combina Precision y Recall para proporcionar una medida general de la calidad del modelo. Se define como la media armónica entre Precision y Recall:

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.28}$$

Un F-score alto significa que el modelo tiene un buen equilibrio de Precision y Recall.

Técnica para evaluación de resúmenes ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) es un paquete de software que utiliza un conjunto de medidas para evaluar la calidad de un resumen generado comparándolo con un resumen de referencia. Estas medidas se destacan por contar el número de unidades superpuestas, como n-gramas, entre un resumen generado y un resumen ideal [31]. A continuación, se presentan cuatro medidas diferentes:

ROUGE-N: cuenta n-gramas entre el resumen candidato y un conjunto de resúmenes de referencia, utilizando una técnica relacionada con el Recall. Esta métrica favorece a los resúmenes que comparten muchas palabras con diversos resúmenes referentes.

ROUGE-L: da importancia a la subsecuencia común más larga entre el resumen de referencia y el resumen generado, considerando el resumen como una secuencia de palabras. Esta métrica ayuda a comparar la similitud entre los dos resúmenes.

ROUGE-W: es una mejora a ROUGE-L, pues además de darle importancia a la subsecuencia más larga, también tiene en cuenta la posición de cada palabra para priorizar aquellos resúmenes con menor diferencia espacial entre la subsecuencia común más larga.

ROUGE-S: mide la superposición de saltos de bi-gramas entre el resumen generado y el resumen de referencia. Normalmente se utiliza para la evalua-

ción de traducciones.

ROUGE-SU: es una extensión de ROUGE-S que da importancia al orden de las palabras en el resumen.

2.5. Antecedentes de investigación

El resumen de documentos se basa en generar un texto conciso que represente de forma coherente la información del documento original. Es una de las tareas más desafiantes en el procesamiento del lenguaje natural (PLN) y la inteligencia artificial (IA) en general [31]. La atención de los investigadores en esta tarea data de 1950, cuando H. P. Luhn creó un sistema de resumen de texto automático basado en la frecuencia de términos [32]. En 1958, Luhn destacó la importancia de observar la frecuencia de las palabras, argumentando que aquellas con frecuencia media son las más importantes. Además, explicó la existencia de las palabras vacías dentro de un texto, como pronombres o preposiciones, que no aportan suficiente información para entender un texto.

Por otro lado, Edmundson [33] sentó las bases para otras tendencias en la generación automática de resúmenes. En 1969, amplió el enfoque de Luhn al proponer factores adicionales que consideraba relevantes dentro de una oración: la frecuencia de la palabra en el texto, la aparición de palabras de la oración en títulos o encabezados de secciones, y la posición de la oración en el texto.

Generalmente, las técnicas de resumen se clasifican en extractivas y abstractivas. Los métodos extractivos tienen como objetivo seleccionar determinadas oraciones destacadas de documentos, mientras que las técnicas abstractivas parafrasean de manera concisa la información del documento [34]. Para el desarrollo de este proyecto, se utilizaron técnicas extractivas.

Algunos trabajos relacionados que utilizan técnicas extractivas para resumir documentos incluyen el clasificador Naïve Bayes desarrollado por Kupiec, Pedersen y Chen, que clasifica las oraciones del documento que se incluirán en el resumen. Conroy y O'leary propusieron un modelo oculto de Markov

(HMM) para el resumen de textos [32]. Hirao, Isozaki, y Maeda demostraron en [18] que las MVS se pueden utilizar para hacer resúmenes, pues tienen un buen desempeño para la extracción de oraciones importantes, y Mani y Bloedorn desarrollaron un modelo basado en grafos [32]. Adicionalmente, con la popularización del aprendizaje profundo, se comenzaron a emplear técnicas basadas en redes neuronales para la construcción de resúmenes.

Estas técnicas se aplican en la producción automática de resúmenes en dominios específicos, como reportes técnicos de diagnósticos médicos, artículos científicos, eventos deportivos, entre otros. En nuestro caso, se ha escogido el enfoque de noticias económicas.

Dentro de las investigaciones enfocadas en redes neuronales está el trabajo de Kageback et al., que en 2014 emplearon un codificador automático recursivo para resumir documentos, logrando el mejor rendimiento en un conjunto de datos de opinión (Opinosis) [34], en 2016, Singh, Kumar, Mangal y Singhal propusieron un método para resumir textos bilingües utilizando la máquina Boltzmann restringida (RBM) [35]. Por otro lado, en el ámbito de redes neuronales enfocadas a portales de noticias, se encuentran los trabajos de Cheng y Lapata, quienes en 2016 propusieron un enfoque basado en redes neuronales y características de oraciones continuas. Desarrollaron una arquitectura codificador-extractor para resúmenes extractivos de un solo documento en un conjunto de datos de CNN/DailyMail [36]. Asimismo, Nallapati, Zhai, y Zhou crearon un modelo de secuencia llamado SummaRuNNer, basado en una red neuronal recurrente para generar resúmenes extractivos de documentos en CNN/DailyMail [37].

Por otra parte, la llegada de la IA generativa representa un hito importante en el campo de la inteligencia artificial, en particular en el ámbito del procesamiento del lenguaje natural (PLN) [38]. La aparición de grandes modelos de lenguaje (LLM), como Generative Pre-trained Transformer 4 (GPT-4) y ChatGPT ha mostrado un rendimiento destacado en tareas de generación de texto, incluyendo la creación de resúmenes. Investigaciones de Goyal [39] y Zhang [40] han demostrado que los resúmenes de noticias generados por LLMs ya alcanzan el nivel de los realizados por humanos.

Lo expuesto anteriormente permite concluir que para la generación de resúmenes extractivos existen enfoques variados, que van desde el uso de algoritmos basados en grafos hasta técnicas como el modelo oculto de Markov (HMM), Naïve Bayes, aprendizaje profundo, IA generativa, entre otras. Algunas de estas técnicas han sido aplicadas en diversos dominios, incluyendo el de las noticias económicas, demostrando su adaptabilidad y efectividad en la extracción de información relevante para formar resúmenes.

Capítulo 3

Etapa de Análisis

3.1. Análisis del problema

El problema de generar resúmenes extractivos de noticias económicas, discutido en el Capítulo 1, presentó algunos desafíos que se abordaron al desarrollar una solución. Dado que las noticias están escritas en lenguaje natural, es común encontrar errores gramaticales, problemas con el uso de los signos de puntuación o símbolos especiales. Por lo tanto, fue crucial realizar un preprocesamiento antes de aplicar las técnicas seleccionadas, para limpiar y preparar adecuadamente las noticias para la generación de resúmenes. Fue fundamental que las técnicas elegidas estuvieran alineadas con el ámbito del problema a resolver, que es la generación de resúmenes extractivos.

3.2. Análisis de la solución

Para solucionar el problema planteado, se propuso diseñar y desarrollar dos modelos capaces de generar resúmenes de noticias económicas. Estos modelos fueron entrenados con un conjunto de resúmenes previamente creado. Se seleccionaron dos técnicas entre diversas opciones de aprendizaje automático y se realizaron experimentos sobre las arquitecturas para determinar los mejores parámetros con el fin de lograr generar resúmenes extractivos.

3.2.1. Lista de actividades

Objetivo 1: Seleccionar y preprocesar noticias económicas de diferentes portales de noticias

Para este objetivo se realizaron las siguientes actividades:

- Recolectar documentos de noticias económicas disponibles en internet, con su respectivo resumen
- Preprocesar noticias económicas (detección y eliminación de símbolos, StopWords, Tokenización y Word Embedding)
- Documentar el proceso realizado en la fase de recolección de datos y preprocesamiento

Objetivo 2: Desarrollar dos modelos de generación automática de resúmenes usando el enfoque de aprendizaje profundo

Para este objetivo se realizaron las siguientes actividades:

- Identificar los requisitos funcionales y no funcionales del sistema
- Analizar los criterios de selección para elegir dos técnicas de aprendizaje profundo entre las de la Sección 2.2.4.
- Diseñar los modelos utilizado dos técnicas de aprendizaje profundo
- Documentar el proceso realizado en la fase de análisis y diseño

Objetivo 3: Evaluar el desempeño de cada modelo haciendo uso de la métrica ROUGE (Recall-Oriented Understudy for Gisting Evaluation) y comparar

Para este objetivo se realizaron las siguientes actividades:

- Definir y seleccionar tres métricas de ROUGE para evaluar los modelos
- Diseñar los experimentos para cada técnica de aprendizaje profundo
- Realizar un análisis en profundidad sobre el desempeño de los modelos de ambas técnicas
- Comparar los resultados de los modelos y seleccionar el mejor en base a los resultados
- Documentar el proceso realizado en la fase de evaluación

3.2.2. Requisitos

Dado que este sistema de generación de resúmenes de noticias económicas no cuenta con interfaz gráfica ni es una aplicación completa, los requisitos se enfocaron más en la lógica del sistema, el rendimiento y la usabilidad en un entorno de desarrollo.

Requisitos funcionales

- El sistema debe cargar y procesar noticias económicas desde archivos TXT y archivos JSON
- El sistema debe preprocesar los textos eliminando caracteres especiales, StopWords, tokenizando las palabras, y realizando Word Embedding
- El sistema debe aplicar dos técnicas de aprendizaje profundo para generar resúmenes automáticos de las noticias
- El sistema debe permitir cargar un conjunto de referencia de resúmenes para la evaluación y comparación con la noticia utilizando ROUGE
- El sistema debe generar un archivo de resultados que contenga los resúmenes generados y las puntuaciones ROUGE correspondientes

Requisitos no funcionales

- El sistema debe poder ajustarse para ejecutar en un entorno de CPU o GPU sin necesidad de cambios significativos en el código
- El sistema debe ser capaz de procesar lotes de hasta 1000 noticias económicas sin superar los límites de memoria de la GPU o CPU disponible
- El sistema debe estar documentado adecuadamente para que cualquier usuario con conocimientos básicos de Python y aprendizaje profundo pueda ejecutarlo sin complicaciones

3.3. Identificación de criterios de selección

Se detalló el proceso de identificación de criterios para seleccionar dos técnicas entre las mencionadas en la Sección 2.2.4. Estas son: Naïve Bayes,

Máquinas de Vectores de Soporte (MVS), Máquina de Boltzmann restringida (MBR), Redes Neuronales Recurrentes (RNN), y la arquitectura codificador-decodificador. Cada técnica fue evaluada según los siguientes criterios:

- Relevancia de la técnica para el problema de generación de resúmenes extractivos
- Desempeño de la técnica en estudios similares
- Disponibilidad de librerías para su implementación
- Tiempo estimado de diseño y desarrollo

Naïve Bayes

- Esta técnica es comúnmente utilizada para problemas de clasificación de texto, lo que la hace adecuada para la generación de resúmenes extractivos mediante la clasificación de cada oración de la noticia como parte del resumen o no.
- Según los estudios realizados en [41] y [16], el clasificador Naïve Bayes asigna una puntuación a cada oración y selecciona aquellas con puntajes altos para incluirlas en el resumen. Aunque los experimentos fueron satisfactorios, Naïve Bayes asume que todas las características son independientes entre sí, lo que puede afectar la precisión del modelo si existe correlación entre dichas características.
- Existen paquetes como Scikit-Learn, NLTK, y NumPy que ofrecen una amplia gama de funcionalidades y pueden ser utilizados en diversos contextos y aplicaciones para implementar y evaluar modelos de Naive Bayes.
- El diseño de esta técnica no es altamente complejo y se considera que tiene un enfoque simplista, lo cual puede limitar su capacidad para capturar relaciones complejas entre las características, a diferencia de modelos más avanzados como las redes neuronales.

Máquinas de Vectores de Soporte (MVS)

- En el contexto de la generación de resúmenes extractivos, las Máquinas de Vectores de Soporte se utilizan para clasificar cada oración de una noticia económica como relevante o no para el resumen. Sin embargo, en conjuntos de datos grandes, el tiempo de entrenamiento puede aumentar significativamente.

- Se ha estudiado un método basado en Máquinas de Vectores de Soporte para el resumen de texto en referencias [42], donde los experimentos demuestran consistentemente un buen rendimiento en tareas de categorización de texto. Además, en [18] se propone un método de extracción de oraciones basado en MVS, que clasifica las oraciones como importantes o no para el resumen. Con tasas de resumen del 30 % y 50 %, los resultados muestran un buen desempeño promedio del modelo de Máquinas de Vectores de Soporte, alcanzando un 51.6 % y 63.5 % respectivamente.

- Una de las librerías más utilizadas para implementar Máquinas de Vectores de Soporte en Python es Scikit-learn. Existen otras bibliotecas como LIBSVM y TensorFlow que también admiten la implementación de MVS.

- Esta técnica requiere un tiempo considerable en el proceso de diseño debido a la necesidad de una selección cuidadosa de parámetros, como el parámetro de regularización C y el tipo de kernel. La elección incorrecta de estos parámetros puede afectar negativamente el rendimiento del modelo.

Máquina de Boltzmann restringida (MBR)

- La máquina de Boltzmann restringida es una red neuronal cuyo enfoque en la generación de resúmenes se centra en modelar la probabilidad de ocurrencia de las oraciones que deben incluirse en el resumen.

- En [35], se propone un enfoque que consta de fases de procesamiento, extracción de características, aprendizaje profundo y posprocesamiento para generar resumen de texto. Se utiliza el algoritmo de aprendizaje MRB, con el cual se logró una precisión cercana al 85 %. En el artículo [43], se analiza el problema del resumen de texto y se plantea una

metodología para generar resúmenes extractivos en el dominio de noticias deportivas en Tamil utilizando Máquinas de Boltzmann restringidas. Según el análisis realizado en el texto, aunque esta técnica puede aplicarse para la generación de resúmenes extractivos, su rendimiento parece depender del preprocesamiento del texto y de la matriz generada de características de la oración.

- Para implementar esta red neuronal, se pueden utilizar librerías como Tensorflow, Keras, y PyTorch, que ofrecen soporte para la implementación y entrenamiento de MRBs.

- El entrenamiento de MRB puede ser complejo y costoso computacionalmente. También pueden surgir dificultades al escalar a conjuntos de datos grandes o modelos con muchas capas.

Redes neuronales recurrentes (RNN)

- Las redes neuronales recurrentes se emplean ampliamente en tareas de aprendizaje automático que implican el modelado de secuencias para detectar patrones, lo cual las hace aptas para la generación de resúmenes extractivos.

- Se destaca la investigación presentada en [37], que introduce un modelo de secuencia llamado SummaRuNNer basado en redes neuronales recurrentes para el resumen extractivo de documentos. Este enfoque trata la generación de resúmenes como un problema de clasificación de secuencias, donde se toma una decisión binaria sobre cada oración para incluirla o no en el resumen. Además, se demuestra que las RNN logran un rendimiento comparable o superior a otros tipos de redes neuronales en esta tarea.

- Entre las principales librerías para trabajar con redes neuronales recurrentes se encuentran Tensorflow, Keras, y PyTorch.

- El tiempo estimado para el diseño y desarrollo de una red neuronal recurrente puede variar significativamente según el tipo de problema a tratar. En muchos casos, se realizan ajustes continuos para mejorar el rendimiento en las métricas utilizadas para evaluar los resultados.

Codificador-Decodificador

- Las arquitecturas codificador-decodificador se utilizan para procesar secuencias de entrada y generar secuencias de salida en aplicaciones de procesamiento del lenguaje natural, por lo que son adecuadas para el problema de generación de resúmenes extractivos.
- El estudio presentado en [36], empleó una arquitectura codificador - decodificador para generar resúmenes extractivos. Estos modelos fueron entrenados en corpus de gran escala que comprenden miles de pares de documentos y resúmenes, como el corpus DailyMail. Los resultados experimentales fueron evaluados utilizando métricas ROUGE, mostrando resultados comparables con los del estado del arte en esta tarea.
- Algunas de las principales librerías en Python para implementar esta técnica son Keras, Tensorflow, y PyTorch.
- El tiempo estimado para el desarrollo está directamente relacionado con la complejidad del modelo, incluyendo el número de capas y unidades tanto en el codificador como en el decodificador. Entrenar arquitecturas con muchas capas o unidades puede ser computacionalmente costoso y requerir un tiempo significativo.

3.4. Elección de las técnicas

Se seleccionaron las dos técnicas utilizadas para el desarrollo de este proyecto, basada en los criterios de selección. El análisis detallado se presenta en el Cuadro 3.1.

Cuadro 3.1: Análisis de las técnicas

Criterios	Análisis	Técnicas
Relevancia de la técnica para el problema de generación de resúmenes extractivos	De acuerdo con las investigaciones realizadas, todas las técnicas se han utilizado para el problema de generación de resúmenes extractivos	Todas
Desempeño de la técnica en estudios similares	Las técnicas con mejor rendimiento en la generación de resúmenes extractivos son las Máquinas de Vectores de Soporte, las Redes Neuronales Recurrentes y la arquitectura Codificador-Decodificador	MVS, RNN, Sequence-to-Sequence
Disponibilidad de librerías para su implementación	Cada una de las técnicas presentadas dispone de una amplia gama de librerías para su diseño e implementación, que incluyen documentación detallada	Todas
Tiempo estimado de diseño y desarrollo	Las técnicas más adecuadas son las Redes Neuronales Recurrentes, la Máquina de Boltzmann restringida, y la arquitectura Codificador-Decodificador. Naïve Bayes y las Máquinas de Vectores de Soporte se descartan debido a la simplicidad en la configuración de sus parámetros	RNN, MBR, Sequence-to-Sequence

Se procede a contar los criterios que cumplen cada una de las técnicas propuestas. Estos se presentan en la Tabla 3.2.

Cuadro 3.2: Criterios cumplidos

Técnicas	Cantidad de criterios cumplidos
Redes Neuronales Recurrentes	4
Arquitecturas Codificador-Decodificador	4
Máquinas de Vectores de Soporte	3
Máquinas de Boltzmann restringidas	3
Naïve Bayes	2

De acuerdo con el análisis que se realizó y la cantidad de criterios propuestos cumplidos, las dos técnicas que se emplearon para el desarrollo de este proyecto son las **Redes Neuronales Recurrentes** y las **Arquitecturas Codificador-Decodificador**.

Capítulo 4

Etapa de Diseño

4.1. Diagrama de flujo

Se diseñó un diagrama de flujo que describe el desarrollo de un sistema para generar resúmenes de noticias económicas. Este se presenta en la Figura 4.1.

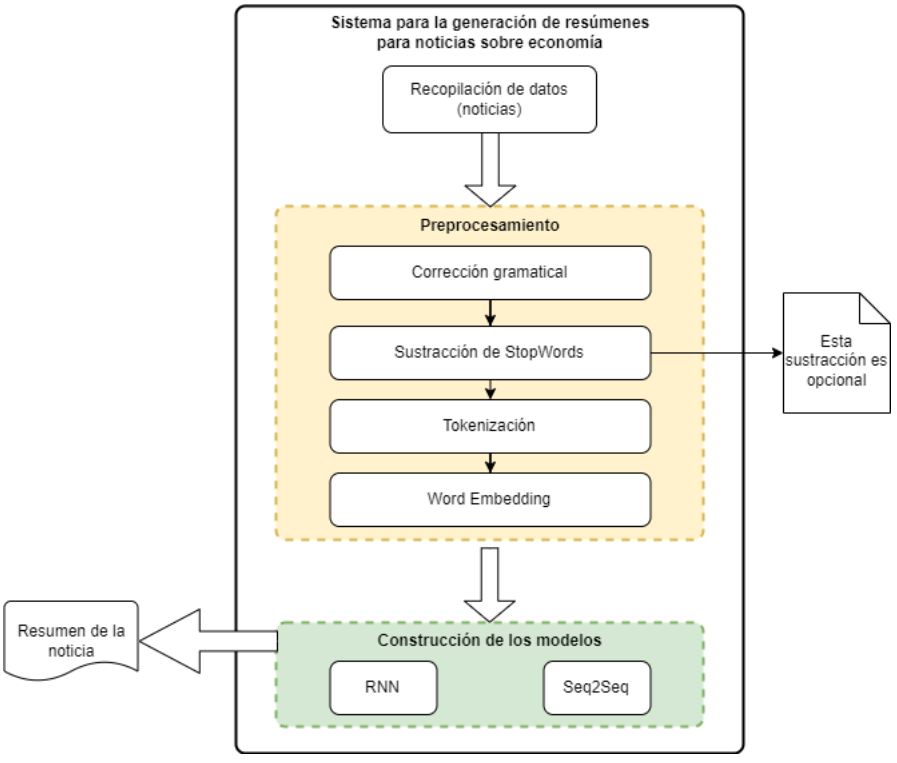

Figura 4.1: Diagrama de flujo para el sistema generador de resúmenes para noticias económicas.

En este diagrama, se pueden observar los dos componentes principales para el desarrollo del sistema: el preprocesamiento y la construcción de los modelos para las dos técnicas. En primer lugar, se recopilaron los datos, específicamente noticias económicas. A partir de estos datos, se desarrolló el componente de preprocesamiento, con cada uno de los módulos que lo componen. Finalmente, se presentó el componente de construcción de modelos para generar el resumen de la noticia.

4.2. Datos y preprocesamiento

Se recopilaron noticias económicas y se desarrollaron cada una de las operaciones realizadas dentro del componente de preprocesamiento, observado en la Figura 4.1.

4.2.1. Recolección de datos

Durante esta fase, se recolectaron 12.031 noticias económicas de diversos portales de noticias y sitios web. Para evitar dificultades gramaticales relacionadas con el idioma, se seleccionaron noticias económicas en inglés. A continuación, se presentan las distintas fuentes de las cuales se obtuvieron los conjuntos de noticias:

- BBC News [44] (510 noticias)
- US Economic News [45] (7326 noticias)
- The New York Times [46] (4195 noticias)

En algunas de estas fuentes existen varias categorías que tratan diferentes temas como ciencia, salud, economía, música, finanzas, educación, entre otros. Por ello, para formar el conjunto de datos a utilizar, se escogieron categorías asociadas con el área de economía.

4.2.2. Preprocesamiento

Se realizan las operaciones pertinentes a la etapa de preprocesamiento, que se muestran en la Figura 4.2.

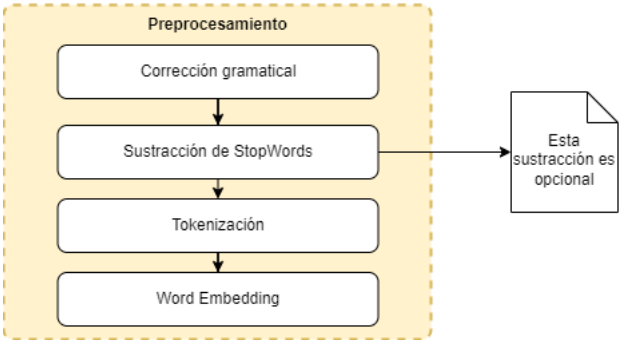

Figura 4.2: Diagrama de flujo para el componente de preprocesamiento.

Las operaciones presentes en la Figura 4.2. se describen a continuación:

- Corrección gramatical: corrige los errores gramaticales que se puedan presentar en las noticias económicas, por ejemplo, poner un espacio después de un punto seguido o el cierre de comillas.
- Sustracción de StopWords: se eliminan palabras vacías que son comunes en el idioma inglés (pronombres, adjetivos demostrativos, entre otros) de las noticias económicas. Este paso puede ser opcional, pues en algunos casos mantener las StopWords puede ser beneficioso para preservar el contexto completo de las oraciones.
- Tokenización: el conjunto de datos fue tokenizado por oraciones, utilizando los puntos (.) como separador entre las frases de las noticias económicas.
- Word Embedding: se transforman las oraciones de cada noticia en vectores que representan su equivalente numérico (ver sección 2.3.3)

4.2.3. Resúmenes de referencia

Tanto para entrenar como para medir el desempeño de los modelos de ambas técnicas, se necesita un conjunto de resúmenes previamente creados para compararlos con los resúmenes generados por cada técnica. Para este propósito, se seleccionaron los conjuntos de resúmenes incluidos en algunas de las fuentes mencionadas en la Sección 4.2.1.
Para aquellas fuentes que no contenían resúmenes se utilizó BERT (*Bidirectional Encoder Representation from Transformers*), un modelo de lenguaje

desarrollado por Google, basado en redes neuronales.

BERT está preentrenado en 3300M de palabras y se utiliza por su capacidad para comprender de manera detallada la semántica y las relaciones en la información. Además, ha demostrado superar significativamente a muchos otros modelos de lenguaje en tareas de PLN [47].

Por lo anterior, se decidió emplear BERT en su versión 0.10.1 para generar los resúmenes de los conjuntos de noticias que no los incluían.

4.3. Diseño de los modelos

Se explica el proceso de creación de las arquitecturas de los modelos correspondientes a las dos técnicas seleccionadas en el Capítulo 3.4. Se describe el diseño de la red neuronal recurrente (RNN) y la arquitectura Codificador-Decodificador, junto con las arquitecturas propuestas para cada una.

4.3.1. Diseño de la red neuronal recurrente

El diseño de esta red se basó en el uso de unidades LSTM (Long Short Term Memory). Se propusieron tres tipos de arquitecturas, ajustadas en función de los resultados obtenidos en los experimentos que se desarrollaron para cada una (Ver Sección 4.5). Estas arquitecturas incluyen diversas capas, como LSTM, densas, entre otras, que se describirán en detalle a lo largo de esta sección.

Las capas LSTM están formadas por cuatro redes neuronales completamente conectadas. Cada una cuenta con un número de unidades que representa una cantidad de neuronas y corresponde a la dimensión del vector de salida. Estas redes se denominan compuerta de entrada (*input gate*), compuerta de olvido (*forget gate*), compuerta de salida (*output gate*), y celda de memoria (*memory cell internal state*).

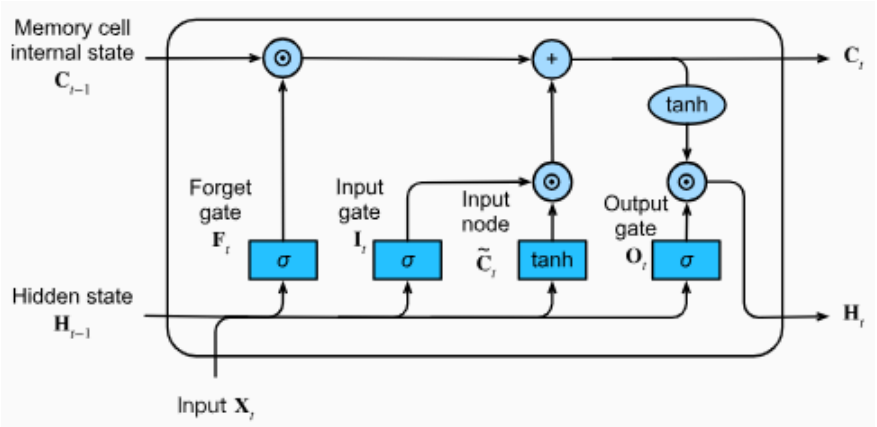

Figura 4.3: Arquitectura de una LSTM [45].

Como se puede observar en la figura anterior, una capa LSTM recibe tres vectores como entrada. Dos de estos vectores son generados por la misma capa LSTM en el instante anterior $t-1$: C (celda de memoria) y H (estado oculto). El tercer vector es la entrada X en el instante t, que en este caso es una oración de una noticia económica representada en un vector numérico.

Dado los vectores de entrada anteriores, la LSTM regula, a través de las compuertas y sus funciones de activación Sigmoid o Tanh, el flujo de información y transforma los valores de los vectores de la celda de memoria y estado oculto, los cuales formarán parte de la entrada en la secuencia del instante $t+1$. Este control de flujo se realiza de modo que la celda de memoria C funcione como una memoria a largo plazo, mientras que el estado oculto H actúe como una memoria a corto plazo, permitiendo que las compuertas decidan si almacenar o eliminar información relevante del vector X.

Arquitectura 1

La primera arquitectura propuesta se presenta en la Figura 4.4. Consiste en una red LSTM, sobre la cual se añade una capa densa, que al final se conecta con otra capa densa de salida que utiliza una función de activación SoftMax. Esta función proporciona una distribución de probabilidades indicando la pertenencia de un elemento de entrada a una categoría u otra. En este caso, la tarea es clasificar la oración de una noticia económica como parte del resumen o no.

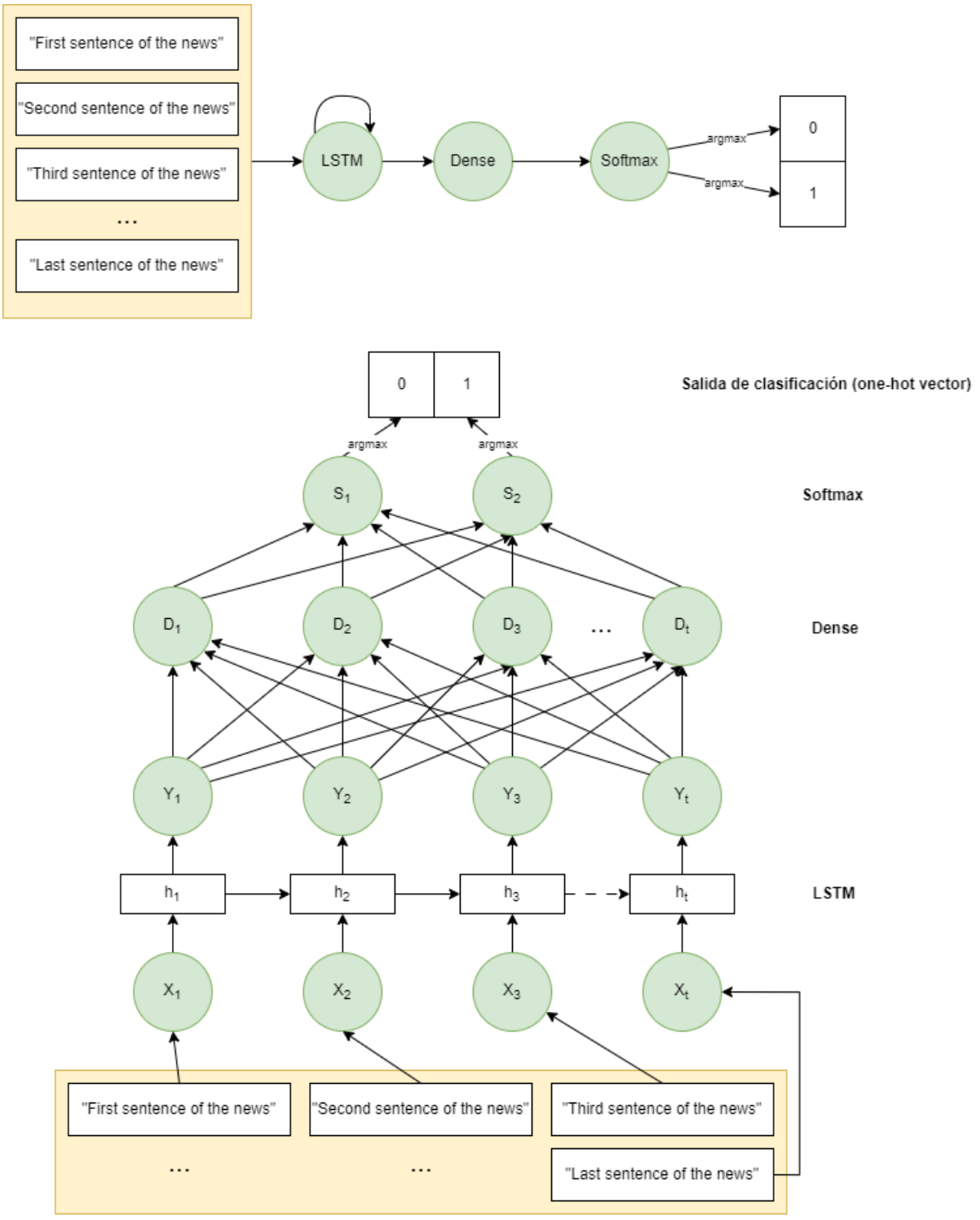

Figura 4.4: Modelo propuesto para la primera arquitectura.

Cada oración de la noticia, representada como una matriz de embeddings de tamaño $n \times d$, donde $n = 200$ corresponde al número de palabras en la oración y $d = 200$ indica la dimensión del vector de embedding, se procesa a través de las unidades de la capa oculta de la LSTM. En esta capa, mediante las compuertas (Ver Sección 2.2), se decide qué información de la oración se almacena, actualiza o elimina. Posteriormente, se obtiene un estado h_t que contiene información relevante y que será considerada en el siguiente instante de tiempo $t + 1$ para procesar la siguiente oración de la noticia.

Una vez se han procesado todas las oraciones de la noticia económica, el vector resultante se pasa como entrada a una red totalmente conectada, conocida como capa densa. Esta capa permite que el modelo aprenda relaciones más complejas entre las características extraídas por la LSTM. Finalmente, el vector resultante se pasa a la capa densa de salida, que transforma la dimensión del vector en un formato adecuado para la clasificación de la oración como parte del resumen o no. El resultado final es un vector de dos posiciones que indica la probabilidad de que una oración de la noticia sea parte del resumen o no.

Arquitectura 1.1

Esta arquitectura es una modificación de la Arquitectura 1, dado que contiene las mismas capas, pero con la adición de capas Dropout para prevenir el sobreajuste (*overfitting*). El sobreajuste suele ocurrir cuando las neuronas de diferentes capas se influyen mutuamente, creando dependencias que hacen que el modelo se ajuste demasiado bien a los datos de entrenamiento, pero no se desempeñe adecuadamente con datos nuevos.

Las capas Dropout se insertan generalmente después de una capa completamente conectada, pues al tener muchos parámetros ocasiona una mayor probabilidad de dependencia entre los nodos. En este caso, se insertaron después de la capa LSTM y la capa densa.

El funcionamiento de las capas Dropout se basa en establecer aleatoriamente algunas unidades de entrada en 0 durante el entrenamiento, eliminándolas temporalmente de la red (Ver Figura 4.5). La cantidad de nodos que se

establecen en 0 depende de la probabilidad especificada en la capa Dropout. Esto obliga a las neuronas a no depender de los nodos de capas anteriores, puesto que no pueden asumir que estarán presentes en cada ejecución del entrenamiento. En las ejecuciones siguientes, la arquitectura de la red se modifica ligeramente debido a la capa Dropout, permitiendo que la red aprenda a producir mejores predicciones.

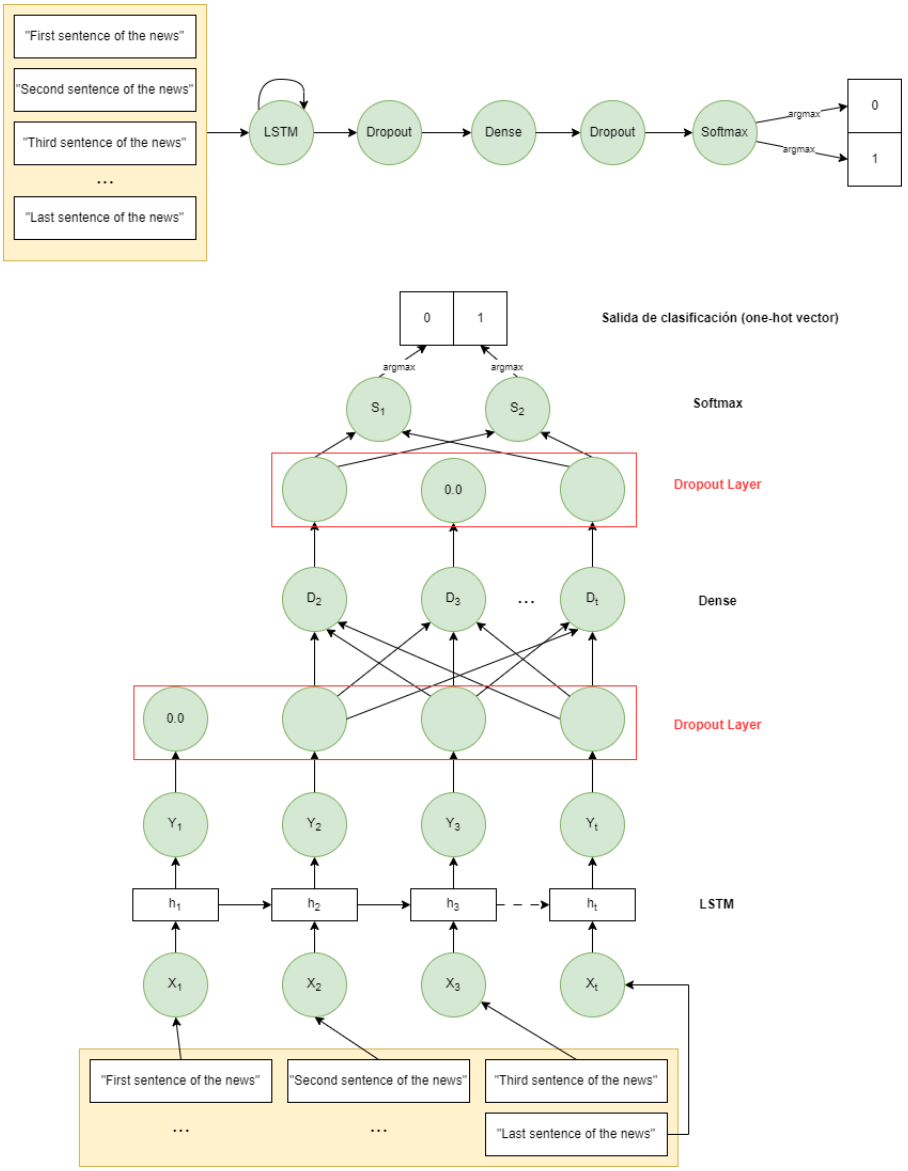

Figura 4.5: Diagrama propuesto para la variación de la primera arquitectura, donde se agrega la capa Dropout.

Arquitectura 2

La segunda arquitectura que se propuso se ilustra en la Figura 4.6. Esta arquitectura varía en el formato de entrada a la red, y consta de dos capas LSTM, donde la última está conectada a una capa densa, que finalmente se conecta con una capa densa de salida.

En esta arquitectura, cada oración de la noticia se representa de manera diferente. Se decidió agrupar las oraciones en fragmentos de siete, de forma que cada fragmento que se pasa como entrada al modelo contiene una oración central y sus tres oraciones vecinas anteriores y tres posteriores. Esto resulta en una matriz de embeddings con la forma $7, 200, 200$. En esta matriz, cada entrada es un grupo de siete oraciones con embeddings dimensionales de 200×200. Esta forma de representación permite a la capa oculta de la LSTM obtener más contexto de la oración central mediante la información de sus oraciones vecinas.

Posteriormente, se utiliza la capa TimeDistributed con el propósito de aplicar la primera capa LSTM a cada una de las 7 secuencias de tamaño $200, 200$ de forma independiente. Esto permite que el modelo capture características locales de cada secuencia sin mezclar información entre ellas. Luego, la segunda capa LSTM procesa la secuencia completa de los 7 vectores resultantes de la primera capa LSTM con TimeDistributed. Tanto en la primera como en la segunda capa LSTM, se decide qué información almacenar, actualizar o eliminar a través de las compuertas de entrada, olvido y salida.

Una vez procesadas todas las oraciones en las capas LSTM, el vector resultante de la segunda capa LSTM se pasa como entrada a una capa densa, la cual facilita que el modelo aprenda relaciones complejas entre las características extraídas por la LSTM. Por último, este vector se pasa a la capa densa de salida, que clasifica la oración de la noticia económica como parte del resumen o no.

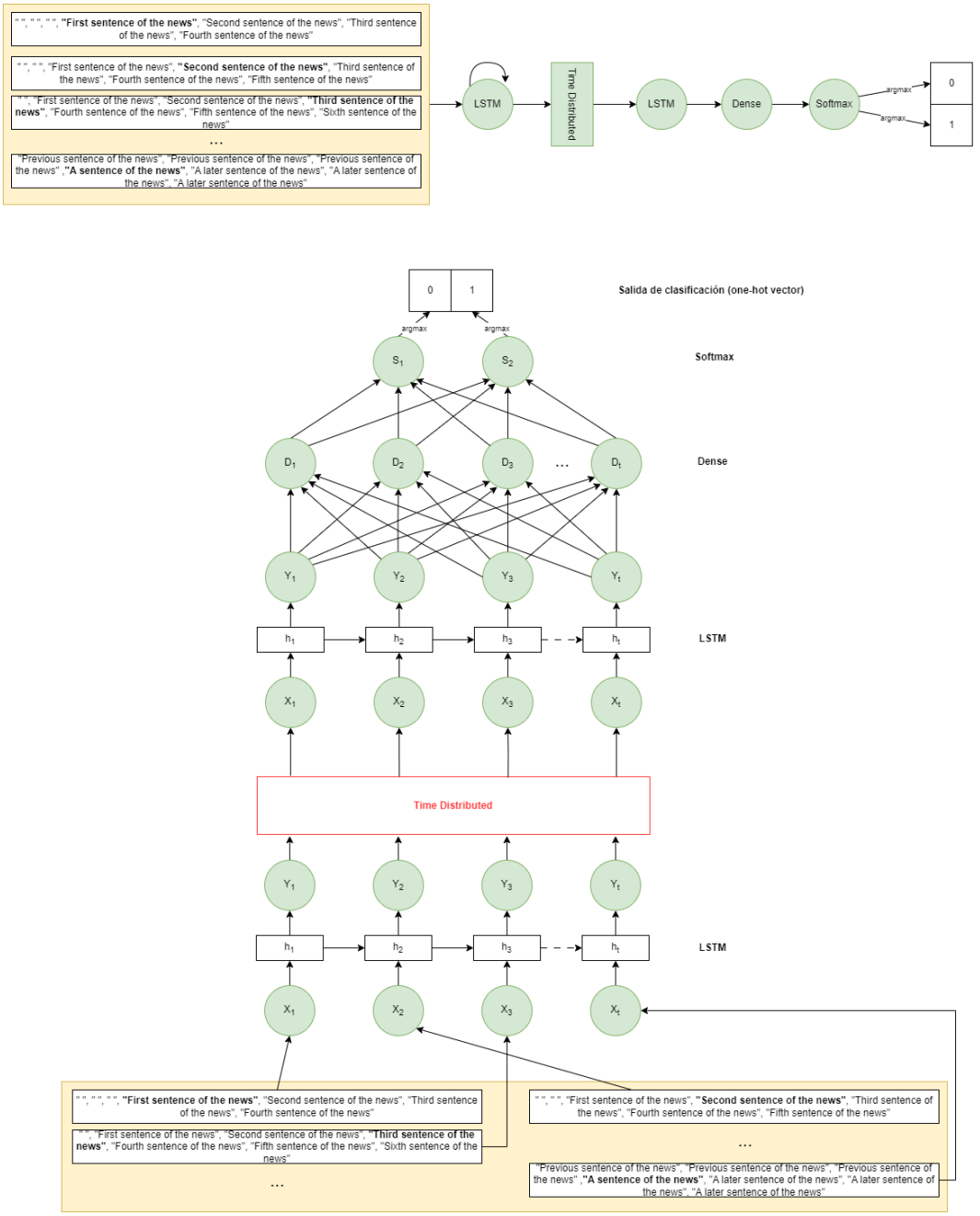

Figura 4.6: Diagrama propuesto para la segunda arquitectura.

Arquitectura 2.1

Al igual que en la Arquitectura 1.1, en la Arquitectura 2 se implementaron variaciones para prevenir el sobreajuste. Para lograr este objetivo, se añadieron capas de Dropout después de las capas LSTM y de la capa densa. Esta técnica de regularización ayuda a evitar que las neuronas dependan de los nodos de las capas anteriores, mejorando la capacidad de la red para generalizar a datos no vistos y, en consecuencia, aumentando su rendimiento en el conjunto de prueba. Esta arquitectura se presenta en la Figura 4.7.

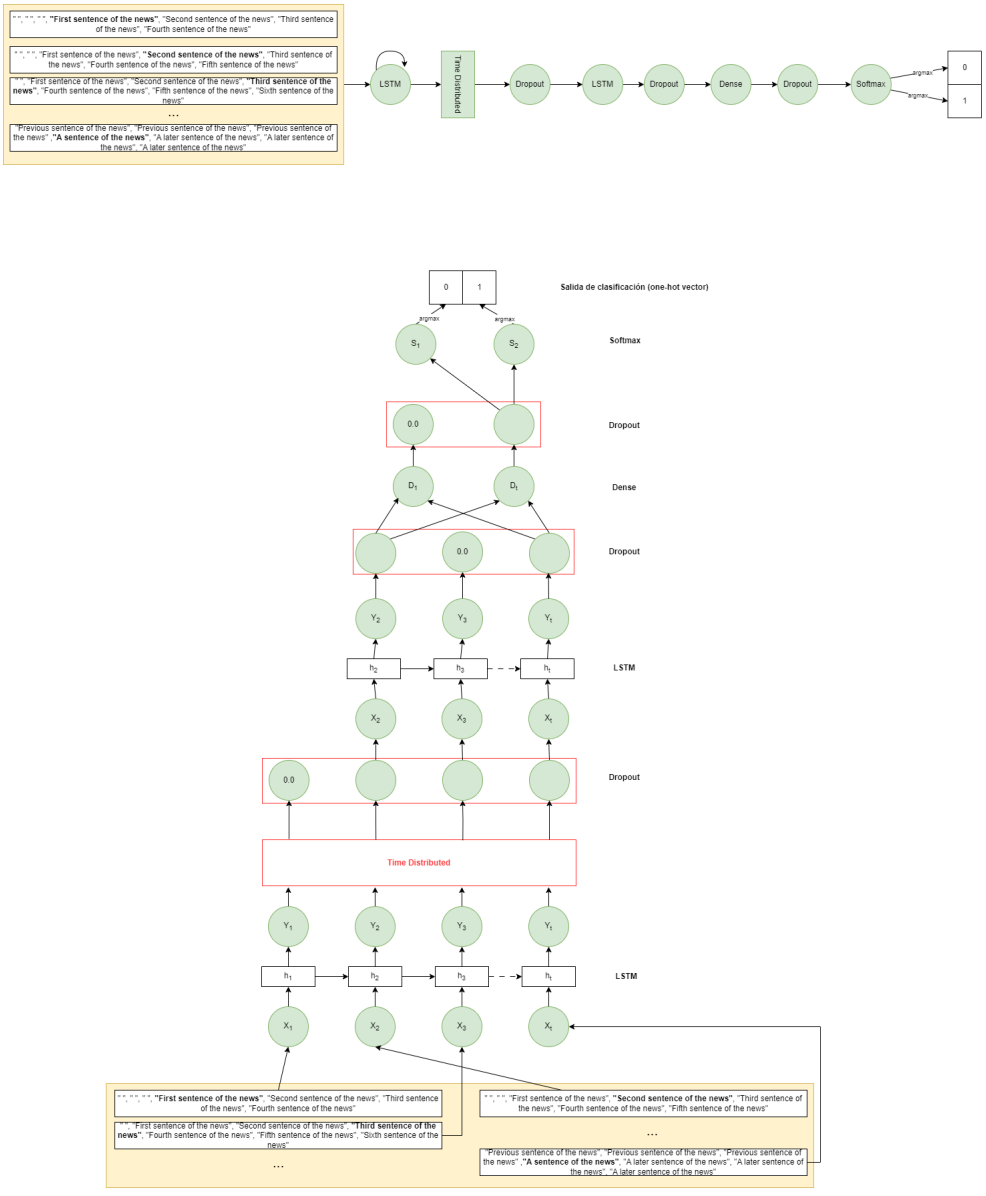

Figura 4.7: Diagrama para la variación de la segunda arquitectura.

Generación del resumen extractivo

Primero se preprocesa la noticia económica y se transforma en una matriz de embeddings. Esta matriz se pasa al modelo de la arquitectura LSTM, que produce como salida un vector de dos posiciones, 0s y 1s. A partir de este vector resultante, se seleccionan las oraciones importantes que están representadas como 1 para formar el resumen extractivo.

4.3.2. Diseño de la arquitectura Codificador - Decodificador

Para generar resúmenes extractivos por medio de esta arquitectura, primero se crearon resúmenes abstractivos a través de un marco codificador-decodificador. Utilizando la distancia euclidiana entre el resumen abstractivo generado y la noticia económica, se producen los resúmenes extractivos. Se propusieron dos tipos de arquitecturas, cada una basada en un tipo de red neuronal diferente, como RNN, GRU, LSTM, y CNN. Estas se explicarán a lo largo de la sección.

La arquitectura codificador-decodificador consta de dos componentes principales: un codificador y un decodificador.

El codificador toma una secuencia de entrada, que en este caso es una oración de la noticia original, y produce una representación vectorial de longitud fija que encapsula información relevante de los elementos de entrada, denominada vector de contexto u oculto. Luego, el decodificador toma el vector de contexto y genera una secuencia de salida basada en este.

Los datos de entrada $x_1, x_2, x_3, ..., x_n$ se pasan a una red de codificador que puede tener varias unidades recurrentes como LSTM o GRU, cada una de las cuales acepta un único elemento de la secuencia de entrada, recopila información para ese elemento y la propaga hacia adelante. Esta produce los estados ocultos h_i a partir de la siguiente ecuación:

$$h_t = g(W^{(hh)} h_{t-1} + W^{(hx)} x_t) \tag{4.1}$$

Donde g puede ser una función de activación o una unidad recurrente, y

se aplican los pesos correspondientes al estado oculto previo h_{t-1}, y al vector de entrada x_t. Este estado oculto se usa como estado oculto inicial en la parte decodificadora del modelo.

La parte decodificadora del modelo también puede tener varias unidades recurrentes. Cada una de estas unidades acepta un estado oculto de la unidad anterior y produce una salida y_t en un paso de tiempo t, la cual se ve de la siguiente forma:

$$y_t = softmax(W^S h_t) \tag{4.2}$$

La salida y_t se calcula a partir del estado oculto en el paso de tiempo actual (h_t) junto al peso respectivo, a través de una función softmax para crear un vector de probabilidad que, en el caso de este proyecto, ayudará a determinar si una palabra de la noticia se incluye o no en el resumen abstractivo.

Arquitectura 1

Esta arquitectura se basó en el trabajo propuesto en [48]. Se considera un ejemplo de arquitectura utilizada para la creación de soluciones basadas en machine learning mediante la técnica Encoder-Decoder. Aunque esta arquitectura no se aplica exactamente para generar resúmenes de texto, se propuso hacer ajustes en su diseño para este fin.

Primero, se prepararon los datos realizando un preprocesamiento similar al explicado en la Sección 4.2. Adicionalmente, se utilizó la librería ktext para construir un vocabulario representado como un mapa de token->entero, conservando solo los 50,000 tokens principales del vocabulario. También se realizó padding para que todas las noticias tengan la misma longitud.

Aparte del preprocesamiento de los datos fuente (noticias) y los datos de destino (resúmenes), se agregaron indicadores especiales como <start>al inicio y <end>al final de cada noticia, al igual que el padding. Esto permite que el modelo sequence-to-sequence sepa cuándo debe empezar y terminar una secuencia.

Modelo Sequence-to-Sequence

Este modelo toma una secuencia de entrada (documentos fuente), y predice una secuencia de salida (documentos de destino). Consta de dos partes principales:

- Codificador: procesa la secuencia de entrada y la convierte en un estado oculto. Los vectores de los documentos de datos fuentes, es decir, las noticias económicas, se utilizan como entrada para este modelo.
- Decodificador: toma el estado oculto del codificador y genera la secuencia de salida. Los vectores de los documentos de destino, es decir, los resúmenes de las noticias económicas, se utilizan como entrada para este modelo y también como objetivo del decodificador.

Modelo Codificador

La función del codificador es extraer características y generar una representación de la secuencia de entrada, que en este caso es una noticia económica. Esta arquitectura consiste en una capa de Embedding que se aplica a las secuencias de palabras de entrada (representadas como índices numéricos) para convertirlas en vectores densos en el espacio vectorial definido por la dimensión del vector de embedding. Los datos de salida de la capa de Embedding se normalizan por lotes, ajustando y normalizando las activaciones antes de pasarlas a la siguiente capa. Posteriormente, se define una capa GRU, que cuenta con una estructura más simple en comparación con la LSTM.

Las capas GRU están formadas por una compuerta de actualización, una compuerta de reinicio y el contenido de la memoria actual (Ver Sección 2.2.4)

La compuerta de actualización z_t se calcula mediante una función de activación Sigmoidal que toma como entrada el estado oculto anterior h_{t-1} y la entrada actual x_t. Esta compuerta determina cuánta información pasada debe transmitirse al futuro. La compuerta de reinicio r_t se calcula de forma similar, pero decide cuánta información pasada se debe olvidar. El contenido de la memoria se equilibra en cada paso de tiempo para evitar saturación o perdida de datos.

Finalmente, el cálculo del estado final de la memoria h_t involucra la combinación de información del estado oculto anterior y el nuevo estado oculto calculado, controlado por las compuertas de reinicio y actualización. Este estado oculto final captura la representación de toda la secuencia de entrada y es la salida del codificador, utilizada como contexto inicial para el decodificador.

Modelo Decodificador

La función del decodificador es generar un resumen abstractivo de la noticia económica, condicionado a las características extraídas por el codificador. Esta arquitectura consta de una capa de Embedding que, al igual que en el codificador, transforma los índices de las secuencias de entrada (los resúmenes originales) en vectores densos del tamaño del vector de embedding. Los datos de salida de la capa de Embedding también se normalizan por lotes. Después, se aplica una capa GRU a la entrada normalizada, utilizando el estado oculto final del codificador como estado inicial. Esto conecta el codificador y el decodificador, permitiendo que la información aprendida durante la codificación influya en la decodificación.

La salida de la GRU correspondiente a una secuencia de salidas para cada paso de tiempo se pasa a una capa densa que captura patrones complejos en los datos. Luego, se aplica una capa Dropout a la salida de la capa densa para prevenir el sobreajuste durante el entrenamiento. La salida del Dropout se normaliza nuevamente. Finalmente, la salida normalizada se pasa a través de una capa densa con activación SoftMax para obtener las probabilidades sobre el vocabulario de salida y formar el resumen abstractivo.

Una representación del modelo Sequence-to-Sequence se muestra en la Figura 4.8 a continuación, esta contiene cada una de las capas que se nombraron anteriormente.

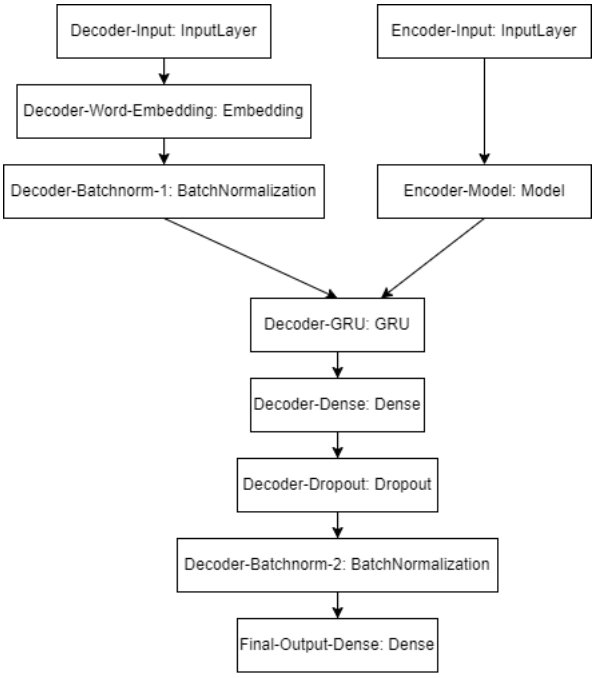

Figura 4.8: Arquitectura Seq2Seq planteada.

Arquitectura 1.1

Esta arquitectura fue una modificación del modelo decodificador de la Arquitectura 1. Mantiene las mismas capas, pero incorporó técnicas adicionales para prevenir el sobreajuste, como Dropout, L1, y L2. El sobreajuste ocurre cuando un modelo se ajusta demasiado bien a las particularidades del conjunto de datos de entrenamiento, en lugar de generalizar a datos nuevos.

Se añadió una capa Dropout después de la capa de Embedding, con el propósito de regularizar las representaciones de las palabras y prevenir el sobreajuste. También se añadieron regularizaciones L1 y L2 en la capa GRU.

La regularización L1 reduce las ponderaciones asociadas a ciertos parámetros, eliminando del modelo las características poco frecuentes o irrelevantes [49]. Por otro lado, la regularización L2 penaliza los pesos grandes, favoreciendo la obtención de pesos pequeños. Ambas técnicas ayudan a mantener un equilibrio entre la complejidad y el rendimiento del modelo, mejorando la precisión y fiabilidad de.

La arquitectura levemente modificada se ilustra en la Figura 4.9.

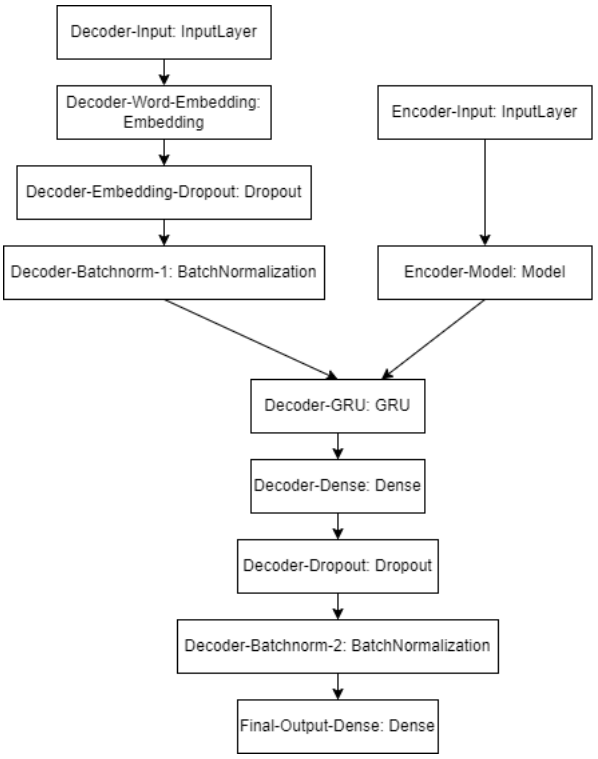

Figura 4.9: Arquitectura 1.1 Seq2Seq planteada, con modificación en decodificador.

Generación del resumen extractivo a partir del resumen abstractivo

Durante el entrenamiento del modelo, el decodificador recibe la secuencia correcta como entrada. Sin embargo, para la generación de resúmenes abstractivos, el modelo debe usar su propia predicción anterior como entrada para generar la siguiente palabra de la secuencia y así formar el resumen. Para lograr esto, se ensambla el modelo de manera que sus pesos entrenados permanezcan intactos y el decodificador utilice la última predicción como entrada.

El proceso que se utilizó para generar un resumen abstractivo consiste en preprocesar y tokenizar la noticia económica de entrada, codificarla utilizando el modelo codificador y luego iniciar el decodificador con el token de inicio que indica el comienzo de la secuencia de salida, la cual se genera palabra por palabra.

Posteriormente, tanto el resumen abstractivo generado como la noticia económica se van a codificar utilizando el modelo codificador previamente entrenado. Estas representaciones vectoriales se utilizan para calcular la distancia euclidiana entre ellas, cuantificando la similitud o diferencia entre las oraciones de la noticia económica y las del resumen abstractivo.

Finalmente, para formar el resumen extractivo, se seleccionan las oraciones más representativas (con las distancias más pequeñas) del resumen abstractivo y la noticia económica, basándose en las distancias euclidianas calculadas previamente.

4.4. Entrenamiento de los modelos

Para las dos arquitecturas propuestas, el entrenamiento de los modelos se basa en las siguientes cinco etapas:

- Lectura del conjunto de datos: se efectúa la lectura de datos cargando tanto la noticia económica original como su resumen correspondiente.
- Limpieza del corpus: se realizan todas las etapas para la limpieza, con el objetivo de preparar el conjunto de datos para su posterior transformación numérica.
- Distribución de datos: los datos se separan en conjunto de entrenamiento, conjunto de validación y conjunto de pruebas.
- Transformación de datos: se convierten los datos de entrada, es decir las noticias, a un formato numérico que pueda ser utilizado por las dos técnicas seleccionadas para la generación de los resúmenes.
- Generación del resumen: se produce el resumen de las noticias económicas a partir de las dos técnicas seleccionadas.

4.4.1. Lectura del conjunto de datos

El conjunto de datos empleado para entrenar los modelos está compuesto por 12.031 noticias económicas (ver sección 4.1.1), las cuales están organizadas en diferentes archivos según su procedencia de portales de noticias o

sitios web. Estas noticias se encuentran almacenadas en dos formatos distintos: archivos TXT y archivos JSON. La diversificación de formatos se debe a que la estructura planteada para cada técnica requiere un formato de entrada diferente.

Las dos técnicas previamente seleccionadas forman parte del aprendizaje supervisado, lo que implica que el modelo se entrena utilizando pares de entrada y salida. En este caso, los pares corresponden a la noticia económica y su resumen, de modo que, durante la lectura del conjunto de datos, se extraen tanto la noticia económica como su resumen asociado.

4.4.2. Limpieza del corpus

Algunas de las noticias extraídas de las fuentes contienen errores gramaticales, como la omisión de espacios después de los puntos seguidos (.) o errores en el uso de comillas. Estos problemas dificultan la etapa de tokenización, por lo que es fundamental corregirlos. Además, las noticias pueden contener caracteres basura o palabras que no contribuyen al entrenamiento del modelo, como las palabras vacías o StopWords. Por lo tanto, esta sección se dedica a limpiar el corpus eliminando aquellas palabras irrelevantes y corrigiendo errores a través de las siguientes etapas:

- Corrección gramatical: se revisan las noticias de los conjuntos de datos y se corrigen los errores gramaticales para evitar inconvenientes en la tokenización.
- Sustracción de StopWords: se eliminan aquellas palabras que no tienen relevancia dentro las noticias económicas.

4.4.3. Distribución de datos

El conjunto de datos utilizado contiene un total de 12.031 noticias económicas. Estas se dividen en tres grupos: datos de entrenamiento, datos de validación, y datos de prueba, cada uno cumpliendo una función específica en el proceso de entrenamiento y evaluación de los modelos.

- Datos de entrenamiento: este conjunto se emplea para entrenar el modelo, permitiéndole aprender de los datos disponibles. Está compuesto por 75 % noticias económicas.

- Datos de validación: estos datos proporcionan una evaluación imparcial del rendimiento del modelo durante el ajuste de hiperparámetros (las configuraciones que se establecen en la arquitectura del modelo antes de empezar el entrenamiento). Para este conjunto se seleccionaron 25 % noticias económicas.

- Datos de prueba: una vez entrenado el modelo, se utilizan los datos de prueba para evaluar su desempeño final. En este caso, se utilizarán 500 noticias económicas.

4.4.4. Transformación de datos

El propósito principal de este proceso es convertir las oraciones de noticias económicas en una representación numérica equivalente. Esto se logra mediante las etapas de Tokenización y Word Embedding (Ver Sección 2.3.).

En primer lugar, las oraciones de las noticias se convierten en secuencias de índices de palabras. Luego, estas secuencias se transforman en vectores de embeddings correspondientes, utilizando vectores preentrenados en grandes conjuntos de datos como GloVe. Estos vectores preentrenados capturan relaciones semánticas entre palabras, lo que permite aprovechar este conocimiento previo en lugar de aprender las representaciones de palabras desde cero.

La matriz de embeddings para una oración de una noticia se representa como nxd, donde n indica el número de palabras en la oración y d es el tamaño del vector de embedding, es decir, cuántos valores numéricos conforman cada vector que representa una palabra. Es importante destacar que se calculó la longitud de la oración más larga de todo el conjunto de datos para determinar el número de palabras a utilizar por oración, con el propósito de formar matrices de dimensiones similares. Dado que existen noticias con diferentes longitudes de oración, se completa con ceros para alcanzar el número total de palabras propuesto para cada oración.

Esta representación mediante matrices de embeddings es fundamental, pues permite capturar significados semánticos y relaciones entre las palabras

de manera numérica y eficiente. Esto facilita el procesamiento posterior para la generación del resumen de las noticias.

Generación del resumen

Se realiza la generación de resúmenes utilizando los modelos de las técnicas seleccionadas: una red neuronal recurrente y una arquitectura codificador-decodificador, también conocida como sequence-to-sequence. Estos modelos reciben las noticias económicas como entrada y generan un resumen basado en la relevancia de las oraciones de la noticia económica.

4.5. Diseño de experimentos

Se describe el diseño de los experimentos propuesto para las arquitecturas de las dos técnicas seleccionadas. El objetivo de esta experimentación es determinar la configuración óptima de parámetros para ambos modelos, mediante el entrenamiento con diferentes configuraciones de parámetros.

4.5.1. Diseño de experimentos RNN

En seguida, se presenta el diseño experimental para determinar la mejor arquitectura de la primera técnica, la red neuronal recurrente LSTM. Los parámetros y rangos utilizados en estos experimentos se detallan en la Tabla 4.1.

Cuadro 4.1: Parámetros y rangos de las configuraciones LSTM

Parámetro	Rango	Intervalo
Tipo de limpieza	Con/Sin StopWords	N/A
Dropout	0.25-0.75	0.25
Número de capas Dropout	0, 2, 3, 4	N/A
Función de activación	ReLu/sigmoid/tanh	N/A
Número de unidades en capa densa	32-256	(*2)
Número de capas densas	1, 2, 3	N/A
Número de unidades en LSTM	32-256	(*2)
Número de épocas	100	N/A

El objetivo de estos experimentos es seleccionar la mejor arquitectura y configuración de parámetros para la red neuronal recurrente LSTM. Este proceso se llevará a cabo para cada una de las arquitecturas mencionadas en la sección 4.5. El criterio de selección de los mejores resultados se basará en el monitoreo del *Accuracy* y el *Validation Loss*.

El Accuracy proporciona una medida clara de qué tan bien el modelo realiza sus predicciones. El Validation Loss, por otro lado, indica la capacidad del modelo para generalizar datos no vistos durante el entrenamiento. Es importante observarlo para detectar sobreajuste y ajustar los hiperparámetros del modelo. Un Validation Loss bajo sugiere una mejor capacidad de generalización del modelo. Idealmente, un Validation Loss bajo junto con un Accuracy alto señala que el modelo está funcionando y generalizando bien.

Para llevar a cabo la experimentación, es fundamental conocer el total de combinaciones posibles a partir de los parámetros listados en la Tabla 4.1. No se consideró el parámetro del tipo de limpieza, pues los experimentos se realizaron tanto con un conjunto de datos sin StopWords como con StopWords. Además, no se tuvo en cuenta el número de épocas, puesto que se utilizó *EarlyStopping* para detener el entrenamiento cuando el desempeño en el conjunto de validación dejaba de mejorar o empeoraba tras un número determinado de épocas. Esto evita el sobreajuste del modelo y mejora su capacidad de generalización, permitiendo seleccionar el modelo con el mejor desempeño en el conjunto de validación.

En total, se pueden generar 1728 combinaciones con los parámetros disponibles. Sin embargo, debido al elevado costo computacional en términos de recursos y tiempo, se seleccionó el 5 % de los experimentos (aproximadamente 80 combinaciones).

Para más detalles sobre el tiempo por experimento, diríjase a la sección de Resultados y Análisis obtenidos en 5.

Finalmente, se seleccionan las 4 mejores configuraciones de los experimentos realizados para las arquitecturas 1 y 2, y sus respectivas variaciones. Estas configuraciones finales para cada arquitectura serán analizadas con mayor detenimiento utilizando métricas adicionales en la Sección 5.2.

4.5.2. Diseño de experimentos arquitectura Codificador - Decodificador

A continuación, se presenta el diseño de experimentos para seleccionar la arquitectura más apropiada de la segunda técnica, la arquitectura codificador-decodificador. Los detalles de este diseño se muestran en la tabla 4.2.

Estos experimentos se realizan con el objetivo de seleccionar la mejor arquitectura y configuración de parámetros para la arquitectura codificador-decodificador. Este proceso se llevará a cabo para cada una de las arquitecturas mencionadas en la sección 4.5. El criterio de selección de los mejores resultados se basará en el monitoreo del *Accuracy* y el *Validation Loss*.

Cuadro 4.2: Parámetros y rangos de las configuraciones de la arquitectura Codificador - Decodificador

Parámetro	Rango	Intervalo
Dropout decodificador	0.25-0.75	0.25
Dropout codificador	0.25-0.75	0.25
Número de capas Dropout	1, 2	N/A
Número de unidades en capa densa	200, 300	N/A
Dimensión Word Embedding codificador	256, 512, 1024	N/A
Dimensión Word Embedding decodificador	256, 512, 1024	N/A
Learning Rate	0.0001, 0.001, 0.01	N/A
Número de épocas	100	N/A

Para continuar con la experimentación, es importante conocer el número de todas las combinaciones posibles a partir de los parámetros listados en la Tabla 4.2. No se tuvo en cuenta el número de épocas, puesto que se utilizó *EarlyStopping* para detener el entrenamiento cuando el desempeño en el conjunto de validación dejaba de mejorar o empeorar tras un número determinado de épocas.

En total, se pueden generar 324 combinaciones con los parámetros disponibles. Sin embargo, debido al elevado costo computacional en términos de recursos y tiempo, se seleccionó el 5 % de los experimentos (aproximadamente 20 combinaciones).

Para más detalles sobre el tiempo por experimento, diríjase a la sección de Resultados obtenidos en 5.2.

Finalmente, se seleccionan las 2 mejores configuraciones de los experimentos realizados para la arquitectura 1 y sus variaciones. Estas configuraciones finales para cada arquitectura serán analizadas con mayor detenimiento utilizando métricas adicionales en el Capítulo 5.2.5.

Capítulo 5

Resultados y Análisis

5.1. Configuración experimentación

5.1.1. Especificaciones de Hardware

Las especificaciones del equipo utilizado para los entrenamientos de los modelos durante la experimentación se encuentran detalladas en la siguiente tabla.

Cuadro 5.1: Especificaciones de Hardware

Componente	Computador
Memoria RAM	16gb DDR4 a 3200Mhz
Procesador	16gb DDR4 a 3200Mhz
Tarjeta de Video/GPU	RTX3070 de 8gb
Disco Duro	Western Digital SSD Nvme 500gb
Sistema Operativo	Windows 11

5.2. Reporte de resultados de la experimentación

5.2.1. Red Neuronal Recurrente

De acuerdo con lo descrito en la Sección 4.5.1, se diseñaron dos arquitecturas principales, cada una con una variación en su modelo mediante la adición de capas de regularización. Las siguientes tablas presentan los mejores resultados obtenidos, considerando el Accuracy (precisión) y el Validation Loss (pérdida en el conjunto de validación).

Es importante recordar que el parámetro de número de épocas variaba según la arquitectura, pues se utilizó Early Stopping para detener el entrenamiento cuando la función de pérdida mejoraba. Es decir, se detenía cuando el conjunto de validación se estancaba o comenzaba a disminuir, evitando así que la red continuara el entrenamiento sin mejorar la generalización de datos, lo que podría llevar al sobreajuste.

Arquitectura 1

Sin StopWords

ID	Dropout	Capas dropout	Función de activación	Unidades en capa densa	Unidades en LSTM	Épocas	Accuracy	Validation Loss
1	N/A	N/A	ReLu	64	64	76	0.7927	0.5316
2	N/A	N/A	Sigmoid	64	64	49	0.7917	0.5339
3	**N/A**	**N/A**	**ReLu**	**64**	**128**	**49**	**0.7926**	**0.5305**
4	N/A	N/A	Tanh	64	64	48	0.7917	0.5399

Figura 5.1: Primera arquitectura para la RNN.

A pesar de tener configuraciones de hiperparámetros similares, las funciones de activación y el número de unidades en LSTM influyen en los resultados. Se observó que la función de activación ReLu parece ser efectiva en términos de precisión, puesto que los modelos 1 y 3 utilizan ReLu, y presentan la mayor precisión. Además, la función de activación Tanh no parece ser tan efectiva, dado que el modelo 4, posee la mayor pérdida de validación (0.5399). Por otro

lado, incrementar el número de unidades LSTM mejora ligeramente los resultados.

El hecho de que el modelo sea relativamente simple puede afectar los resultados. Por un lado, es más eficiente en términos de tiempo y recursos computacionales, y por el otro, puede tener una capacidad limitada para capturar relaciones complejas entre los datos, lo que lleva a un bajo rendimiento en datos nuevos.

Sin embargo, se destacó que el modelo 1 tiene la mayor exactitud con un valor de 0.7927, seguido del modelo 3. En contraste, el modelo 3 tiene la menor pérdida en el conjunto de validación con un valor de 0.5305, seguido por el modelo 1.

En términos generales, el modelo 3 es el mejor, puesto que tiene la menor pérdida de validación (0.5305) y una exactitud (0.7626) casi igual a la de modelo 1.

Se decidió realizar un análisis para el caso de los datos que no tienen StopWords con el modelo 3, pues representa la mejor arquitectura. A continuación, se compara el Accuracy y Validation Loss para cada uno:

Cuadro 5.2: Resultados sin StopWords y con StopWords

Métrica	Modelo 3 sin SW	Modelo 3 con SW
Accuracy	0.7939	0.7926
Validation Loss	0.5567	0.5305

Se observó que la eliminación de StopWords puede mejorar ligeramente la exactitud del modelo. No obstante, incluir StopWords parece ayudar al modelo a generalizar mejor, como se evidencia por la menor pérdida de validación.

Finalmente, de acuerdo con los resultados y análisis realizados en los modelos de la arquitectura 1, se seleccionó el **modelo 3**. El tiempo promedio

para llevar a cabo los experimentos de la Arquitectura 1 fue de 2 minutos por época. Es decir, que el modelo con mayor número de épocas (76) tuvo una duración aproximada de 152 minutos.

Arquitectura 1.1

Sin StopWords

ID	Dropout	Capas dropout	Función de activación	Unidades en capa densa	Unidades en LSTM	Épocas	Accuracy	Validation Loss
1	**0.25**	**2**	**Tanh**	**64**	**64**	**60**	**0.7932**	**0.5300**
2	0.25	2	Sigmoid	128	32	63	0.7628	0.5483
3	0.25	2	Sigmoid	64	32	85	0.7645	0.5445
4	0.25	2	ReLu	64	64	44	0.7662	0.5383

Figura 5.2: Variación de la primera arquitectura para la RNN.

En esta arquitectura, se incluyó la capa de dropout, lo que mejoró la capacidad de generalización del modelo, pues al apagar aleatoriamente una fracción de neuronas durante el entrenamiento, se forzó al modelo a aprender representaciones más robustas.

Esto se evidenció en el modelo 1, que presentó la mejor exactitud (0.7932) y menor pérdida de validación (0.5300), indicando una mejor generalización y precisión en comparación con los otros modelos. Cabe destacar que, en la Arquitectura 1, este modelo presentaba la mayor pérdida de validación, pero en esta nueva arquitectura posee la menor perdida de validación.

Se decidió realizar un análisis para el caso de los datos que no tienen StopWords utilizando el modelo 1, puesto que representaba la mejor arquitectura. A continuación, se compara la exactitud y la pérdida de validación para cada caso:

Se observó un comportamiento similar al de la Arquitectura 1. La eliminación de StopWords mejoró ligeramente la exactitud del modelo, mientras que la inclusión de StopWords ayudó al modelo a generalizar mejor, como se evidenció por la menor pérdida de validación en el modelo 1.

Cuadro 5.3: Resultados sin StopWords y con StopWords

Métrica	Modelo 1 sin SW	Modelo 1 con SW
Accuracy	0.7941	0.7932
Validation Loss	0.5541	0.5300

Finalmente, de acuerdo con los resultados y análisis realizados en los modelos de la arquitectura 1.1, se selecciona el **modelo 1**. El tiempo promedio para llevar a cabo los experimentos de la Arquitectura 1.1 fue de 2 minutos por época. Es decir, que el modelo con mayor número de épocas (85) tuvo una duración de aproximadamente 170 minutos.

Arquitectura 2

Sin StopWords

ID	Dropout	Capas dropout	Función de activación	Unidades en capa densa	Unidades en LSTM	Épocas	Accuracy	Validation Loss
1	**N/A**	**N/A**	**ReLu**	**64**	**64**	**56**	**0.7640**	**0.5421**
2	N/A	N/A	Tanh	64	64	7	0.7522	0.6018

Figura 5.3: Segunda arquitectura para la RNN.

En esta arquitectura se utilizaron dos capas LSTM y se varió el formato de entrada a la red, agrupando las oraciones en fragmentos de siete. Esto repercutió de manera directamente proporcional en el costo computacional del entrenamiento, puesto que el tiempo promedio para llevar a cabo los experimentos fue de 10 minutos por época. Es decir, el modelo con mayor número de épocas (56) duró aproximadamente 560 minutos. Sin embargo, esto no implicó que un mayor número de capas y una entrada más compleja mejoraran significativamente el desempeño de los modelos. Como se observó en la Tabla 5.3, en comparación con la Arquitectura 1 que contenía una sola capa LSTM, la exactitud de los modelos disminuyó.

De igual forma, se destaca el modelo 1 como el mejor de esta arquitectura. Presentó la menor pérdida de validación (0.5421) y una mayor exactitud (0.7640).

Se decidió realizar un análisis para el caso de los datos que no tienen StopWords con el modelo 1, pues representaba la mejor arquitectura. A continuación, se compara el Accuracy y Validation Loss para cada uno:

Cuadro 5.4: Resultados sin StopWords y con StopWords

Métrica	Modelo 1 sin SW	Modelo 1 con SW
Accuracy	0.7945	0.7640
Validation Loss	0.5564	0.5421

La eliminación de StopWords mejoró significativamente la exactitud del modelo, esto puede deberse a que se concentra más en las palabras clave, resultando en una representación más precisa de los datos. No obstante, la inclusión de StopWords ayudó al modelo a generalizar mejor, como se evidenció por la menor pérdida de validación en el modelo 1. Finalmente, de acuerdo con los resultados y análisis realizados en los modelos de la arquitectura 2, se seleccionó el **modelo 1**.

Arquitectura 2.1

Sin StopWords

ID	Dropout	Capas dropout	Función de activación	Unidades en capa densa	Unidades en LSTM	Épocas	Accuracy	Validation Loss
1	0.25	2	Sigmoid	64	32	17	0.7845	0.5807
2	**0.25**	**2**	**Tanh**	**64**	**32**	**95**	**0.7851**	**0.5677**

Figura 5.4: Variación de la segunda arquitectura para la RNN.

Esta arquitectura se configuró con capas de dropout, lo que contribuyó a una ligera mejora en la exactitud del modelo. Sin embargo, este ajuste incrementó la complejidad de la arquitectura, afectando el desempeño general de

los modelos. Como se observa en la Tabla 5.4., la pérdida de validación en comparación con la Arquitectura 2 sigue siendo bastante alta.

De los modelos evaluados en esta arquitectura, se seleccionó el modelo 2 como el mejor. Tiene la menor pérdida de validación (0.5677) y una exactitud mayor (0.7851).

Se decidió realizar un análisis adicional utilizando el modelo 2 en el contexto de datos sin StopWords, dado que representa la mejor arquitectura. A continuación, se compara la exactitud y la pérdida de validación para ambos escenarios:

Cuadro 5.5: Resultados sin StopWords y con StopWords

Métrica	Modelo 2 sin SW	Modelo 2 con SW
Accuracy	0.7879	0.7851
Validation Loss	0.5824	0.5677

Al igual que en la Arquitectura 2, la eliminación de StopWords en estos experimentos mostró una mejora significativa en la exactitud del modelo. Sin embargo, la inclusión de StopWords permitió al modelo generalizar mejor, como lo evidencia la menor pérdida de validación en el modelo 2.

Finalmente, de acuerdo con los resultados y análisis de los modelos de la arquitectura 2.1, se seleccionó el **modelo 2**. El tiempo promedio requerido para llevar a cabo los experimentos de la Arquitectura 2.1 fue de 10 minutos por época. Por lo tanto, el modelo con mayor número de épocas (95) tuvo una duración aproximada de 950 minutos.

Los mejores modelos de cada una de las arquitecturas nombradas fueron analizados usando métricas adicionales, los resultados se muestran en la siguiente sección.

5.2.2. Comparación entre arquitecturas RNN

En esta sección se presenta el análisis de desempeño en la generación de resúmenes de noticias económicas de los mejores modelos seleccionados en la Sección 5.2.1. de la arquitectura RNN, con el objetivo de elegir el modelo más adecuado para representar la técnica RNN.

De acuerdo con los resultados obtenidos y presentados en la Tabla 5.10, el modelo de la arquitectura 1.1 mostró una mayor exactitud (0.7932) y una menor pérdida de validación (0.5300), por lo que se consideró el mejor en términos generales en relación con estas métricas.

Cuadro 5.6: Comparación entre modelos

Modelos	Accuracy	Validation Loss
Arquitectura 1	0.7926	0.5305
Arquitectura 1.1	0.7932	0.5300
Arquitectura 2	0.7640	0.5421
Arquitectura 2.1	0.7851	0.5677

El conjunto de datos utilizado para el entrenamiento y validación de las arquitecturas no presentó una distribución uniforme entre las oraciones de la noticia económica incluidas o no en el resumen. Por tal motivo, se realizó un análisis más detallado utilizando matrices de confusión para las cuatro arquitecturas.

Como se observa en las Figuras 5.5 a 5.8, los cuatro modelos demostraron un buen desempeño en la clasificación de las oraciones que no debían incluirse en el resumen. Sin embargo, no clasificaron adecuadamente aquellas oraciones que sí debían formar parte del resumen.

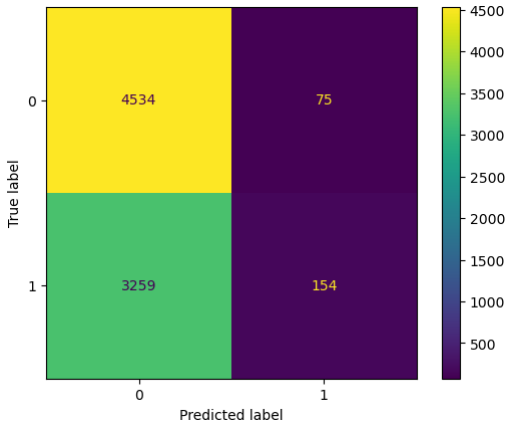

Figura 5.5: Matriz de confusión Arquitectura 1

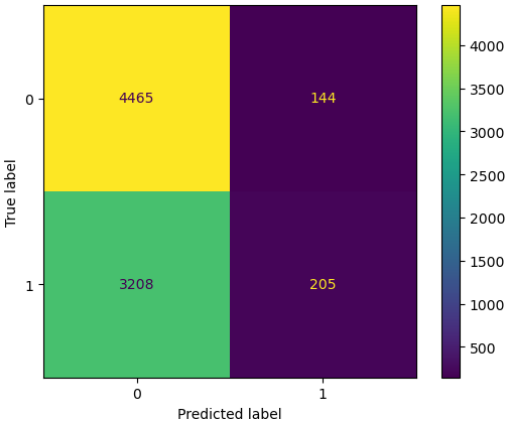

Figura 5.6: Matriz de confusión Arquitectura 1.1

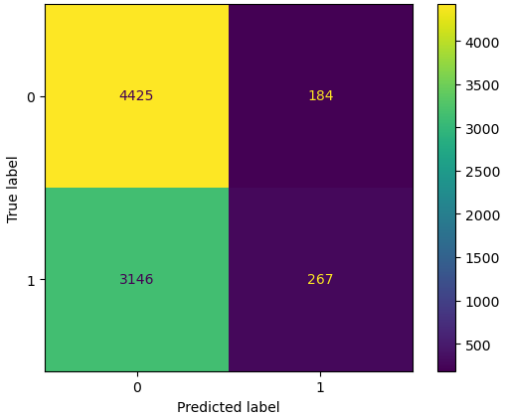

Figura 5.7: Matriz de confusión Arquitectura 2

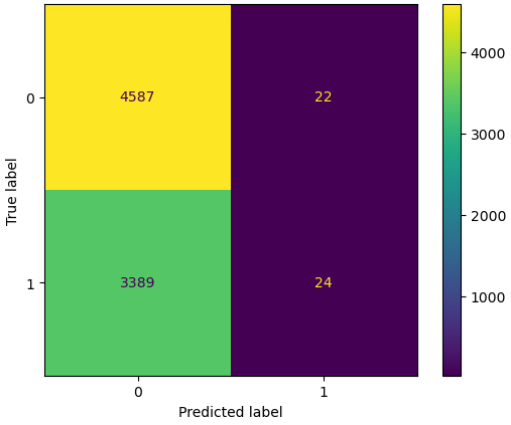

Figura 5.8: Matriz de confusión Arquitectura 2.1

Se utilizó ROUGE para analizar la calidad de los resúmenes generados en comparación con los resúmenes de referencia. En esta evaluación, se consideraron ROUGE-1, que mide la cantidad de n-gramas (palabras) comunes entre los resúmenes generados y los resúmenes de referencia, y ROUGE-L que se basa en la subsecuencia común más larga entre ambos resúmenes.

	ROUGE-1			**ROUGE-L**		
	F-score	Precision	Recall	F-score	Precision	Recall
Modelo arquitectura 1	0.0988	0.2951	0.0616	0.0936	0.2801	0.0583
Modelo arquitectura 1.1	0.1267	0.3309	0.0846	0.1160	0.3059	0.0771
Modelo arquitectura 2	**0.1779**	**0.4644**	**0.1161**	**0.1649**	**0.4326**	**0.1073**
Modelo arquitectura 2.1	0.0182	0.0508	0.0103	0.0165	0.0466	0.0103

Figura 5.9: Métricas ROUGE para modelos RNN.

Como se puede observar en la Tabla 5.9, todos los modelos, excepto el de la arquitectura 2.1, tienden a mostrar una precisión mayor que el recall. Se destaca que el modelo con mejor desempeño es el de la Arquitectura 2, pues tiene los valores más altos en todas las métricas de ROUGE-1 y ROUGE-L.

Discusión de Resultados

Los cuatro modelos propuestos no evidencian resultados significativos en la clasificación de oraciones, dado que no logran identificar correctamente todas las oraciones que deben incluirse en el resumen. Esto se puede observar en la matriz de confusión, donde el número de oraciones no clasificadas como pertenecientes al resumen es mayor que el de las clasificadas correctamente. Esto problema puede deberse al desbalance en los datos y a la posible insuficiencia del conjunto de noticias utilizado para el entrenamiento, lo que limita la capacidad del modelo para generalizar a datos nuevos.

Aunque la Arquitectura 1.1. mostró el mejor desempeño en términos de exactitud y pérdida de validación, esto no garantiza que sea la métrica más adecuada para evaluar la calidad del contenido generado. Por lo tanto, se utilizaron métricas ROUGE para analizar este aspecto. Los valores de ROUGE-L tienden a ser ligeramente inferiores a los de ROUGE-1, sugiriendo que los modelos son más efectivos capturando palabras individuales (n-gramas) que la estructura de la frase completa (subsecuencia más larga). Además, se observa que todos los modelos, excepto el modelo de la Arquitectura 2.1, presentan una mayor precisión que recall. Esto indica que las palabras que se predicen como relevantes son correctas la mayoría de las veces, aunque el modelo no

siempre identifica todas las palabras como importantes, resultando en un recall bajo.

El modelo que destaca en las métricas ROUGE-1 y ROUGE-L, es el de la Arquitectura 2, considerado el más efectivo en capturar tanto las palabras como la estructura de los resúmenes de referencia. El modelo de la arquitectura 2.1 mostró el peor rendimiento en todas las métricas, posiblemente debido a la complejidad de su arquitectura, que requiere más datos de los utilizados. Además, su formato de entrada incluye cada oración central junto con sus tres oraciones vecinas anteriores y posteriores, lo que incrementa el consumo de recursos computacionales.

Finalmente, de acuerdo con el análisis realizado y con el objetivo de continuar la comparación de la técnica RNN con la arquitectura codificador - decodificador, se seleccionó el modelo de la Arquitectura 2 para la técnica de redes neuronales recurrentes, debido a su superior desempeño frente a los demás. Aunque también se consideró el modelo de la Arquitectura 1.1 por su rendimiento en exactitud y pérdida de validación, sus resultados en la métrica ROUGE fueron significativamente inferiores a los de la Arquitectura 2.

A continuación, se detallan las especificaciones de la Arquitectura 2.

Cuadro 5.7: Comparación entre modelos

Parámetro	Valor
Unidades capa densa	64
Núm. capas LSTM	2
Unidades LSTM	64
Función de activación	ReLu
Épocas	56

5.2.3. Arquitectura Codificador-Decodificador

Conforme a lo expuesto en la Sección 4.5.2, se diseñó una arquitectura principal, con una variación en su modelo añadiendo capas de regularización. Se eligen los mejores modelos observando el Accuracy (precisión) y el Validation Loss (pérdida de validación). Así como en la Red Neuronal Recurrente, se utilizó Early Stopping, por lo que la cantidad de épocas puede variar.

Arquitectura 1

ID	Dropout	Dimensión Word Embedding	Unidades en capa densa	Learning Rate	Épocas	Accuracy	Validation Loss
1	0.75	1024	200	0.0001	110	0.7015	2.0602
2	0.25	256	200	0.01	27	0.6918	2.1745
3	0.75	256	200	0.01	27	0.6599	2.2861
4	**0.25**	**512**	**300**	**0.0001**	**45**	**0.7829**	**2.0534**

Figura 5.10: Primera arquitectura para el Codificador-Decodificador.

De las configuraciones de los hiperparámetros se puede observar que una mayor representación del embedding puede ser beneficiosa para capturar características más detalladas del lenguaje. Esto se evidencia en los modelos 1

y 5, que tienen una exactitud mayor. Además, la combinación de un dropout más bajo (0.25) y una mayor dimensión de embedding (512) parece mejorar el desempeño, como se observa en el modelo 5. Por otro lado, el learning rate o taza de aprendizaje, influye de forma inversamente proporcional, pues los modelos con mayor exactitud presentan un learning rate menor. Esto sucede porque los pequeños ajustes le permiten al modelo adaptarse más precisamente a los patrones presentes en los datos de entrenamiento.

De acuerdo con los resultados presentados, se seleccionó el **modelo 5**. El tiempo promedio para llevar a cabo los experimentos de la Arquitectura 1 fue de 2 minutos por época. Por lo tanto, el modelo con mayor número de épocas (110) tuvo una duración aproximadamente de 220 minutos.

Arquitectura 1.1

ID	Dropout	Dimensión Word Embedding	Unidades en capa densa	Learning Rate	Épocas	Accuracy	Validation Loss
1	0.25	256	200	**0.01**	43	0.7383	2.4854
2	**0.25**	**256**	**200**	**0.0001**	**55**	**0.7481**	**2.0678**
3	0.75	256	300	0.0001	152	0.6772	2.1652
4	0.25	512	300	0.0001	45	0.7827	2.0509

Figura 5.11: Variación de la primera arquitectura para el Codificador-Decodificador.

En esta arquitectura, se incorporó una capa de dropout después de la capa de embedding y se agregó regularización en la capa GRU. Los resultados muestran que modelo 5, que comparte los mismos parámetros que el modelo de la arquitectura 1, no presentó una mejora significativa. En contraste, el modelo 4, que utiliza un dropout más alto, evidenció una menor exactitud (0.6772). Similar a lo observado en la arquitectura 1, la tasa de aprendizaje se destaca como un parámetro crítico que afecta significativamente el rendimiento del modelo. Es un parámetro que debe variarse con discreción para evitar que se ajuste demasiado bien al conjunto de entrenamiento y no generalice bien a datos nuevos.

Se decidió seleccionar el **modelo 3**, que representa el segundo mejor desempeño de la Tabla 5.11, puesto que el modelo 5 ya se había escogido para la arquitectura 1. Esto con el objetivo de compararlo con un modelo cercano a su desempeño. Por otro lado, en los resultados de los experimentos de la arquitectura RNN, se evidenció que la inclusión de StopWords permitía generalizar mejor el modelo, a diferencia de utilizar conjunto de datos sin StopWords. Por esta razón, se decidió no experimentar con conjuntos de datos sin StopWords.

El tiempo promedio para llevar a cabo los experimentos de la Arquitectura 1.1 fue de 2 minutos por época. Por lo tanto, el modelo con mayor número de épocas (152) tuvo una duración aproximadamente de 304 minutos.

5.2.4. Comparación entre arquitecturas Codificador - Decodificador

Se analizó detalladamente el desempeño de los mejores modelos del Codificador-Decodificador para la generación de resúmenes de noticias económicas. El objetivo fue identificar el modelo más adecuado para representar eficazmente esta técnica.

Los modelos evaluados fueron seleccionados con base en sus resultados en términos de precisión (Accuracy) y pérdida de validación (Validation Loss), como se muestra en la Tabla X. Los resultados indican que el modelo de la Arquitectura 1 exhibió un rendimiento notablemente superior en comparación con el modelo de la Arquitectura 1.1.

Cuadro 5.8: Comparación entre modelos

Modelos	Accuracy	Validation Loss
Arquitectura 1	0.7829	2.0534
Arquitectura 1.1	0.7481	2.0678

La métrica ROUGE también fue empleada para analizar y comparar la calidad de los resúmenes generados con respecto a los resúmenes de referencia.

En particular, se consideraron ROUGE-1 y ROUGE-L.

	ROGUE-1			ROUGE-L		
	F-score	Precision	Recall	F-score	Precision	Recall
Modelo arquitectura 1	**0.6381**	**0.6486**	**0.6578**	**0.4376**	**0.4418**	**0.4542**
Modelo arquitectura 1.1	0.5701	0.6357	0.5468	0.3740	0.4156	0.3604

Figura 5.12: Métrica ROUGE para la arquitectura Codificador-Decodificador.

El modelo de la Arquitectura 1, que emplea únicamente una capa de dropout, muestra un mejor desempeño en todas las métricas ROUGE en comparación con el Modelo 1.1, que incorpora dos capas de dropouts y técnicas adicionales de regularización en la capa GRU. Además, el modelo de la Arquitectura 1 no solo alcanzó una mayor precisión, indicando una mejor capacidad para generar resultados correctos, sino que también presentó una menor pérdida de validación, lo que sugiere una mayor capacidad de generalización.

Discusión de resultados

La arquitectura 1 muestra una mayor exactitud y una mejor pérdida de validación, lo que indicaría una buena señal para considerar el modelo como el del mejor desempeño. Sin embargo, estas métricas no garantizan la calidad del contenido generado. Por tal razón, se emplea la métrica ROUGE para una evaluación más profunda y específica.

En términos de ROUGE, los dos mejores modelos seleccionados para la arquitectura Codificador-Decodificador, muestran una diferencia muy leve entre ellos. Se destaca que el modelo de la arquitectura 1, que solo contiene una capa de dropout, muestra un mejor desempeño en todas las métricas ROUGE en comparación con el modelo de la arquitectura 1.1, que incorpora un dropout adicional y técnicas de regularización en la capa GRU. Esto sugiere que, aunque las técnicas de regularización son útiles para evitar el sobreajuste, su implementación excesiva puede llevar a una disminución en el rendimiento del modelo.

El F-score, que combina precisión y recall en una sola medida, también re-

fuerza esta observación. Tanto en ROUGE-1 como en ROUGE-L, el F-score del modelo de la arquitectura 1.1 es menor al de la arquitectura 1.

Finalmente, de acuerdo con el análisis realizado y con el objetivo de continuar la comparación de la arquitectura Codificador-Decodificador con la técnica RNN, se seleccionó el modelo de la arquitectura 1. Aunque no tiene el mejor desempeño en métricas de precisión y pérdida de validación como el de la arquitectura 1.1, sí representa resultados significativos en la métrica ROUGE, que se encarga de evaluar tareas de procesamiento de lenguaje natural como la generación de resúmenes o la traducción automática.

A continuación, se detallan las especificaciones de la arquitectura 1.

Cuadro 5.9: Comparación entre modelos

Parámetro	Valor
Unidades capa densa	300
Dropout	0.25
Dim. word embedding encoder decoder	512
Learning Rate	0.0001
Épocas	45

5.2.5. Comparación de las técnicas

De acuerdo con los resultados obtenidos, la Red Neuronal Recurrente muestra una ligera superioridad en términos de exactitud al clasificar una oración como parte del resumen o no. Estos resultados son evidentes en la Tabla 5.10.

Cuadro 5.10: Comparación entre modelos

Modelos	Accuracy	Validation Loss
RNN	0.7932	0.5300
Codificador - Decodificador	0.7829	2.0534

Sin embargo, en las métricas ROUGE, se observa que la arquitectura Codificador-Decodificador presenta un buen equilibrio entre precisión y recall en comparación con la RNN.

	ROUGE-1			ROUGE-L		
	F-score	Precision	Recall	F-score	Precision	Recall
Codificador-Decodificador	0.6381	0.6486	0.6578	0.4376	0.4418	0.4542
RNN	0.1779	0.4644	0.1161	0.1649	0.4326	0.1073

Figura 5.13: Métrica ROUGE para ambas arquitecturas.

En términos generales, tanto la Red Neuronal Recurrente como la arquitectura Codificador-Decodificador enfrentan desafíos para determinar qué oraciones corresponden al resumen. Este problema puede estar relacionado con el desbalance en los datos, dado que hay mayor cantidad de oraciones que no se deben incluir en comparación con las que sí se deben incluir. Además, la insuficiencia de datos limita la capacidad de los modelos para generalizar a datos nuevos.

A pesar de estas dificultades, cada una de las arquitecturas presenta fortalezas en distintas métricas. La Red Neuronal Recurrente destaca en exactitud y presenta una menor pérdida en el conjunto de validación, lo que sugiere una buena capacidad para clasificar correctamente las oraciones. Por otro lado, la arquitectura Codificador-Decodificador supera a la RNN en las métricas ROUGE, lo que puede atribuirse a los estados ocultos producidos por el codificador, que le permiten al modelo capturar y mantener información contextual rele-

vante de la secuencia de entrada y utilizarla adecuadamente en la generación de la secuencia de salida.

Es importante señalar que este proyecto enfrentó limitaciones en términos de hardware, pues los recursos computacionales no permitieron experimentar con grandes cantidades de datos o con arquitecturas demasiado complejas. Esta restricción se reflejó en las predicciones realizadas por los modelos. No obstante, se logró implementar un generador de resúmenes de noticias económicas con cada arquitectura.

5.2.6. Ejemplo de generación de resúmenes

A continuación, se presentan ejemplos de resúmenes generados por las arquitecturas: Red Neuronal Recurrente y Codificador-Decodificador, a partir de una noticia económica. Además, se realiza una comparación con resúmenes producidos por GPT-4, con el objetivo de evaluar y analizar el rendimiento y la calidad de los resúmenes generados por estas arquitecturas en contraste con un modelo avanzado de lenguaje como GPT-4.

Para esta evaluación, se utilizó la plataforma ChatGPT. Dado que esta no permite procesar grandes volúmenes de datos de entrada, las noticias incluidas en esta sección fueron evaluadas manualmente. Para evitar cualquier influencia de los chats previos, se borró la memoria después de la generación de cada resumen.

Primer ejemplo

Noticia:

The dollar has hit its highest level against the euro in almost three months after the Federal Reserve head said the US trade deficit is set to stabilise.

And Alan Greenspan highlighted the US government's willingness to curb spending and rising household savings as factors which may help to reduce it. In late trading in New York, the dollar reached $1.2871 against the euro, from $1.2974 on Thursday. Market concerns about the deficit has hit the green-

back in recent months. On Friday, Federal Reserve chairman Mr Greenspan’s speech in London ahead of the meeting of G7 finance ministers sent the dollar higher after it had earlier tumbled on the back of worse-than-expected US jobs data. Ï think the chairman’s taking a much more sanguine view on the current account deficit than he’s taken for some time,"said Robert Sinche, head of currency strategy at Bank of America in New York. "He’s taking a longer-term view, laying out a set of conditions under which the current account deficit can improve this year and next."

Worries about the deficit concerns about China do, however, remain. China’s currency remains pegged to the dollar and the US currency’s sharp falls in recent months have therefore made Chinese export prices highly competitive. But calls for a shift in Beijing’s policy have fallen on deaf ears, despite recent comments in a major Chinese newspaper that the "time is ripe"for a loosening of the peg. The G7 meeting is thought unlikely to produce any meaningful movement in Chinese policy. In the meantime, the US Federal Reserve’s decision on 2 February to boost interest rates by a quarter of a point - the sixth such move in as many months - has opened up a differential with European rates. The half-point window, some believe, could be enough to keep US assets looking more attractive, and could help prop up the dollar. The recent falls have partly been the result of big budget deficits, as well as the US’s yawning current account gap, both of which need to be funded by the buying of US bonds and assets by foreign firms and governments. The White House will announce its budget on Monday, and many commentators believe the deficit will remain at close to half a trillion dollars.

Resumen original:

The dollar has hit its highest level against the euro in almost three months after the Federal Reserve head said the US trade deficit is set to stabilise. Market concerns about the deficit has hit the greenback in recent months. Ï think the chairmans taking a much more sanguine view on the current account deficit than hes taken for some time,s̈aid Robert Sinche, head of currency strategy at Bank of America in New York. Ḧes taking a longer-term view, laying out a set of conditions under which the current account deficit can improve this year and next.C̈hinas currency remains pegged to the dollar and the US

currencys sharp falls in recent months have therefore made Chinese export prices highly competitive. The recent falls have partly been the result of big budget deficits, as well as the USs yawning current account gap, both of which need to be funded by the buying of US bonds and assets by foreign firms and governments.

Resumen generado RNN:

The dollar has hit its highest level against the euro in almost three months after the Federal Reserve head said the US trade deficit is set to stabilise. And Alan Greenspan highlighted the US governments willingness to curb spending and rising household savings as factors which may help to reduce it. On Friday, Federal Reserve chairman Mr Greenspans speech in London ahead of the meeting of G7 finance ministers sent the dollar higher after it had earlier tumbled on the back of worse-than-expected US jobs data. In the meantime, the US Federal Reserves decision on 2 February to boost interest rates by a quarter of a point - the sixth such move in as many months - has opened up a differential with European rates.

ROUGE-1: (precision=0.496, recall=0.3780, fmeasure=0.4290)
ROUGE-L: (precision=0.344, recall=0.2621, fmeasure=0.2975)

Resumen extractivo generado por ChatGPT:

The dollar reached its highest level against the euro in almost three months after Federal Reserve Chairman Alan Greenspan suggested the US trade deficit might stabilize. Greenspan pointed to the US government’s plans to curb spending and increased household savings as positive factors. The dollar rose to $1.2871 against the euro, from $1.2974, reversing recent concerns about the deficit. Greenspan’s comments, made ahead of the G7 finance ministers meeting, sparked optimism about long-term improvements in the current account deficit. Meanwhile, China’s pegged currency continues to fuel competitiveness concerns, but no policy shifts are expected soon. Additionally, the US Federal Reserve’s recent interest rate hikes could maintain the dollar’s appeal against European rates. However, the US budget deficit remains a significant issue.

ROUGE-1: (precision=0.4574, recall=0.3598, fmeasure=0.4027)
ROUGE-L: (precision=0.3178, recall=0.2500, fmeasure=0.2799)

Resumen generado Codificador-Decodificador:

The dollar has hit its highest level against the euro in almost three months after the Federal Reserve head said the US trade deficit is set to stabilise. On Friday, Federal Reserve chairman Mr Greenspan's speech in London ahead of the meeting of G7 finance ministers sent the dollar higher after it had earlier tumbled on the back of worse-than-expected US jobs data. The recent falls have partly been the result of big budget deficits, as well as the US's yawning current account gap, both of which need to be funded by the buying of US bonds and assets by foreign firms and governments. In the meantime, the US Federal Reserve's decision on 2 February to boost interest rates by a quarter of a point - the sixth such move in as many months - has opened up a differential with European rates. But calls for a shift in Beijing's policy have fallen on deaf ears, despite recent comments in a major Chinese newspaper that the ẗime is ripeẗfor a loosening of the peg.

ROUGE-1: precision=0.565, recall=0.5823, fmeasure=0.5739
ROUGE-L: precision=0.445, recall=0.4588, fmeasure=0.4521

Resumen abstractivo generado por ChatGPT:

The US dollar reached its highest level against the euro in nearly three months following remarks by Federal Reserve Chairman Alan Greenspan, who indicated that the US trade deficit may stabilize. Greenspan highlighted government efforts to curb spending and increased household savings as factors that could reduce the deficit. Despite recent concerns over the deficit, especially due to China's currency being pegged to the dollar, Greenspan's comments offered a more optimistic long-term view. The Federal Reserve's recent interest rate hike also contributed to the dollar's strength by making US assets more attractive. However, the large budget and current account deficits remain challenges for the US economy.

Resumen extractivo generado a partir del resumen abstractivo anterior, calculando la distancia euclidiana entre las oraciones de la noticia original y las oraciones del resumen abstractivo:

The dollar has hit its highest level against the euro in almost three months after the Federal Reserve head said the US trade deficit is set to stabilize.

Alan Greenspan highlighted the US government's willingness to curb spending and rising household savings as factors which may help to reduce it. China's currency remains pegged to the dollar, and the US currency's sharp falls in recent months have made Chinese export prices highly competitive. The half-point window, some believe, could be enough to keep US assets looking more attractive and help prop up the dollar. The recent falls have partly been the result of big budget deficits, as well as the US's yawning current account gap, both of which need to be funded by the buying of US bonds and assets by foreign firms and governments.

ROUGE-1: (precision=0.6929, recall=0.5915, fmeasure=0.6382)
ROUGE-L: (precision=0.6571, recall=0.5610, fmeasure=0.6053)

Finalmente, se observa que la arquitectura Codificador-Decodificador tiende a predecir de manera más efectiva las oraciones que deben incluirse en el resumen, ya que en la mayoría de los casos, identifica al menos dos oraciones correctas. En contraste, la Red Neuronal Recurrente (RNN) a menudo solo predice una o dos oraciones correctamente, e incluso en ocasiones no acierta ninguna. Esta diferencia se refleja en las métricas ROUGE-1 y ROUGE-L, donde la arquitectura Codificador-Decodificador muestra una precisión más alta y un recall más equilibrado. En cambio, la RNN presenta una alta precisión pero un bajo recall, lo que sugiere que, aunque el modelo puede identificar algunas frases relevantes, no es exhaustivo y omite varias frases importantes.

También podemos observar que el resumen extractivo generado a partir del resumen abstractivo por ChatGPT muestra mejores métricas en comparación con los resultados obtenidos por arquitecturas como la Codificador-Decodificador y la Red Neuronal Recurrente. Se destaca que ChatGPT tiende a predecir al menos tres oraciones correctas, además de incluir otras oraciones que considera relevantes. Sin embargo, el resumen extractivo generado por ChatGPT muestra un puntaje ROUGE menor. Esto podría deberse a que el resumen abstractivo proporciona un mayor contexto sobre la noticia en general.

Todos los códigos implementados para el desarrollo de este proyecto se encuentran aquí.

Capítulo 6

Conclusiones

El desarrollo de este proyecto permitió ampliar conocimientos sobre el aprendizaje automático, específicamente, aprendizaje profundo. Durante su ejecución, se profundizó en las diversas disciplinas que lo integran, lo que permitió alcanzar el objetivo general planteado: desarrollar un sistema para la generación automática de resúmenes de noticias económicas utilizando técnicas de aprendizaje profundo.

Las arquitecturas escogidas surgieron en base al análisis de diversas técnicas de aprendizaje automático utilizadas para la generación de resúmenes extractivos. Cada una de estas técnicas fue evaluada en base a criterios de selección. Las seleccionadas fueron: Redes Neuronales Recurrentes y la arquitectura Codificador - Decodificador.

Durante la experimentación, se observó que, en tareas de procesamiento de lenguaje natural mediante técnicas de aprendizaje profundo, es crucial disponer de grandes cantidades de datos, pues esto influye en la capacidad del modelo para aprender y generalizar adecuadamente. Asimismo, grandes volúmenes de datos requieren recursos computacionales robustos, lo cual significó un desafío en el desarrollo de este proyecto. A pesar de las limitaciones computacionales, se llevaron a cabo experimentaciones para tratar alternativas y configuraciones de modelos que permitieron generar resúmenes de noticias económicas utilizando aprendizaje profundo.

En la etapa de evaluación con métricas, se analizó que en tareas de procesamiento de lenguaje natural, las métricas de exactitud y perdida de validación no son suficientes para garantizar la calidad del resumen generado. Por ello, las métricas ROUGE resultan ser herramientas valiosas para evaluarlos.

Capítulo 7

Trabajo Futuro

Ampliar la investigación sobre diferentes diseños y variantes de las Redes Neuronales Recurrentes, especialmente aquellas enfocadas en la generación de resúmenes abstractivos. Esto con el objetivo de comparar los resultados obtenidos con los de la arquitectura Codificador-Decodificador utilizada en este proyecto, para determinar cuál técnica ofrece un mejor desempeño y es útil usarla como medio para crear un resumen extractivo.

Implementar técnicas modernas como GPT-4, T5, y LLMs para evaluar su efectividad en la generación de resúmenes y comparar su rendimiento con las arquitecturas planteadas en este proyecto.

Explorar la técnica de Sentence Embedding y aplicarla a las arquitecturas desarrolladas en este proyecto, con el objetivo de evaluar el desempeño de los modelos al utilizar una técnica alternativa para representar su contenido.

Utilizar mejores recursos computacionales para analizar en mayor detalle las capacidades de los modelos con conjuntos de datos más grandes y realizar experimentos más extensivos para mejorar las métricas de las técnicas empleadas.

Implementar técnicas de transferencia de aprendizaje utilizando modelos previamente entrenados en grandes corpus de datos, con el objetivo de adaptarlos a tareas de generación de resúmenes extractivos y evaluar si el conocimiento adquirido durante el preentrenamiento puede mejorar la calidad de estos.

Bibliografía

[1] Prensa Escrita. Todos los periódicos diarios, 2022. [En línea]. Disponible en: https://www.prensaescrita.com/america/colombia.php

[2] Contribuidores de Wikipedia. Periodismo económico, 2020. Wikipedia, La Enciclopedia Libre. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Periodismo_econ%C3%B3mico.

[3] Salton, G., Singhal, A., Mitra, M., & Buckley, C. (1997). Automatic text structuring and sum- marization. Information Processing and Management, 33 (2)

[4] A. Echeverría. Generación Automática de Resúmenes extractivos genéricos de múltiples documentos basado en Word2Vec. 2018.

[5] A. Al Munzir, M. L. Rahman, S. Abujar, Ohidujjaman and S. A. Hossain. Text analysis for Bengali Text Summarization using Deep Learning. 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, pp. 1-6, 2019.

[6] Montilla, A. Algoritmo para Generación Automática de Resúmenes Extractivos Genéricos de un Documento basado en el Procedimiento de Búsqueda del Pescador, 2016.

[7] M. E. Mendoza. Generación automática de resúmenes extractivos de múltiples documentos basada en algoritmos meméticos. 2015.

[8] Corvi, J. Resumen extractivo de documentos. Un análisis comparativo de técnicas de puntuación, 2019.

[9] S. Russell and P. Norving. Artificial Intelligence a Modern Approach. Prentice Hall, 3 edition, 2010, pp. 649-651.

[10] M. Mohri, A. Rostamizadeh and A. Talwalkar. Foundations of Machine Learning. Massachusetts Institute of Technology, 2 edition, 2018.

[11] K. P. Sinaga and M. -S. Yang. Unsupervised K-Means Clustering Algorithm. vol. 8, pp. 80716-80727, 2020.

[12] S. Naeem, A. Ali, S. Anam y M. Munawar. An Unsupervised Machine Learning Algorithms: Comprehensive Review. International Journal of Computing and Digital Systems, 2023.

[13] M. Kaloev y G. Krastev. Experiments Focused on Exploration in Deep Reinforcement Learning. 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, pp. 351-355, 2021.

[14] J. Jia and W. Wang. Review of reinforcement learning research. 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC), Zhanjiang, China, pp. 186-191, 2020.

[15] W. Xiaofang, L. Lan, Z. Qianyin, L. Fengyu, L. Jiawei Y H. Di. Constructing Naive Bayesian Classification Model by Spark for Big Data. 2020.

[16] J. Kupiec, J. Pedersen y F. Chen. A Trainable Document Summarizer. 1995.

[17] T. Joachims. Text categorization with Support Vector Machines: learning with many relevant features. 1998.

[18] T. Hirao, H. Isozaki y E. Maeda. Extracting Important Sentences with Support Vector Machines. 2002.

[19] C. Ruíz and M. Basualdo. Redes Neuronales: Conceptos Básicos y Aplicaciones. 2001.

[20] Boltzmann machine. `https://en.wikipedia.org/wiki/Boltzmann_machine.Acceso:Abril29,2024.`

[21] Overview of Restricted Boltzmann Machine. `https://medium.com/@nibeditadas9/overview-of-restricted-boltzmann-machine-a1981feb37f2.Acceso:Abril29,2024.`

[22] J. Chung, C. Gulcehre, K. Cho and Y. Bengio. Gated Feedback Recurrent Neural Networks. 2015.

[23] H. Salehinejad, S. Sankar, J. Barfett, E. Colak and S. Valaee. Recent Advances in Recurrent Neural Networks. 2017.

[24] F. Mortezapour Shiri, T. Perumal, N. Mustapha and R. Mohamed. A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU. 2023.

[25] D. Jurafsky y J.H Martin. "Encoder-Decoder Models, Attention, and Contextual Embeddings", en Speech and Language Processing. 2019, cap. 10.

[26] Tokenization in NLP. https://medium.com/@abdallahashraf90x/tokenization-in-nlp-all-you-need-to-know-45c00cfa2df7. Acceso:Junio10,2024.

[27] T. Islam, M. Hossain y M. F. Arefin. Comparative Analysis of Different Text Summarization Techniques Using Enhanced Tokenization. 3rd International Conference on Sustainable Technologies for Industry 4.0 (STI), Dhaka, Bangladesh, pp. 1-6, 2021.

[28] A. A. A. Rafat, M. Salehin, F. R. Khan, S. A. Hossain y S. Abujar. Vector Representation of Bengali Word Using Various Word Embedding Model. 8th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, pp. 27-30, 2019.

[29] D. Jurafsky y J.H Martin. "Vector Semantics and Embeddings", en Speech and Language Processing. 2019, cap. 6.

[30] Comprensión de la Matriz de Confusión. https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python. Acceso:Junio15,2024.

[31] W. S. El-Kassas, C. R. Salama, A. A. Rafea y H. K. Mohamed. Automatic text summarization: A comprehensive survey. 2021.

[32] N. S. Shirwandkar y S. Kulkarni. Extractive Text Summarization Using Deep Learning. Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-5, 2018.

[33] C. D. Paice. The automatic generation of literature abstracts: An approach based on the identification of self-indicating phrases. 1981

[34] R. Nallapati, B. Zhou y M. Ma. Classify or Select: Neural Architectures for Extractive Document Summarization. 2016.

[35] S. P. Singh, A. Kumar, A. Mangal y S. Singhal. Bilingual Automatic Text Summarization Using Unsupervised Deep Learning. 2016.

[36] J. Cheng y M. Lapata. Neural Summarization by Extracting Sentences and Words. 2016.

[37] R. Nallapati, F. Zhai y B. Zhou. SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents. 2016.

[38] E. Wilson, A. Saxena, J. Mahajan, L. Panikulangara, S. Kulkarni y P. Jain. "FIN2SUM: Advancing AI-Driven Financial Text Summarization with LLMs". 2024 International Conference on Trends in Quantum Computing and Emerging Business Technologies, Pune, India, 2024, pp. 1-5.

[39] T. Goyal, J. Jessy Li, y G. Durrett. News summarization and evaluation in the era of gpt3. 2022.

[40] T. Zhang, F. Ladhak, E. Durmus, P. Liang, K. McKeown, y T. B Hashimoto. Benchmarking large language models for news summarization. 2023.

[41] H. Nguyen Thi Thu. An Optimization Text Summarization Method Based on Naïve Bayes and Topic Word for Single Syllable Language. 2014.

[42] M. S. Patil, M. S. Bewoor y S. H. Patil. A Hybrid Approach for Extractive Document Summarization Using Machine Learning and Clustering Technique. 2014.

[43] T. Priyadharshan y S. Sumathipala. Text Summarization for Tamil Online Sports News Using NLP. 2018.

[44] BBC News Summary. `https://www.kaggle.com/datasets/pariza/bbc-news-summary?resource=download`. Acceso: Junio 20, 2024.

[45] US Economic News Articles (Useful for NLP). https://www.kaggle.com/datasets/heeraldedhia/us-economic-news-articles?resource=download&select=US-Economic-News.csv. Acceso: Junio 20, 2024.

[46] Z. Wang, X. Shan, X. Zhang y J. Yang. N24News: A New Dataset for Multimodal News Classification. 2022.

[47] M. Ramina, N. Darnay, C. Ludbe y A. Dhruv. Topic level summary generation using BERT induced Abstractive Summarization Model. 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India. 2020. –

[48] Long Short-Term Memory (LSTM). https://d2l.ai/chapter_recurrent-modern/lstm.html. Acceso: Junio 20, 2024.

[49] Understanding Encoder-Decoder Sequence to Sequence Model. https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346. Acceso: Junio 21, 2024.

[50] Feature Extraction and Summarization with Sequence-to-Sequence Learning. https://www.kdd.org/kdd2018/hands-on-tutorials/view/feature-extraction-and-summarization-with-sequence-to-sequence-learning. Acceso: Junio 21, 2024.

[51] Understanding GRU Networks. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be. Acceso: Junio 21, 2024.

[52] Explaining L1 and L2 regularization in machine learning. https://medium.com/@fernando.dijkinga/explaining-l1-and-l2-regularization-in-machine-learning-2356ee91c8e3. Acceso: Junio 21, 2024.