



Síntesis de textos usando Modelos de Lenguaje

Alumno: Alejo Torres

Profesor Guía: María Lorena Talamé

Carrera: Ingeniería en Informática

Unidad Académica: Facultad de Ingeniería

Año: 2024

Índice

Índice.....	2
Índice de Ilustraciones.....	4
Índice de Tablas.....	6
Abstract.....	7
Capítulo 1: Introducción.....	8
Descripción del Problema.....	8
Motivación.....	9
Pasos a seguir.....	9
Capítulo 2: Estado de la cuestión.....	11
¿Qué es la inteligencia artificial?.....	11
¿Qué es una red neuronal?.....	12
Función de Activación.....	14
Regla de la cadena.....	15
Descenso del Gradiente.....	16
Algoritmo de retropropagación.....	20
Proceso de Optimización.....	23
¿Qué es Deep Learning ?.....	24
IAs Generativas.....	25
Autoencoders.....	26
Limitación de los Autoencoders.....	28
Autoencoders Variacionales.....	28
¿Qué es el Procesamiento del Lenguaje Natural?.....	31
Embeddings de palabras.....	34
Redes neuronales aplicadas al NLP.....	36
Redes LSTM.....	37
¿Qué es un LLM?.....	38
Arquitectura Transformer.....	39
Mecanismo de self-attention.....	41
Codificación posicional.....	43
Encoders.....	44
Decoders.....	45
¿Qué es el Clustering?.....	48
Algoritmo K-means.....	49
Cómo elegir K.....	51
Posibles Alternativas.....	52
Capítulo 3: Definición de Objetivos.....	54
Objetivo general.....	54
Objetivos Específicos.....	54
Restricciones o límites.....	54

Alcances del proyecto.....	55
Capítulo 4: Solución Propuesta.....	56
Frameworks y herramientas a implementar en el desarrollo del proyecto.....	57
API de inferencia.....	57
Manejo de los modelos.....	58
Librería de Machine Learning.....	59
Framework Backend.....	60
Framework Frontend.....	60
Metodología a implementar.....	61
Análisis de Requerimientos.....	63
Planificación del proyecto.....	64
Análisis de Costos.....	66
Análisis de Factibilidad.....	68
Análisis de Riesgos.....	74
Análisis y Diseño del Prototipo.....	76
Desarrollo del Prototipo.....	80
Desarrollo del Prototipo de la Interfaz de Usuario.....	89
Capítulo 6: Resultados.....	91
Capítulo 7: Conclusiones.....	100
Anexo I: Resultado de las pruebas.....	102
Referencias.....	114

Índice de Ilustraciones

Ilustración 1 visitas diarias en el periodo de Enero 2023 a Marzo 2023 de ChatGPT y Bing	7
Ilustración 2: Representación formal de una neurona	12
Ilustración 3: gráfico ejemplo de una recta que aproxima una serie de mediciones	16
Ilustración 4: gráfico que indica el error entre la aproximación y las mediciones	17
Ilustración 5: comparación de la gráfica de Error contra la grafica de la recta de aproximación	19
Ilustración 6: Esquema de un perceptrón multicapa	21
Ilustración 7: Esquema simplificado de un Autoencoder	26
Ilustración 8: Esquema de un VAE	29
Ilustración 9: Dos distribuciones Normales comparadas en el mismo sistema de ejes	29
Ilustración 10: Recta numérica con ejemplo de edades.	32
Ilustración 11: Ejemplos de palabras con relación a un identificador numérico.	32
Ilustración 12: Clusters de palabras	34
Ilustración 13: Esquema de una neurona LSTM.	36
Ilustración 14: Diagrama de la arquitectura Transformer original	41
Ilustración 15: proyección en 2D de vector llave y vector cerradura	42
Ilustración 16: Visualización de una matriz de atención.	43
Ilustración 17: senos y cosenos de la codificación posicional.	45
Ilustración 18: observaciones en un plano cartesiano.	51
Ilustración 19: centroides y observaciones en un plano cartesiano.	51
Ilustración 20: observaciones agrupadas en un plano cartesiano.	52
Ilustración 21: centroides y observaciones agrupadas en un plano cartesiano.	52
Ilustración 22: Comparación entre modelos GPT.	58
Ilustración 23: Planificación del proyecto.	66
Ilustración 24: Esquema de precios para GPT-4 32K.	68
Ilustración 25: Cálculo de VAN y TIR.	71
Ilustración 26: Diagrama de flujo del prototipo para la aplicación de síntesis de texto.	78
Ilustración 27: Fragmento de código I.	82
Ilustración 28: Fragmento de código II.	82
Ilustración 29: Fragmento de código III.	83
Ilustración 30: Fragmento de código IV.	83
Ilustración 31: Fragmento de código V.	84
Ilustración 32: Fragmento de código VI.	84
Ilustración 33: Visualización 2D de los Embeddings.	85

Ilustración 34: Fragmento de código VII.	85
Ilustración 35: Fragmento de código VIII.	86
Ilustración 36: Fragmento de código IX.	86
Ilustración 37: Fragmento de código X.	87
Ilustración 38: Fragmento de código XI.	87
Ilustración 39: Fragmento de código XII.	87
Ilustración 40: Fragmento de código XIII.	88
Ilustración 41: Fragmento de código XIV.	88
Ilustración 42: función encargada de paralelizar la síntesis de documentos.	89
Ilustración 43: Interfaz de usuario I.	90
Ilustración 44: Interfaz de usuario II.	91
Ilustración 45: Diseño del sistema.	92
Ilustración 46: Tema no presente en el texto original aparece en la síntesis.	96
Ilustración 47: perceptrón multicapa representación matricial de sus parámetros.	103
Ilustración 48: Ejemplo de cálculos numéricos.	106

Índice de Tablas

Tabla 1: Costos de recursos humanos	65
Tabla 2: Costos de Hardware	65
Tabla 3: Configuración de las pruebas	92

Abstract

Los modelos de inteligencia artificial enfocados a la generación de contenido, también llamadas Inteligencias Artificiales (IA) generativas, sin duda alguna han revolucionado el paradigma de trabajo de sectores completos. La tecnología es un hecho y existen numerosos productos y servicios que integran su potencial. Los *Large Language Models (LLMs)* que generan contenido del tipo *text-to-text*, son el motor que impulsa numerosas aplicaciones de tendencia actual. ChatBots, asistentes virtuales y motores de búsqueda personalizados son algunos ejemplos de éstos. Sin embargo, los *LLMs* comprenden una potente herramienta para el tratamiento de la información: al ser capaces de procesar lenguaje natural es posible usar estos modelos generalistas en tareas de extracción y síntesis de contenido. El propósito de este proyecto de grado es el de desarrollar una aplicación que permita resumir el contenido de un archivo en formato PDF, sin importar su tamaño, aprovechando el poder de inferencia de los *LLMs* presentando estos servicios de *machine learning* a través de una interfaz intuitiva y amigable para un usuario final.

Palabras claves: *Large Language Models, Machine Learning, Síntesis de texto, Transformer*

Capítulo 1: Introducción

Descripción del Problema

Hoy en día, inteligencia artificial, modelos generativos, *chatGPT* son frases tendencia que están en la boca de todo el mundo. Usuarios con formación técnica como aquellos que no la poseen ya se benefician de las ventajas que brindan estas nuevas tecnologías al alcance de todos. Se puede observar en la Ilustración 1 como en el mes de Febrero del año 2023, la página de *ChatGPT*, el asistente impulsado por IA más popular del momento, sobrepasa la cantidad de visitas diarias de Bing, un motor de búsqueda con más 500 millones de visitantes únicos mensuales (Silverio, s. f.).

ChatGPT vs. Bing

Daily Visits, Desktop & Mobile Web, Worldwide (main bing.com domain only)

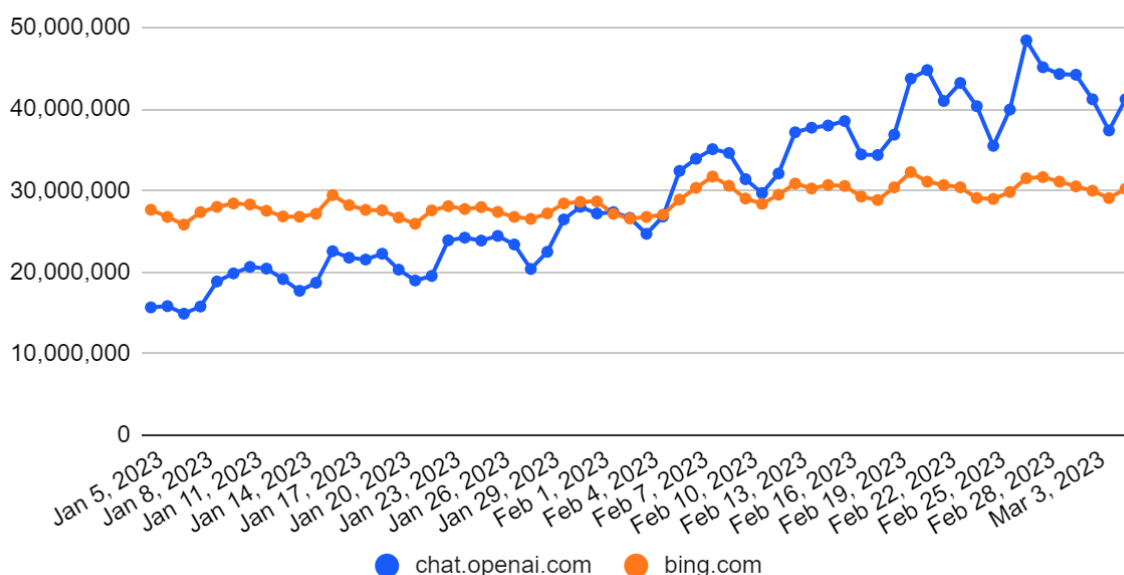


Ilustración 1: visitas diarias en el periodo de Enero 2023 a Marzo 2023 de ChatGPT y Bing Fuente: (Carr F. David, 2023, similarweb)

Se ha visto el surgimiento de startups tecnológicas basadas completamente en esta esfera de conocimiento generando aún más especulación y entusiasmo por el alcance que pueden lograr los productos y servicios que integran exitosamente las virtudes de la inteligencia artificial (Law, 2022). Sin embargo, un área interesante para explotar es la del tratamiento de información de manera automática aprovechando las capacidades del procesamiento del lenguaje natural que brindan los Grandes Modelos de Lenguaje, del inglés Large Language Models (en adelante, LLMs). Estos algoritmos de deep learning son capaces de sintetizar, traducir, predecir y generar contenido usando grandes conjuntos de

datos (Nvidia, 2023). El poder condensar la información relevante es fundamental para un correcto proceso de estudio, sobre cualquier tema. En la era de la información que se vive actualmente, donde se cuenta con acceso casi instantáneo a grandes volúmenes de datos, la capacidad de síntesis se vuelve aún más importante. El problema surge cuando la cantidad de texto a procesar demanda más tiempo del que se cuenta para esa tarea.

Motivación

El interés por basar mi proyecto de grado alrededor de esta temática surge gracias a la oportunidad que me fue brindada por la *Universidad Católica de Salta* en conjunto con la *Universidad Politécnica de Madrid* donde pasé un semestre cursando las asignaturas de *Sistemas Inteligentes*, *Aprendizaje automático II* y *Minería de datos*, las cuales fueron dictadas por los profesores *Alberto Díaz Alvarez* y el *Dr. Francisco Serradilla García*. Sus excelentes cátedras me sirvieron a modo de introducción y pude adquirir las bases necesarias para continuar mi aprendizaje de forma autónoma, donde decidí inclinarme por las IAs generativas, en específico aquellos modelos basados en la arquitectura *Transformer*.

Fue durante este proceso de investigación y de asimilación de conocimientos que se despertó en mí la curiosidad por el desarrollo de aplicaciones que utilicen estas tecnologías. Así fue como pude tomar noción del alcance real que se puede lograr si aprovechamos estas técnicas para agilizar y modernizar los procesos de enseñanza y estudio. Mi intención es generar una potente herramienta de estudio que permita facilitar la asimilación de conocimiento, tanto para personas involucradas en tareas de investigación como estudiantes de cualquier nivel.

La comunidad *Open Source*, quién innumerables veces ha aportado en mis desarrollos personales, es otro gran factor por el que he decidido abordar este tema. Cuento con la intención de liberar todo el material resultante de este proyecto con la esperanza de poder aportar significativamente al proceso de formación de cualquiera que así lo desee.

“Si hemos podido ver más lejos, es porque nos hemos parado sobre hombros de gigantes”

- Isaac newton

Pasos a seguir

El trabajo implica, en primera instancia, una investigación y familiarización con las técnicas y tecnologías involucradas, como lo es la arquitectura Transformer. En pocas palabras, esta arquitectura puede definirse como una red neuronal que aprende contexto y por lo tanto, también significado, mediante el seguimiento de relaciones en datos secuenciales como las palabras de esta oración (Merritt, 2022). La misma da sustento a los *LLMs*. La revisión de librerías, proyectos y estudios relacionados con el Procesamiento del Lenguaje Natural (*NLP*), técnicas de *clustering* y *machine learning*, desglosando los conceptos hasta el punto que puedan ser comprendidos y utilizados para la construcción del software de síntesis de texto.

Lo anterior mencionado se complementará con el diseño de la arquitectura de los servicios de inferencia del sistema donde se integrarán todas estas herramientas para lograr el fin último del software. Sobre este diseño se evaluará su rendimiento a través de la generación de resúmenes de temas específicos que luego serán evaluados por profesionales en la materia. Gracias a ésto se podrá perfeccionar las directivas y los parámetros de los modelos.

Teniendo en cuenta estos resultados se producirán las modificaciones pertinentes a la arquitectura, acompañado del desarrollo de una interfaz web amigable e intuitiva con la que el usuario podrá interactuar con los servicios de inteligencia artificial mencionados anteriormente.

Capítulo 2: Estado de la cuestión

¿Qué es la inteligencia artificial?

Se puede definir la Inteligencia Artificial (IA), de manera general, como la capacidad de artilugios artificiales de realizar tareas propias de una inteligencia humana (Sáenz de Cabezón, 2021). Cuestiones como la capacidad de cálculo y la memoria resultan atributos fáciles de asignar tanto a un computador como a una persona. Sin embargo, cualidades como la capacidad de aprendizaje, la creatividad e, incluso, la autoconciencia son cualidades humanas sobre las que, poco a poco, la inteligencia artificial ha avanzado.

La Inteligencia Artificial es un campo de la informática y la ciencia de la computación que se enfoca en desarrollar sistemas y programas capaces de realizar tareas que requieran inteligencia humana. En esencia, la IA busca crear máquinas que puedan emular y simular procesos de pensamiento, razonamiento y toma de decisiones propios de los seres humanos (Gobierno de España, s. f.). Es un campo multidisciplinario y a lo largo de las décadas, ha evolucionado desde conceptos teóricos hasta aplicaciones prácticas que están transformando rápidamente diversas industrias.

En el núcleo de la IA se encuentran los algoritmos y modelos que permiten a las máquinas aprender de los datos y adaptarse para resolver problemas complejos. El aprendizaje automático (*machine learning*) es una subárea fundamental de la IA que implica el desarrollo de técnicas que permiten a las máquinas mejorar su rendimiento a medida que se exponen a más datos.

Se puede definir *machine learning* como un conjunto de técnicas mediante las cuales un algoritmo que debe realizar una tarea en específico puede modificar su comportamiento basándose en los datos que dispone, en una medida de que tan bien o mal ha realizado su tarea o que tan bien ha actuado según lo que le informe un tercero. Es decir, pueden aprender de sus errores. Este es el concepto fundamental que da origen a todas las técnicas, algoritmos y arquitecturas. Respecto a cómo aprenden estos algoritmos existen tres enfoques fundamentales (Gonzalez, 2020):

- *Aprendizaje supervisado*: útil cuando se cuenta con datos etiquetados, es decir, se conoce de antemano la solución inherente al conjunto de datos, por lo que se puede optimizar el algoritmo comparando su rendimiento con lo que debería resultar.
- *Aprendizaje no supervisado*: de importancia cuando no se tienen etiquetas en los datos, es decir no se conocen el resultado a priori para el conjunto de datos. Relacionado con tareas de agrupamiento.

- *Aprendizaje por refuerzo*: actúa mucho por prueba y error. Generalmente utilizado para entrenar agentes, usando las mediciones tomadas del entorno y su rendimiento para calcular un puntaje.

Una vez en contexto, las *redes neuronales* son una de estas técnicas englobadas dentro de la gran esfera del machine learning, presumiblemente la más potente y versátil de todas. Una concepción errónea que se tiene sobre la redes de neuronas es que está completamente inspirada en el cerebro humano, cuando en realidad lo único en lo que se asemejan estas dos es en la forma como los nodos o *neuronas* de una red se interconectan entre sí, como también lo hacen las células del cerebro. Además se puede asemejar la sinapsis que ocurre en el mismo con los cálculos que se producen dentro de cada uno de estos pequeños nodos. Aparte de eso, no presentan más similitudes.

¿Qué es una red neuronal?

Una red neuronal es un sistema computacional que involucra hardware y software para tratar de imitar las capacidades computacionales de los sistemas biológicos usando elementos simples interconectados llamados neuronas. Estas efectúan una serie de cálculos simples sobre los datos de entrada para luego comunicar el resultado a las demás neuronas relacionadas. Algunas ventajas que presentan frente a otras técnicas de machine learning son:

- *Aprendizaje adaptativo*: esto quiere decir que aprende de un conjunto de experiencias, es decir, de los datos de entrada sin necesidad de que un experto en la materia deba estar indicando cómo debe ser ese proceso de aprendizaje.
- *Autoorganización*: la propia red, a través de sus parámetros, genera una interpretación de la información que recibe durante el entrenamiento.
- *Tolerancia a fallos*: puede trabajar con información parcial, dañada o ruidosa. Si bien es cierto que un modelo será tan bueno como los datos sobre los cuales fue entrenado, esto no quiere decir que no tenga cierto nivel de tolerancia al ruido y a la información faltante al aprender las características más relevantes del conjunto de datos.
- *Flexibilidad*: pueden adaptarse a nuevos entornos sin ser re-entrenadas gracias a su capacidad de aprendizaje.
- *Tiempo real*: debido a la naturaleza matricial de los cálculos en una red se puede aprovechar el poder de cómputo especializado de las unidades de procesamiento de gráficos junto con técnicas de loteo para poder usar máquinas paralelas, alcanzando más velocidad y reduciendo los tiempos de entrenamiento e inferencia.

- Generalización: una vez entrenada, puede clasificar elementos desconocidos y también se puede determinar qué cosa la red todavía no ha aprendido a hacer.

Por muy extraño que parezca, las redes de neuronas no son más que expresiones matemáticas. Tomando las entradas de datos como inputs de la función, aplicando los pesos de la red y las desviaciones, se obtienen las salidas que son las predicciones o inferencias de la red. Las redes de neuronas son una clase de función matemática y pueden ser estudiadas a partir del diagrama de la Ilustración 2 (García Serradilla, 2023).

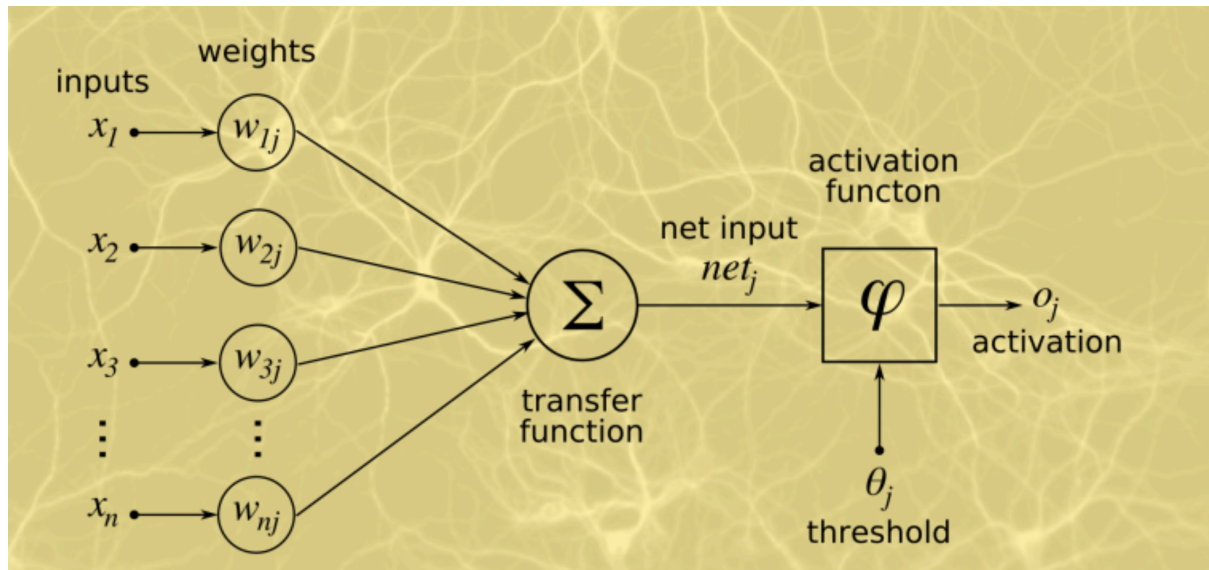


Ilustración 2: Representación formal de una neurona Fuente: (García Serradilla, 2023)

Una explicación intuitiva es la siguiente: cada uno de los datos de entrada x_i sufrirán un escalado dictado por un parámetro entrenable llamado *peso* denotado por W_{ij} . Este parámetro indica la relevancia de esa entrada para la tarea en cuestión. Que un parámetro sea entrenable significa que el mismo será optimizado durante el proceso de entrenamiento. Luego, cada uno de estos resultados serán sumados para obtener el resultado parcial dentro de la red. Sin embargo, lo que se tiene hasta ahora es una suma de términos escalados respecto a un valor arbitrario. Existe otro parámetro entrenable llamado *desviación* o *bias*, el cual constituye un término independiente. Si se piensa en una representación del modelo a construir en un par de ejes cartesianos, este término permite realizar una traslación en el eje vertical. Tomando como ejemplo una serie de observaciones graficadas en un plano y teniendo en cuenta el objetivo de realizar un trabajo de regresión sobre ellos, ajustando una recta lo mejor que se pueda. Sin un término independiente no se podría ajustar bien la recta para estos datos particulares, por lo que su función es agregar libertad al modelo. De la mano al mismo concepto, se sabe que al sumar términos lineales el resultado siempre será lineal (por la propiedad de cerradura bajo la adición). Sin

embargo, es muy complejo encontrar datos del mundo real que presenten una naturaleza lineal (obviando fenómenos físicos puntuales), por lo que interesa hallar una forma de realizar regresión o clasificaciones con un modelo más libre que permita ajustar una curva arbitraria y desprenderse de una simple regresión lineal. Esto es justo lo que sucede en cada una de las neuronas, una pequeña regresión lineal (Abhigyan, 2020) sobre el dato en cuestión. Para poder romper con la linealidad se introduce el concepto de función de activación. Teniendo en cuenta la Ilustración 2 y la explicación anterior, se puede presentar el concepto de perceptrón, el cual es uno de los modelos más básicos y fundamentales dentro de las redes neuronales. El mismo consiste en una sola neurona que toma una entrada, realiza una suma ponderada y luego aplica una función de activación para producir una salida. Las redes neuronales, en esencia, están compuestas por múltiples perceptrones organizados en capas: la primera es una capa de entrada que recibe los datos iniciales, aunque si bien recibe el nombre de capa de entrada o *input layer* la realidad es que no realiza ningún tipo de cálculo significativo para el modelo, más bien cumple la tarea de “marco” o *placeholder* para estructurar la información que ingstará el modelo. Seguido a ésta se ubican una o más capas ocultas que procesan las características internas, y por último, una capa de salida que produce la predicción final. Cada una de las neuronas de una capa está conectada a cada neurona de la capa siguiente, formando una estructura compleja que permite a la red aprender representaciones no lineales de los datos. Al ajustar los pesos de las conexiones entre estas neuronas mediante un proceso llamado entrenamiento, se pueden modelar relaciones complejas y patrones que no son percibibles a simple vista en los datos (García Serradilla, 2023).

Función de Activación

Una función de activación es un componente fundamental en las redes neuronales. Se utiliza para determinar si y en qué medida una neurona debe activarse (enviar una señal) en función de la suma ponderada de las entradas que recibe. Las funciones de activación introducen no linealidades en el modelo, lo que permite a las redes neuronales aprender relaciones complejas en los datos.

Como se vio, las neuronas reciben entradas y las multiplican por sus correspondientes pesos, suman las desviaciones y luego se aplica una función de activación al resultado. Existen diversas funciones de activación pensadas para tareas específicas. La elección de la función de activación depende del problema y las características del modelo. Sin embargo, las más comunes y que se usarán luego para explicar el resto de los temas, son las siguientes:

- *Función Rectified Linear Unit (ReLU)*: La función ReLU es más popular en la actualidad debido a su eficiencia computacional y la prevención del problema de gradientes que afecta otras funciones. Simplemente devuelve cero si el valor de entrada es negativo y el mismo valor, si es positivo. Su fórmula es: $f(x) = \max(0, x)$
- *Función Softmax*: La función Softmax es usada especialmente en la capa de salida de redes neuronales para convertir las salidas en probabilidades. Es útil en problemas de clasificación multiclase¹. Su fórmula es: $\text{softmax}(Z_i) = \frac{e^{Z_i}}{\sum_{j=1}^k e^{Z_j}}$ donde Z_i

es la entrada de la función softmax para i

- *Función Sigmoide*: La función sigmoide tiene una forma de "S" y transforma cualquier valor en un rango entre 0 y 1. Fue ampliamente utilizada en redes neuronales debido a la facilidad de poder expresar la derivada en términos de la misma función, pero está siendo reemplazada, en muchas aplicaciones, debido a problemas de *desvanecimiento* y *explotación* del gradiente del cual se hablará más adelante. Su fórmula es: $\frac{1}{1+e^{-x}}$.

Mediante la *no linealidad* que brinda este artilugio matemático se pueden construir modelos que se ajusten a cualquier comportamiento que presentan los datos. Pero sigue sin resolverse algo crucial, la forma en la que estos sistemas aprenden (Bag, 2023).

Regla de la cadena

Antes de hablar sobre el mecanismo utilizado para el aprendizaje dentro de las redes neuronales es necesario comentar otro artilugio matemático que se aplica en este algoritmo. La *regla de la cadena* (también conocida como el teorema de las funciones compuestas) es una fórmula explícita de la derivada de una función compuesta por dos funciones derivables. Esta regla permite conocer la j -ésima derivada parcial de la i -ésima aplicación parcial de la composición de dos funciones de varias variables. Un ejemplo conceptual sería el siguiente: ***si un automóvil puede viajar el doble de rápido que una bicicleta y una bicicleta es cuatro veces más rápida que un ser humano entonces un auto es 8 veces más rápido que un ser humano.***

¹ La clasificación es un tipo de problema en *machine learning* donde el objetivo es predecir la clase o categoría a la que pertenece un nuevo dato basándose en el conocimiento adquirido a partir de un conjunto de datos de entrenamiento previamente etiquetado. A su vez un problema multiclase es un subtipo de problema de clasificación donde existen tres o más clases posibles en las que se pueden clasificar los datos. El objetivo es asignar una de múltiples etiquetas posibles a cada muestra de datos. Por ejemplo, en un problema de clasificación de imágenes, las categorías pueden ser "perro", "gato", "coche", "avión". El modelo debe ser capaz de clasificar cada imagen en una de estas categorías.

$$\frac{dz}{dt} = \frac{dz}{dy} * \frac{dy}{dt} = 2 * 4 = 8$$

Como es conocido, una derivada es una razón de cambio, una proporción que indica cuánto cambia una variable en función de un cambio producido en otra. Esta es la idea fundamental detrás del entrenamiento de una red neuronal: en definitiva, se desea observar qué tanto ha influido cada parámetro en el resultado obtenido para así ajustarlo adecuadamente.

Descenso del Gradiente

La utilidad de la regla de la cadena en este campo viene acompañada por el algoritmo de descenso del gradiente (*DG*). Intrínsecamente, la inteligencia artificial tiene sus bases en un problema de optimización. El entrenamiento de una red de neuronas es un problema en donde no se optimizan costos ni se trata de maximizar ganancias, sino que se busca reducir un error el cual resulta de comparar la inferencia y las etiquetas originales (García Serradilla, 2023) . El error producido es el resultado de la diferencia entre *lo que realmente es* y *lo que el modelo infiere*. En definitiva, se busca que esta función de pérdida o costo sea la menor posible.

Las variables independientes que intervienen en esta función de error tendrán origen en todos los parámetros entrenables de la arquitectura de redes neuronales: los pesos, los cuales formarán matrices en cada capa y las desviaciones o *bias*, que se agrupan vectorialmente. Estos dos tipos de parámetros se ajustarán iterativamente hasta alcanzar valores óptimos.

Dicho esto, la pregunta ahora es: ¿de qué manera se puede optimizar estos parámetros? La forma más común, o en la que la mayoría de las técnicas se basan, es en el método del descenso del gradiente, aunque también pueden ajustarse los pesos de la red mediante técnicas que no utilicen *DG* como, por ejemplo, un *algoritmo de computación evolutiva*. Como este tipo de algoritmos de aprendizaje escapa a los modelos y temas que se tratarán a continuación, si se desea profundizar sobre el tema puede referirse a la Notebook Jupyter.² para ejecutar el entrenamiento de un perceptrón multicapa utilizando un algoritmo de computación evolutiva.

Para poder comprender el mecanismo de descenso del gradiente, se tomará un ejemplo sencillo, utilizando la curva de la Ilustración 3. El objetivo será el de aproximar una recta (como la de la imagen) a los datos u observaciones los cuales están representados

² En el siguiente link <https://github.com/AlejoTorres2001/AI/tree/windows/ProyectoDeGrado/Anexos/Anexo1-Neuroevolucion> encontrará todos los archivos e instrucciones necesarios para ejecutar el notebook Jupyter.

por puntos verdes. Este aproximador se utilizará para poder predecir valores de la altura de una persona según la información de su peso.

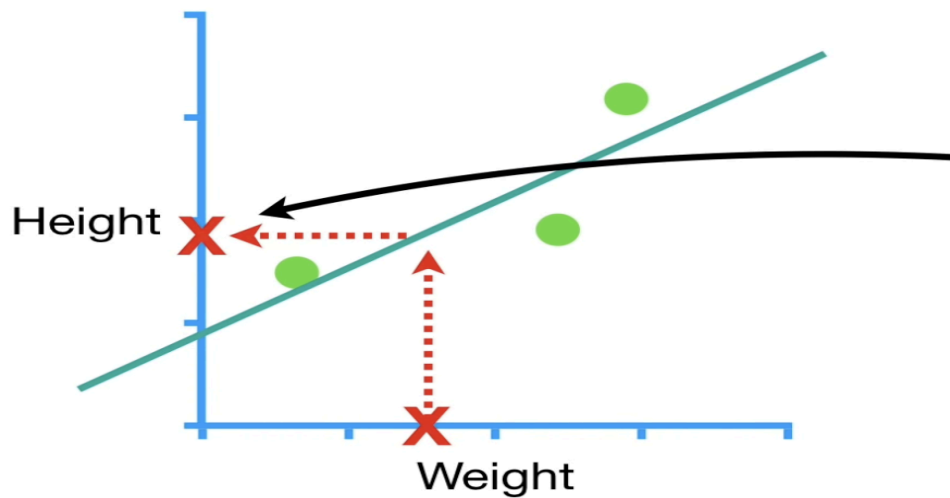


Ilustración 3: gráfico ejemplo de una recta que aproxima una serie de mediciones, Fuente: (Starmer J, 2019, StatQuest)

La recta tiene la siguiente ecuación:

$$\text{Predicted Height} = \text{intercept} + \text{slope} * \text{Weight}$$

Se optimizará, únicamente, la pendiente de la recta (*slope*) usando el método del descenso del gradiente y se introducirá un valor numérico a la intersección obtenido por el método de cuadrados mínimos. *Cuadrados mínimos* se refiere a un método útil en estadísticas y matemáticas para encontrar la mejor aproximación a través de una línea o curva que minimiza la suma de los cuadrados de las diferencias entre los valores observados y los valores predichos por la línea o curva. Este método es comúnmente usado en el análisis de regresión y se emplea para ajustar modelos matemáticos a conjuntos de datos (Starmer, 2019).

Puede surgir ahora la siguiente pregunta: *¿por qué no usar el método de mínimos cuadrados para optimizar los parámetros en vez del descenso del gradiente?* El método de mínimos cuadrados resuelve este problema encontrando la solución más óptima o la solución perfecta. Lo que sucede es que este método se basa en resolver la derivada igualando la ecuación a 0, para buscar los puntos mínimos. Esto vuelve al método de mínimos cuadrados poco flexible. En cambio, el método del descenso del gradiente estima muy bien esta solución perfecta sin la necesidad de este paso limitante, volviéndolo mucho más flexible en escenarios donde no se puede resolver dicha ecuación de manera analítica.

Como se verá a continuación, el método del descenso del gradiente es un método numérico e iterativo. El *error*, en este caso, surgirá de la aproximación generada por la recta en el punto en cuestión y la medición tomada que forma parte del conjunto de datos.

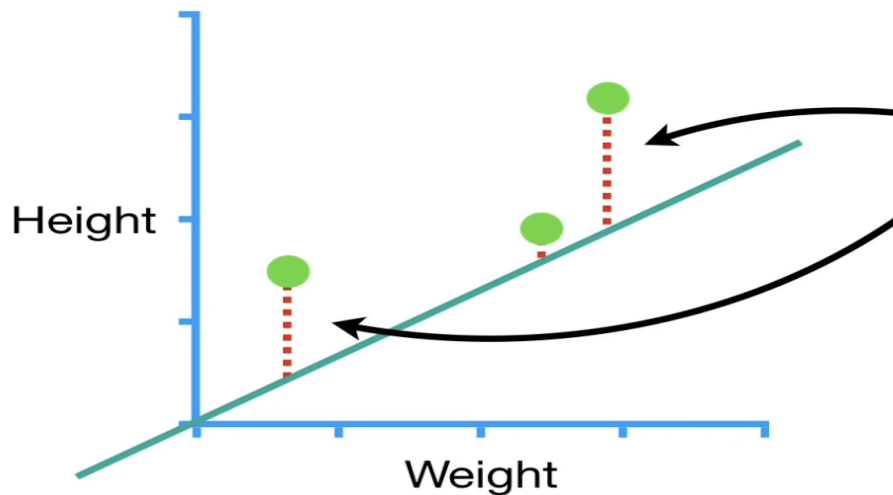


Ilustración 4: gráfico que indica el error entre la aproximación y las mediciones Fuente: (Starmer J, 2019, StatQuest)

Es decir, el error será la diferencia entre lo observado y lo que el modelo ha predicho. Si luego a este error se le calcula su cuadrado, se obtiene el *error cuadrático*. Algunas de las razones por las que se opta por utilizar el cuadrado del error son:

- *Propiedades Matemáticas:* El error cuadrático produce una función de pérdida que es continua, diferenciable y convexa en la mayoría de los casos, lo que es esencial para el proceso de optimización usando el descenso de gradiente, que requiere de las derivadas, como se verá más adelante.
- *Sensibilidad a Diferencias Grandes:* El error cuadrático penaliza de manera más significativa las diferencias grandes entre las predicciones y los valores reales. Esto puede ser útil cuando se desea que el modelo sea más sensible a los casos en los que se cometen errores significativos.
- *Solución Analítica:* En algunos casos, el error cuadrático puede tener soluciones analíticas para los valores óptimos de los parámetros del modelo. Esto facilita el cálculo de los valores que minimizan el error cuadrático.
- *Derivadas Claras:* Las derivadas del error cuadrático en relación con los parámetros del modelo son simples y pueden calcularse directamente. Esto facilita el cálculo de los gradientes necesarios para los métodos de optimización (Frost, 2021).

Siguiendo el ejemplo, se optimizará el parámetro de la intersección de la recta. Si se repite el proceso de calcular los errores de cada recta para un valor de intersección distinto, se obtienen los siguientes gráficos donde se ve que desplazando la intersección de la recta produce un valor distinto para el error. Gráficamente se puede ver en el gráfico de puntos donde se mide el error respecto de la interacción que existe un mínimo valor.

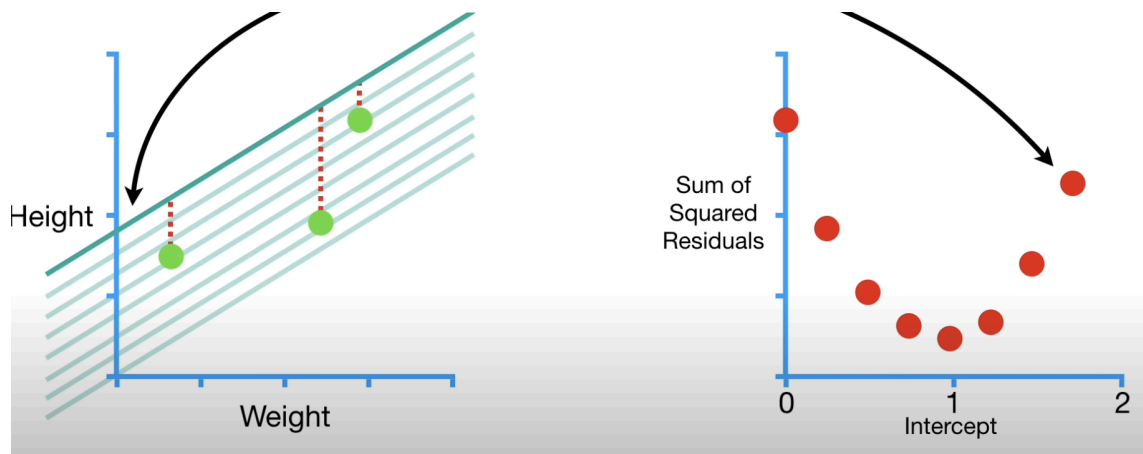


Ilustración 5: comparación de la gráfica de Error contra la grafica de la recta de aproximación, Fuente: (Starmer J, 2019, StatQuest)

A continuación se verá cómo aplicando el algoritmo de descenso. Lo que se busca es:

$$\frac{dSumSqrResiduals}{dIntercept} = \sum \frac{d(RealValue - predictedValue)^2}{dIntercept}$$

Es decir, se busca saber cuánto influye la intersección en el error del modelo, que en este caso es la recta. Por esto se sabe que *predictedValue* está dado por la ecuación de la recta $intercept + slope * weight$. Por lo que es posible aplicar la regla de la cadena y obtener la derivada respecto del parámetro para luego sustituir las variables necesarias por las mediciones y así obtener un número que representa el error para ese modelo en concreto.

Esta información indica la pendiente de la curva de error. Como el objetivo es minimizar el error, interesa recorrer la curva de manera descendente. Además, como se busca aproximar esta derivada a 0, mientras más lejos se está del 0, se tomarán incrementos más grandes en dirección descendente y mientras más cerca se esté del 0, los pasos se volverán más pequeños. Esto se logra con otro parámetro intrínseco del algoritmo llamado factor de aprendizaje o *Learning Rate* (α). Gracias a la derivada que brinda la pendiente de la curva, se sabe que cuando la misma es 0 (o muy aproximada) se encuentra un punto de inflexión, en este caso, uno mínimo.

Dado que las redes neuronales suelen tener múltiples capas y una gran cantidad de parámetros, el espacio de búsqueda es extremadamente complejo y suele contener muchas curvas y valles. Esto da lugar a la presencia de mínimos locales, que son puntos donde la función de pérdida es menor que en sus alrededores inmediatos, pero no necesariamente el valor más bajo en todo el espacio. El término "mínimo absoluto" se refiere al punto de la función de error que tiene el valor más bajo en todo el espacio. Debido a la complejidad y no linealidad del espacio de búsqueda en las redes neuronales, es difícil asegurar que el proceso de descenso del gradiente convergerá exactamente al mínimo absoluto. En cambio, es más común que el descenso del gradiente pueda quedar atrapado en mínimos locales (Keim, 2020).

Sin embargo, es importante destacar que aunque los mínimos locales son un desafío en el proceso de optimización, en la práctica, las redes neuronales a menudo logran un rendimiento aceptable mediante la exploración y la búsqueda de soluciones en dichos espacios. Además, hay técnicas avanzadas, como el uso de *tasas de aprendizaje adaptativas*, la *inicializaciones adecuadas de los pesos* y la *exploración de arquitecturas* de redes diferentes, que pueden ayudar a mitigar el problema de los mínimos locales en el entrenamiento (García Serradilla, 2023).

A modo de resumen, se puede decir que el algoritmo del descenso del gradiente está compuesto por los siguientes pasos:

1. Calcular la derivada de la función de error para cada parámetro involucrado en ella, Cuando se habla de un parámetro en términos de la función de error se refiere a una variable independiente que influye de alguna manera y con cierto peso en el resultado de error final
2. Inicializar los parámetros con valores aleatorios
3. Utilizar estos valores en el gradiente calculado en el paso 1
4. Calcular el tamaño del paso $StepSize = Slope * LearningRate$
5. Calcular el nuevo valor del parámetro
 $NewParameter = OldParameter - StepSize$

Las condiciones de cierre de este algoritmo, en teoría son dos: puede suceder que el que el tamaño de paso sea muy cercano a 0, o que hayamos alcanzado un número fijo de iteraciones (Starmer, 2019). En la práctica puede detenerse el entrenamiento de manera manual en tres circunstancias: luego de una serie de iteraciones (*epochs*), cuando alcancemos un valor de error aceptable o luego de un tiempo fijo.

Algoritmo de retropropagación

Se sabe que la unidad básica de las redes neuronales es el *perceptrón* y que la unión de varias de ellas forman una red que permite resolver problemas de clasificación y regresión mayores. Generalmente, una red neuronal está formada por capas de varias neuronas. La primera, la capa de entrada, recibe los datos. Por ejemplo los síntomas de una posible enfermedad. Luego, una serie de capas, llamadas *capas ocultas*, procesarán los datos de entrada y una última, la *capa de salida*, devolverá la salida, en el caso del ejemplo planteado si el paciente tiene la enfermedad o no.

El deber del creador de una red neuronal es encontrar cuántas capas y qué cantidad de neuronas serán necesarias para que un algoritmo cambie los pesos y pueda transformar la entrada que haya recibido en la salida esperada.

Existen muchos algoritmos que cumplen la tarea de ajustar los parámetros de una red, pero el más popular es *Backpropagation* siendo este uno de los más usados y fáciles de entender. Existen diversas formas de medir el error. En específico, *Backpropagation* busca minimizar el error cuadrático medio producido por la capa de salida (Karpthy, 2022). Como se dijo, el error está dado por la siguiente fórmula:

$$Error = \sum_{k=1}^J \frac{1}{2} (ObservedValue_k - PredictedValue_k)$$

donde J es el número de neuronas de salida.

Por otra parte, el algoritmo necesita un conjunto de datos para que la red pueda aprender, es decir, contar con las entradas y las salidas esperadas. Al conjunto de pares de datos de entrada y salida (denotados por las variables de *ObservedValue* y *PredictedValue* en la fórmula anterior) se lo denomina *conjunto de entrenamiento*. No está demás decir que la cantidad de neuronas de la capa de entrada y de salida, deberá coincidir con las dimensiones del conjunto de datos de entrenamiento. En el diagrama de la Ilustración 6 se puede ver una arquitectura del perceptrón multicapa con 2 neuronas en la capa de entrada, 2 capas de 2 y 3 neuronas respectivamente y otra neurona en la salida.

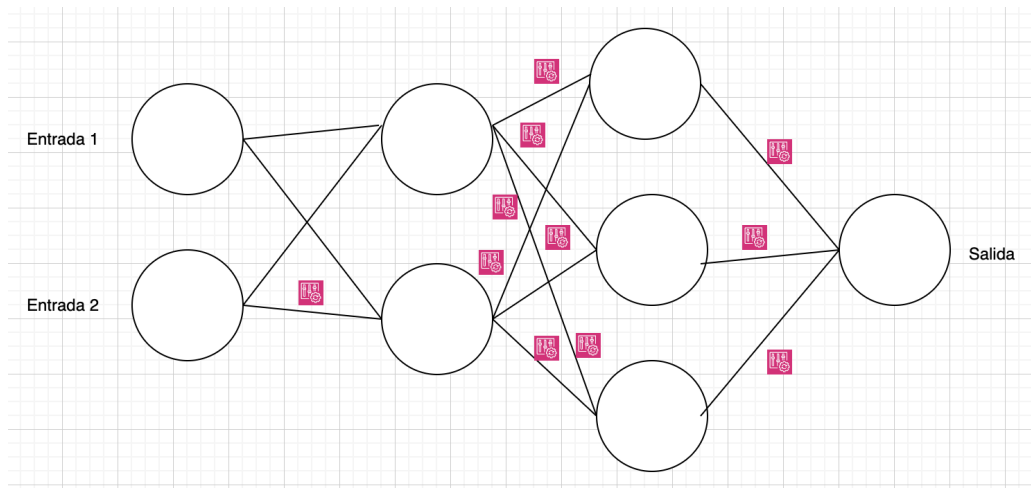


Ilustración 6: Esquema de un perceptrón multicapa, Fuente: Elaboración propia

Los pasos de este algoritmo son los siguientes:

1. *Inicialización de pesos*: En el inicio, todos los pesos de las conexiones entre las neuronas se establecen aleatoriamente o usando alguna técnica de inicialización. Son estos los que determinan la influencia que tiene cada conexión en la salida de una neurona.
2. *Paso hacia adelante (Forward Pass)*: Cuando se presenta una entrada a la red, esta se pasa a través de la red capa por capa. Cada neurona toma una combinación ponderada de las salidas de las neuronas en la capa anterior, las suma y pasa el resultado a través de una función de activación. Esto finalmente produce una predicción en la salida de la última capa.
3. *Cálculo del error*: Comparar la predicción realizada por la red con el valor real que debería ser. Esto es lo que se llama error o "pérdida" ya que mide cuán incorrecta es la predicción de la red para esa entrada en particular.
4. *Paso hacia atrás (Backward Pass)*: Aquí es donde comienza el algoritmo de backpropagation propiamente dicho. El objetivo es entender cómo los pesos de las conexiones entre las neuronas deben ajustarse para reducir el error. Para hacerlo, se calcula la contribución de cada peso al error total.
5. *Cálculo de gradientes*: El algoritmo calcula la derivada parcial del error con respecto a cada peso. Esto indica cómo cambiar cada peso para disminuir el error. Utiliza la

regla de la cadena para propagar el error desde la capa de salida hasta las capas anteriores.

6. *Actualización de parámetros*: Una vez que se calculan las derivadas, se puede ajustar los parámetros de la red. La idea es mover los parámetros en la dirección que reduzca el error. Esto se hace mediante un proceso de optimización como el visto en el descenso de gradiente. Los pesos se ajustan proporcionalmente a las derivadas calculadas, utilizando una tasa de aprendizaje, que puede o no variar. Esto también se conoce como la *regla Delta* ³
7. *Iteración*: Repetir los pasos 2 al 6 para muchas instancias del conjunto de datos de entrenamiento, una y otra vez, para que la red se ajuste gradualmente a los datos. Cada iteración mejora la red haciendo que los pesos se ajusten en la dirección que minimiza el error total.

Este proceso de ajustar los pesos en función del error se repite iterativamente hasta que la red neuronal pueda hacer predicciones con un error aceptablemente bajo en nuevos datos que no ha visto durante el entrenamiento. En definitiva, el algoritmo de backpropagation es la base matemática que permite a las redes neuronales aprender y mejorar su rendimiento a través de la optimización de sus conexiones ponderadas (BitBoss, 2020).

Proceso de Optimización

Tanto el algoritmo de retropropagación (*Backpropagation*) como el algoritmo de *descenso del gradiente* son conceptos fundamentales en el campo del aprendizaje profundo y el aprendizaje automático en general. Aunque están relacionados y a menudo se usan juntos en el entrenamiento de redes neuronales, se refieren a aspectos diferentes del proceso de optimización.

El descenso del gradiente es un método de optimización que busca encontrar el valor mínimo de una función mediante la iteración de ajustar los parámetros en la dirección opuesta al gradiente de la función en un punto dado. Es importante recordar que el gradiente es el vector que apunta en la dirección de mayor aumento de la función, por lo que moverse en la dirección opuesta al gradiente con pasos pequeños lleva al valor mínimo de la función. El algoritmo de descenso del gradiente se utiliza para ajustar los parámetros de un modelo de manera que se minimice la función de pérdida, la cual mide qué tan bien

³ La regla delta, en el contexto de redes neuronales y aprendizaje supervisado, se refiere al ajuste de los pesos de la red para minimizar el error de predicción. Específicamente, los pesos W se actualizan restando un valor proporcional a la derivada del error respecto a los pesos $E(W)$. Este ajuste se realiza multiplicando la derivada por una tasa de aprendizaje, que controla la magnitud del cambio. Matemáticamente, se expresa como: $W \leftarrow W - E(W)$

se ajusta el modelo a los datos. Por último, la tasa de aprendizaje controla el tamaño de los pasos tomados en la dirección del gradiente.

Cuando se menciona la retropropagación se refiere al proceso específico utilizado en el entrenamiento de redes neuronales para calcular los gradientes de la función de pérdida con respecto a los pesos de las conexiones de la red. Implica el cálculo eficiente de las derivadas parciales de la función de pérdida en función de los pesos y las entradas de la red. La retropropagación se basa en la regla de la cadena del cálculo diferencial, que permite descomponer el gradiente de la función de pérdida en términos de gradientes más pequeños de las capas de la red. Esto permite calcular los gradientes para ajustar los pesos de cada capa de la red utilizando el algoritmo de descenso del gradiente.

La retropropagación es el proceso que permite calcular los gradientes necesarios para el descenso del gradiente en el contexto de redes neuronales. La retropropagación calcula cómo cada peso de la red contribuye al error final y cómo esos errores se propagan hacia atrás en la red. Una vez calculados estos gradientes, el descenso del gradiente se aplica para ajustar los pesos y minimizar la función de pérdida.

El algoritmo de retropropagación es una técnica específica dentro del entrenamiento de redes neuronales que calcula gradientes, mientras que el algoritmo de descenso del gradiente es un enfoque general de optimización utilizado en diversos problemas de aprendizaje automático, incluyendo el entrenamiento de redes neuronales (García Serradilla, 2023).

Los algoritmos mencionados en esta sección se desarrollaron en las Notebook Jupyter junto con una implementación ad-hoc de un perceptrón multicapa⁴.

¿Qué es Deep Learning ?

Cuando las redes de neuronas tienen más de una capa oculta se las conoce como redes profundas. Estos modelos dan lugar a toda una rama del machine learning llamada deep learning. Hoy en día, se ha evolucionado al punto de pensar las arquitecturas de estos sistemas en capas y no en neuronas individuales, por lo que generalmente se trabaja con varias de ellas, entrando automáticamente en la categoría de deep learning. Las IAs generativas y en específico los LLMs pertenecen a esta categoría (Asaxena, 2018).

Cuando se trabaja con redes profundas se presentan algunos inconvenientes, entre ellos:

⁴ En el siguiente enlace encontrara el código junto con la explicación de una implementación ad hoc de un perceptrón multicapa utilizando el kernel de Jupyter Notebooks https://github.com/AlejoTorres2001/AI/blob/windows/ProyectoDeGrado/Anexos/AnexoII-Perceptron%26MLP-adhoc-implementation.ipynb?short_path=2e18507

- *El proceso de entrenamiento se vuelve arduo* y requiere de mucha “prueba y error”. Al no existir una única forma de hacerlo, existen diversos enfoques que pueden resolver mejor o peor el problema a trabajar. Si bien existe un conjunto de buenas prácticas, no existe un “libro dorado” que de la solución exacta.
- *La elección de la arquitectura* se basa plenamente en prueba y error, depende netamente del problema a resolver y de la cantidad de datos disponibles.
- *La “explicabilidad”* del modelo se pierde, es decir no se sabe con certeza cómo toma las decisiones y en base a qué lo hace. Éste es el mayor impedimento a la hora de utilizar el aprendizaje profundo y las redes de neuronas en campos como la medicina, justicia y seguridad, por ejemplo, donde técnicas como *Random Forest* continúan siendo ampliamente usadas debido a que estos modelos cuentan con buenas características de explicabilidad.
- *El volumen de datos necesario* aumenta en gran medida. Para entrenar las arquitecturas de deep learning es necesario un número considerable de registros; mientras más profunda la red mayor será la cantidad de datos necesarios (García Serradilla, 2023).

IA's Generativas

La Inteligencia Artificial Generativa (*IAG*) es una rama de la inteligencia artificial que se enfoca en la generación de contenido original a partir de datos existentes. Nombradas por *Yann LeCun* en 2014 como “*la idea más interesante en Machine Learning en los últimos diez años*”, estas técnicas son capaces de generar imágenes, música, texto y demás contenido con un potente componente creativo en ellos y simulan ser creados por humanos. Se basan en modelos y algoritmos que pueden aprender a capturar y modelar la distribución subyacente de un conjunto de datos de entrenamiento para luego generar instancias nuevas y auténticas que se asemejen a esos datos. La forma más popular de construir y entrenar una Inteligencia artificial de generación de contenido es a través de las Redes Neuronales Generativas Adversariales (*GAN*) (Granieri, 2023).

Una *GAN* consiste en dos redes neuronales: el *generador* y el *discriminador*. El generador toma ruido aleatorio como entrada y genera datos sintéticos, mientras que el discriminador toma datos reales y datos producidos por el generador y trata de distinguir entre ellos. Ambas redes son entrenadas en un proceso competitivo: el generador busca generar datos que engañen al discriminador, mientras que el discriminador intenta mejorar en su capacidad de discernir entre datos reales y generados. Un bloque fundacional en este tipo de inteligencias artificiales son los *espacios latentes*. A continuación se detalla en profundidad este concepto, la forma de crearlos y sus variantes.

Autoencoders

Un autoencoder es una arquitectura de red neuronal perteneciente a la categoría de modelos no supervisados. Están pensados para la extracción y compresión de características, así como la posterior reconstrucción de los datos originales a partir de estas representaciones latentes; todo esto sucede en un proceso que se denomina *entrenamiento auto-supervisado*. Estos modelos son utilizados como una técnica de *reducción de la dimensionalidad* y compresión de datos, pero con pérdida.

A nivel técnico, un autoencoder consta de dos componentes principales: el codificador (*encoder*) y el decodificador (*decoder*). Estos dos componentes trabajan juntos para aprender una representación más compacta de los datos de entrada sin etiquetas, permitiendo que el modelo capture las características más significativas y subyacentes del dataset.

Se puede resumir el comportamiento de un autoencoder en un alto nivel de la siguiente forma: primero, el codificador toma los datos de entrada y los transforma en un espacio de características de menor dimensión, conocido como *espacio latente*. Este proceso de codificación implica aplicar transformaciones lineales y no lineales a los datos originales para capturar sus patrones más relevantes. La reducción de la dimensión en el espacio latente implica una compresión de los datos. Cabe mencionar que se trata de una compresión con pérdida de información. Sin embargo se entiende que el espacio latente es capaz de capturar los atributos más relevantes.

Segundo, el decodificador, toma la representación en el espacio latente generada por el codificador y la reconstruye de nuevo en el espacio original de alta dimensión. Este proceso implica transformaciones inversas de las aplicadas por el codificador. El objetivo es que la reconstrucción sea lo más cercana posible a los datos originales.

Durante el entrenamiento, la función de pérdida del autoencoder compara los datos de entrada con las reconstrucciones generadas por el decodificador. El objetivo es minimizar la diferencia entre los datos originales y las reconstrucciones, lo que conduce a que el autoencoder aprenda a capturar las características más relevantes en su espacio latente. Son muy útiles en una gran variedad de aplicaciones, entre ellas, la reducción de la dimensionalidad de datos de alta complejidad y la eliminación de ruido de señales y generación de datos nuevos y similares a los de entrada. Es por esto último que se conocen también como la primer arquitectura de red de generación de contenido (Bank et al., 2021).

Un autoencoder no se diferencia mucho de un perceptrón multicapa como los que ya se explicó anteriormente. Se entrenan a través del método de descenso del gradiente usando un algoritmo de backpropagation para ajustar los pesos (A. D. Díaz Álvarez et al., 2023).

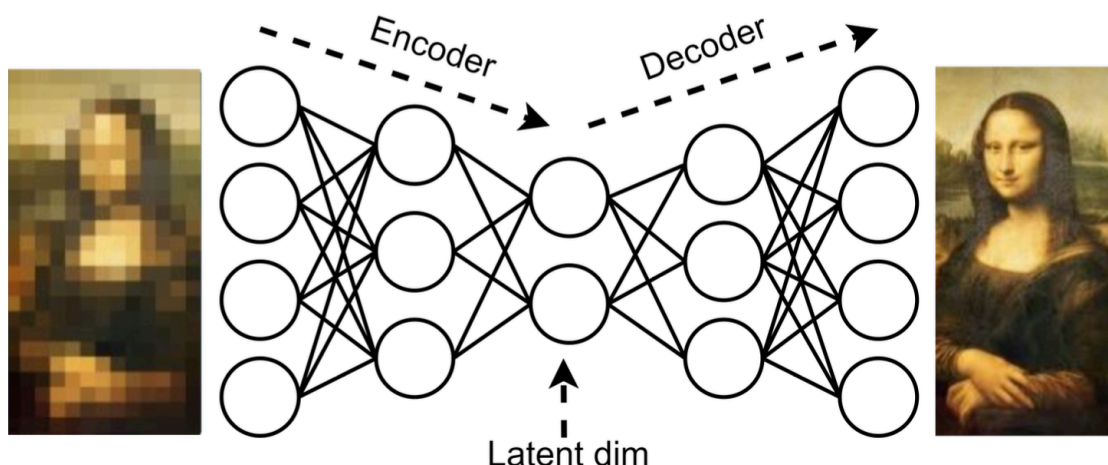


Ilustración 7: Esquema simplificado de un Autoencoder Fuente: (A. D. Díaz Álvarez et al., 2023, UPM)

El entrenamiento de un autoencoder involucra el aprendizaje simultáneo del codificador y del decodificador en función de un objetivo común el cual es comprimir los datos de entrada en una representación latente y luego reconstruirlos con la mayor fidelidad posible. Más en profundidad, los pasos involucrados son los siguientes:

- **Codificación:** Durante el entrenamiento, se presentan datos de entrada al codificador que es la parte inicial del autoencoder, donde se aplican transformaciones lineales y no lineales a los datos para mapearlos a una representación en el espacio latente de menor dimensión. El término espacio latente se utiliza para describir los parámetros (pesos y desviaciones) que resultan de esta etapa, se entiende, que los mismos están cargados con las características más relevantes de los datos a los cuales fueron expuestos. Como las redes neuronales tienden a tener una forma de embudo, supone además una reducción de las dimensiones en comparación con los datos originales.
- **Cálculo de la Pérdida de Reconstrucción:** Luego, la representación latente generada por el codificador se pasa al decodificador. El decodificador intenta generar una reconstrucción que sea lo más similar posible a los datos originales, para esto, se calcula la pérdida de reconstrucción, que mide la discrepancia entre los datos originales y las reconstrucciones generadas por el decodificador con el objetivo de minimizar esta pérdida.
- **Retropropagación y Actualización:** El gradiente de la pérdida de reconstrucción se propaga hacia atrás a través del decodificador y el codificador, lo que permite ajustar los pesos y sesgos de ambas partes de la red. El codificador aprende a generar representaciones latentes que sean útiles para la reconstrucción precisa en el decodificador.

Gracias a que se fuerza el aprendizaje de las características más relevantes, elementos como el ruido o las distorsiones, tienden a ser eliminadas automáticamente (A. D. Díaz Álvarez et al., 2023).

Limitación de los Autoencoders

Los autoencoders tienen mucho potencial, pero en la generación de *datos sintéticos* no son muy eficientes. El principal problema se debe a que el espacio latente generado no es continuo, sino que se encuentra formado por regiones separadas unas de otras las cuales agrupan las características de ejemplos similares. Entre estas regiones no hay un espacio continuo de representaciones intermedias coherentes. Aunque técnicamente sí exista una continuidad entre regiones, la misma carece de sentido para la generación de ejemplos, así que para el objetivo en concreto, se puede considerar que no es continuo.

Para comprender mejor esto se puede pensar en que se asigna la tarea de reconocer las características más relevantes de diferentes tipos de archivos de texto a un autoencoder. El mismo agrupa dichas características en zonas específicas del espacio latente. Pregúntese ahora, ¿de verdad no hay un espacio intermedio entre un archivo de código fuente en C++ y un archivo de código fuente en Python? ¿o entre un archivo HTML y un archivo XML? Intuitivamente se puede reconocer características y similitudes entre los tipos de archivos como su estructura, su formato o la similitud entre lenguajes de alto nivel. El problema está en que los autoencoders generan un espacio latente continuo sin una estructura específica en términos de distribuciones probabilísticas. Esto significa que las interpolaciones entre puntos en el espacio latente (como pensar en la relación entre el archivo en C++ y el archivo en Python o el archivo HTML y el archivo XML) no tienen una interpretación semántica directa y no se pueden realizar operaciones probabilísticas confiables, como la generación de nuevas muestras de manera coherente. Esto hace que la interpolación entre ejemplos sea imposible. En definitiva, si el espacio intermedio no es continuo, las salidas del decoder no son realistas. Se verá a continuación una variante de esta técnica que resuelve el problema, especialmente formada para la generación de contenido (A. D. Díaz Álvarez et al., 2023).

Autoencoders Variacionales

Los Autoencoders Variacionales (VAEs) abordan la limitación de los autoencoders al introducir un componente probabilístico en el espacio latente. Estos son una extensión de los autoencoders tradicionales donde utilizamos conceptos de inferencia probabilística para modelar las distribuciones en el espacio latente. Esto hace que el espacio latente en VAEs tenga una estructura probabilística, lo que permite realizar interpolaciones suaves y operaciones generativas más flexibles. En definitiva, se cuenta con un *espacio latente*

continuo. En los VAEs, el espacio latente está definido por dos vectores, uno de *medias* y el otro de *desviaciones típicas*, formando un vector de distribuciones normales.

$$(N(\mu_1, \sigma_1), \dots, N(\mu_n, \sigma_n))$$

Con este artilugio matemático es que se logra que el *decoder* asocie áreas enteras y no solo puntos concretos a ligeras variantes de la misma salida. Esto, en definitiva, genera un espacio mucho más interpolado y suave.

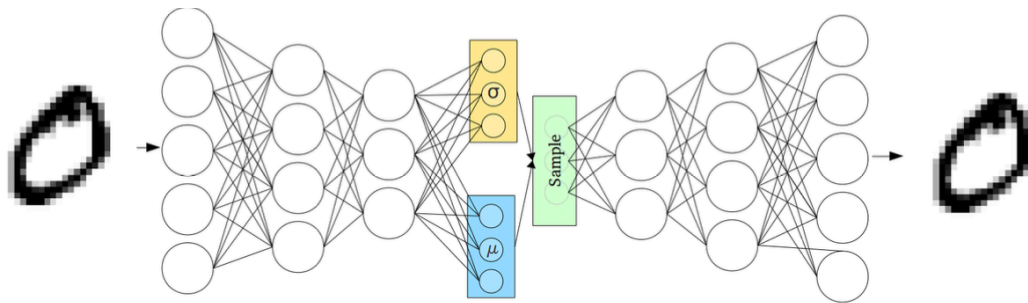


Ilustración 8: Esquema de un VAE, Fuente: (A. D. Díaz Álvarez et al., 2023, UPM)

Al hacer esto se necesita ajustar las funciones de pérdida individualmente, utilizando una función de pérdida o error convencional para medir la distancia del objeto generado y una divergencia KL o divergencia de Kullback-leibler entre la distancia latente aprendida y la distribución previa, que actúa como término de regularización (A. D. Díaz Álvarez et al., 2023).

La *divergencia KL* es una medida utilizada en teoría de la información para medir la diferencia entre distribuciones de probabilidad. Dicho de otra forma es el mecanismo que se debe utilizar para poder cuantificar una medida de error sobre este nuevo espacio latente continuo, es decir, una medida de cuán bien se ajusta la distribución latente aprendida a la distribución latente real de los datos.

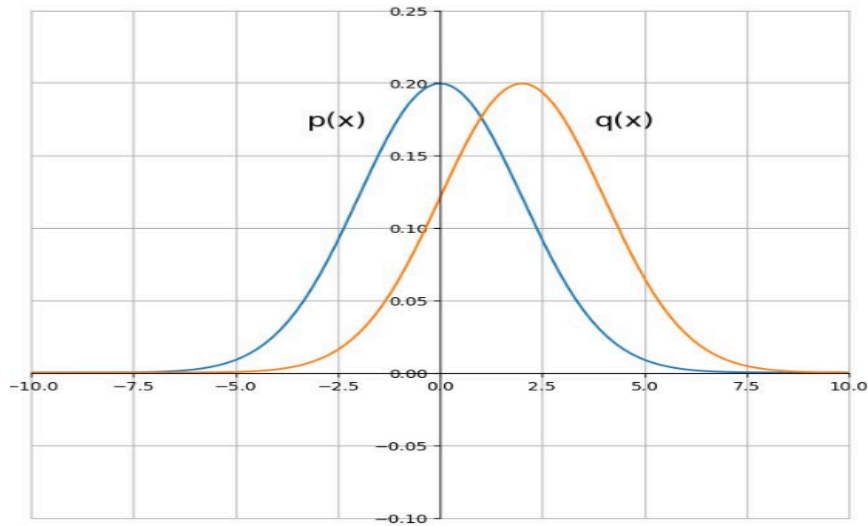


Ilustración 9: Dos distribuciones Normales comparadas en el mismo sistema de ejes. Fuente: (A. D. Díaz Álvarez et al., 2023, UPM)

Se puede pensar que $P(x)$ es la distribución normal conocida y $Q(x)$ es la distribución, también normal pero aprendida por el VAE. Se divide la función de pérdida en dos términos, el término de reconstrucción y el término de regularización, donde entra a jugar la divergencia KL. Se le llama término de regularización porque trata de evitar que el modelo colapse en un espacio latente poco informativo.

$$\mathcal{L}(y, \hat{y}) = \mathcal{L}_r(y, \hat{y}) + \mathcal{L}_{KL}(y, \hat{y})$$

A modo de resumen, se pueden distinguir las capacidades y limitaciones de los VAE. Estos modelos son capaces de generalizar gracias al uso de una función de densidad, lo que les permite generar datos indistinguibles de los de entrada y crear nuevos datos sin copiar directamente los ejemplos. Sin embargo, las principales desventajas incluyen la baja calidad de los datos generados, lo que limita su uso en aplicaciones reales, y la poca diversidad en los datos generados debido a su dependencia, entre otros factores, de la función de densidad aprendida.

Con esto se cubren las bases teóricas que dan origen a la generación de contenido usando inteligencia artificial. Ahora se hablará de una rama de la IA y la lingüística computacional que se encarga de trabajar la interacción entre las computadoras y el lenguaje humano. Los avances en IAs generativas, junto con las técnicas de modelado secuencial que fueron evolucionando con los años resultaron en la arquitectura que dará sustento al producto en torno al cual gira este trabajo: los *LLMs* y la arquitectura *Transformer*.

¿Qué es el Procesamiento del Lenguaje Natural?

El procesamiento del lenguaje natural o *natural language processing (NLP)* se ha desarrollado a través de dos grandes campos: por un lado se cuenta con aquellas técnicas que se apoyan más en la codificación manual de todas las reglas gramaticales que resulten interesantes para la tarea en cuestión; por el otro, sobre las que se hará especial énfasis, se encuentran aquellas técnicas que se basan en el análisis de grandes *corpus* de texto con el objetivo de que todas estas reglas sean aprendidas, es decir técnicas de machine learning. Siguiendo este enfoque, el *NLP* es el área de la *IA* que cubre la comprensión y generación de lenguaje natural. No está limitado únicamente al texto, también abarca lenguaje de señas y lenguajes hablados, aunque está mucho más avanzado en texto escrito por la cantidad de datos disponibles. Uno de sus objetivos es conseguir que los computadores trabajen con nuestro lenguaje en lugar de trabajar nosotros con el suyo (A. Díaz Álvarez, 2023)

Podemos dividir el NLP en dos áreas de trabajo bien diferenciadas:

- *Natural language generation (NLG)*: Generación de lenguaje a partir de representaciones internas. Puede considerarse como un componente de traducción entre los datos y el lenguaje natural y se subdivide en múltiples tareas: planificación del discurso, selección léxica, etc.
- *Natural language understanding (NLU)*: Comprensión lectora de las máquinas. Transforman el lenguaje natural en una representación adecuada para la máquina. Normalmente, requiere diferentes niveles de análisis: morfológico, sintáctico, semántico, etc. (Chomsky, 1955).

Como ya se ha explicado, las redes neuronales son una potente herramienta las cuales parecen ser unas excelentes candidatas para la tarea de captar las características y reglas de un corpus de texto. Sin embargo, se sabe bien que una red neuronal, por su naturaleza matemático-estadística, solamente puede operar con números, no con palabras o caracteres. El primer gran desafío a resolver es encontrar una forma de *codificar* las palabras en números operables matemáticamente.

Se podría pensar en una primera aproximación a este problema, tratando de utilizar el código ASCII de cada carácter que conforme una palabra. Esto es, en efecto, una forma de codificación para representar textos que puede utilizarse.

Es importante aclarar que la codificación puede darse a nivel de palabra o a nivel de carácter o en un nivel intermedio, conocido como sub-palabra o *sub-word*. Cada forma de codificación tiene sus ventajas y desventajas. Por ejemplo, codificar a nivel de carácter da mayor flexibilidad, permitiendo al modelo incluso generar palabras nunca antes vistas. Pero si lo que se busca es generar texto realista, sería mejor un enfoque más abarcativo,

entrenando al modelo sobre palabras ya construidas. En definitiva, dependerá de la tarea en cuestión. Como un punto aparte, los grandes modelos como GTP-2 Y GPT-3 usan una codificación sub-word. Es a cada uno de estos bloques (ya sean caracteres, palabras o subpalabras) lo que se conoce como *tokens* (la unidad a procesar) y al proceso de crearlos, *tokenización* (Santana Vega, 2020).

Retomando la idea de codificar los caracteres utilizando su representación ASCII, sucede que con este tipo de codificación se está atentando contra el problema en sí. A modo de explicación: se piensa en un caso donde se cuenta con un conjunto de datos de personas, entre estos atributos, se cuenta con la edad de los individuos, algunos tienen 15 años, otros 20 y otros 30 años. Una red neuronal que ingestaría estos datos aprendería las diferencias de un valor y otro sobre la dimensión de ese atributo, es decir, entendería que 30 es el doble de 15 y que 20 es mayor que 15 pero menor que 30. Se puede ver más claramente si se grafica en una recta.

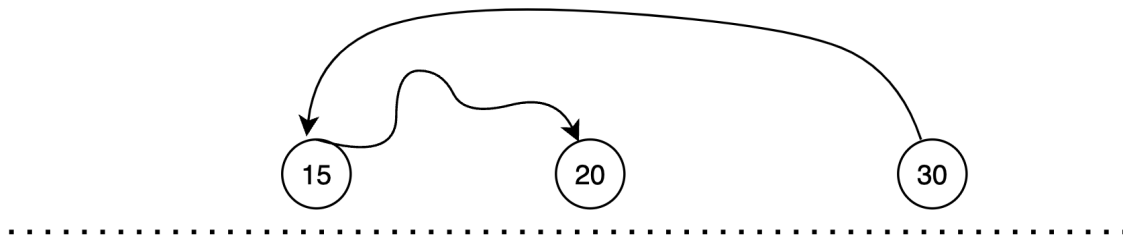


Ilustración 10: Recta numérica con ejemplo de edades. Fuente: Elaboración propia

Este tipo de información es la que habitualmente utiliza una red neuronal para resolver el problema en cuestión. Sin embargo, en este caso, no es lo que se busca. Como la red únicamente comprende números, no sabe que ésto está relacionado con la edad de una persona, o que, por ejemplo, se trate de etiquetas que representan caracteres o incluso palabras. Tal sería el caso en el que se asigna el número 15 para representar la palabra “teléfono”, el número 20 para “tostador” y el 30 para “animal”.

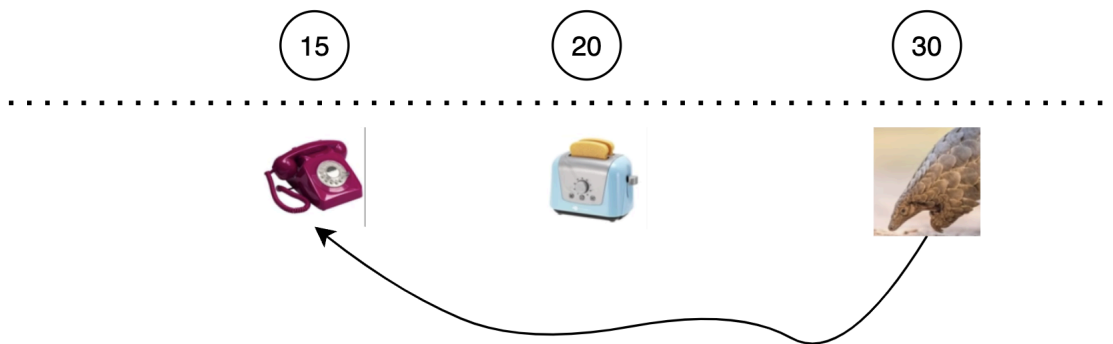


Ilustración 11: Ejemplos de palabras con relación a un identificador numérico. Fuente: Elaboración propia

Si se usara esta forma de codificación, la red aprendería que la palabra “animal”, representada por el 30 es mayor que la palabra “teléfono”, representada por el 15 y esto no tiene sentido. Una solución a este problema se basa en vectorizar la palabra creando representaciones vectoriales de cada palabra, usando un vector del tamaño del *vocabulario*, llenándolo de 0 excepto en la posición que simbolice dicha palabra. Esta forma de codificación de datos nominales o no numéricos se conoce como *one-hot encoding*. Si en el ejemplo la palabra “teléfono” tenía un id 15, entonces se busca la componente 15 del vector que la representa y se marca un 1. Esto resuelve el problema de la distancia entre palabras. Remitiéndose al ejemplo anterior donde solo se contaba con tres palabras en el vocabulario, la dimensión de dichos vectores resultan de tres componentes por lo que se podría graficar. Cada uno ocuparía su propia dimensión y se estaría trabajando con versores con una distancia constante entre ellos.

Sin embargo, esto constituye un gasto enorme de memoria, para representar 20 palabras pertenecientes a un vocabulario de 1000 palabras tendríamos una *matriz de 20x1000* y en su mayoría estaría ocupada de ceros. Sería el equivalente de almacenar en todo un diccionario, la definición de sólo una palabra.

De momento se asume que la distancia entre todas las palabras es la misma, pero resulta intuitivo pensar en que existen palabras que deberían estar más cercanas unas de otras, como por ejemplo los meses del año o los días de la semana, ~~o cierta gama de electrodomésticos~~. Por lo que se debe buscar una forma de capturar la información semántica de cada palabra e infundirla en estos vectores de manera tal que palabras relacionadas se agrupan cercanas unas de otras en el espacio vectorial.

Esta solución viene de la mano de un concepto llamado *embeddings* (Santana Vega, 2020).

Embeddings de palabras

Como seres humanos se entiende que hay palabras que conceptualmente son más similares a otras palabras. Esta información resultaría interesante poder modificarla en los algoritmos de NLP. Esto es posible y puede lograrse compactando la información de los vectores creados con la técnica de one-hot encoding, usando el concepto anteriormente visto de autoencoders para generar un espacio latente que no solo codifique las palabras, sino que capture su información conceptual y semántica. Además, se ha visto que los espacios latentes reducen la dimensionalidad de los datos los cuales serán agrupados de la forma que permita a la red acercarse lo más posible a la solución óptima de la tarea en cuestión. Gracias a ésto se obtienen representaciones vectoriales en un espacio de alta dimensionalidad, permitiendo operar vectorialmente las palabras. En el siguiente ejemplo se puede ver cómo las representaciones vectoriales de palabras pueden ser operables entre sí. Estas pueden capturar relaciones semánticas. En este caso, la relación de género se refleja en el espacio vectorial mediante la diferencia entre los vectores de "rey" y "hombre" que es similar a la diferencia entre los vectores de "reina" y "mujer" (Álvarez Díaz, 2023), resultando ambas en la representación vectorial para “género”, demostrando que existe una relación semántica entre estas palabras.

$$V(rey) - V(hombre) \approx V(reina) - V(mujer)$$

En última medida, un embedding de palabras es una matriz de vectores de palabras apilados.

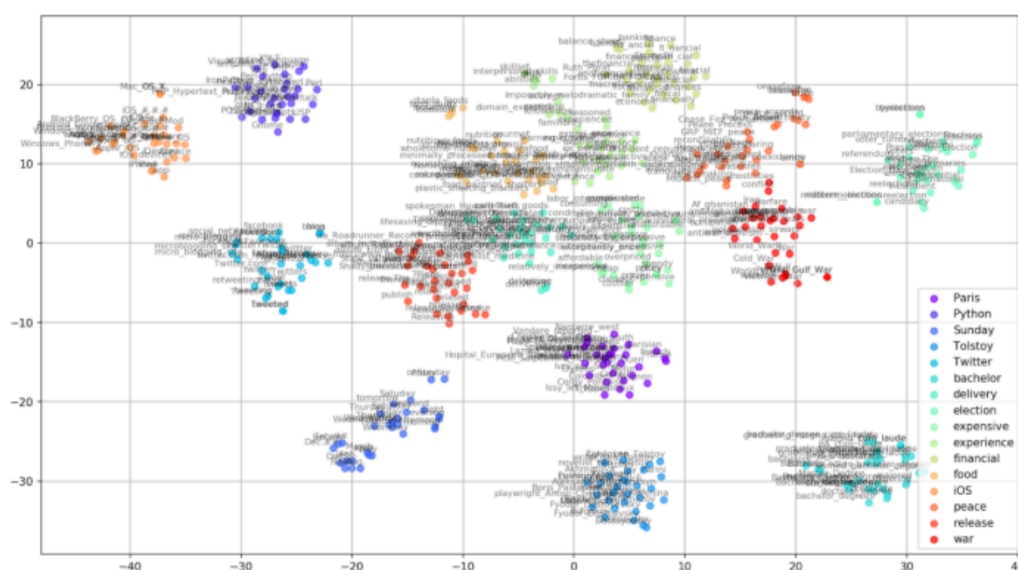


Ilustración 12: Clusters de palabras, Fuente: (A. D. Díaz Álvarez et al.,2023,UPM)

Los embeddings de palabras se ven beneficiados en gran medida por la técnica de entrenamiento conocida como aprendizaje por transferencia o, en inglés, *Transfer Learning*. El aprendizaje por transferencia es una técnica que consiste en aprovechar el conocimiento adquirido por un modelo entrenado en una tarea específica para mejorar el rendimiento en otra tarea relacionada. En lugar de entrenar un modelo desde cero para cada tarea individual que se desee, se pueden reutilizar y adaptar los conocimientos ya adquiridos en tareas previas. Esto es especialmente valioso en situaciones donde se tiene acceso a un gran conjunto de datos y un modelo pre entrenado en una tarea general. La idea básica es que, al entrenar inicialmente un modelo en una tarea amplia y luego ajustarlo para tareas específicas, se puede beneficiar de las características y representaciones aprendidas durante el entrenamiento inicial de la tarea más general (Brownlee, 2017).

Por lo general, es mediante el transfer learning que podemos utilizar los embeddings más conocidos como lo son *Word2Vec* o *Glove*⁵. Como ya se ha visto, las operaciones algebraicas entre vectores permiten de alguna manera jugar con sus significados.

Cuando se habla de embeddings existen dos arquitecturas centrales, que dependen de su cometido: *Skip-Gram* y *continuous bag of words* (CBow). Con *Skip-Gram*, el objetivo es capturar las relaciones semánticas y sintácticas entre las palabras al predecir las palabras circundantes de una palabra objetivo en un contexto dado. En este enfoque, el proceso se revierte en comparación con la forma en que se presenta el texto. Dado un corpus de texto, se considera una palabra objetivo y se intenta predecir las palabras circundantes o de contexto. Por ejemplo, en la oración "El gato está durmiendo en el sofá", si tomamos "durmiendo" como palabra objetivo, Skip-Gram tratará de predecir "El", "gato", "está" y "en" como palabras contextuales.

Con *Continuous Bag of Words* (CBow), el objetivo es predecir una palabra objetivo a partir de un contexto dado, tomando en cuenta un número arbitrario de palabras antes y después de la palabra objetivo dentro de la secuencia. En CBow, dada la oración "El gato está durmiendo en el sofá" y si elegimos "El", "gato", "está" y "en" como palabras de contexto, entonces el modelo intentará predecir "durmiendo" basándose en las palabras circundantes (Álvarez Díaz, 2023).

⁵ Cuando se menciona a Word2Vec o Glove se habla de los modelos de incrustaciones de palabras en sí mismos, y no a las técnicas utilizadas en su creación, las cuales comparten el mismo nombre. La siguiente es una guía, la cual no es de mi autoría, sobre como descargar y utilizar el embedding Word2Vec usando la api de gensim <https://github.com/harmanpreet93/load-word2vec-google>

Si desea profundizar en el tema puede revisar el Notebook Jupyter donde el autor implementa ad-hoc un embedding de palabras⁶.

Redes neuronales aplicadas al NLP

Las redes neuronales en el procesamiento del lenguaje natural se enfrentan a un desafío particular: la información no se presenta como datos independientes, sino como secuencias ligadas temporalmente. Esto significa que los datos están profundamente conectados con el concepto de tiempo. Un ejemplo sencillo es intentar recitar el abecedario. Resulta fácil porque lo aprendemos de pequeños. Sin embargo, hacerlo al revés es mucho más difícil, a menos que se haya practicado repetidamente. Los humanos procesan la información de esta forma, lo que se conoce como memoria secuencial.

Pensando en la forma en que trabaja el cerebro, miles de millones de neuronas se interconectan sin una dirección única. Los seres humanos razonan usando información previa, es decir una decisión que se toma en un instante de tiempo no se basa únicamente en lo que se ha percibido en ese momento, sino también en eventos previos. Hasta ahora sólo se han analizado redes de neuronas que entran en la categoría de *feed-forward*, es decir, aquellas redes neuronales donde la información se mueve en una sola dirección: desde las capas de entrada, a través de las capas ocultas, y finalmente hacia las capas de salida. En este tipo de arquitectura, las conexiones entre las neuronas no forman ciclos o bucles, lo que significa que los datos no regresan a una neurona anterior en el proceso. Un gran impedimento de éstas es que no pueden manejar información secuencial variable. En otras palabras, para producir la salida sólo se puede usar información de la entrada actual ya que no cuentan con la capacidad de memorizar entradas pasadas para predicciones.

A partir del uso de Autoencoders para generar espacios latentes se ve cómo interactúan las herramientas y técnicas de cómputo del Deep Learning con la disciplina del Procesamiento del Lenguaje Natural. Sin embargo, existen arquitecturas de redes neuronales que de alguna manera se especializan en el tratamiento de información de naturaleza secuencial, como lo son las secuencias de palabras que terminan por formar los corpus de texto. Estas son las redes neuronales que siguen una arquitectura *recurrente*.

⁶ En el siguiente enlace podrá encontrar el código junto con la explicación paso por paso para crear una implementación adhoc de un modelo de incrustaciones de palabras
<https://github.com/AlejoTorres2001/AI/blob/windows/ProyectoDeGrado/Anexos/AnexoIV-Embedding%20creation%20and%20visualization.ipynb>

Redes LSTM

Las Redes LSTM (Long Short-Term Memory) o Memorias de Largo y Corto Plazo son un tipo de red recurrente utilizada para abordar el problema de las dependencias a largo plazo en el procesamiento de secuencias. Estas se diseñaron para superar las limitaciones de las unidades recurrentes tradicionales, que tenían dificultades para capturar relaciones a largo plazo debido al desvanecimiento del gradiente (García Serradilla, 2023).

Estas redes pueden tener una memoria de largo plazo tanto como de corto plazo ya que cuentan con la capacidad de añadir o eliminar la información de su memoria.

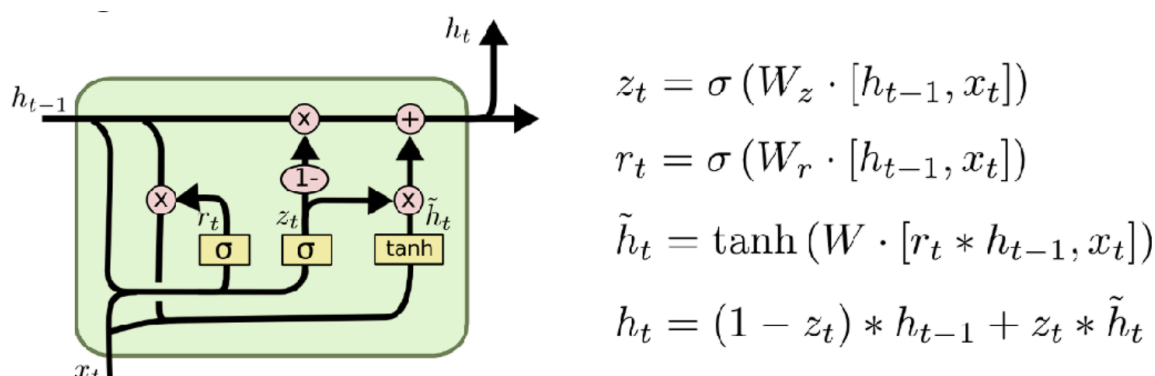


Ilustración 13: Esquema de una neurona LSTM. Fuente: (García Serradilla F., 2023, UPM)

La celda LSTM, tiene una entrada adicional conocida como el estado de la celda, que es fundamental para su funcionamiento. El estado de la celda actúa como una banda transportadora, permitiendo agregar o eliminar información a lo largo del tiempo. Este proceso se controla mediante compuertas, que son elementos clave en la LSTM. Las compuertas son pequeñas redes neuronales que regulan el flujo de la información. Cuando una compuerta está abierta, permite que la información pase libremente, y cuando está cerrada, bloquea su flujo. Cada compuerta se compone de una red neuronal que procesa la entrada, una función sigmoide que transforma la salida de la red neuronal en un valor entre 0 y 1, y un elemento multiplicador que ajusta la información entrante en función de ese valor.

La primera compuerta, (forget gate), determina qué parte de la información anterior debe descartarse. Toma el estado oculto anterior y la entrada actual, los procesa, y luego pasa el resultado por la función sigmoide. Esto genera un vector que decide, para cada elemento del estado anterior, si debe ser olvidado o retenido. Si un valor es cercano a 0, esa parte de la información será descartada; si es cercano a 1, se retendrá en el estado de la celda. Por ejemplo, si se está analizando texto y se detecta un cambio en el sujeto de la oración de plural a singular, la forget gate puede ser entrenada para eliminar la referencia plural que ya no es relevante.

La siguiente compuerta es la puerta de entrada (*input gate*), que permite añadir nueva información al estado de la celda. De manera similar, se procesan el estado oculto anterior y la entrada actual, se pasan por una función sigmoideal, y el resultado es un vector que decide qué parte de la nueva información debe ser almacenada. Este proceso permite actualizar la memoria de la LSTM, por ejemplo, para reflejar que el sujeto ahora es singular.

Con la información de las puertas de olvido e ingreso, se actualiza el estado de la celda. Primero, la información irrelevante se elimina multiplicando el estado anterior por el vector generado por la puerta de olvido. Luego, se genera un vector de candidatos para formar parte de la nueva memoria, que se filtra a través de la puerta de entrada, y se añade al estado de la celda, actualizando así su contenido. Finalmente, el nuevo estado oculto se calcula utilizando la puerta de salida (*output gate*). Esta puerta filtra el estado de la celda para decidir qué parte de la información debe ser enviada al siguiente paso temporal o utilizada como salida. Primero, el estado de la celda se pasa por la función tangente hiperbólica para garantizar que sus valores estén en el rango de -1 a 1. Luego, la puerta de salida aplica su filtro para producir el nuevo estado oculto, que representa la memoria a corto plazo de la LSTM en ese momento (Hochreiter & Schmidhuber, 1997).

¿Qué es un LLM?

Un enorme modelo de lenguaje o *large language models* (LLM) es un tipo de modelo de procesamiento de lenguaje natural que utiliza el aprendizaje automático para entender y generar lenguaje humano. Son versátiles y pueden ser adaptados a diversas tareas de procesamiento de lenguaje natural, como traducción automática, resumen de texto, respuesta a preguntas, análisis de sentimientos, generación de código y mucho más. Esto se logra ajustando las capas finales del modelo para que se adapten a una tarea específica. Tienen aplicaciones en una amplia gama de industrias, desde asistentes virtuales y chatbots hasta análisis de texto a gran escala.

Antes de su existencia, un problema de clasificación como por ejemplo el análisis de sentimientos de una serie de comentarios de una página web, se podía resolver tomando un diccionario con palabras positivas y otro con palabras negativas y analizando la polaridad de las palabras para asignar puntajes a cada frase. Este es un análisis insuficiente si se tiene en cuenta cuestiones del lenguaje como son la ironía y el doble sentido. Luego, gracias al Deep Learning, se entrenaban redes neuronales que aprendían la relación entre las palabras (entrada) y su sentimiento (salida). Como se trata de una tarea supervisada, es necesario tener un dataset concreto para cada tipo de tarea, sin importar que tan parecidas sean unas con otras. Es decir, por ejemplo, si en principio se clasificaba la frase en positivo y negativo y después se quería valorar con una escala del 1 al 5, se debía contar con un

dataset distinto y volver a entrenar un modelo desde el inicio. Una nueva tarea implicaba un nuevo dataset y un entrenamiento completo (Santana Vega, 2022).

Para resolver esta dependencia y con la llegada de los modelos Transformers, surgieron los LLMs que se independizan de la necesidad de contar con modelos específicos para cada tarea en particular. La idea subyacente consiste en tener modelos pre-entrenados que supieran comprender el lenguaje, para luego entrenarlos usando una menor cantidad de datos para darle un ajuste final al modelo según la tarea específica. Esto se conoce como *fine-tuning*. Este cambio de paradigma da origen al aprendizaje autosupervisado donde ya no se necesitan etiquetar grandes conjuntos de datos, sino que se toman pequeños datasets donde se ocultan ciertas palabras o se separan frases para que el modelo las complete. La facilidad con la que ahora se pueden crear conjuntos de datos, impulsó a que la tendencia fuera escalar el tamaño de estos a razón de órdenes de magnitud, lo que dio origen a la era de los LLMs.

El término Large se refiere a la gran cantidad de parámetros que estos modelos tienen. Los LLMs tienen cientos de millones, o incluso miles de millones, de parámetros entrenables. En 2020 la empresa OpenAI decidió crear el modelo GPT-3 (Generative pre-trained transformers) con un tamaño 117 veces mayor que GPT-2. GPT-3 se convirtió en la primera IA en imitar la forma en la que los humanos se comunican. Fue justamente durante el entrenamiento de este modelo que se confirmó que, al escalarlos a grandes dimensiones, en su tarea de completar texto, aprendían otras tareas como traducción, análisis de sentimiento, generación de código, etc. A continuación, se explicará la arquitectura que da soporte a estos modelos.

Arquitectura Transformer

La arquitectura Transformer constituye un hito fundamental en el campo del procesamiento del lenguaje natural. Este modelo de deep learning se basa en un mecanismo esencial llamado atención (*attention*), el cual permite a la red modelar las relaciones entre las palabras en una secuencia pero evitando tener que procesar una palabra a la vez. A diferencia de las arquitecturas anteriores, como las redes LSTM, que dependían en gran medida de la secuencia de entrada, el *Transformer* aprovecha la atención para procesar simultáneamente todas las palabras en una secuencia. Esto significa que es altamente paralelizable donde las operaciones de atención y transformación pueden realizarse en paralelo, aprovechando mucho más el hardware actual para acelerar significativamente el entrenamiento y la inferencia.

Esta arquitectura se compone principalmente de dos elementos claves: el *codificador* y el *decodificador*. El codificador toma la entrada de texto y la procesa en una representación numérica rica en información llamada embedding. A partir de esta

incrustación, se procesan múltiples capas de atención y transformaciones. Cada capa consta de dos subcapas principales:

- Capa de atención múltiple (*Multi-Head Attention*): Calcula la importancia relativa de cada palabra en relación con las demás palabras en la secuencia de entrada. Utiliza múltiples *cabezas* de atención para capturar diferentes tipos de relaciones.
- Capa de alimentación hacia adelante (*Feed-Forward Layer*): Después de la capa de atención múltiple, se aplica una red neuronal completamente conectada para procesar cada posición por separado y en paralelo. Contempla el cómputo propiamente dicho dentro del modelo.

El decodificador es similar al codificador, pero tiene una capa adicional de atención llamada atención encubierta (*masked self-attention*) que se utiliza para predecir cada palabra en la secuencia de salida en función de las palabras anteriores generadas. Ayuda a prevenir que el modelo vea información del futuro durante la generación de palabras (Vaswani et al., 2023).

A continuación se explicará más a fondo cada una de estas partes, junto con el mecanismo de atención que implementa el modelo.

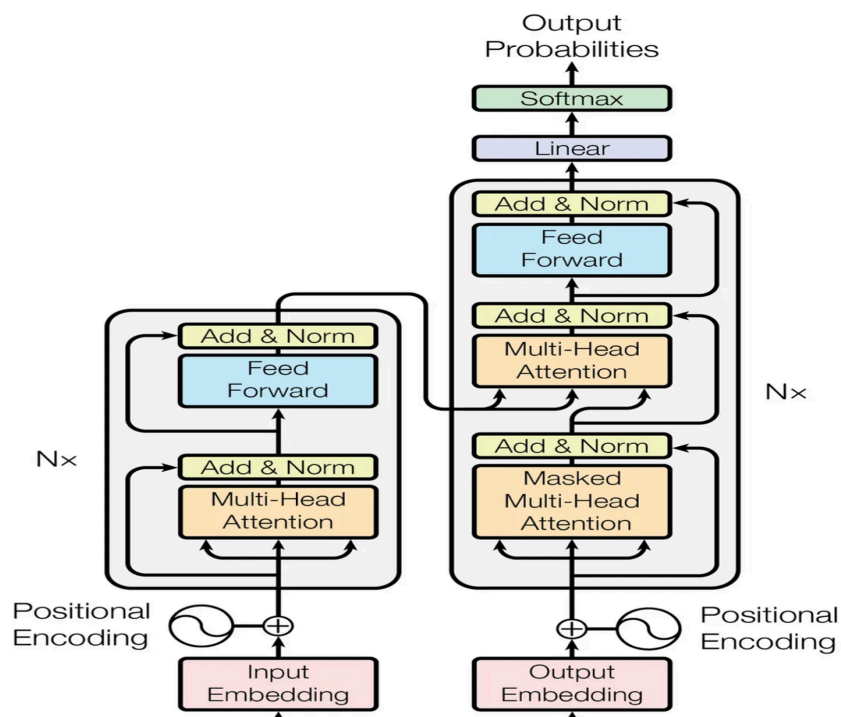


Ilustración 14: Diagrama de la arquitectura Transformer original, Fuente: (Vaswani et al., 2023)

Mecanismo de self-attention

Se sabe que aunque se trate de secuencias largas las palabras se relacionan unas con otras mediante semántica, contexto, significado, etc. Esas relaciones son el objetivo por alcanzar por el mecanismo de atención. Este mecanismo de atención está conformado por dos redes neuronales: una se encargará de identificar las características importantes de la palabra, mientras que la otra se encargará de identificar las propiedades interesantes que esa palabra busca en la secuencia. Cabe aclarar que el resultado de estas redes serán vectores numéricos, como bien se vio anteriormente, la forma de trabajar con secuencias de texto en Deep Learning es a través de representaciones embeddings. Se puede entender esto mediante la analogía de una llave y una cerradura, para cada palabra de la secuencia, estas dos redes crearán una llave (lo que la palabra busca) y una cerradura (lo que la palabra ofrece a las demás). A partir de ahora se nombrarán a estos vectores *key* y *query*. (Vaswani et al., 2023).

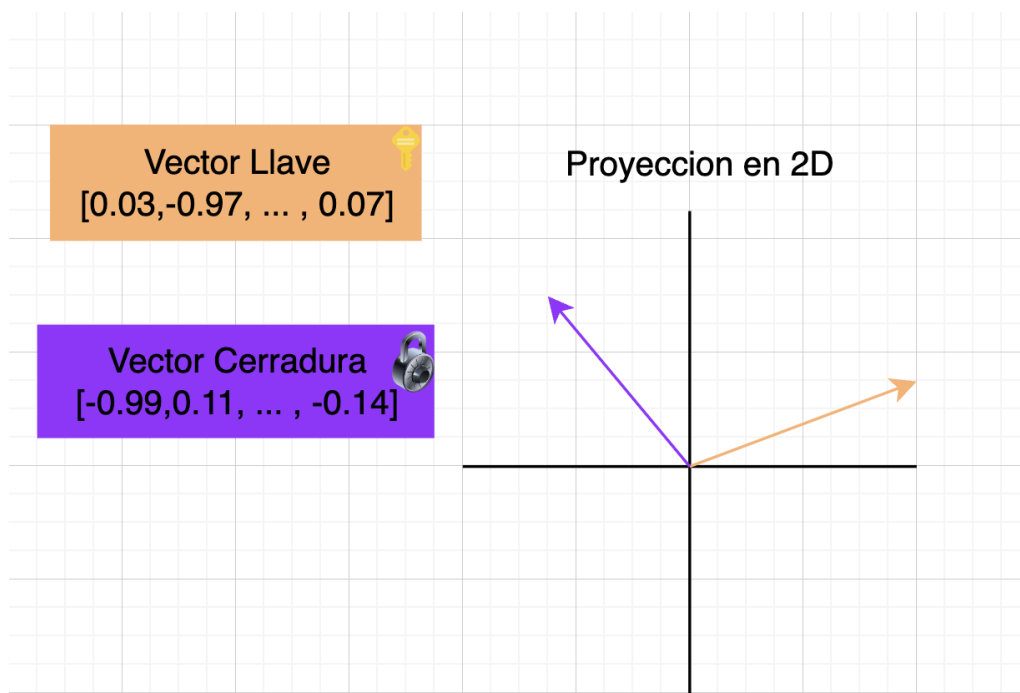


Ilustración 15: proyección en 2D de vector llave y vector cerradura, Fuente: Elaboración propia

Estas llaves funcionan en mejor o peor medida con cada cerradura. Matemáticamente se usará el producto escalar en estos dos vectores para calcular qué tan similar es la dirección de estos dos vectores, es decir, si las redes que generaron estos vectores quieren indicar que son compatibles, deberán asignarle a cada palabra vectores que tengan una dirección similar. Entonces, para cada palabra, se calculará una puntuación

de atención usando el vector *query* y los vectores *key* de cada una, formando así lo que se conoce como vector de atención. Esto indica qué tan relevante es cada palabra para esa palabra en cuestión. Si se calcula este vector para cada una de las palabras de la secuencia, obtenemos lo que se conoce como matriz de atención, donde se puede ver la importancia que está asignando cada palabra al resto de la frase.

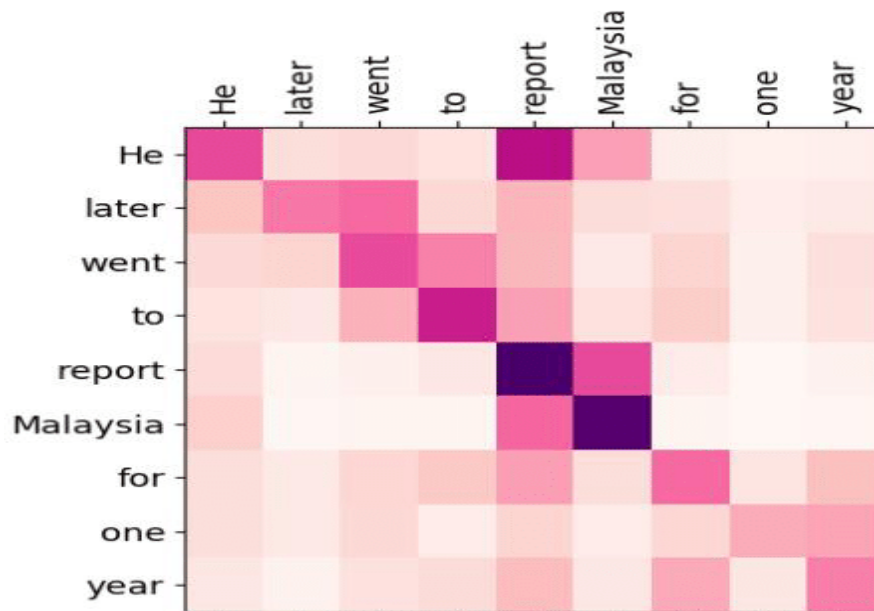


Ilustración 16: Visualización de una matriz de atención. Fuente: (Du, Siyuan & Wang, Hao, 2022)

En otras palabras, indican a qué le presta atención la red neuronal en cada entrada para tomar sus decisiones. Gracias a esto se puede relacionar palabras dentro de la secuencia sin importar que tan alejadas estén. Ahora, usando otra red neuronal se procesa cada una de las palabras y transforma cada palabra a un vector de salida necesario para cumplir la tarea que se busca. Este vector de salida es al que se llama vector *value* (Vaswani et al., 2023). Las atenciones calculadas anteriormente servirán para realizar una suma ponderada de estos vectores *value* usando el puntaje de atención como un factor de mezcla de cada uno de los vectores de valor. De esta manera, la salida tendrá el contexto necesario de todas las otras palabras que conforman a la frase, dando mayor importancia a aquellas a las que le haya prestado *atención*. Con esto se resuelve el principal problema al que se enfrentaban las RNN, la falta de memoria. En el contexto de la arquitectura, este procedimiento sucede en las attention layers o también conocidas como attention heads. Las multi-head attention layers son simplemente varias de estas capas apiladas. En concreto, la arquitectura presentada en el paper original cuenta con seis *attention heads* en el decoder.

Se puede notar también una clara distinción entre el procesamiento de la semántica de la secuencia y el cómputo propiamente dicho, siendo que cada componente principal en la arquitectura cuenta con una sola capa de *FeedForward*, es decir, neuronas *full-connected*. El secreto del éxito de esta arquitectura está en su mecanismo de atención (Alammar, 2020).

Codificación posicional

Como se ha visto, esta arquitectura puede extraer la información más relevante de cada palabra en la secuencia. Este procesamiento y cálculo de vectores de atención se realiza de forma paralela sobre todas las palabras de la secuencia, lo que representa una mejora importante. Sin embargo, se pierde el sentido de la secuencialidad, entonces, ¿cómo se puede saber cuál es la posición de cada palabra en la frase? Para ello se introduce una nueva forma de codificación que, combinada con los embeddings de palabras, aporta esa componente posicional faltante. Se hablará ahora sobre este proceso de codificación previo.

Cuando se habla de la arquitectura transformer se hace énfasis en analizar secuencias de datos, en concreto, centrado en el análisis de secuencias de palabras. Como se mencionó anteriormente, el primer paso es necesariamente obtener una representación numérica de estas palabras a través de la técnica de los espacios latentes para convertirlas en vectores de alta dimensionalidad. Para recuperar el sentido de orden en la secuencia se debe buscar otra forma de codificación que sea compatible con los embeddings de palabras. Para ello, se genera otro vector por cada palabra, pero este contendrá únicamente información posicional de cada una, para luego sumarlos componente a componente.

Una primera aproximación en la que se podría pensar es en usar un posicionamiento absoluto, es decir, sumar a cada vector de cada palabra un vector de igual tamaño con el número de la posición que ocupa en la secuencia. Esto presenta un problema: si se tiene una secuencia de 200 palabras, la información del vector original, por comparación relativa, sería casi insignificante en relación con la posicional recordando que las redes neuronales *sólo ven* números. Una posible solución podría ser normalizar estos vectores respecto del tamaño de la secuencia, pero esto conlleva a que la información posicional que se muestra a la red varíe con respecto de la longitud de la secuencia.

Por lo tanto, para otorgar a cada palabra un valor único dependiendo de la posición que ocupa en la secuencia, la codificación que se utiliza en la arquitectura transformer se basa en una combinación de senos y cosenos a diferentes frecuencias. Se usan tantas ondas como palabras en la secuencia.

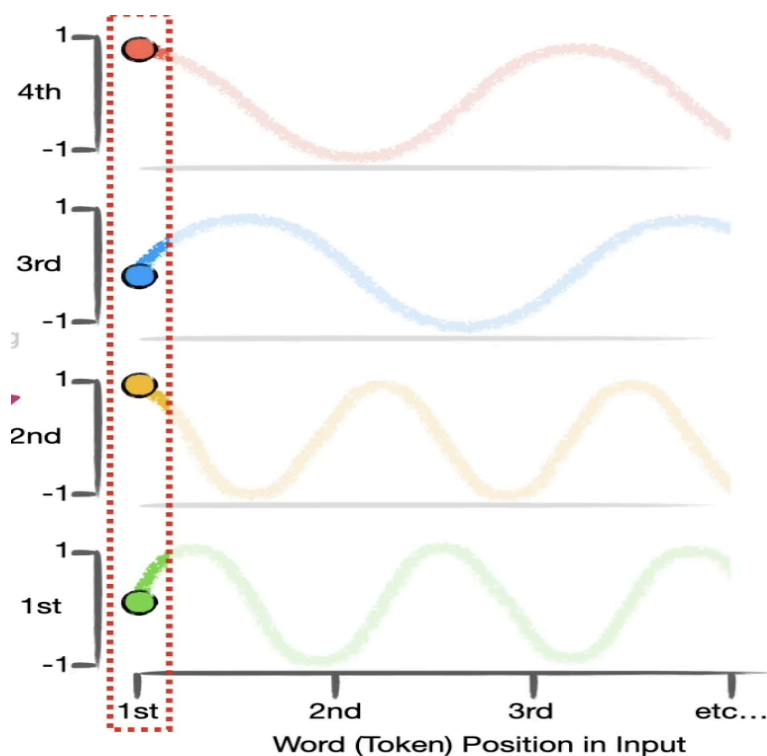


Ilustración 17: senos y cosenos de la codificación posicional, Fuente:(Startmer J., 2023, StatQuest)

Como se puede ver en la Ilustración 17, para la representación vectorial de la primera palabra el resultado de la codificación posicional sería 0,1,0,1. Para las demás palabras, el proceso es el mismo. Es posible que, debido a la naturaleza periódica de las funciones, dos incrustaciones presenten los mismos valores de codificación posicional en una de sus componentes. Sin embargo, debido a que las funciones aumentan su periodo respecto a la posición dentro de la incrustación, incluso con algunos pares de valores repetidos, el resultado final es una secuencia única de valores de posición por cada token o palabra.

Encoders

El bloque encoder en la arquitectura Transformer es una pieza esencial en la modelación de la secuencia. Este está diseñado para procesar y generar representaciones continuas a partir de una secuencia de entrada. El bloque encoder en la arquitectura original se compone de seis capas idénticas, cada una de las cuales incorpora dos subcapas principales: un mecanismo de *multi-headed self-attention* o autoatención de múltiples entradas (donde se desarrollan los cálculos de self-attention) y una red neuronal *full-connected* o *feed-forward*.

El mecanismo de multi-headed self attention es el componente principal que permite al encoder evaluar y ponderar la importancia de diferentes posiciones en la secuencia de

entrada de forma simultánea. Utilizando las representaciones vectoriales de cada token de la secuencia junto a la codificación posicional explicadas anteriormente, el Encoder puede capturar dependencias a largas distancias dentro de la entrada, gracias a los cálculos que conforman el mecanismo de self-attention. Esto es una capacidad crucial para tareas como la traducción automática y el procesamiento del lenguaje natural. Cada subcapa de self-attention en el encoder está seguida por una red neuronal feed-forward que se aplica de manera independiente a cada token en la secuencia. Esta red consta de dos transformaciones lineales separadas por una función de activación ReLU, lo que permite al modelo aprender representaciones más complejas de la entrada (Vaswani et al., 2023).

Para asegurar la estabilidad y la eficiencia del entrenamiento, cada subcapa dentro del bloque encoder incorpora una conexión residual seguida de una normalización por capas. La conexión residual es una técnica crucial en el diseño de redes neuronales profundas y se implementa sumando la entrada de la subcapa a su salida antes de aplicar la normalización. Ambas se introdujeron inicialmente para mitigar el problema del desvanecimiento del gradiente, lo que impedía que los pesos de las capas más profundas se actualizaran adecuadamente, dificultando el aprendizaje efectivo (Vaswani et al., 2023).

Al incorporar una conexión residual, se crea un camino directo para la propagación del gradiente desde la salida de la subcapa hasta su entrada. Esto significa que, incluso si los gradientes a través del camino principal se desvanecen, el camino residual asegura que una parte del gradiente siempre se transmita, mejorando la eficiencia del entrenamiento y permitiendo que las redes sean más profundas sin sufrir problemas de degradación.

La normalización por capas, que sigue a la suma de la conexión residual, también desempeña un papel vital en la estabilidad del entrenamiento. Esta ajusta y escala las activaciones de manera que cada capa en el modelo reciba datos con una distribución similar. Esto ayuda a prevenir problemas como la covarianza de cambio interno, donde los cambios en las distribuciones de entrada a medida que el modelo se entrena pueden ralentizar el proceso de aprendizaje y hacer que la red sea más difícil de entrenar. Al normalizar las activaciones en cada subcapa, se asegura que la distribución de las entradas a cada capa se mantenga consistente. Ambos elementos son fundamentales al trabajar con este tipo de arquitecturas complejas.

Decoders

El bloque *decoder* en la arquitectura Transformer juega un papel crucial en la generación de la secuencia de salida a partir de las representaciones proporcionadas por el encoder. Al igual que el *encoder*, el *decoder* está compuesto por una pila de seis capas idénticas, cada una con tres subcapas. La primera, al igual que el *encoder* implementa un mecanismo de *multi-headed self-attention*; la segunda es una capa de atención

multi-entrada pero la cual toma las salidas del encoder como entradas; y la tercera es una red *feed-forward* o *full-connected*. Cada una de estas subcapas está rodeada por una conexión residual seguida de una normalización por capas, al estilo del bloque encoder (Vaswani et al., 2023).

La primera subcapa del decoder, permite que cada posición en la secuencia de salida actual se enfoque en todas las posiciones anteriores en la secuencia de salida generada hasta el momento. Esto es posible gracias a un mecanismo de enmascaramiento (*masking*) que impide que los cálculos de atención consideren posiciones futuras, ya que se debe recordar que esta arquitectura ingesta la totalidad de la secuencia en una sola pasada y no token por token como lo haría una red *feed-forward*. La estrategia de enmascarado es fundamental para mantener la propiedad autorregresiva del modelo. Este enmascaramiento se implementa estableciendo en $-\infty$ los valores en la entrada de la función softmax que corresponden a conexiones que se consideran ilegales para una predicción en concreto, asegurando que las predicciones para una posición i solo dependan de las posiciones anteriores a i .

La segunda subcapa en el bloque decoder es aquella que implementa el mecanismo de atención multi-entrada, pero sobre la salida producida por el *encoder*. La misma permite que cada posición en la secuencia de salida generada se enfoque en todas las posiciones de la secuencia de entrada original. Esta capacidad de atención cruzada es fundamental para que el modelo integre información relevante de la entrada al generar la salida, mejorando así la alineación y traducción entre las secuencias de entrada y salida (Vaswani et al., 2023).

El proceso de atención cruzada es esencial para que el decoder pueda acceder a las representaciones codificadas de la secuencia de entrada (producidas por el encoder) y determinar qué partes de la entrada son más relevantes para generar cada token de la salida. Durante un proceso de traducción, por ejemplo, cada palabra en la oración de salida debe alinearse correctamente con las palabras correspondientes en la oración de entrada para que la traducción sea precisa y coherente. La atención cruzada facilita esta alineación al permitir que el decoder evalúe todas las posiciones de la secuencia de entrada mientras genera cada posición de la secuencia de salida.

La atención cruzada es un componente vital que permite la interacción efectiva entre el encoder y el decoder. A través de este mecanismo, el decoder puede acceder directamente a la información contextualmente rica de la secuencia de entrada, asegurando que cada paso en la generación de la secuencia de salida esté bien informado y alineado con la entrada original. Este proceso optimiza la capacidad del modelo para manejar tareas complejas de procesamiento de lenguaje natural. La misma se implementa utilizando las representaciones generadas por el encoder como los vectores vistos anteriormente, claves

(K) y valores (V). Estas representaciones contienen la información procesada de la secuencia de entrada, capturando las relaciones contextuales entre las palabras de entrada. Mientras tanto, los vectores de consultas (Q) provienen de la capa anterior del decoder y representan la información generada hasta ese punto en la secuencia de salida. Al calcular la atención, se utilizan las consultas para ponderar las claves, determinando así qué valores deben ser enfocados. Esto permite incorporar información relevante de la entrada a medida que genera cada nuevo token en la secuencia de salida. En términos matemáticos, el producto punto escalado de los vectores Q y los vectores K se calcula primero, seguido de la aplicación de una función softmax para obtener las ponderaciones de atención. Estas ponderaciones se utilizan luego para combinar los vectores V, produciendo una representación ponderada que se pasa a la siguiente capa del decoder. Este mecanismo asegura que la salida generada en cada paso no solo dependa de las palabras generadas anteriormente, sino también de la totalidad de la secuencia de entrada, proporcionando así una base sólida para tareas de secuencia a secuencia como la traducción automática (Vaswani et al., 2023).

Finalmente, en la tercera subcapa, la red *feed-forward* aplica transformaciones no lineales a cada posición individualmente. Al igual que en el bloque encoder, consiste en dos transformaciones lineales separadas por una activación ReLU, lo que permite que el modelo capture y transforme características complejas de las representaciones de la secuencia. Las conexiones residuales en cada subcapa del decoder facilitan la propagación del gradiente, permitiendo que la red aprenda de manera más eficiente y mantenga información relevante a través de las capas (Vaswani et al., 2023).

Una implementación ad-hoc cercana a la arquitectura original del paper (Vaswani et al., 2023) pero creando tokens a nivel de carácter junto con una explicación detallada de las herramientas matemáticas que permiten el alto paralelismo del modelo, se encuentra disponible en el repositorio de github⁷.

⁷

<https://github.com/AlejoTorres2001/AI/blob/windows/ProyectoDeGrado/Anexos/AnexoVI-GPT-from-scratch.ipynb> en el siguiente enlace encontrara una implementacion cercana a la arquitectura original del paper con tokens a nivel de carácter, junto con las herramientas matematicas que lo soportan.

¿Qué es el Clustering?

El clustering es una técnica de análisis de datos utilizada en el campo de la minería de datos y el aprendizaje automático. Su objetivo principal es dividir un conjunto de datos en grupos o clústeres, donde los elementos dentro de un mismo clúster comparten características similares, mientras que aquellos en diferentes clústeres presentan diferencias significativas. Los algoritmos de clustering entran en la categoría de no supervisados debido a que no se requiere de un dataset etiquetado para su uso y las características, similitudes y diferencias serán inferidas por el algoritmo (Huanca, 2021).

El proceso de clustering implica varias etapas claves:

- *Selección de Características:* Antes de realizar el clustering, es importante seleccionar las características relevantes que definirán la similitud entre los elementos del conjunto de datos. Estas características pueden ser numéricas, categóricas o incluso texto, según el tipo de datos en cuestión. Es un paso fundamental debido a que no todas las características pueden ser relevantes para la clasificación, incluso algunas pueden ser derivadas o calculadas de otras. Resulta interesante eliminar estas redundancias para evitar caer en el problema de las dimensionalidades que en definitiva desemboca en que el modelo no sea capaz de agrupar adecuadamente.
- *Elección del Algoritmo:* Existen varios algoritmos de clustering, cada uno con sus propias ventajas y limitaciones. Algunos de los algoritmos más comunes incluyen el K-Means, que se describe a continuación. También existen el Clustering Jerárquico o el Clustering Basado en Densidad (DBSCAN), entre otros. La elección del algoritmo adecuado depende de las características de los datos y de los objetivos específicos del análisis.
- *Medición de la similitud:* Para agrupar los elementos en clústeres, se requiere una medida de similitud o distancia entre ellos. Esta medida puede variar según el algoritmo utilizado y el tipo de datos. Por ejemplo, en el K-Means, la distancia euclidiana es comúnmente utilizada para medir la similitud entre puntos numéricos.
- *Inicialización:* En el caso del algoritmo K-Means, se requiere una inicialización de los centroides (puntos representativos de cada clúster). Esto puede afectar la convergencia y el resultado final del algoritmo. Las estrategias de inicialización incluyen seleccionar puntos aleatorios, utilizar heurísticas o técnicas más avanzadas como el *K-Means++* (Penumudy, 2021).
- *Evaluación de Clústeres:* Después de la finalización del proceso de clustering, es importante evaluar la calidad de los clústeres generados. Esto puede implicar métricas como la coherencia intraclúster y la separación interclúster, así como

inspección visual para determinar si los clústeres tienen sentido y son útiles para el análisis.

Algoritmo K-means

El algoritmo K-Means o K-Medias es uno de los algoritmos de clustering más populares y ampliamente utilizados. Su objetivo principal es dividir un conjunto de datos en K clústeres, donde K es un valor predefinido por el usuario. Cada clúster está representado por un punto llamado centroide, y los elementos se asignan al clúster cuyo centroide está más cercano en función a alguna medida de distancia, generalmente la distancia euclidiana. A continuación, se explicará mediante un ejemplo las bases y fundamentos del algoritmo.

Suponga que se tiene un conjunto de datos que se puede graficar en un plano, como el que se ve en la ilustración 18:

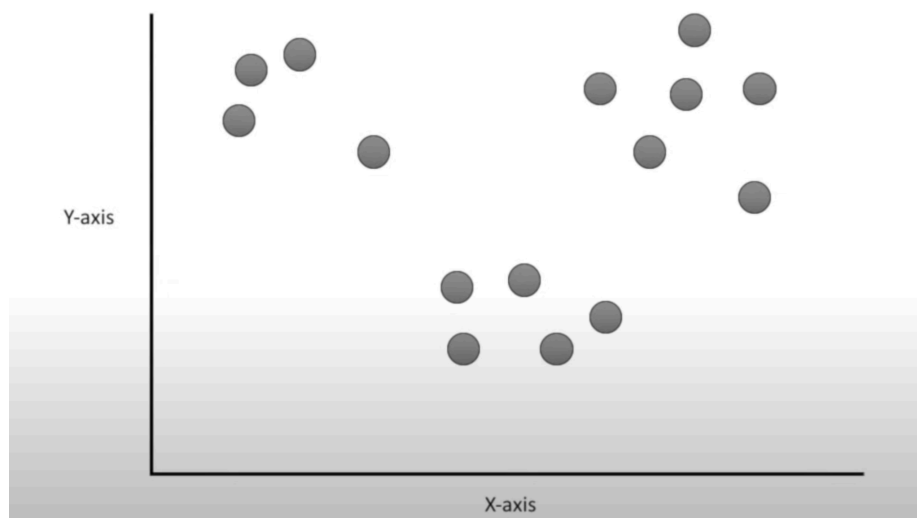


Ilustración 18: observaciones en un plano cartesiano, Fuente: (Startmer J., 2018, StatQuest)

El Algoritmo de K-Medias comienza tomando k puntos aleatorios en el plano cartesiano para ubicar los centroides. Los centroides son un punto geométrico que indica el centro de cada una de las agrupaciones, no tiene por qué coincidir con un registro del conjunto, sino que es una idea matemática que representa el lugar donde la distancia hacia todos los puntos que conforman la agrupación es menor, y la distancia respecto de las otras agrupaciones es mayor, es decir, *se quiere* encontrar una forma de juntar lo más posible estos datos, pero alejarlos lo más que se pueda del resto.

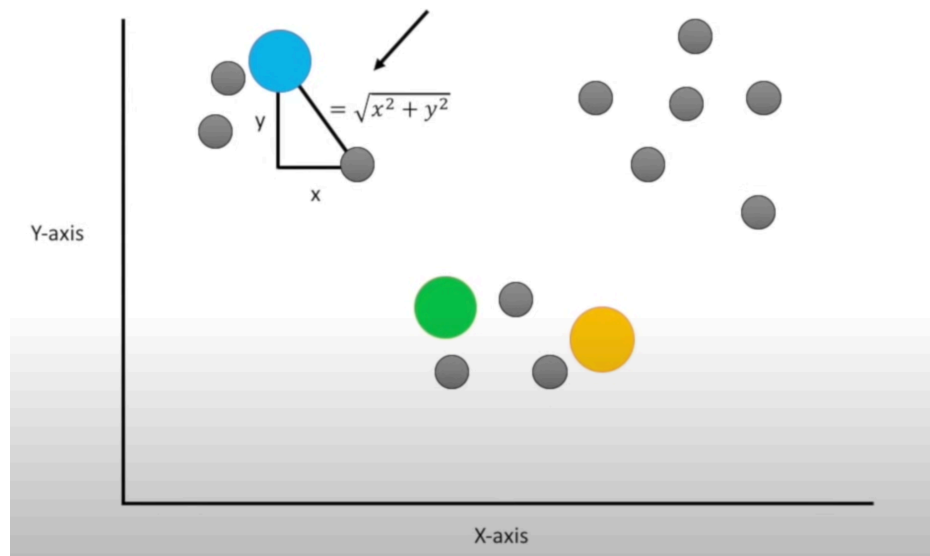


Ilustración 19: centroides y observaciones en un plano cartesiano, Fuente: (Startmer J., 2018, StatQuest)

Para el ejemplo se toma $k=3$ y se usa la distancia euclidiana para calcular qué tan lejos está cada punto de cada centroide

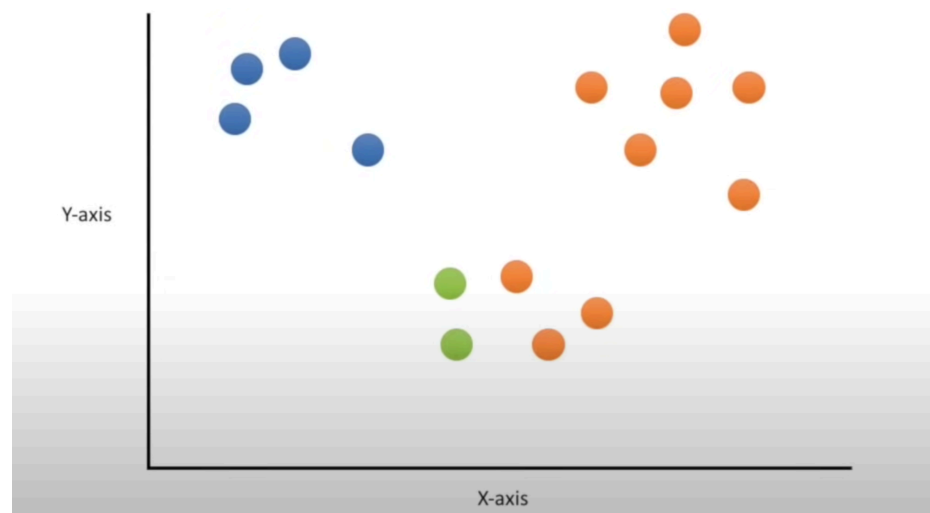


Ilustración 20: observaciones agrupadas en un plano cartesiano, Fuente: (Startmer J., 2018, StatQuest)

Se repite este paso hasta asignar todos los puntos a un cluster en concreto. El siguiente paso es calcular la media de las distancias de cada uno de los puntos a su

centroide correspondiente y éste nuevo punto se usará para actualizar el centroide del clúster.

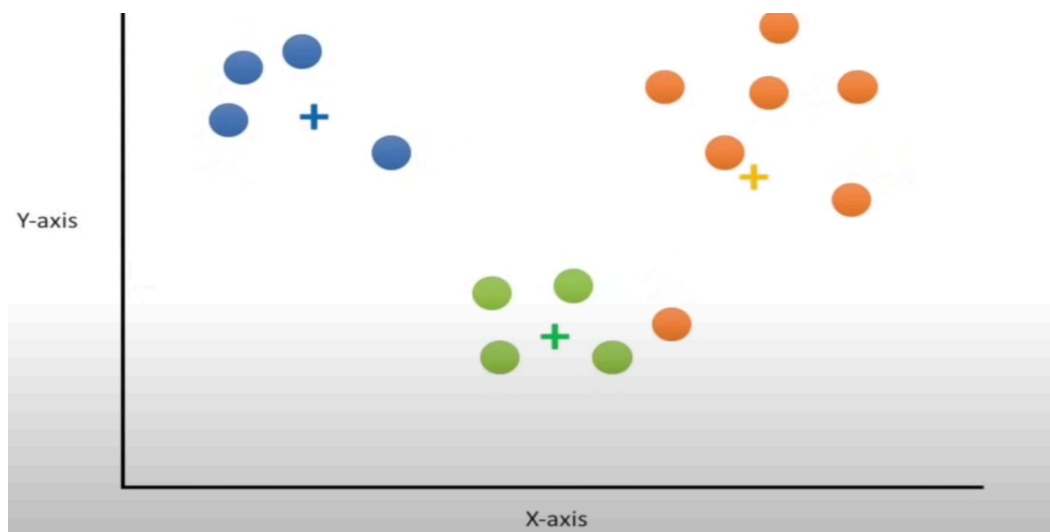


Ilustración 21: centroides y observaciones agrupadas en un plano cartesiano, Fuente: (Startmer J., 2018, StatQuest)

Con el nuevo centroide se repiten los pasos anteriores. El algoritmo converge cuando los centroides ya no cambian entre iteraciones o cuando se alcanza un número máximo de iteraciones predefinido. Como la posición inicial de los K centroides influye mucho en el algoritmo, a veces es recomendable realizar varias iteraciones del algoritmo y tomar el mejor agrupamiento.

Cómo elegir K

Elegir el mejor valor de K es un desafío importante ya que puede afectar significativamente la calidad y la interpretación de los resultados. No existe una regla definitiva para determinar el valor óptimo de K, pero hay varias técnicas que pueden utilizarse:

- **Método del codo (Elbow Method):** Este es un enfoque visual en el que se traza la variación de la suma de las distancias al cuadrado de los puntos dentro de sus centroides frente al número de clústeres K. A medida que K aumenta, es probable que la suma de las distancias disminuya, pero eventualmente se alcanzará un punto en el que la disminución se vuelve más gradual, formando una curva que se asemeja a un "codo". El valor de K en el punto del codo puede ser considerado como una buena elección.
- **Método de la Silueta (Silhouette Method):** Este método evalúa la calidad de los clústeres calculando la distancia promedio entre un punto y los otros puntos en su mismo clúster, que se denominará a , y la distancia promedio entre el punto y los

puntos en el clúster más cercano diferente, que se denotará con b . La silueta para cada punto se calcula como $\frac{(b-a)}{\max(a,b)}$, y luego se calcula el valor promedio de silueta para todos los puntos. Un valor más alto de silueta indica que los puntos están más cerca de los miembros de su propio clúster y más lejos de los puntos en otros clústeres.

Posibles Alternativas

Actualmente existen técnicas disponibles para realizar resúmenes de texto aplicando técnicas que no necesariamente involucran Deep Learning pero sí son consideradas técnicas de Machine Learning y NLP. Sin embargo, implementar grandes modelos de lenguaje ofrece ventajas significativas en términos de precisión, coherencia y capacidad para capturar el contexto semántico del texto de manera más efectiva que las técnicas convencionales. Algunas de ellas son:

- *Resumen de centroides*: Consiste en calcular el vector promedio de todas las frases en el texto utilizando representaciones vectoriales de palabras (como Word2Vec). Luego, selecciona las frases más cercanas al vector promedio como un resumen.
- *Método LSA (Latent Semantic Analysis)*: Utiliza técnicas de reducción de dimensionalidad para encontrar relaciones semánticas entre palabras y frases en un texto. Luego, selecciona las frases más significativas para formar el resumen.
- *Algoritmos de ranking de oraciones*: Se basa en asignar puntajes a cada oración en función de su importancia dentro del texto. Luego, selecciona las oraciones con los puntajes más altos para construir el resumen.
- *Algoritmos de agrupamiento de texto*: Consiste en conformar agrupaciones con las frases o párrafos similares en temas y seleccionar representantes de cada grupo como parte del resumen.
- *Uso de estructuras gramaticales*: Donde se identifica la estructura gramatical del texto, como encabezados, subencabezados y listas numeradas, luego estas estructuras se utilizan para crear un resumen estructurado.
- *Método TF-IDF (Term Frequency-Inverse Document Frequency)*: Calcula la importancia relativa de cada palabra en el texto y luego selecciona frases que contengan palabras con valores altos de TF-IDF.

Las herramientas que incorporan las ventajas del deep learning en conjunto con los LLMs son producto de la creciente popularidad de esta tecnología y, por lo tanto, relativamente nuevas. Gracias a la naturaleza de estos modelos y a su capacidad de simular el lenguaje humano, los resultados tienden a ser de mejor calidad, mostrando una

mayor atención en los detalles y en el contexto del cual se extrae el texto a resumir. Esto establece a los sintetizadores de texto basados en modelos del lenguaje como una fuerte alternativa aportando, además, una fácil incorporación en aplicaciones web ya existentes.

Capítulo 3: Definición de Objetivos

En esta sección se definirán los aspectos y características principales del problema a resolver identificando los objetivos y alcances del proyecto, analizando su contexto y evaluando las alternativas existentes para resolverlo.

Objetivo general

El objetivo general de este proyecto es construir una solución tecnológica que permita sintetizar el contenido de un archivo en formato PDF usando técnicas de Machine Learning y modelos de lenguajes.

Objetivos Específicos

Los objetivos específicos que permitirán alcanzar el objetivo general son:

- Analizar las bases teóricas y fundamentos que sustentan las herramientas y servicios usados en la síntesis de texto.
- Diseñar la arquitectura que soporte los servicios para permitir a un usuario resumir el contenido de un archivo en formato PDF.
- Verificar la eficacia a través de la generación de síntesis sobre diferentes tipos de textos.
- Implementar un prototipo de sistema en la plataforma web que consuma los servicios de inferencia contruidos a través de una interfaz web responsiva.

Restricciones o límites

Es bien sabido que el término *Large* en *Large Language Models* hace referencia al tamaño de estos, llegando incluso a contener billones y trillones de parámetros, entrenados sobre conjuntos de datos y corpus de texto que se asemejan a fracciones considerables de la internet completa. La principal limitante es que resulta prácticamente imposible en términos de tiempo, poder de cómputo y presupuesto construir un LLM competente para la tarea en cuestión. Es más, incluso el ejecutar alguna variante OpenSource, con las mismas prestaciones de modelos freemium o de pago, resulta casi imposible con hardware de grado de consumidor final y con el que se cuenta actualmente para este proyecto. Por lo que las opciones se limitan a utilizar servicios que tercerizan el cómputo de estos modelos, manejando un esquema de pagos por peticiones o número de tokens, como lo son los servicios ofrecidos por empresas como *OpenAI* o *Streamlit*, entre otros.

Alcances del proyecto

El proyecto estará conformado por las siguientes tareas generales:

- Reunir y analizar información relevante relacionada a la Síntesis de texto en el marco del Machine Learning.
- Analizar, diseñar y construir un prototipo de aplicación que permita resumir el texto de un archivo PDF con una extensión que no supere alrededor de las 300 páginas, usando modelos del lenguaje
- Evaluar su rendimiento mediante la generación de resúmenes sobre diferentes documentos.

Capítulo 4: Solución Propuesta

El problema de la síntesis de texto es un desafío importante en el campo del Procesamiento del Lenguaje Natural y el Aprendizaje Profundo. Antes de la aparición de las técnicas de Deep Learning, la generación de texto estaba impulsada principalmente por enfoques basados en reglas y modelos de lenguaje estadísticos. Sin embargo, la llegada del Deep Learning revolucionó la forma en que se aborda este problema, permitiendo generar un texto más coherente y contextualmente relevante.

Hoy en día la tendencia está en el uso de LLMs. Existen una gran variedad de modelos de tipo open source sobre los cuales la comunidad ha realizado *fine tunnings* con esta tarea en particular en mente. *Hugging Face*, una plataforma donde la comunidad comparte sus implementaciones, modelos pre-entrenados y datasets, se ha vuelto de vital importancia para el desarrollo de la IA de código abierto. Sin embargo, la limitante más grande en esta tendencia tiene que ver con el poder de procesamiento requerido para la correcta inferencia de estos grandes modelos del lenguaje siendo casi imposible alcanzar las velocidades y rendimiento de servicios freemium de inferencia utilizando hardware convencional. Si bien es cierto que técnicas como la *cuantización*⁸ de los modelos o su ejecución a través del protocolo de *Petals*⁹ acerca un poco al poder de computo requerido, todavía constituye la mayor limitante.

El objetivo de este proyecto es construir una herramienta de software con una interfaz amigable al usuario que permita sintetizar el contenido textual de un archivo PDF utilizando LLMs.

⁸ Cuantización de los modelos se puede definir como el proceso de mapear valores de un gran conjunto de números reales a valores de un pequeño conjunto discreto. Normalmente, esto implica asignar entradas continuas a valores fijos en la salida. Una forma común de lograrlo es redondeando o truncando. la principal motivación detrás de la cuantificación de redes neuronales profundas es mejorar la velocidad de inferencia.

⁹ Petals es una red distribuida que permite a cualquier persona con una computadora contribuir con su poder de procesamiento inactivo para entrenar y ejecutar modelos de IA que generan texto.

Frameworks y herramientas a implementar en el desarrollo del proyecto

A continuación, se explicará el conjunto de herramientas software, junto con posibles alternativas evaluadas para cada una de ellas, que serán necesarias para poder desarrollar el proyecto.

API de inferencia

Existe una amplia gama de modelos de lenguaje disponibles hoy en día, tanto de naturaleza freemium como OpenSource. Si se habla de modelos de código abierto, otra gran interrogante a resolver es dónde se va a ejecutar dicho modelo, ya sea bajo infraestructura on-premise o utilizando un servicio de infraestructura *cloud*. Siguiendo este último enfoque, se encuentran las siguientes herramientas:

- *API de inferencia de Hugging Face*: Proporciona la capacidad de comunicarse con modelos alojados en su infraestructura *cloud* a través de una API¹⁰ web. Es importante aclarar que no se trata de una solución integral debido a que los modelos disponibles (todos de carácter OpenSource) requieren un proceso de *fine-tuning* previo. Su sistema de cobranzas está medido en horas de cómputo, lo cual no lo hace óptimo para este proyecto.
- *API de inferencia de TextBlob*: Aunque no es tan potente como los modelos de OpenAI, TextBlob ofrece una solución más ligera y fácil de implementar para tareas básicas de procesamiento de lenguaje natural, por lo que no es factible a la hora de sintetizar grandes volúmenes de texto.

Sin embargo, para este proyecto se optará por utilizar la API de inferencia ofrecida por OpenAI, utilizando los modelos GPT-3.5 Y GPT-4, dos de los modelos más potentes a la fecha. Esta solución sigue un enfoque cloud de facturación por petición realizada, más específicamente, por token tanto en la petición como en la respuesta. La potencia de estos modelos, junto con una clara documentación de su interfaz web sumado al conveniente sistema de facturación hacen de esta solución la más factible para el proyecto.

¹⁰ API significa “interfaz de programación de aplicaciones” por sus siglas en inglés. Las API son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos (AWS, s. f.).

	GPT-3 DaVinci	GPT-3.5 Turbo	GPT-4(8k span)
Parameters	125 - 175 Billion	154 Billion	1 Trillion
Max Token Span	2,049 tokens	4,096 tokens	8,192 tokens
Pricing	\$0.03 - 0.12 /1k Token	\$0.002 / 1k tokens	\$0.03 - 0.06 /1k Token
Training Cutoff	Oct 2019	Sep 2021	Sep 2021
Optimization Techniques	Does not use RLHF	Uses RLHF	Uses RLHF

Ilustración 22: Comparación entre modelos GPT. Fuente: (Patel, 2022)

Estos modelos serán la pieza fundamental del sistema y se encargará de realizar la síntesis del texto y todas las tareas de inferencia necesarias para lograr dicha síntesis total y adecuada tanto en contexto como en semántica. Como se mencionó anteriormente, el tamaño de estos modelos hace que su ejecución requiera de mucho poder computacional y recursos de hardware. Por lo tanto, la solución más factible para alcanzar la mejor calidad posible en los resúmenes es la de optar por un servicio de inferencia a través del consumo de alguna API para tercerizar el poder de procesamiento necesario. La elección de la API de inferencia de OpenAI se fundamenta en su destacada calidad y precisión en la generación de texto, respaldada por el renombre de OpenAI en el campo de la inteligencia artificial. Además, su amplia cobertura de idiomas y dominios, junto con una documentación detallada y un sólido soporte comunitario hacen que sea una opción confiable y versátil para aplicaciones de síntesis y resumen de texto.

Manejo de los modelos

Para incorporar el poder de inferencia de los modelos del lenguaje en el flujo de aplicaciones las siguientes opciones pueden ser de utilidad:

- *Guidance*: un lenguaje específico para controlar modelos de lenguaje a través de sus directivas.
- *TensorFlow Serving*: una alternativa popular para la implementación de modelos de aprendizaje automático en producción. Ofrece herramientas para desplegar modelos de TensorFlow de manera eficiente.
- *ONNX Runtime*: permite la implementación de modelos de aprendizaje automático en diferentes frameworks, incluyendo PyTorch y TensorFlow, con un enfoque en la eficiencia y la velocidad de inferencia.

Si bien es cierto que estas herramientas se podrían adaptar al desarrollo que se busca construir en este proyecto, estas opciones no cuentan con abstracciones tan

poderosas y específicamente orientadas al tratamiento con modelos del lenguaje como lo hace LangChain. LangChain resulta ser la opción más madura y robusta para incorporar el poder de inferencia de los modelos del lenguaje en el flujo de aplicaciones que existe a la fecha.

Langchain es una poderosa librería con implementaciones en Python y Javascript que cuenta con abstracciones para facilitar la implementación de aplicaciones impulsadas por IA, introduce el concepto de cadenas o *chains* que permiten agrupar los elementos involucrados sobre la inferencia del modelo, personalizar parámetros, directivas e incluso el propio motor de inferencia, generando así una interfaz común que brinda soporte para los clientes específicos de cada uno de los servicios de inferencia más populares como lo son OpenAI, StreamLit, HuggingFace, etc. La elección de Langchain se justifica por su capacidad para simplificar la implementación de aplicaciones basadas en IA a través de abstracciones que facilitan el manejo de los elementos relacionados con la inferencia del modelo. Su soporte para múltiples servicios de inferencia populares te permite flexibilidad en la elección de proveedores y optimización de parámetros, lo que contribuye a una implementación más eficiente y adaptable.

Librería de Machine Learning

Sklearn es una de las librerías de Machine Learning más utilizadas en el lenguaje de programación Python. Contiene implementaciones de los algoritmos más populares y para el propósito de este proyecto es más que suficiente. Las posibles alternativas a esta librería son:

- *PyTorch*: otro framework de aprendizaje automático de código abierto que se ha vuelto muy popular debido a su flexibilidad y facilidad de uso, especialmente en la investigación.
- *XGBoost*: una biblioteca optimizada para la implementación de algoritmos de *gradient boosting*, que es ampliamente utilizada en competencias de ciencia de datos debido a su eficiencia y precisión.

Sin embargo, Sklearn es usada en mucha mayor medida que sus alternativas, ofreciendo documentación y códigos de ejemplo con gran disponibilidad que sus competidores. Además, cuenta con una extensa comunidad de autodidactas dispuestos a compartir sus conocimientos. También conocido como *Scikit-learn*, es una de las bibliotecas más populares y ampliamente utilizadas del lenguaje de programación Python para realizar tareas de aprendizaje automático. Proporciona herramientas eficientes y fáciles de usar para llevar a cabo tareas como clasificación, regresión, agrupamiento, reducción de dimensionalidad, etc. Esta constituye la elección más sólida para realizar tareas de

aprendizaje automático en Python debido a su amplia gama de herramientas y su facilidad de uso. Al proporcionar implementaciones eficientes de algoritmos de aprendizaje automático, *Scikit-learn* simplifica el proceso de desarrollo y experimentación con modelos de machine learning, lo que lo convierte en la opción más razonable.

Framework Backend

Respecto a la solución para implementar el servidor web, se requiere un marco de trabajo flexible que permita incorporar los servicios desarrollados en python y soportando las librerías mencionadas anteriormente. Algunas alternativas son:

- *Flask*: un micro framework de Python para el desarrollo web que es popular por su simplicidad y facilidad de uso, aunque se encuentra algo desactualizado.
- *Django*: un framework de desarrollo web de alto nivel que sigue el principio de "baterías incluidas" y es adecuado para aplicaciones web más complejas.

La opción a utilizar será *FastAPI*, uno de los frameworks de backend más rápidos escritos en Python que aprovecha todas las ventajas que brinda una ASGI (Asynchronous Server Gateway Interface), como lo es Starlette, en cuanto a manejo de peticiones no bloqueantes. Su sistema de tipado permite crear soluciones robustas y su flexibilidad se adapta a patrones de desarrollo web modernos, lo cual es fundamental para trabajar efectivamente sobre este tipo de aplicaciones.

FastAPI es un marco de desarrollo web de Python de alto rendimiento y fácil de usar que se utiliza para crear aplicaciones web y APIs. Fue diseñado para ayudar a los desarrolladores a construir aplicaciones rápidas y escalables con una sintaxis clara y una documentación automática gracias al tipado estático. Aprovecha el poder de librerías de modelado de datos como Pydantic y posee una serie de *middlewares* que facilitan la construcción de funcionalidades web. Su elección se justifica por su alto rendimiento, facilidad de uso y capacidad para crear aplicaciones web y APIs de forma rápida y escalable. Su sintaxis clara y documentación automática, junto con el aprovechamiento de librerías como Pydantic para el modelado de datos, facilitan la construcción de funcionalidades web, permitiendo así enfocarse en la lógica de negocio sin preocuparse por detalles de implementación.

Framework Frontend

En cuanto a la tecnología utilizada para construir la interfaz web, existen una gran cantidad de librerías y frameworks modernos. Algunas alternativas que se tuvieron en cuenta fueron:

- *Angular*: un framework de desarrollo web de TypeScript desarrollado por Google que ofrece una estructura robusta y un conjunto completo de herramientas para la construcción de aplicaciones web.
- *Vue.js*: una biblioteca progresiva para la construcción de interfaces de usuario que se centra en la facilidad de uso y la integración gradual en proyectos existentes.

La librería que se utilizará es *ReactJS*, una de las soluciones más populares que cuenta con un gran ecosistema de dependencias que solucionan la mayoría de los problemas comunes a la hora de encarar el desarrollo de una UI web. Su enfoque basado en componentes web y su paradigma funcional hacen mucho más sencillas las tareas como maquetar y estilar, manejo de estados internos de la aplicación y tratamiento de peticiones asíncronas.

ReactJs es una biblioteca de JavaScript de código abierto utilizada para construir interfaces de usuario interactivas y dinámicas en aplicaciones web. Desarrollada y mantenida por Facebook, React se ha convertido en una de las herramientas más populares en el desarrollo web moderno debido a su enfoque en la construcción eficiente y modular de componentes reutilizables, en lugar de abordar la creación de una aplicación web como un conjunto de páginas separadas. Ésta se centra en la creación de componentes individualmente encapsulados y autónomos. Estos componentes pueden ser pequeñas piezas de la interfaz de usuario, como botones, formularios o elementos más grandes, como encabezados y paneles de navegación. Cada componente puede tener su propia lógica y estado interno, lo que facilita la gestión y la reutilización en diferentes partes de la aplicación. Se destaca por su capacidad para construir interfaces de usuario interactivas y dinámicas en aplicaciones web de manera eficiente y modular. Su enfoque en la creación de componentes reutilizables y autónomos simplifica el desarrollo de la interfaz de usuario de tu aplicación, permitiéndote construir una experiencia de usuario consistente y atractiva. Además, su popularidad y gran comunidad de desarrolladores proporcionan un amplio soporte y recursos para el desarrollo de tu proyecto.

Metodología a implementar

La elección de la metodología de desarrollo en cascada está respaldada por varias razones claves. En primer lugar, la naturaleza secuencial y estructurada de esta metodología proporciona una hoja de ruta clara para el proceso de desarrollo. En un entorno donde la incertidumbre es común, como en el trabajo con modelos de inteligencia artificial, esta claridad y predictibilidad es un punto fuerte para considerar. Esta metodología asegura que cada etapa del proyecto se complete antes de pasar a la siguiente, lo que minimiza el riesgo de errores y garantiza una progresión ordenada.

Además, la metodología en cascada se alinea perfectamente con la naturaleza definida y los objetivos claros de este proyecto desde el principio. Con requisitos bien establecidos, no se esperan cambios significativos en el curso del proyecto. Esto significa que la metodología en cascada puede desplegar sus fortalezas, permitiendo una planificación y ejecución eficientes sin la necesidad de adaptaciones frecuentes. Asimismo, la generación de una cantidad sustancial de documentación en cada etapa del proceso es beneficiosa para un proyecto que implica la implementación de algoritmos complejos de inteligencia artificial.

El modelo en cascada es uno de los enfoques más antiguos y tradicionales para la gestión de proyectos de desarrollo de software. Se caracteriza por su enfoque lineal y secuencial en el que cada fase debe completarse antes de que pueda comenzar la siguiente. Sus etapas son:

- *Relevar Requisitos:* En esta etapa, se recopilan y documentan los requisitos del software. Se define claramente qué debe hacer el software y cómo debe funcionar.
- *Diseño:* Una vez que los requisitos se han establecido, se procede al diseño del sistema. Esto implica crear un diseño detallado de cómo se estructurará y funcionará el software, incluidos aspectos como la arquitectura, la interfaz de usuario y las bases de datos.
- *Implementación:* En esta fase, se lleva a cabo la programación real del software. Los desarrolladores traducen el diseño en código fuente utilizando lenguajes de programación específicos.
- *Pruebas:* Después de que se haya implementado el software, se realizan pruebas exhaustivas para garantizar que cumple con los requisitos y funciona correctamente. Esto incluye pruebas de unidad, pruebas de integración y pruebas de sistema.
- *Despliegue:* Una vez que el software ha pasado las pruebas con éxito, se despliega en el entorno de producción.

- *Mantenimiento*: Esta es la fase en la que se realizan actualizaciones, correcciones de errores y mejoras en el software a lo largo de su ciclo de vida, la cual no será contemplada en este proyecto.

Aunque tiene ventajas que fueron comentadas anteriormente, la metodología de desarrollo en cascada presenta algunas desventajas. Una de ellas es su rigidez ya que la inflexibilidad inherente puede ser problemática si se descubren cambios en los requisitos a medida que avanza el proyecto. Otra desventaja es la demora en la retroalimentación. En este enfoque, la retroalimentación de los interesados generalmente ocurre después de que se haya completado gran parte del desarrollo, lo que dificulta la corrección de errores conceptuales tempranos. Por último, esta metodología carece de adaptabilidad lo que significa que dificulta la incorporación de cambios significativos una vez que se ha iniciado el desarrollo.

Análisis de Requerimientos

La primera fase de la metodología de desarrollo en cascada consiste en detallar los requerimientos del sistema. Para este proyecto, los requerimientos que deben ser alcanzados por el sistema son:

- El sistema debe permitir al usuario seleccionar un solo archivo en formato PDF únicamente, desde el sistema de archivos de su computador.
- El sistema debe poder procesar y extraer todo el texto del archivo.
- El sistema debe permitir al usuario modificar los hiper parámetros involucrados en el proceso de inferencia y síntesis del texto.
- El sistema debe presentar el resumen al usuario como un archivo en formato PDF
- El sistema debe informar de manera clara y en un lenguaje comprensible las excepciones en el flujo de inferencia causadas por terceros o mala configuración de los hiper parámetros.
- El sistema debe ser accesible tanto en computadores como en dispositivos móviles.

Planificación del proyecto

La gestión de proyectos es una disciplina fundamental para asegurar que los proyectos se completen de manera eficiente y eficaz. Se ha decidido utilizar una adaptación de la metodología del PMBOK (Project Management Body of Knowledge) para ciertas etapas de análisis y definición del proyecto. Esta decisión se justifica por varios factores.

Naturaleza y trabajo del Proyecto

El mismo involucra una cantidad significativa de investigación y trabajo con modelos de IA, áreas que requieren una planificación rigurosa y estructurada en su comienzo, pero debe dejarse margen de acción para la investigación. PMBOK es conocido por su enfoque sistemático y detallado que ayuda a manejar proyectos complejos y técnicamente exigentes.

Definición de Requisitos

Desde el inicio del proyecto, se espera que los requisitos estén claramente definidos, respecto al alcance y objetivos del prototipo a desarrollar. PMBOK es particularmente adecuado para proyectos con requisitos bien establecidos, ya que su metodología fomenta una planificación detallada y exhaustiva desde el principio, lo que minimiza las incertidumbres y cambios inesperados en el transcurso del proyecto.

Equipo de trabajo

El equipo de desarrollo está compuesto por tres personas. En equipos pequeños, una estructura de gestión de proyectos formal como la del PMBOK puede ser demasiado abrumadora ya que asegurar que todas las actividades están bien coordinadas y que los recursos se utilizan de manera óptima, por lo que se optó por adaptar las partes más útiles de la metodología para este caso.

Duración del Proyecto

El proyecto está planeado para durar 110 días. Dada esta duración relativamente corta, es crucial contar con una planificación sólida desde el principio para asegurar que todas las tareas se completen a tiempo y dentro del presupuesto. PMBOK ofrece un marco que facilita la planificación y el control riguroso del cronograma.

Planificación detallada

Se prefiere una planificación detallada y exhaustiva desde el inicio del proyecto. La metodología se basa en la creación de planes detallados que cubren todos los aspectos del proyecto, incluyendo el alcance, el tiempo, el costo, la calidad, los recursos humanos, la comunicación, los riesgos, y las adquisiciones. La planificación se ha centrado principalmente en los aspectos relacionados con el alcance, el tiempo, y el costo, dado que estos son los factores clave para garantizar el éxito del desarrollo. La exclusión de la planificación detallada sobre la calidad, los recursos humanos, la comunicación, los riesgos, y las adquisiciones se debe a la naturaleza controlada y predecible de los objetivos. Dado que los requisitos están claramente definidos y se espera que se mantengan estables a lo largo del proyecto. La planificación del proyecto se representará por medio de un Diagrama

de Gantt, en el cual se mostrará el orden de las actividades necesarias para poder llevar a cabo el proyecto y el tiempo estimado para cada actividad.

	Name	Duration	Start	Finish
1	☐ Software para sintetizar el texto de un archivo PDF	110 days?	9/4/23 8:00 AM	2/2/24 5:00 PM
2	☐ Analisis e inicio del proyecto	9 days?	9/4/23 8:00 AM	9/14/23 5:00 PM
3	Identificación del Problema	3 days?	9/4/23 8:00 AM	9/6/23 5:00 PM
4	Definición del alcance	2 days?	9/7/23 8:00 AM	9/8/23 5:00 PM
5	Definición de Objetivo General y Especifico	2 days?	9/9/23 8:00 AM	9/12/23 5:00 PM
6	Inicio del Proyecto	2 days?	9/13/23 8:00 AM	9/14/23 5:00 PM
7	☐ Definicion y Planificacion del Proyecto	7 days?	9/14/23 8:00 AM	9/22/23 5:00 PM
8	Analisis de Requerimientos	2 days?	9/14/23 8:00 AM	9/15/23 5:00 PM
9	Analisis de Riesgos	3 days?	9/15/23 8:00 AM	9/19/23 5:00 PM
10	Analisis de Factibilidad	3 days?	9/19/23 8:00 AM	9/21/23 5:00 PM
11	Analisis de Costos - Beneficios	1 day?	9/22/23 8:00 AM	9/22/23 5:00 PM
12	☐ Investigacion	17 days?	9/27/23 8:00 AM	10/19/23 5:00 PM
13	☐ Conceptos bases de Machine learning	11 days?	9/27/23 8:00 AM	10/11/23 5:00 PM
14	Funcionamiento de un MLP	3 days?	9/27/23 8:00 AM	9/29/23 5:00 PM
15	AI's generativas	3 days?	9/29/23 8:00 AM	10/3/23 5:00 PM
16	Modelos del lenguaje	4 days?	10/4/23 8:00 AM	10/9/23 5:00 PM
17	Algoritmos de clustering	2 days?	10/10/23 8:00 AM	10/11/23 5:00 PM
18	☐ Estudio sobre LLMs	6 days?	10/12/23 8:00 AM	10/19/23 5:00 PM
19	Redes Recurrentes	2 days?	10/12/23 8:00 AM	10/13/23 5:00 PM
20	Arquitectura Transformer	3 days?	10/13/23 8:00 AM	10/17/23 5:00 PM
21	Mecanismo de Self-Attention	3 days?	10/17/23 8:00 AM	10/19/23 5:00 PM
22	Desarrollo del flujo de inferencia de la aplicacion	2 days?	10/20/23 8:00 AM	10/23/23 5:00 PM
23	☐ Estudio de Software y librerias a utilizar	4 days?	10/23/23 8:00 AM	10/26/23 5:00 PM
24	LangChain	2 days?	10/23/23 8:00 AM	10/24/23 5:00 PM
25	Servicios de Inferencia terciarizados	2 days?	10/23/23 8:00 AM	10/24/23 5:00 PM
26	ReactJs	3 days?	10/24/23 8:00 AM	10/26/23 5:00 PM
27	☐ Desarrollo del Software	72 days?	10/26/23 8:00 AM	2/2/24 5:00 PM
28	☐ Analisis y diseño	25 days?	10/26/23 8:00 AM	11/29/23 5:00 PM
29	Analisis y diseño de los servicios de inferencia	15 days?	10/26/23 8:00 AM	11/15/23 5:00 PM
30	Analisis y diseño de la interfaz de usuario	10 days?	11/16/23 8:00 AM	11/29/23 5:00 PM
31	☐ Desarrollo	30 days?	11/30/23 8:00 AM	1/10/24 5:00 PM
32	Desarrollo de lo servicios del lado del servidor	20 days?	11/30/23 8:00 AM	12/27/23 5:00 PM
33	Desarrollo de la interfaz de usuario	10 days?	12/28/23 8:00 AM	1/10/24 5:00 PM
34	☐ Testing	4 days?	1/11/24 8:00 AM	1/16/24 5:00 PM
35	Testing del Backend	2 days?	1/11/24 8:00 AM	1/12/24 5:00 PM
36	Testing del Frontend	2 days?	1/13/24 8:00 AM	1/16/24 5:00 PM
37	☐ Evaluacion	13 days?	1/17/24 8:00 AM	2/2/24 5:00 PM
38	Generacion de resúmenes	4 days?	1/17/24 8:00 AM	1/22/24 5:00 PM
39	Recolectar retroalimentacion de Expertos	7 days?	1/23/24 8:00 AM	1/31/24 5:00 PM
40	Comparacion de Resultados	2 days?	2/1/24 8:00 AM	2/2/24 5:00 PM

Ilustración 23: Planificación del proyecto Fuente: Elaboración propia

Análisis de Costos

A continuación se realiza un análisis de los costos involucrados en el proyecto, en materia humana como así también en hardware y en software.

Costos de Recursos Humanos

Los costos de recursos humanos son los costos asociados con el personal necesario para poder llevar a cabo el proyecto. Para poder estimar los costos de recursos humanos, se utilizó como referencia la escala de honorarios de Informática de COPAIPA (Consejo

Profesional de Agrimensores, Ingenieros y Profesionales Afines), consultada en el mes de noviembre del año 2023, convirtiendo los valores a dólares estadounidenses. tomando como referencia el precio del dólar paralelo promedio para dicho mes.

Cargo	Cantidad Necesaria	Costo por Hora	Total, de Horas en el Proyecto	Subtotal
Lider de Proyecto	1	8 USD	120	960 USD
Desarrollador Full Stack	1	7 USD	600	4200 USD
Data Scientists	1	9 USD	160	1440 USD

Tabla 1: Costos de recursos humanos, Fuente: Elaboración propia

Costos de Hardware

Los costos de hardware son los costos relacionados a los equipos y dispositivos que serán necesarios para poder llevar a cabo el proyecto. En un contexto empresarial sería necesario una computadora de gama media para cada integrante del equipo, la cual permita llevar a cabo las tareas de desarrollo y gestión pertinentes.

Debido a que ya se disponía del equipamiento necesario para afrontar el proyecto en su totalidad, no se requiere la adquisición de nuevos equipos y dispositivos relacionados al Hardware. Se realizó una búsqueda del costo actual del hardware empleado para la realización del proyecto, tomando como precio de referencia la publicación de este en la plataforma de ventas de *Amazon US*.

Concepto	Cantidad	Costo por unidad
Lenovo ThinkPad L14 - Laptop empresarial, pantalla táctil FHD de 14 pulgadas, Intel Core i5-1135G7, almacenamiento SSD de 1 TB, memoria de 16 GB, Bluetooth y Wi-Fi 6, Thunderbolt 4, HDMI, Windows 11 Pro	3	680 USD

Tabla 2: Costos de Hardware, Fuente: Elaboración propia

Costos de Software

La mayoría de las librerías y marcos de trabajo que se utilizaran en el proyecto son de código abierto, bajo una licencia que permite su utilización en proyecto tanto con fines educativos como con fines comerciales. El único elemento de software que posee un

sistema de facturación serán los servicios de inferencia tercerizados que se utilizarán para interactuar con los modelos del lenguaje. Como se comentó, esta decisión está basada en la realidad que implica implementar y servir estos modelos de gran tamaño y alta demanda de recursos de manera *on-premise* por lo que la tercerización de estos servicios implica una reducción de costos considerable como así también una mejora en el rendimiento del flujo de trabajo de la aplicación.

Estos servicios poseen un esquema de facturación conocido como *pay-as-you-use* en el cual se factura únicamente el tiempo de cómputo utilizado, mediante un sistema de créditos. Específicamente relacionado a modelos de inteligencia artificial generativos de tipo *text-to-text*, el cobro se realiza por token. Para calcular el costo de una petición a este servicio se debe tener en cuenta el precio del token en la directiva del modelo y el precio del token en la salida del modelo, para luego sumarlos y obtener el precio por petición.

Es evidente que el precio de servicio es variable respecto al número de tokens, el tamaño de las respuestas y el modelo seleccionado. Si bien es cierto que no se brinda una información exacta sobre cómo cada uno de estos servicios realiza la tokenización de las secuencias de texto de entrada y salida, puede aproximarse usando la fórmula mencionada anteriormente. Existen servicios de calculadoras de precios por petición como es el caso de *gptforwork.com*. Teniendo en cuenta el modelo GPT-4 con una ventana de contexto de 32K tokens, los precios por 100 peticiones variando la cantidad de tokens, tanto de directiva como de respuesta. Se pueden ver en la Ilustración 24:

Tokens per execution	Words per execution	Price for 1 execution ⓘ	Price for 100 executions ⓘ
10	8	~\$0.00090	~\$0.09
20	15	~\$0.00180	~\$0.18
50	38	~\$0.00450	~\$0.45
100	75	~\$0.00900	~\$0.90
200	150	~\$0.01800	~\$1.80
500	375	~\$0.04500	~\$4.50
1000	750	~\$0.09000	~\$9.00
2000	1500	~\$0.18000	~\$18.00
4000	3000	~\$0.36000	~\$36.00

Ilustración 24: Esquema de precios para GPT-4 32K, Fuente: (*gptforwork.com*, 2023)

En un contexto empresarial, se puede suponer que el producto se brinda a través de un sistema de suscripciones para los usuarios. Se podría toma como referencia la cantidad de documentos utilizados para la elaboración de este proyecto de grado, promediado con la cantidad de documentos de bibliografía utilizados para el desarrollo de proyectos de extensión y propósito similar. Con el objetivo de calcular un costo aproximado por el uso del

servicio, por usuario. Es decir tratar de estimar el uso promedio de los servicios de un usuario individual. Suponiendo que, por mes, un usuario procesa alrededor de 25 documentos usando peticiones con el máximo número de tokens posibles, teniendo en cuenta el flujo de inferencia del sistema y los servicios necesarios para manejar adecuadamente la síntesis del documento se estaría hablando de un costo base de aproximadamente **7 USD mensual**, por usuario lo que genera un costo anual para la empresa de **84 USD** por usuario activo.

Costos Totales

En un contexto empresarial, teniendo en cuenta lo anteriormente mencionado, el capital humano representa un costo de **6600 USD**; el hardware necesario para el proyecto supone una inversión de **2040 USD**, por lo que en total significa una inversión inicial de **8640 USD**. El poner en producción el sistema contempla un costo base de **7 USD** en promedio por cada suscripción activa, considerando las restricciones mencionadas anteriormente.

Análisis de Factibilidad

Factibilidad Económica

Cuando se habla de la factibilidad económica de un proyecto se refiere a la evaluación para determinar si el desarrollo del proyecto es viable financieramente y si el mismo, puede generar beneficios suficientes para cubrir los costos asociados. Si bien es cierto que no es un objetivo principal de este proyecto generar ingreso a través de una aplicación o servicio, un modelo de negocio basado en suscripciones de usuarios activos sería fácilmente implementable. Para ello se recurrirá al análisis TIR y VAN. El análisis de TIR (Tasa Interna de Retorno) y VAN (Valor Actual Neto) es una técnica comúnmente utilizada en la evaluación de proyectos de inversión y en la toma de decisiones financieras. Ambos son indicadores claves en la gestión financiera y tienen como objetivo determinar la viabilidad y rentabilidad de un proyecto. El VAN es una medida que calcula el valor presente de los flujos de efectivo futuros generados por una inversión, descontados a una tasa de descuento específica, menos la inversión inicial. La TIR es la tasa de descuento que iguala el valor presente de los flujos de efectivo futuros de una inversión con su costo inicial. En otras palabras, es la tasa que hace que el VAN sea cero. Un proyecto es considerado aceptable si su TIR es mayor que la tasa de descuento o el costo de capital de la empresa, ya que indica que el proyecto debería generar un retorno superior al costo del capital.

Sabiendo que el proyecto requiere una inversión inicial de **8640 dólares** y si se cobrase una suscripción mensual de **10 dólares** a cada usuario, supondría un retorno de **3 dólares** por usuario activo al mes.

Si se supone una tasa de inversión alternativa del **10%** y considerando un periodo de **24 meses** para recuperar la inversión inicial, se puede obtener el número de usuarios necesarios para que el proyecto sea mínimamente rentable de la siguiente ecuación:

$$VAN = \sum_{t=1}^{24} \frac{CF_t}{(1+r)^t} - I_0$$

donde:

- CF_t : Es el flujo de caja en el periodo t.
- r : Es la tasa de descuento mensual (0.83%).
- I_0 : Es la inversión inicial (8640 USD).

Para que el proyecto sea rentable, el VAN debe ser cero o positivo. Se despeja el número de usuarios (U):

$$0 = \sum_{t=1}^{24} \frac{U \times 3}{(1+0.0083)^t} - 8640$$

Utilizando métodos numéricos aproximados se obtiene que aproximadamente se necesitan **135 usuarios activos** constantes por mes. Con este supuesto se presenta el análisis de VAN y TIR del proyecto:

Cálculo de TIR y VAN

	SynthAI		
TIR Anual	0.97%		
VAN	\$136.70		

Ilustración 25: Calculo de VAN y TIR, Fuente: Elaboración propia

El proyecto es factible económicamente ya que con un mínimo de **135 usuarios activos por mes**, y un mínimo precio en la suscripción de **10 dólares**, en el plazo de **24 meses** además de recuperar la inversión inicial, el proyecto presenta rentabilidad.

Factibilidad Operativa

La factibilidad operativa hace referencia a la evaluación que se debe realizar para determinar si se dispone de los recursos necesarios para poder llevar a cabo el proyecto. En este caso, el proyecto sí es operativamente factible, debido a que se dispone del capital humano y de la capacitación correspondiente. A continuación, se describen las aptitudes y cualidades necesarias en cada uno de roles requeridos para el proyecto

El Líder de Proyecto será el encargado de gestionar la planificación, desarrollo, control y evaluación de las diferentes etapas del proyecto, algunas de las cualidades deseadas para la persona a ocupar este cargo son:

- Experiencia en gestión de proyectos de desarrollo de software, preferiblemente en proyectos de inteligencia artificial.
- Habilidades de comunicación efectiva para coordinar al equipo y comunicarse con los interesados del proyecto.
- Capacidad para establecer y mantener plazos realistas y gestionar eficazmente los recursos disponibles.
- Conocimiento técnico general para comprender las necesidades del proyecto y guiar al equipo en la dirección correcta.
- Capacidad para liderar, motivar y desarrollar al equipo, fomentando un ambiente de trabajo colaborativo y productivo.

El Desarrollador de Software se encargará de la construcción del sistema web desde la implementación del backend y su integración con los servicios de inferencia tercerizados, hasta el diseño y desarrollo de la interfaz de usuario. Las cualidades deseadas en este rol son:

- Experiencia en desarrollo de aplicaciones web, con conocimientos en tecnologías como Python, JavaScript, y frameworks como FastAPI.
- Familiaridad con la integración de API, especialmente con la API de inferencia de OpenAI u otras APIs de inteligencia artificial.
- Habilidad para escribir código limpio, modular y bien documentado, siguiendo las mejores prácticas de desarrollo de software.
- Capacidad para trabajar en equipo, colaborar con otros desarrolladores y adaptarse a los cambios en los requisitos del proyecto.

El Data Scientist se encargará de los servicios que componen el pipeline de procesamiento de datos y la integración de las librerías para manejo de los LLMs. Las habilidades deseadas para este rol son:

- Experiencia en análisis de datos y modelado estadístico, con habilidades en lenguajes como Python o R.
- Conocimiento profundo de técnicas de aprendizaje automático y procesamiento del lenguaje natural, necesarias para el desarrollo de la funcionalidad de generación de síntesis de texto.
- Familiaridad con técnicas de clustering y análisis de agrupamiento para identificar patrones y estructuras en los datos.
- Habilidad para evaluar, seleccionar e implementar modelos de inteligencia artificial adecuados para las necesidades del proyecto.
- Destrezas en la visualización de datos para comunicar de manera efectiva los hallazgos y resultados obtenidos durante el análisis.

Factibilidad Técnica

Cuando se habla de la Factibilidad Técnica, se refiere a la evaluación de la capacidad técnica y tecnología para la implementación del proyecto, es decir, si a día de hoy, existen técnicas y conocimientos que posibiliten la implementación y desarrollo del proyecto. En este caso, el proyecto si es factible desde un punto de vista técnico debido a:

- El hardware requerido es accesible y puede ser gestionado eficientemente mediante servicios de cómputo en la nube.
- Los modelos GPT-3.5 y GPT-4 que la API de inferencia de OpenAI expone, cumplen satisfactoriamente con los requisitos de procesamiento de lenguaje natural necesarios para el proyecto.
- Las bibliotecas sklearn y LangChain son adecuadas para implementar las técnicas de clustering y el manejo de los modelos de lenguaje en un contexto de sistema distribuido.
- La utilización de ReactJS para el frontend asegura un desarrollo eficiente y escalable de la interfaz de usuario.
- La capacidad de escalar la infraestructura en la nube garantiza una gestión eficiente de los recursos y un modelo de negocio sostenible.

Se sabe entonces que los requisitos de hardware son obtenibles y los servicios de inferencia tercerizados se encuentran disponibles. Al ser un tema que gira en torno a una esfera de conocimiento ampliamente trabajada y que durante los últimos años ha constituido una tendencia tecnológica, actualmente se cuenta con todos los elementos necesarios a nivel tecnológico para desarrollar el proyecto en su totalidad. Incluso, si se deseara implementar un control sobre los usuarios que acceden al mismo, al tratarse de una aplicación basada en la plataforma web, se podría utilizar un sistema de autenticación basado en sesiones, junto con algún mecanismo de persistencia de datos.

La estructura de costos basada en suscripciones mensuales permite un modelo de negocio sostenible ya que se podría escalar la infraestructura necesaria dinámicamente y en tiempo real, utilizando cualquier servicio de cómputo elástico en la nube, servicio ofrecido por la mayoría de proveedores cloud actualmente. Por lo antedicho se puede afirmar que el proyecto es técnicamente factible.

Factibilidad Legal

La factibilidad legal se refiere a la evaluación que se debe realizar para determinar si el proyecto a realizar cumple con todas las leyes y regulaciones vigentes. A continuación, se detallan los puntos más importantes.

Respecto al uso de herramientas y bibliotecas de código abierto: como todas las herramientas y bibliotecas utilizadas en el desarrollo del proyecto están publicadas bajo licencias de código abierto como MIT, Apache 2.0, GNU GPL, entre otras, se permiten el uso, modificación y distribución del software tanto en ámbitos educativos como comerciales, siempre que se cumplan los términos específicos de cada licencia, como la atribución y, en algunos casos, la redistribución del código modificado. Este uso de software de código abierto asegura que no hay impedimentos legales relacionados con derechos de propiedad intelectual sobre las herramientas y bibliotecas empleadas.

Respecto al cumplimiento con los Términos de Uso de la API de inferencia de OpenAI: se cuenta con una suscripción válida a la API de inferencia de OpenAI y se cumplen con todos los términos de uso y políticas establecidas por la empresa. Esto incluye no utilizar la API para actividades prohibidas, respetar los límites de uso establecidos y adherirse a las condiciones de uso comercial. Al cumplirse con estos términos, se garantiza que el uso de la API de OpenAI es legal y acorde con las políticas del proveedor del servicio.

Respecto al uso de material de libre acceso: las pruebas de síntesis de texto se realizan utilizando material que es libre de derechos de autor y que se encuentra públicamente disponible en internet, como notas de clase y libros de libre acceso. No se utilizan datos personales en estas pruebas. El uso de material de libre acceso asegura que no se violan derechos de propiedad intelectual ni se infringen leyes de derechos de autor. Al no utilizar datos personales, también se elimina cualquier preocupación relacionada con la privacidad y la protección de datos personales.

Respecto a las regulaciones locales: el proyecto se desarrolla dentro de territorio argentino. Se debería tener en cuenta entonces la Ley de Protección de los Datos Personales, también conocida como Ley N° 25.326. Esta ley establece las normas sobre la protección de los datos personales y tiene como objetivo garantizar el derecho a la privacidad de las personas. Sin embargo, el sistema no persiste ni requiere datos personales de los usuarios para su funcionamiento completo. Es importante aclarar, que, si bien se utilizó un modelo basado en suscripciones de usuarios activos para representar la factibilidad económica del proyecto, el mismo enfoque escapa al alcance del proyecto, aunque es totalmente factible su implementación y monetización. El proyecto no involucra datos financieros ni de salud, lo que permite evitar las complejidades legales asociadas con la gestión y protección de estos tipos de datos sensibles.

Dicho esto, se puede decir que el proyecto es legalmente factible porque cumple con todas las leyes y regulaciones vigentes en Argentina. El uso de herramientas de código abierto y la adhesión a los términos de uso de la API de OpenAI aseguran que no hay impedimentos legales relacionados con software y servicios de terceros. Además, el uso de material de libre acceso y la ausencia de datos personales eliminan cualquier preocupación relacionada con la propiedad intelectual y la protección de datos.

Análisis de Riesgos

Para realizar un análisis de riesgos completo del proyecto, es necesario considerar una variedad de factores que podrían impactar negativamente su desarrollo y operación. A continuación, se presenta una lista de riesgos potenciales, junto con su análisis y posibles estrategias de mitigación.

Limitantes por parte de la empresa OpenAI:

- Descripción: OpenAI podría imponer restricciones en el uso de sus APIs, como cambios en las políticas de uso, limitaciones de acceso, o discontinuación de ciertos modelos.
- Impacto: 5-Alto. La funcionalidad central del proyecto depende de la API de OpenAI.
- Probabilidad: 3
- Puntaje: 15
- Mitigación: Utilizar alguna de las alternativas exploradas

Aumento inesperado en los Costos:

- Descripción: Los costos de los servicios en la nube y de la API de OpenAI pueden aumentar inesperadamente.
- Impacto: 5-Alto. Podría afectar la viabilidad financiera del proyecto.
- Probabilidad: 4
- Puntaje: 20
- Mitigación: Establecer contratos a largo plazo con proveedores de infraestructura cloud para fijar precios. Implementar un sistema de monitoreo de costos en tiempo real.

Cambios en las formas de Tokenización:

- Descripción: Modificaciones en cómo OpenAI o las bibliotecas de NLP manejan la tokenización.
- Impacto: 3-Medio. Podría requerir ajustes significativos en el procesamiento de texto.

- Probabilidad: 3
- Puntaje: 9
- Mitigación: Mantener el código flexible y modular para adaptarse rápidamente a cambios en las bibliotecas de tokenización. Revisar periódicamente las actualizaciones de OpenAI y bibliotecas utilizadas.

Cambios del tipo breaking changes en las herramientas:

- Descripción: Actualizaciones en las bibliotecas utilizadas (ReactJS, sklearn, LangChain) que introducen cambios que rompen la compatibilidad con el código existente.
- Impacto: 4-Medio-Alto. Puede causar interrupciones en el desarrollo y funcionamiento de la aplicación.
- Probabilidad: 3
- Puntaje: 12
- Mitigación: Seguir una estrategia de versionado cuidadosa, testear actualizaciones en entornos de desarrollo antes de implementarlas en producción, y mantener una buena documentación del código para facilitar los ajustes necesarios.

Personal abandonando el proyecto antes de su finalización:

- Descripción: Situaciones en las que miembros clave del equipo de desarrollo deciden dejar el proyecto por motivos personales, profesionales u otras oportunidades, lo que puede resultar en una pérdida de conocimiento y experiencia crítica para el proyecto.
- Impacto: 5-Alto. Puede causar retrasos significativos en el cronograma del proyecto, pérdida de conocimiento técnico y una reducción en la calidad del producto final.
- Probabilidad: 3
- Puntaje: 15
- Mitigación: Implementar un plan de retención de personal que incluya incentivos y reconocimiento, fomentar un ambiente de trabajo positivo y colaborativo y documentar detalladamente el progreso del proyecto y el conocimiento técnico clave.

Problemas de escalabilidad:

- Descripción: La infraestructura actual podría no ser suficiente para manejar un gran aumento en el número de usuarios.
- Impacto: 3-Medio. Podría afectar la experiencia del usuario.
- Probabilidad: 3
- Puntaje: 9

- Mitigación: Utilizar servicios de cómputo en la nube que permitan una escalabilidad dinámica y real, y realizar pruebas de estrés para identificar y solucionar posibles cuellos de botella.

Compatibilidad y actualizaciones del Software:

- Descripción: Necesidad de mantener la compatibilidad con múltiples versiones de bibliotecas y herramientas de software.
- Impacto: 3-Medio. Podría requerir un esfuerzo continuo en mantenimiento y pruebas.
- Probabilidad: 3
- Puntaje: 9
- Mitigación: Utilizar herramientas de gestión de dependencias y automatización de pruebas para asegurar que las actualizaciones no introduzcan incompatibilidades.

Análisis y Diseño del Prototipo

A continuación, se muestra un diagrama del flujo de trabajo del prototipo para la aplicación de síntesis de texto junto con las diferentes tecnologías y herramientas involucradas que fueron descritas en el análisis de tecnologías de la solución.

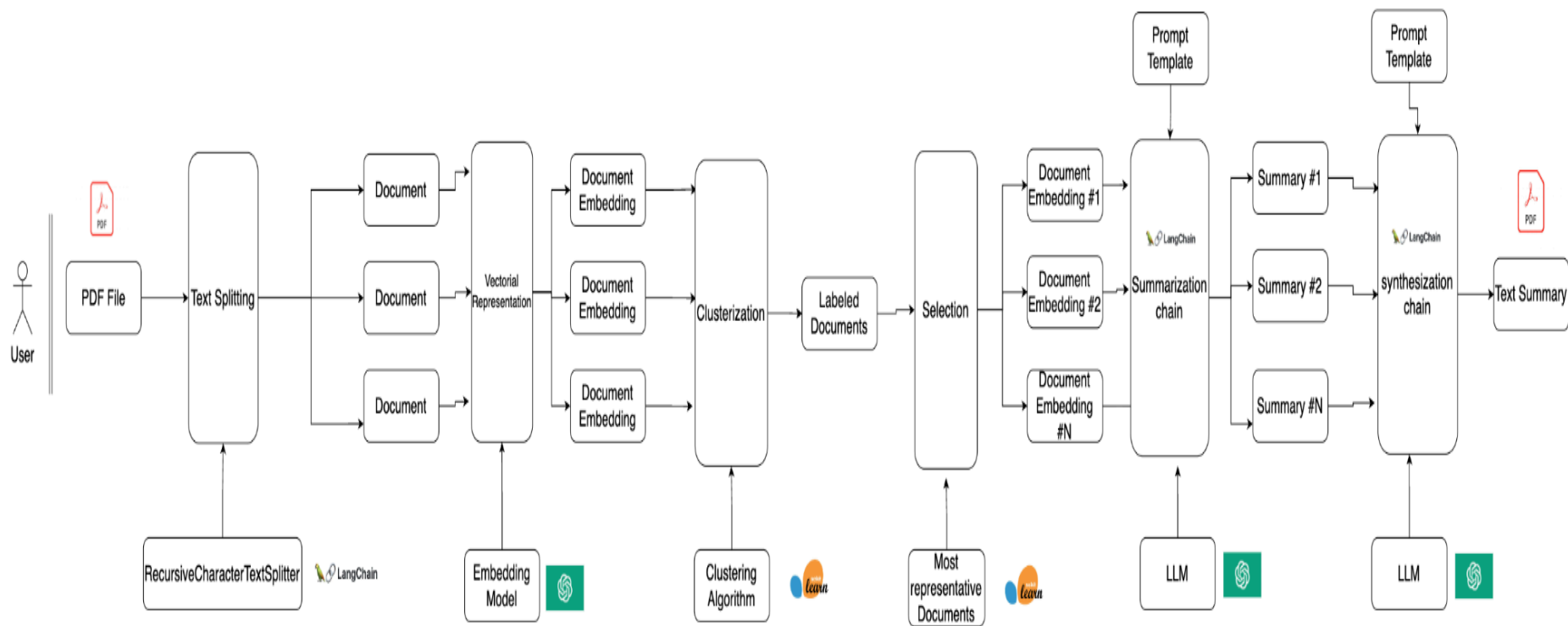


Ilustración 26: Diagrama de flujo del prototipo para la aplicación de síntesis de texto. Fuente: Elaboración propia

Como se puede ver en la Ilustración 26, partiendo de un archivo en formato PDF suministrado por el usuario, se procederá a leer y extraer el texto plano del mismo. Esta tarea se lleva a cabo utilizando una biblioteca especializada en la manipulación y extracción de contenido de archivos PDF denominada PYPDF2 que facilita la manipulación programática de la información en archivos estructurados.

Una vez extraído, el texto se prepara para su procesamiento dividiéndolo en fragmentos manejables denominados documentos. Este proceso de segmentación se realiza utilizando la implementación de Langchain llamada RecursiveCharacterTextSplitter, que permite dividir el texto en partes más pequeñas mientras se asegura de conservar el contexto y la continuidad del contenido mediante un solapamiento calculado entre los fragmentos. El tamaño de los documentos y la cantidad de caracteres compartidos entre documentos adyacentes son parámetros configurables dentro del flujo que pueden afectar la calidad de la síntesis final. El solapamiento es esencial para garantizar que las transiciones entre los fragmentos no interrumpen la coherencia del contenido. Por ejemplo, si un documento A finaliza en medio de una oración, el solapamiento garantiza que la continuación de esa oración aparezca también al comienzo del documento B, permitiendo mantener la continuidad narrativa y contextual. Esto también ayuda a mejorar la precisión del modelo de embeddings y del algoritmo de clustering, ya que las representaciones vectoriales de los fragmentos conservan las dependencias contextuales necesarias para una agrupación y posteriormente una inferencia efectiva.

El siguiente paso crucial es la obtención de una representación en un espacio vectorial denso de cada uno de estos documentos. Para ello, se emplea un modelo de embeddings que transforma los textos en vectores de características. En este proyecto, se utilizó el servicio de OpenAI expuesto a través de su API, el cual es de los más robustos y menos costosos actualmente. El proceso comienza enviando los fragmentos de texto al servicio de OpenAI, seleccionando alguno de los modelos pre entrenados disponibles. En este proyecto se optó por text-embedding-ada-002, conocido por su eficacia en la generación de representaciones vectoriales de alta calidad. La elección de este modelo se basó en su capacidad para capturar significados semánticos complejos y su rendimiento comprobado en una variedad de tareas de procesamiento de lenguaje natural.

Con los vectores de los documentos listos, se procede a aplicar un proceso de clustering para agrupar documentos similares. Este paso se realiza utilizando la biblioteca sklearn, que ofrece una variedad de algoritmos de agrupamiento. En este caso, se optó por el algoritmo K-means debido a su eficacia y simplicidad para trabajar con grandes conjuntos

de datos vectoriales. K-means es especialmente adecuado para esta tarea porque busca minimizar la variación dentro de cada cluster, agrupando documentos que son similares entre sí en un mismo grupo. La configuración del número de clusters es también un parámetro configurable del flujo que puede modificarse a través de la interfaz web.

Una vez que los documentos han sido agrupados, se seleccionan los documentos más representativos de cada agrupación. Este proceso de selección se basa en la proximidad de los documentos al centroide de su respectivo cluster, asegurando que los documentos elegidos sean los más representativos de cada grupo.

La selección de los documentos representativos comienza con la identificación de los centroides de cada cluster, que son los puntos en el espacio vectorial que representan el centro geométrico de los documentos dentro del cluster. Estos centroides son calculados durante la ejecución del algoritmo de K-means y convenientemente son expresados por la implementación utilizada en sklearn como una lista, lo que permite iterarla. Una vez que se tienen los centroides, se mide la distancia de cada documento dentro de un cluster a su respectivo centroide. Esta distancia generalmente se calcula utilizando la distancia euclidiana. Para cada cluster, el documento más representativo se determina seleccionando aquel cuya distancia al centroide es mínima. Este documento, al estar más cercano al centroide, es considerado el que mejor encapsula las características y el contenido del cluster en su conjunto. Este enfoque asegura que los documentos seleccionados no solo sean centrales en términos de posición en el espacio vectorial, sino también representativos del tema o contenido dominante en ese cluster. Este proceso de selección es crítico porque los documentos representativos son los que se utilizarán en las etapas posteriores de síntesis de texto. Al elegir los documentos más cercanos a los centroides, se maximiza la probabilidad de que estos documentos contengan la información más relevante y representativa del clúster, lo que a su vez mejora la calidad y la coherencia de los resúmenes generados.

La generación de resúmenes individuales de cada documento seleccionado se realiza a través de la abstracción Chains provista por el framework de Langchain. Utilizando un modelo de lenguaje, también proporcionado por la API de OpenAI, se genera un resumen conciso y coherente para cada documento. La configuración del modelo de lenguaje juega un papel importante en la optimización de la calidad del resumen. Los parámetros como *max_tokens* para controlar la longitud del output y *temperature* para ajustar la creatividad del texto generado pueden ser configurados a través de la interfaz web.

Finalmente, las síntesis individuales se combinan para formar la síntesis final del texto original. Este proceso implica varios pasos cruciales para asegurar que el resumen final sea coherente, fluido y representativo del contenido original en su totalidad. Para

abordar este problema, se aplica un modelo de lenguaje adicional, como GPT-4, manejándolo a partir de la librería LangChain. Este modelo se encarga de refinar y sintetizar el contenido combinado. La refinación implica la eliminación de redundancias, la reestructuración de frases y la mejora de la cohesión entre los diferentes resúmenes parciales. Esto se logra mediante la reevaluación de las conexiones lógicas y semánticas entre las oraciones y párrafos, asegurando que la información fluya de manera natural y coherente. Gracias a la arquitectura de éste y su mecanismo de atención, estas resultan tareas idóneas para el modelo. Un aspecto importante de este enfoque es que el modelo de lenguaje no solo combina las síntesis parciales, sino que también realiza una síntesis integral del contenido. Esto significa que el modelo evalúa el texto globalmente para asegurarse de que todas las ideas principales del documento original estén representadas de manera adecuada y equilibrada en el resumen final. El objetivo es crear una síntesis que no solo sea una simple agregación de partes, sino una representación cohesiva y coherente del conjunto.

Desarrollo del Prototipo

El desarrollo del prototipo explicado en el apartado anterior se realizó creando un entorno virtual del lenguaje de programación python en su versión 3.11 utilizando la herramienta de Jupyter Notebooks. Esta herramienta permite prototipar y explorar soluciones de manera dinámica gracias a que posibilita ejecutar código python en formato de celdas, compartiendo la misma sesión del kernel y acompañándolo de celdas especiales a modo de documentación en formato *markdown*. Se comenzó por poner en práctica diferentes técnicas de síntesis de texto, en diferentes problemas variando el tamaño del corpus de texto y la complejidad requerida en el resumen, hasta llegar a la solución final. A continuación, se muestran imágenes de cada uno de los bloques de código, correspondientes a cada etapa en el flujo de inferencia completo correspondiente a la solución final. El archivo .ipynb (Jupyter Notebook) que contiene el resto de las pruebas de concepto puede encontrarse en el repositorio de github¹¹.

¹¹ En el siguiente enlace encontrara el codigo fuente del prototipo del sistema en formato Jupyter Notebook
<https://github.com/AlejoTorres2001/AI/blob/windows/ProyectoDeGrado/Anexos/AnexoVIII-Prototipo.ipynb>

Sintetizar textos completos usando clustering

En el método anterior pasamos todos los tokens al modelo, ya que su capacidad lo permitía, ahora que sucede si tenemos un texto mas largo? Necesitamos encontrar una forma que nos permita identificar los párrafos mas importantes y los conceptos claves que formaran parte de la directiva para crear el resumen Usaremos el libro [Into Thin Air](#) acerca del accidente de 1996 en el Everest

```
1 from langchain.document_loaders import PyPDFLoader
2
3 # Load the book
4 loader = PyPDFLoader("../data/IntoThinAirBook.pdf")
5 pages = loader.load()
6
7 # Cut out the open and closing parts
8 pages = pages[26:277]
9
10 # Combine the pages, and replace the tabs with spaces
11 text = ""
12
13 for page in pages:
14     text += page.page_content
15
16 text = text.replace('\t', ' ')
```

Ilustración 27: Fragmento de código I, Fuente: Elaboración propia

Como se ve en la Ilustración 27 el proceso de carga y extracción del texto del archivo PDF se realiza utilizando la libreria PyPDFLoader, la cual es invocada a través de un *wrapper* incluido dentro de Langchain. Se procede a dejar de lado las páginas que no contienen información útil para luego concatenar el texto y eliminando carácter especial de tabulación .

```
1 num_tokens = llm.get_num_tokens(text)
2
3 print (f"This book has {num_tokens} tokens in it")
```

... This book has 139472 tokens in it

Lo que haremos sera, dividir el texto en documentos manejables, luego obtendremos los vectores de cada documento a través de un modelo de embeddings y los agruparemos en clusters. Luego seleccionaremos los documentos que representen mejor a cada cluster obteniendo aquellos que estén mas cercanos a cada uno de los respectivos centroides y finalmente crearemos un resumen con esos documentos. Las etapas del proceso son las siguientes:

1. Cargar el libro en un único archivo de texto
2. Separar el texto en documentos manejables
3. Obtener los vectores de cada documento a través de un modelo de embeddings
4. Agrupar los vectores en clusters para identificar los conceptos clave
5. Obtener los documentos mas cercanos a cada uno de los centroides
6. Crear un resumen con los documentos seleccionados

Ilustración 28: Fragmento de código II, Fuente: Elaboración propia

Instanciando el modelo del lenguaje se puede obtener una aproximación de la cantidad de tokens en los que se separa el texto.

```

1 # Loaders
2 from langchain.schema import Document
3
4 # Splitters
5 from langchain.text_splitter import RecursiveCharacterTextSplitter
6
7 # Model
8 from langchain.chat_models import ChatOpenAI
9
10 # Embedding Support
11 from langchain.vectorstores import FAISS
12 from langchain.embeddings import OpenAIEmbeddings
13
14 # Summarizer we'll use for Map Reduce
15 from langchain.chains.summarize import load_summarize_chain
16
17 # Data Science
18 import numpy as np
19 from sklearn.cluster import KMeans

```



```

1 text_splitter = RecursiveCharacterTextSplitter(separators=["\n\n", "\n", "\t"],
2 chunk_size=10000, chunk_overlap=3000)
3 docs = text_splitter.create_documents([text])

```

Ilustración 29: Fragmento de código III, Fuente: Elaboración propia

Como se ve en la Ilustración 29, se instancia el separador recursivo de caracteres, especificando los parámetros de tamaño y overlap y los posibles separadores a tener en cuenta, los cuales son caracteres especiales.

```

1 text_splitter = RecursiveCharacterTextSplitter(separators=["\n\n", "\n", "\t"],
2 chunk_size=10000, chunk_overlap=3000)
3 docs = text_splitter.create_documents([text])

```



```

1 num_documents = len(docs)
2
3 print (f"Now our book is split up into {num_documents} documents")

```

Now our book is split up into 78 documents

obtenemos las representaciones vectoriales de estos 78 documentos

```

1 embeddings = OpenAIEmbeddings(openai_api_key=openai_api_key)
2
3 vectors = embeddings.embed_documents([x.page_content for x in docs])

```

Para agrupar los documentos usaremos el algoritmo de KMedias

```

1 num_clusters = 11
2
3 # Perform K-means clustering
4 kmeans = KMeans(n_clusters=num_clusters, random_state=42).fit(vectors)

```

Ilustración 30: Fragmento de código IV, Fuente: Elaboración propia

Utilizando estos documentos, se carga un modelo de embeddings para luego generar los vectores correspondientes a cada fragmento de texto. El paso siguiente es definir el número de clusters y aplicar el algoritmo de K-medias sobre los vectores.

```

visualizamos las las agrupaciones formadas

1 kmeans.labels_
[48]
array([ 2,  2,  2,  2,  2,  2,  6,  9,  9,  4,  4,  4,  4,  9,  9,  9,  9,
        4,  4,  4,  4,  9,  9,  5,  4,  0,  0,  0,  5,  5,  5,  9,  9,  3,
        3,  9,  3,  3,  3,  3,  9,  9,  9,  3,  3,  3,  3,  7,  7,  7,
        7,  7,  6,  6,  6,  6,  6,  6,  9,  1,  1,  3,  4,  4,  1,  1,  1,
        1,  9, 10, 10, 10,  8,  8,  8,  8,  8], dtype=int32)

```

Ilustración 31: Fragmento de código V, Fuente: Elaboración propia

Se puede visualizar en la Ilustración 31 que cada uno de los documentos fue categorizado con una etiqueta correspondiente al número de cluster.

```

Vamos a realizar una reducción de la dimensionalidad de estos vectores para poder visualizarlos en un plano 2D

1 from sklearn.manifold import TSNE
2 import matplotlib.pyplot as plt
3
4 # Taking out the warnings
5 import warnings
6 from warnings import simplefilter
7
8 # Filter out FutureWarnings
9 simplefilter(action='ignore', category=FutureWarning)
10
11 # Perform t-SNE and reduce to 2 dimensions
12 tsne = TSNE(n_components=2, random_state=42)
13 reduced_data_tsne = tsne.fit_transform(vectors)
14
15 # Plot the reduced data
16 plt.scatter(reduced_data_tsne[:, 0], reduced_data_tsne[:, 1], c=kmeans.labels_)
17 plt.xlabel('Dimension 1')
18 plt.ylabel('Dimension 2')
19 plt.title('Book Embeddings Clustered')
20 plt.show()

```

Ilustración 32: Fragmento de código VI, Fuente: Elaboración propia

El siguiente paso, sirve únicamente a modo de verificación y no está incluido en el desarrollo del sistema final, se aplica una reducción de la dimensionalidad a los vectores para poder graficarlo en un plano de ejes cartesianos.

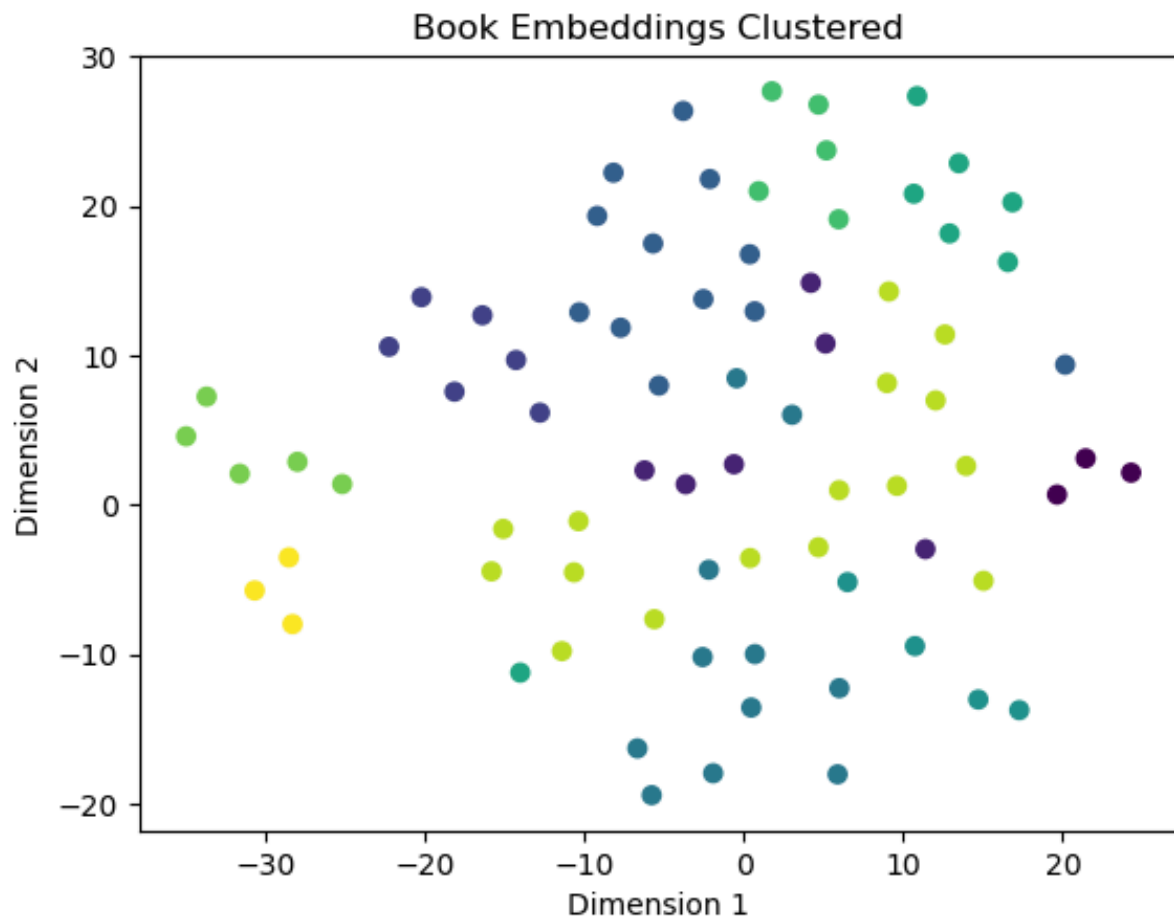


Ilustración 33: Visualización 2D de los Embeddings, Fuente: Elaboración propia

Podemos observar algunas agrupaciones claras

El siguiente paso es obtener aquellos documentos mas representativos de cada cluster

```

1 closest_indices = []
2
3 # Loop through the number of clusters you have
4 for i in range(num_clusters):
5
6     # Get the list of distances from that particular cluster center
7     distances = np.linalg.norm(vectors - kmeans.cluster_centers_[i], axis=1)
8
9     # Find the list position of the closest one (using argmin to find the
    smallest distance)
10    closest_index = np.argmin(distances)
11
12    # Append that position to your closest indices list
13    closest_indices.append(closest_index)

```

[43]

MagicPython

Ilustración 34: Fragmento de código VII, Fuente: Elaboración propia

En la Ilustración 34 se puede observar el algoritmo para obtener la distancia de cada uno de los vectores al cluster para luego obtener los índices de aquellos que se encuentren más cerca de su respectivo centroide.

ahora debemos ordenarlos para evitar perder el orden del texto original

```
1 selected_indices = sorted(closest_indices)
2 selected_indices
```

[44] MagicPython

```
... [0, 12, 26, 29, 39, 41, 51, 54, 65, 71, 75]
```

Lo que haremos a continuación sera resumir cada uno de los documentos seleccionados, para luego combinarlos y obtener el resumen final

```
1 llm3 = ChatOpenAI(temperature=0,
2         openai_api_key=openai_api_key,
3         max_tokens=1000,
4         model='gpt-3.5-turbo'
5     )
```

[45] MagicPython

Ilustración 35: Fragmento de código VIII, Fuente: Elaboración propia

```

1 map_prompt = """
2 You will be given a single passage of a book. This section will be enclosed in
3 triple backticks (````)
4 Your goal is to give a summary of this section so that a reader will have a
5 full understanding of what happened.
6 Your response should be at least three paragraphs and fully encompass what was
7 said in the passage.
8
9 ```{text}```
10 FULL SUMMARY:
11 """
12 map_prompt_template = PromptTemplate(template=map_prompt, input_variables=
13 ["text"])

```

[46] MagicPython

```

1 map_chain = load_summarize_chain(llm=llm3,
2                                  chain_type="stuff",
3                                  prompt=map_prompt_template)

```

[48] MagicPython

obtenemos los documentos en base a los indices seleccionados

```

1 selected_docs = [docs[doc] for doc in selected_indices]

```

[49] MagicPython

Ilustración 36: Fragmento de código IX Fuente: Elaboración propia

Se define entonces la plantilla para la directiva del modelo de lenguaje que se encargará de sintetizar cada fragmento de texto. Luego utilizando los índices anteriores, se recuperan los fragmentos de texto propiamente dichos.

iteramos sobre ellos y almacenamos los resúmenes individuales en una lista

```
1 summary_list = []
2
3 # Loop through a range of the length of your selected docs
4 for i, doc in enumerate(selected_docs):
5
6     # Go get a summary of the chunk
7     chunk_summary = map_chain.run([doc])
8
9     # Append that summary to your list
10    summary_list.append(chunk_summary)
11
12    print (f"Summary #{i} (chunk #{selected_indices[i]}) - Preview:
13          {chunk_summary[:250]} \n")
```

MagicPython

Ilustración 37: Fragmento de código X, Fuente: Elaboración propia

En la Ilustración 37 se ve cómo se ejecuta la cadena por cada uno de los documentos para luego agregarlos a una lista.

```
[50]
... Summary #0 (chunk #0) - Preview: The passage describes the author's experience of reaching the summit of Mo
Summary #1 (chunk #12) - Preview: In this passage, the author and his team are at Lobuje, a village at 16,2
Summary #2 (chunk #26) - Preview: In this passage, the author describes two events that occurred on Everest
Summary #3 (chunk #29) - Preview: In this passage, the author describes the equipment and personal items th
Summary #4 (chunk #39) - Preview: In this passage, the author describes his ascent up the Lhotse Face towar
Summary #5 (chunk #41) - Preview: In this passage, the author describes their experience waiting on a ledge
Summary #6 (chunk #51) - Preview: This passage describes the harrowing experience of a group of climbers on
Summary #7 (chunk #54) - Preview: In this passage, the author recounts the events of May 10, 1996, when Sco
Summary #8 (chunk #65) - Preview: The passage describes the aftermath of the tragic events that occurred du
Summary #9 (chunk #71) - Preview: The passage discusses the controversy surrounding Jon Krakauer's book, "I
Summary #10 (chunk #75) - Preview: In this passage, the author addresses the controversy surrounding his bo
```

MagicPython

Ilustración 38: Fragmento de código XI, Fuente: Elaboración propia

concatenamos los resúmenes en orden

```
1 summaries = "\n".join(summary_list)
2
3 # Convert it back to a document
4 summaries = Document(page_content=summaries)
5
6 print (f"Your total summary has {llm.get_num_tokens(summaries.page_content)}
7       tokens")
```

[51]

MagicPython

```
... Your total summary has 4002 tokens
```

Ilustración 39: Fragmento de código XII, Fuente: Elaboración propia

```
1 llm4 = ChatOpenAI(temperature=0,
2                 openai_api_key=openai_api_key,
3                 max_tokens=3000,
4                 model='gpt-4',
5                 request_timeout=120
6                 )

1 combine_prompt = """
2 You will be given a series of summaries from a book. The summaries will be
3 enclosed in triple backticks (```)
4 Your goal is to give a verbose summary of what happened in the story.
5 The reader should be able to grasp what happened in the book.
6 ```{text}```
7 VERBOSE SUMMARY:
8 """
9 combine_prompt_template = PromptTemplate(template=combine_prompt,
10                                         input_variables=["text"])

1 reduce_chain = load_summarize_chain(llm=llm4,
2                                     chain_type="stuff",
3                                     prompt=combine_prompt_template,
4                                     # verbose=True # Set this to true if you want to
5                                     # see the inner workings
6                                     )
```

Ilustración 40: Fragmento de código XIII, Fuente: Elaboración propia

Finalmente se define la plantilla de la directiva para realizarla síntesis general, uniendo los resultados de la etapa anterior y ejecutando la cadena.

```
ejecutamos la cadena de síntesis

1 output = reduce_chain.run([summaries])

1 print (output)

In this story, the author recounts their experience as part of a New Zealand-based team attempting to summi
The story also highlights the challenges and dangers of high-altitude mountaineering, as well as the import
Throughout the story, the author describes various events and challenges faced by the climbers, including m
As the climbers continue their ascent, they face physical and mental challenges, as well as the individuali
In the aftermath of the tragedy, the author reflects on the emotional impact, including survivor's guilt an
```

Ilustración 41: Fragmento de código XIV, Fuente: Elaboración propia

Basándose en este prototipo se construyeron los servicios de inferencia para ser expuestos a través de una API web que será consumida por la interfaz de usuario. Los mismos se desarrollaron y documentaron siguiendo las buenas prácticas y principios de la programación con el objetivo de conseguir un módulo que facilita el trabajo a lo largo de la construcción e integración del flujo de datos completo.

Para afrontar los extensos tiempos de espera en la generación de una síntesis, se propuso una solución basada en hilos de ejecución paralelos, encargados de interactuar con la API de OpenAI de forma asíncrona para así paralelizar la síntesis de los documentos seleccionados como se puede ver la Ilustración 42.

```
async def summarize_docs(llm: BaseLanguageModel, prompt: str, input_variables:
list[str], documents: list[Document]):
    try:
        summary_list = []
        map_chain = create_map_summarization_chain(
            llm=llm, prompt=prompt, input_variables=input_variables)

        with concurrent.futures.ThreadPoolExecutor() as executor:
            future_to_doc = {executor.submit(
                run_summarization, map_chain, doc): doc for doc in documents}

            for future in concurrent.futures.as_completed(future_to_doc):
                doc = future_to_doc[future]
                try:
                    chunk_summary = future.result()
                    summary_list.append(chunk_summary)
                except Exception as e:
                    print(
                        f"Error processing document: {doc.page_content}, error:
                        {e}")

        return summary_list
    except Exception as e:
        raise Exception("Could not summarize documents.")
```

Ilustración 42: función encargada de paralelizar la síntesis de documentos, Fuente: Elaboración propia.

El componente crucial para la paralelización es el uso de *concurrent.futures.ThreadPoolExecutor*. Este módulo de Python permite ejecutar llamadas a funciones de forma asíncrona y paralela utilizando un pool de hilos. En este caso, se crea un *ThreadPoolExecutor* y se utiliza un diccionario para mapear cada documento a un futuro. Cada futuro es una instancia de *concurrent.futures.Future* que representa una abstracción que encapsula una llamada a la función *run_summarization* para un documento específico, pudiendo obtener su resultado o manejar su excepción, enviándola al pool de hilos.

La ejecución paralela se maneja mediante un bucle que itera sobre una construcción permite procesar los futuros a medida que van completándose, sin necesidad de esperar a que todas las tareas finalicen. Para cada futuro completado, se obtiene el documento asociado y se intenta extraer el resultado de la síntesis. Si la llamada es exitosa, el resumen generado se añade a la lista. En caso de que ocurra una excepción durante el procesamiento de un documento, se captura e imprime un mensaje de error detallando el contenido del documento y la naturaleza de este.

Este enfoque tiene varias ventajas en términos de optimización de tiempo de ejecución. La paralelización mediante hilos permite que múltiples documentos se procesen simultáneamente, reduciendo significativamente el tiempo total necesario para sintetizar grandes conjuntos de documentos. Además, el uso de futuros simplifica la gestión de la concurrencia y el manejo de excepciones, haciendo el código más robusto y eficiente.

Desarrollo del Prototipo de la Interfaz de Usuario

A continuación, se muestran capturas del prototipo de la interfaz web de usuario, construída utilizando la librería de desarrollo web ReactJs donde se puede ver la configuración de los diferentes parámetros involucrados en el flujo de inferencia y la posibilidad de descargar la síntesis final en formato PDF o copiándola al portapapeles del usuario conectado.

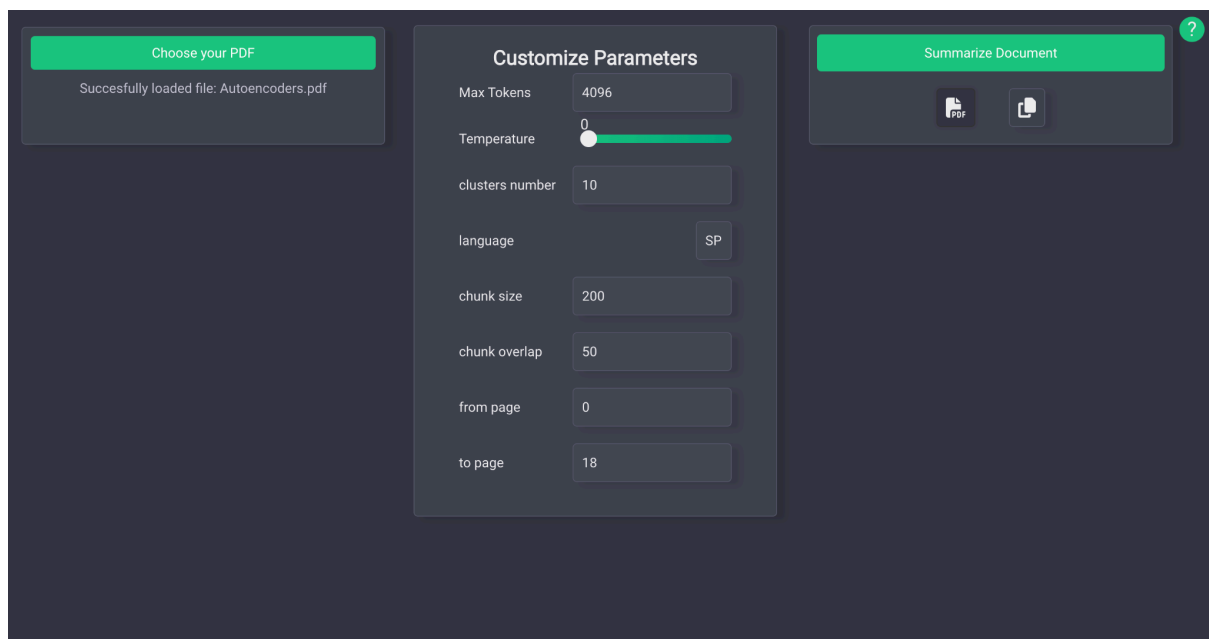
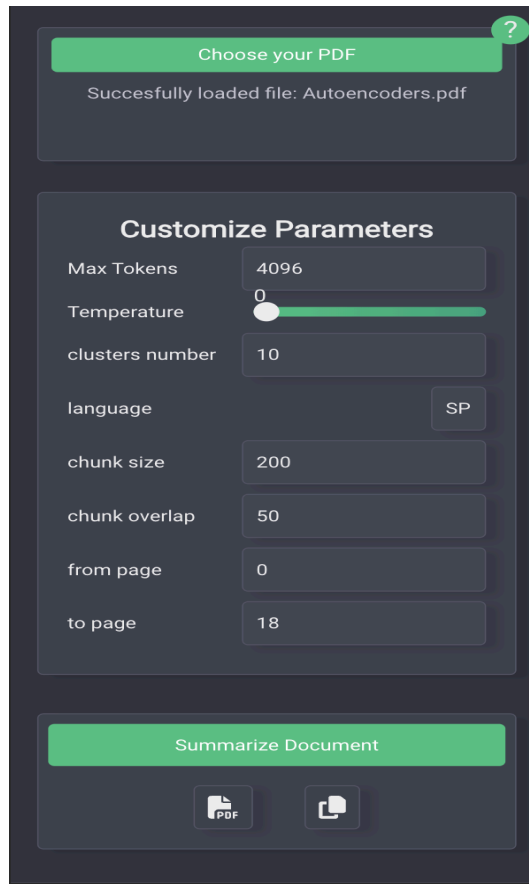


Ilustración 43: Interfaz de usuario / Fuente: Elaboración propia

La interfaz también cuenta con un modo responsivo para adaptarse a cualquier tamaño de pantalla del dispositivo.



The image shows a dark-themed web application interface. At the top, there is a green button labeled "Choose your PDF" with a small green circle containing a question mark to its right. Below this button, a message states "Succesfully loaded file: Autoencoders.pdf". The main section is titled "Customize Parameters" and contains several input fields and a slider. The parameters are: "Max Tokens" (4096), "Temperature" (a slider set to 0), "clusters number" (10), "language" (a dropdown menu showing "SP"), "chunk size" (200), "chunk overlap" (50), "from page" (0), and "to page" (18). At the bottom of this section is a green button labeled "Summarize Document". Below the button are two icons: a PDF icon and a copy icon.

Parameter	Value
Max Tokens	4096
Temperature	0
clusters number	10
language	SP
chunk size	200
chunk overlap	50
from page	0
to page	18

Ilustración 44: Interfaz de usuario II Fuente: Elaboración propia

Capítulo 6: Resultados

En este capítulo se detallan las pruebas realizadas sobre el sistema en un ambiente pre-productivo que puede resumirse mediante el diagrama de la Ilustración 45. Se tomaron los servicios desarrollados en el prototipo para construir una API web la cual expone el flujo de inferencia a través del protocolo de comunicación HTTP. Si desea revisar el código fuente del sistema completo, tanto del *backend* como el *frontend*, puede referirse al link del repositorio github¹².

Como se puede observar en la ilustración 45, a través de la interfaz web, el usuario puede cargar el documento a resumir en formato PDF junto con los parámetros *max_okens*, *Temperature*, *clusters_number*, *language*, *chunk_size*, *chunk_overlap* y *from_page*, *to_page* los cuales delimitan la sección del documento a sintetizar. Estos parámetros corresponden a la configuración de las distintas etapas del proceso. Estos datos son comunicados al servidor web a través de una AP a través del protocolo de comunicación de sistemas HTTP. El servidor web, valiéndose de los diferentes servicios, evalúa la petición y procesa la información a través del pipeline de datos, el cual producirá las estructuras necesarias para ejecutar los servicios de inferencia, comunicándose con el proveedor de servicios tercerizado, para luego construir una respuesta a la petición original.

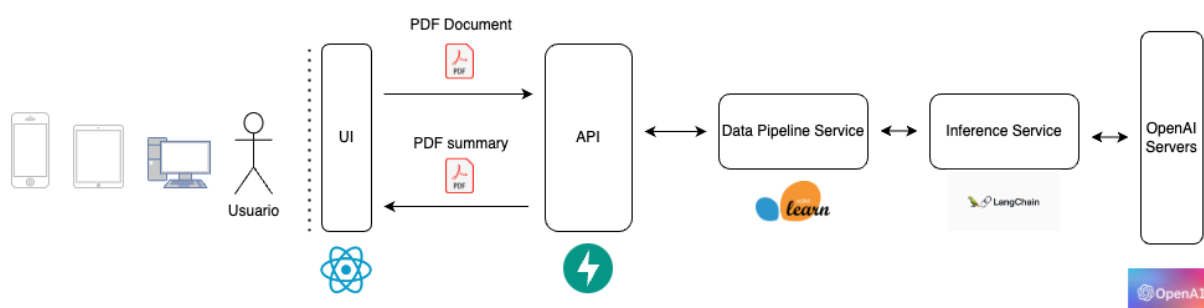


Ilustración 45: Diseño del sistema, Fuente: Elaboración propia

Las pruebas fueron realizadas para evaluar el desempeño del sistema en un ambiente pre productivo. Se seleccionaron una serie de documentos representativos para cada tipo de contenido a resumir, los cuales fueron procesados utilizando diferentes conjuntos de parámetros. Es importante destacar que estas pruebas se realizaron como una valoración subjetiva basada en criterios predefinidos, y no buscan reemplazar la percepción ni el estudio humano del contenido. Se considera que las pruebas realizadas son más que

¹² <https://github.com/AlejoTorres2001/AI/tree/windows/apps/synthai> en el siguiente enlace encontrara el código del sistema completo contemplando los servicios del lado del servidor construidos para poder operar el flujo de datos completo, junto con la aplicación web que funciona de interfaz de usuario

suficientes para evaluar la efectividad del sistema como una herramienta de apoyo en la generación de síntesis de texto.

Los documentos utilizados en las pruebas fueron seleccionados cuidadosamente para representar una variedad de tipos de contenido, incluyendo documentos técnicos, artículos académicos, textos de divulgación y obras de literatura clásica. Se eligieron parámetros específicos para cada prueba basados en el tipo de contenido y los objetivos del resumen esperado.

Cada prueba se basó en criterios específicos relevantes para el tipo de contenido del documento y los objetivos del resumen. Se consideraron aspectos como la capacidad del sistema para reconocer la idea principal, identificar las partes clave del documento, explicar las tecnologías involucradas, y comprender el objetivo del texto. Estos criterios se utilizaron como base para evaluar el rendimiento del sistema y determinar la calidad de los resúmenes generados.

A la hora de la evaluación de los resultados se tuvo en cuenta como guía general los aspectos que se detallan a continuación, teniendo en cuenta que esta herramienta no busca reemplazar la lectura y análisis que una persona puede llegar a realizar sobre un material literario cualquiera. Los criterios de evaluación son de carácter general y no buscan funcionar como una justificación lingüística profesional sobre la calidad de los resultados.

- *Coherencia*: La síntesis mantiene una estructura lógica y coherente en relación con el texto original.
- *Relevancia*: La síntesis incluye información relevante y significativa del texto original.
- *Consistencia*: La síntesis presenta consistencia en el uso del lenguaje y el tono en comparación con el texto original.
- *Precisión*: La síntesis refleja con precisión las ideas principales y los detalles importantes del texto original.
- *Comprensión*: La síntesis demuestra una comprensión clara y completa del contenido del texto original.
- *Veracidad*: La síntesis presenta las ideas del texto original de manera original y no simplemente copia y pega fragmentos del texto.

Las opciones de parametrización del flujo completo están sujetas a los siguientes parámetros:

- *Max Tokens*: hace referencia a la cantidad de tokens máxima con la que contará el Documento resumido
- *Temperature*: se refiere a un parámetro de ajuste que se utiliza durante la generación de texto. La temperatura es un valor numérico que controla la aleatoriedad en la generación de contenido

- *Clusters number*: número de centroides a utilizar en el agrupamiento de documentos
- *Language*: lenguaje de salida para el resumen
- *Chunk size*: tamaño de los fragmentos en los que se dividirá el documento original para ser procesado
- *Chunk overlap*: solapamiento entre fragmentos de texto. Con el propósito de conservar contexto entre cada uno de ellos
- *From page*: página del documento desde la cual comenzar la extracción de contenido
- *To page*: página del documento donde finalizar la extracción de contenido

Tabla 3: Configuración de las pruebas, Fuente: Elaboración propia

Prueba N°	Documento utilizado	Extensión del Documento (páginas)	Parámetros utilizados
1	Anteproyecto referido a este Trabajo de fin de Grado	5	<i>Max Tokens</i> : 4096 <i>Temperature</i> : 0.2 <i>clusters number</i> : 8 <i>language</i> : SP <i>chunk_size</i> : 140 <i>chunk_overlap</i> : 30 <i>from_page</i> : 1 <i>to page</i> : last
2	Attention is all you need: artículo que introduce el mecanismo de atención basado en similitud de cosenos	15	<i>Max Tokens</i> : 4096 <i>Temperature</i> : 0.0 <i>clusters number</i> : 10 <i>language</i> : EN <i>chunk_size</i> : 200 <i>chunk_overlap</i> : 50 <i>from_page</i> : 1 <i>to_page</i> : 9

3	EERP:Empresa extendida. Estrategia e informática.Notas de la cátedra	60	<i>Max Tokens:</i> 4096 <i>Temperature:</i> 0.0 <i>clusters number:</i> 11 <i>language:</i> SP <i>chunk_size:</i> 220 <i>chunk_overlap:</i> 50 <i>from_page:</i> 1 <i>to_page:</i> last
4	Protección,Seguridad y Privacidad en los datos de los metaversos: artículo científico.	7	<i>Max Tokens:</i> 4096 <i>Temperature:</i> 0.0 <i>clusters number:</i> 9 <i>language:</i> SP <i>chunk_size:</i> 170 <i>chunk_overlap:</i> 40 <i>from_page:</i> 1 <i>to_page:</i> 6
5	Estrategia e informática:Gestión del Cambio.Notas de la cátedra	45	<i>Max Tokens:</i> 4096 <i>Temperature:</i> 0.0 <i>clusters number:</i> 12 <i>language:</i> SP <i>chunk_size:</i> 250 <i>chunk_overlap:</i> 70 <i>from_page:</i> 0 <i>to_page:</i> 44
6	Mi planta de naranja lima	203	<i>Max Tokens:</i> 4096 <i>Temperature:</i> 0.0 <i>clusters number:</i> 10 <i>language:</i> SP <i>chunk_size:</i> 450 <i>chunk_overlap:</i> 70 <i>from_page:</i> 0 <i>to_page:</i> 202

7	La metamorfosis: Franz Kafka	40	<i>Max Tokens:</i> 4096 <i>Temperature:</i> 0.0 <i>clusters number:</i> 11 <i>language:</i> SP <i>chunk_size:</i> 150 <i>chunk_overlap:</i> 30 <i>from_page:</i> 0 <i>to_page:</i> 39
8	Estrategia e Informática: Introducción. Notas de Clase	36	<i>Max Tokens:</i> 4096 <i>Temperature:</i> 0.0 <i>clusters number:</i> 10 <i>language:</i> SP <i>chunk_size:</i> 100 <i>chunk_overlap:</i> 40 <i>from_page:</i> 0 <i>to_page:</i> 35
9	Educating to Compete: Pandemic-Era Patterns of Technology Incorporation in the Southern Cone. Trabajo de fin de grado de M. Candelaria Torres	127	<i>Max Tokens:</i> 4096 <i>Temperature:</i> 0.0 <i>clusters number:</i> 12 <i>language:</i> EN <i>chunk_size:</i> 250 <i>chunk_overlap:</i> 50 <i>from_page:</i> 0 <i>to_page:</i> 127
10	Autoencoders por Dor Bank, Noam Koenigstein, Raja Giryes. Artículo científico	22	<i>Max Tokens:</i> 4096 <i>Temperature:</i> 0.0 <i>clusters number:</i> 10 <i>language:</i> SP <i>chunk_size:</i> 200 <i>chunk_overlap:</i> 50 <i>from_page:</i> 0 <i>to_page:</i> 18

A continuación, se describen los comentarios sobre cada una de las síntesis generadas en cada caso de prueba. Si desea leer el documento completo puede referirse al Anexo I en la sección de anexos del documento.

Prueba N1: Anteproyecto

Los resultados de la prueba de síntesis muestran una estructura coherente y lógica, identificando claramente las partes del proyecto y su idea principal. Se destaca la relevancia al explicar las tecnologías involucradas y el objetivo del trabajo. Sin embargo, se introduce información no presente en el texto original, lo que afecta la veracidad de la síntesis. En términos de consistencia, la redacción es adecuada, y la precisión se refleja en la transmisión clara de las ideas principales. La comprensión del contenido original se demuestra al explicar detalladamente las tecnologías involucradas y el objetivo del proyecto.

Una vez desarrollada la aplicación, los resultados obtenidos serán expuestos a expertos en el tema para su evaluación y se diseñará una interfaz amigable e intuitiva para facilitar el acceso y uso de los resultados obtenidos a través de la aplicación.

Además, el artículo también analiza la importancia de la **inteligencia emocional en el ámbito laboral**. La inteligencia emocional, que se refiere a la capacidad de reconocer, comprender y gestionar nuestras

propias emociones y las de los demás, es fundamental para el éxito y el bienestar tanto individual como organizacional. La inteligencia emocional influye en la toma de decisiones, la productividad laboral y contribuye a la creación de un ambiente laboral positivo.

Ilustración 46: Tema no presente en el texto original aparece en la síntesis, Fuente: Elaboración propia

Se puede mencionar que el uso de un valor nulo en el parámetro de *temperatura* del modelo, el cual restringe la libertad, permite obtener resultados mucho más acertados. Por ejemplo, en el caso de prueba *Nº1* se utilizó 0.2 como valor para la *temperatura* del modelo obteniendo así un resumen que mencionaba temas de inteligencia emocional, cuando la idea principal del documento era la inteligencia artificial.

Prueba N2: Attention is all you need

Se puede ver una síntesis coherente y relevante que explica el paralelismo en el cómputo y compara la secuencialidad de las redes recurrentes con la naturaleza paralela del mecanismo de atención. Además, se incluyen detalles sobre las capas de *self-attention* en el *decoder* y el *encoder*, así como las ventajas del mecanismo de atención. Sin embargo,

la síntesis no aborda la arquitectura completa en detalle, lo que podría afectar la precisión y la veracidad de la representación del texto original. Aunque hay consistencia en el uso del lenguaje y el tono, la omisión de esta información podría limitar la comprensión integral del lector.

Prueba N3: EERP: Empresa extendida. Estrategia e informática

La síntesis realizada demuestra una comprensión clara y completa del contenido del texto original sobre la Empresa Extendida, Estrategia e Informática. Se destacan los niveles de integración, resaltando la importancia de la comunicación entre las diferentes áreas de la empresa. Además, se subraya la relevancia de los procesos de negocio principales en cada área, evidenciando una selección precisa de la información significativa del texto original. La consistencia en el uso del lenguaje y el tono se mantiene, y se presenta de manera coherente, reflejando la estructura lógica del texto original. La precisión se manifiesta en la captura precisa de las ideas principales y los detalles importantes, como la mención de SCM, CRM y el Marco Empresarial Extendido. Además, la síntesis muestra originalidad al presentar las ideas del texto original de manera propia, evitando simplemente copiar y pegar fragmentos.

Prueba N4: Protección, Seguridad y Privacidad en los datos de los metaversos: artículo científico

Se destaca la coherencia en la estructura, manteniendo una secuencia lógica. La relevancia se evidencia en la inclusión de información significativa, como la definición de metaversos y la importancia de la búsqueda de vulnerabilidades en estos entornos virtuales. La consistencia en el uso del lenguaje y el tono se mantiene en relación con el texto original, mientras que la precisión se refleja en la captura precisa de las ideas principales y los detalles importantes, como la implicancia del gran volumen de información compartida y la importancia de la educación para los usuarios de metaversos. La veracidad se manifiesta en la presentación original de las ideas, evitando la copia de fragmentos del artículo.

Prueba N5: Estrategia e informática: Gestión del Cambio

La relevancia se evidencia en la inclusión de información significativa, como la importancia de la gestión del cambio como vínculo clave y las implicancias asociadas a esta gestión. Además, se presenta con consistencia en el uso del lenguaje y el tono, reflejando la terminología y el estilo del texto original. La precisión se manifiesta en la captura precisa de las ideas principales y los detalles importantes, como las fases del modelo de gestión del

cambio y el énfasis en comunicar el cambio de manera efectiva y la estrategia del Océano Azul. En cuanto a la veracidad, la síntesis presenta las ideas de manera original.

Prueba N6: Mi planta de naranja lima

Se destaca la coherencia en la estructura, que refleja la disposición del texto original al identificar al autor y ubicar a los personajes en relación entre sí. La relevancia se evidencia al detallar las escenas más importantes y reconocer el modismo regional en los diálogos. La consistencia en el uso del lenguaje y el tono se mantiene, al igual que la precisión al reflejar con exactitud las ideas principales y detalles relevantes, como el carácter surrealista del sueño de Zezé. Sin embargo, la síntesis no contempla el final de la historia, lo que podría afectar su integridad. En cuanto a la veracidad, presenta las ideas de manera original, respetando la esencia del texto original.

Prueba N7: La metamorfosis, Franz Kafka

Se evidencia una secuencia lógica que identifica a los personajes, rescata escenas importantes y detalla los sentimientos del protagonista. La relevancia se evidencia en la inclusión de información significativa y relevante para comprender la trama y el desarrollo del personaje principal. La consistencia en el uso del lenguaje y el tono se mantiene en comparación con el texto original, mientras que la precisión se refleja en la captura precisa de las ideas principales y los detalles importantes. Sin embargo, la síntesis no incluye interpretaciones del relato, lo que podría haber enriquecido aún más su profundidad. En cuanto a la veracidad, presenta las ideas del texto original de manera original, evitando simplemente copiar y pegar fragmentos.

Prueba N8: Estrategia e Informática, Introducción

Se observa una estructura coherente, que destaca la importancia del modelo de las 5 fuerzas de Porter, así como de la diferenciación y la visión estratégica. Se incluye información relevante sobre las figuras citadas y se enfatiza la importancia de la estrategia para obtener ventajas competitivas. La consistencia en el lenguaje y el tono se mantiene, y se refleja con precisión la esencia de las ideas principales y los detalles importantes. Además, se comprende claramente el proceso de planificación estratégica para una organización. La síntesis presenta estas ideas de manera original, evitando simplemente repetir fragmentos del texto original.

Prueba N9: Educating to Compete, Trabajo de fin de Grado

Muestran una comprensión clara y completa del contenido original. Se destaca la coherencia en la estructura, resumiendo eficazmente la investigación empírica, la creación de variables y las fuentes de datos utilizadas. Además, se realiza una compilación coherente del análisis comparativo entre Uruguay y Argentina, así como de las conclusiones sobre la resiliencia educativa frente al impacto del COVID-19 en América Latina. La relevancia se evidencia en la inclusión de puntos clave sobre dicho impacto en el sistema educativo regional. Sin embargo, se observa una falta de resumen de la teoría internacional de competencia entre naciones, limitándose sólo a la economía del conocimiento. La consistencia en el uso del lenguaje y el tono se mantiene en relación con el texto original, al igual que la precisión en la captura de ideas principales y detalles importantes. En cuanto a la veracidad, la síntesis presenta las ideas de manera original, evitando simplemente copiar y pegar fragmentos del texto original.

Prueba N10: Autoencoders, paper científico

Los resultados de la síntesis del artículo científico "Autoencoders" reflejan una comprensión clara y completa del contenido original. Se observa una estructura coherente, destacando la ecuación de log-verosimilitud y la importancia del error de reconstrucción, así como la mención de la distancia de Wasserstein y las capacidades de extracción de características de los autoencoders. La relevancia se evidencia en la inclusión de información significativa para comprender el tema. La consistencia en el uso del lenguaje y el tono se mantiene, y se refleja con precisión la esencia de las ideas principales y los detalles importantes. Además, se comprende claramente la conclusión del artículo.

Capítulo 7: Conclusiones

El prototipo desarrollado permite la creación de resúmenes competentes sobre cualquier documento o corpus de texto que le sea suministrado. Es capaz de identificar actores y conceptos, destacar ideas principales y presentar la información en un formato coherente. Gracias al mecanismo de atención utilizado es capaz de retener contexto extenso cuando trabaja con obras literarias, procesarlas y brindar una síntesis de las partes relevantes. Los resúmenes producidos y analizados no buscan ser una sustitución directa al documento original puesto que se puede apreciar una pérdida de información durante el flujo de inferencia, la cual es aceptable considerando la forma de operar el sistema. Sin embargo, comprende una potente herramienta para utilizarse como una primera aproximación al contenido o un repaso a posteriori de la lectura completa del mismo.

Otro aspecto interesante es la capacidad de generar resúmenes en un idioma independiente al idioma del documento original, sin perder calidad, gracias a la arquitectura subyacente del LLM utilizado. El idioma objetivo es un parámetro configurable dentro de la aplicación.

Por último, destacar que el sistema fue probado en un ambiente pre-productivo de la plataforma web, donde se pudo observar que los tiempos de inferencia tienden a ser considerablemente extensos.

A manera de conclusión, se puede decir que el objetivo general del proyecto, que fue el de construir una solución tecnológica que permitiera sintetizar el contenido de un archivo en formato PDF usando técnicas de Machine Learning y modelos de lenguajes, se cumplió con éxito, respetando el alcance planteado. Se logró desarrollar una herramienta eficaz y competente para la síntesis de texto. El sistema demostró ser una solución valiosa para la creación de síntesis, tanto en el idioma original del documento como en otros idiomas, manteniendo una alta calidad y coherencia en la presentación de la información.

Las pruebas realizadas en el ambiente pre-productivo permitieron identificar áreas de mejora, especialmente en términos de tiempo de inferencia. Si bien desde un inicio el desarrollo del sistema se enfocó en paralelizar la mayor cantidad de líneas de ejecución posibles utilizando técnicas de *multi procesamiento* y *pools* de procesos, la performance del mismo depende en gran medida de la disponibilidad de los servicios tercerizados, tanto de los *embeddings* de palabras como de los *LLMs* involucrados. Una forma de mitigarlo, como una mejora a futuro, consistiría en la utilización de tareas en segundo plano y notificaciones *push* del lado del cliente, por ejemplo, una solución basada en *ServiceWorkers*.

En cuanto a futuras líneas de trabajo, se podría explorar la optimización del sistema para reducir aún más los tiempos de inferencia, así como la integración con otros servicios y herramientas para ampliar las capacidades del sistema y ofrecer una solución aún más robusta y eficiente. Respecto a la mejora en la calidad y coherencia de los resúmenes, se pueden realizar pruebas con arquitecturas como MAMBA o técnicas de *selective state modeling*, las cuales se desprenden del mecanismo de atención y las capas *feed forward* tradicionales y plantean una alternativa más eficiente y de manejo de secuencias mucho más amplias que la arquitectura transformer.

Además, integrar un sistema de *feedback* donde los usuarios puedan calificar la calidad de los resúmenes generados puede ser muy útil. Este *feedback* puede ser utilizado para entrenar modelos en un enfoque de aprendizaje supervisado continuo, ajustando los parámetros del modelo y mejorando su rendimiento con el tiempo basado en las preferencias y necesidades reales de los usuarios. También se pueden implementar técnicas de análisis semántico y post-procesamiento para mejorar la coherencia y fluidez de los resúmenes. Por ejemplo, el uso de herramientas de procesamiento de lenguaje natural para corregir errores gramaticales, ajustar la estructura de las frases y asegurar la cohesión semántica en los resúmenes generados, como una capa de verificación adicional al resultado obtenido del modelo del lenguaje.

Anexo I: Resultado de las pruebas

El siguiente anexo contiene las síntesis finales obtenidas para cada uno de los casos de prueba.

Prueba N1: Anteproyecto

La Mg. Lic. María Lorena Talamé de la Universidad Católica de Salta ha sido seleccionada para liderar un proyecto que busca desarrollar una aplicación web capaz de resumir información contenida en archivos PDF utilizando un modelo de lenguaje de aprendizaje profundo (LLM). Talamé, reconocida por su experiencia en inteligencia artificial, supervisará el progreso del proyecto y brindará orientación a los estudiantes involucrados. El objetivo principal del proyecto es crear una herramienta que agilice y mejore los procesos de estudio e investigación. La aplicación será capaz de analizar el contenido de un archivo PDF, identificar las ideas principales y generar un resumen conciso. Se espera que sea fácil de usar, permitiendo a los usuarios cargar sus archivos, seleccionar las opciones de resumen y obtener un resumen completo en segundos. Además, se espera que la aplicación pueda adaptarse a diferentes tipos de documentos y ofrecer resúmenes personalizados según las necesidades del usuario. Para desarrollar esta aplicación, el autor explorará el ecosistema de desarrollo de aplicaciones impulsadas por IA, utilizando herramientas y librerías como LangChain, CTransformer, BitsAndBytes, StreamLit y ChainLit. También se explorarán las técnicas de Machine Learning a través de la revisión de papers originales, artículos científicos, seminarios y charlas informativas para obtener un conocimiento completo y profundo sobre estas técnicas. El desarrollo del software de síntesis de texto considerará varios aspectos, incluyendo la generación automática de texto coherente, una interfaz intuitiva y herramientas de edición y personalización. El software deberá ser capaz de analizar y comprender el contenido de diferentes fuentes de información y sintetizar la información de manera clara y concisa. Una vez desarrollada la aplicación, los resultados obtenidos serán expuestos a expertos en el tema para su evaluación y se diseñará una interfaz amigable e intuitiva para facilitar el acceso y uso de los resultados obtenidos a través de la aplicación. Además, el artículo también analiza la importancia de la inteligencia emocional en el ámbito laboral. La inteligencia emocional, que se refiere a la capacidad de reconocer, comprender y gestionar nuestras propias emociones y las de los demás, es fundamental para el éxito y el bienestar tanto individual como organizacional. La inteligencia emocional influye en la toma de decisiones, la productividad laboral y contribuye a la creación de un ambiente laboral positivo.

Prueba N2:Attention is all you need

The article discusses the application of parallel computing in models that compute hidden representations for all input and output positions simultaneously. These models are designed to process large amounts of data efficiently and quickly, making them particularly useful in tasks that involve processing large amounts of data, such as natural language processing or machine translation. The author mentions that the training process for their model took 3.5 days using 8 P100 GPUs, suggesting that the training process was computationally intensive and required the use of powerful GPUs to complete within a reasonable timeframe. The article also introduces the concept of aligned recurrence, a specific type of recurrent neural network architecture that has been proven to perform well in simple-language question answering and language modeling tasks. Recurrent neural networks can process sequential data by utilizing feedback connections, making them particularly useful in tasks that involve understanding and generating human language. However, the article also acknowledges that despite advancements in computing technology, there is still a limitation when it comes to sequential computation, which can slow down the overall speed and efficiency of computing systems. The use of self-attention layers in the decoder of an encoder-decoder model is also discussed. These layers enable each position in the decoder to focus on different parts of the input sequence, allowing the decoder to better understand the context and dependencies between different positions in the input sequence. This attention mechanism allows the decoder to assign different weights to different positions in the input sequence, depending on their relevance to the current position in the decoder, which enhances the performance and effectiveness of the encoder-decoder model. The article also highlights the successful performance of a big model on the WMT 2014 English-to-French translation task, as evidenced by its high BLEU score of 41.0. This achievement demonstrates the model's ability to accurately translate English text into French and suggests that it may employ advanced neural network techniques. The potential benefits of using self-attention in models are also discussed, with one of these benefits being that self-attention could lead to more interpretable models. By inspecting attention distributions, researchers and practitioners can gain a better understanding of how the model is attending to different parts of the input, which can help in interpreting its behavior and decision-making process. Finally, the article introduces the concept of Scaled Dot-Product Attention, a key component of the Transformer model, which is widely used in natural language processing tasks. This attention mechanism allows the model to focus on relevant parts of the input sequence while disregarding irrelevant information, improving its performance in various natural language processing tasks.

Prueba N3:ERP:Empresa extendida. Estrategia e informática

El artículo aborda la importancia de la integración en el ámbito empresarial, tanto en términos de negocios como de tecnología. Se destaca que las empresas están trabajando de manera más estrecha y colaborativa, lo que les permite aprovechar al máximo las oportunidades y enfrentar los desafíos de manera más efectiva. Esta integración se refleja en una percepción compartida entre los actores empresariales, lo que significa que todos tienen una comprensión común de los objetivos y las estrategias de la empresa. Se menciona una aplicación diseñada para apoyar la integración en el nivel de negocio y también en el nivel de sistemas físicos e infraestructura. Esta aplicación facilita la comunicación y la colaboración entre diferentes áreas de una empresa, permitiendo una mejor coordinación y eficiencia en las operaciones comerciales. Además, puede conectarse y trabajar en conjunto con los dispositivos y la infraestructura física de una organización, como sensores, máquinas y redes de comunicación. El artículo también destaca la importancia de los diferentes procesos de negocio relacionados con cada una de las áreas funcionales de una empresa. Estos procesos permiten la coordinación y el flujo eficiente de información y recursos entre los distintos departamentos. Cada área funcional tiene sus propios procesos de negocio, diseñados para cumplir con los objetivos y metas específicas de esa área. Es fundamental que exista una buena comunicación y colaboración entre las diferentes áreas funcionales para asegurar que los procesos de negocio se ejecuten de manera eficiente y efectiva. En el contexto de la cadena de suministro, se aborda la gestión de la demanda y el control de la distribución. La gestión de la demanda implica predecir y satisfacer las necesidades de los clientes de manera eficiente, mientras que el control de la distribución se refiere a la supervisión y gestión de los procesos de entrega de los productos a los clientes. También se mencionan los horizontes de acción y SCM, que se refieren a la planificación y ejecución de las actividades relacionadas con la cadena de suministro. El artículo también aborda la integración en el Marco Empresarial Extendido, en relación a tres aspectos clave: EE (Enterprise Engineering), CRM (Customer Relationship Management) y SCM (Supply Chain Management). La integración en este marco implica la implementación de estrategias y tecnologías que permitan una gestión más eficiente de los recursos, una mejor relación con los clientes y una mayor coordinación en la cadena de suministro. Se habla sobre EAI (Enterprise Application Integration), una solución que permite la integración de los sistemas de una empresa. A diferencia de una solución ERP, que proporciona una plataforma única para gestionar todas las operaciones de una empresa, EAI se enfoca en la integración de los sistemas existentes de una empresa para mejorar la comunicación y el intercambio de datos entre ellos. El uso de un ERP en una empresa es fundamental para mejorar la eficiencia y la productividad de los diferentes departamentos, al eliminar interfaces, agilizar procesos y garantizar la consistencia de los datos. Además, se destaca

el concepto de dato único, lo que significa que toda la información de la empresa se encuentra centralizada en un solo lugar. En el concepto de CRM (Customer Relationship Management), se menciona la tendencia de centrarse en el cliente. Esta tendencia implica que las empresas están poniendo un mayor énfasis en comprender y satisfacer las necesidades y deseos de sus clientes. Finalmente, se aborda el tema de la integración en diferentes niveles, incluyendo la integración en el nivel de sistemas físicos e infraestructura, la integración socio-humana, la integración de procesos de negocio y la integración de estrategias. Además, se habla sobre la importancia de la conexión de diferentes aplicaciones y sistemas heterogéneos a través de una infraestructura de integración de software, conocida como EAI (Enterprise Application Integration).

Prueba N4:Protección,Seguridad y Privacidad en los datos de los metaversos: artículo científico

El artículo aborda la importancia de los metaversos como una solución a problemas no especificados en el mundo físico. Los metaversos son entornos virtuales que utilizan tecnología para expandir nuestra realidad, proporcionando experiencias más inmersivas y efectivas. Sin embargo, el análisis de los metaversos también debe considerar las posibles vulnerabilidades del sistema, ya que los atacantes podrían explotarlas para acceder a los dispositivos y realizar acciones no autorizadas. Por lo tanto, es crucial realizar una búsqueda exhaustiva de posibles vulnerabilidades para garantizar la seguridad de los dispositivos utilizados en los metaversos. El artículo también menciona una serie de nombres que parecen ser los autores o colaboradores de la investigación. Aunque no se proporciona información específica sobre el estudio, la diversidad de nombres sugiere un enfoque multidisciplinario e internacional. Además, se hace referencia a un artículo publicado en el Internet of Things Journal en 2019, que revisó los estudios de evaluación experimental para despliegues de redes inalámbricas de malla en entornos urbanos. Este estudio proporciona una visión general de los despliegues de redes inalámbricas de malla y destaca la importancia de realizar evaluaciones experimentales para comprender mejor su rendimiento. El artículo "The social metaverse: Battle for privacy" de Loeb y Neff, publicado en la revista IEEE Technology and Society Magazine, se menciona para abordar el tema de la privacidad en el metaverso social. Los autores destacan que los usuarios comparten una gran cantidad de información personal en el metaverso social, lo que plantea preocupaciones sobre cómo se utiliza y protege esta información. Proponen la implementación de medidas de seguridad y privacidad, incluyendo políticas de privacidad claras y transparentes, y la educación de los usuarios sobre los riesgos y las mejores prácticas para proteger su información personal. Finalmente, se enfatiza la necesidad de priorizar la seguridad informática en el metaverso, generando una cultura de seguridad

entre los usuarios. Esto implica educar a los usuarios sobre las medidas de seguridad que deben tomar para proteger su información personal y evitar ser víctimas de ataques cibernéticos. Los usuarios también deben estar conscientes de los riesgos que pueden enfrentar al interactuar en este entorno virtual, como el robo de identidad, el fraude o la suplantación de identidad.

Prueba N5:Estrategia e informática:Gestión del Cambio

El Lic. Jorge A. Perdiguero, en su artículo, destaca la importancia de la gestión del cambio en las organizaciones. Según él, la gestión del cambio es un enfoque que ayuda a comprender el vínculo entre el proceso estratégico y el cambio organizacional. Este proceso es fundamental para que las organizaciones puedan adaptarse y responder de manera efectiva a los cambios en su entorno. La gestión del cambio implica identificar y gestionar los riesgos asociados con el cambio, así como también comunicar y capacitar a los empleados para que puedan adaptarse a los nuevos procesos y formas de trabajo. Perdiguero también enfatiza que simplemente cambiar las estructuras y sistemas no es suficiente para lograr un cambio efectivo. Es necesario también revisar y modificar las políticas existentes, ya que estas pueden ser un obstáculo para el cambio deseado. Al hacerlo, se busca aumentar la credibilidad del cambio y asegurar que se implementen las medidas necesarias para lograr los objetivos propuestos. Además, se enfatiza la importancia de promover a aquellos que aplican una visión de cambio, reconociendo y recompensando a aquellos individuos que están comprometidos con el cambio y que trabajan activamente para implementarlo. El autor propone un modelo de gestión del cambio basado en tres fases: diagnóstico, diseño e implementación. En la fase de diagnóstico, se analiza la situación actual de la organización y se identifican las áreas que requieren cambios. En la fase de diseño, se elabora un plan detallado para implementar los cambios necesarios, teniendo en cuenta los recursos disponibles y los objetivos de la organización. Finalmente, en la fase de implementación, se lleva a cabo el plan diseñado, asegurando la participación y el compromiso de todos los miembros de la organización. Perdiguero también habla sobre el proceso de desprendimiento personal que se extiende al ámbito social y llega hasta los ecosistemas para la creación de nuevos escenarios. Este proceso implica que el individuo debe liberarse de sus propias limitaciones y prejuicios para poder interactuar de manera más efectiva con los demás y con el entorno natural. El autor también destaca la importancia de implementar estrategias para lograr la competitividad y el rendimiento superior en cualquier ámbito. Estas estrategias son fundamentales para alcanzar el éxito y destacarse en un mercado cada vez más competitivo. Además, se menciona que la implementación de estas estrategias requiere de un enfoque integral, que involucre a todos los niveles de la organización. Perdiguero también aborda la importancia

de contar con un liderazgo estratégico en las organizaciones. Este tipo de liderazgo es fundamental para lograr los objetivos y metas de la empresa, ya que implica la capacidad de formular y ejecutar estrategias efectivas. Finalmente, el autor habla sobre la estrategia del Océano Azul y las cuatro barreras organizacionales que pueden dificultar su implementación. Estas barreras incluyen la mentalidad de la competencia, la falta de liderazgo y visión estratégica, la resistencia al cambio y la falta de recursos y capacidades. Superar estas barreras es fundamental para que las organizaciones puedan adoptar la estrategia del Océano Azul y encontrar nuevos espacios de mercado a través de la innovación y la diferenciación.

Prueba N6: Mi planta de naranja lima

El artículo se centra en la obra del autor José Mauro de Vasconcelos, destacando su habilidad para retratar la fidelidad al habla y los modismos propios de la zona en la que se desarrollan sus historias. Se resalta su destreza en el uso del diálogo, el cual se caracteriza por ser abundante y acertado. Vasconcelos solo escribe cuando siente que la novela fluye naturalmente a través de él, creando personajes y situaciones auténticas. Se describen varias escenas y diálogos de sus obras. En una de ellas, el narrador y su hermano Luis pasan una noche en casa con su padre, quien parece estar distraído y no interactúa con ellos como solía hacerlo. En otra escena, Zezé, un personaje de la obra, se siente maltratado y cree que Gum, otro personaje, no lo quiere realmente. Zezé revela que ha hecho algo que ha resultado en una herida en su pie, lo que preocupa a Gum.

En otra parte de la historia, Minguito, un personaje que pronto cumplirá seis años, tiene una conversación con otro personaje sobre alguien que constantemente toca la bocina cuando lo ve. Minguito ignora a esta persona, creyendo que le tiene miedo. En otra escena, el narrador se encuentra trabajando en la calle, pero es interrumpido por una mujer enfadada que lo arrastra hasta el comedor y lo tira contra la mesa, lo que provoca que el narrador se niegue a comer. El narrador, Zezé, se encuentra en un estado de tristeza y desesperación después de la muerte de alguien cercano. Busca consuelo en un árbol, considerándolo como su amigo y confidente. Sin embargo, Totoca, otro personaje, lo encuentra y trata de consolarlo. En otra escena, el narrador visita la casa de Dindinha y se encuentra con su tío Edmundo durmiendo en una hamaca. El diablo tienta al narrador a hacer una travesura con una caja de fósforos.

El narrador se siente culpable y remordido mientras camina por la calle, sintiéndose diferente a los demás niños que están mostrando y comparando sus juguetes. Se detiene cerca de un cafetín llamado "Miseria y Hambre", reflejando la situación de pobreza en la que

se encuentra. En otra escena, el narrador es amenazado y humillado por un individuo llamado el Portugués, pero responde con rabia que lo matará cuando crezca. En una experiencia surrealista, Minguito se transforma en un caballo volador y el narrador lo acompaña mientras vuelan. Sin embargo, la tranquilidad se ve interrumpida por el sonido de un tren a lo lejos. En otra escena, el protagonista se encuentra con su padre, quien es comparado con un viejo árbol de raíces oscuras. El protagonista revela que su planta de naranja-lima será cortada pronto, pero él ya no le importa. Ambos lloran por la pérdida de la planta.

Finalmente, Zezé se niega a llevar un paquete de castañas y otras cosas a su hermanito, Sergito, porque tiene que terminar de trabajar. Sergito descubre que Zezé ha estado trabajando como lustrador y se sorprende de que alguien se lustre en Navidad. Zezé explica que ha pasado todo el día trabajando y solo ha ganado diez cruzeiros, de los cuales cinco fueron dados como limosna. Aún necesita ganar dos cruzeiros más, pero no se menciona para qué los necesita.

Prueba N7:La metamorfosis, Franz Kafka

El artículo narra la vida de Gregorio Samsa, un hombre que, inexplicablemente, se ha transformado en un insecto gigante. Gregorio, que antes era un comerciante viajero, se encuentra en una situación de completo aislamiento y marginación, escondiéndose debajo del canapé para evitar los ruidos y movimientos bruscos que ahora le resultan aterradores. A pesar de su extraña nueva existencia, Gregorio comienza a acostumbrarse a su condición, aunque esto también le lleva a ver con mayor claridad su creciente desconexión emocional y física con su familia. La relación con su hermana, que inicialmente lo cuida, se deteriora cuando ella empieza a tratarlo con indiferencia, descuidando sus necesidades.

La transformación de Gregorio afecta profundamente la dinámica familiar. Su padre, frustrado con su propio trabajo y con el peso de la situación, se refugia en mantener su uniforme puesto todo el tiempo, quizás como una forma de aferrarse a algún sentido de control o dignidad. A pesar de la presencia de Gregorio como un insecto en la casa, la familia intenta seguir con su vida cotidiana, aunque la tensión y el desconcierto son palpables.

En una escena del relato, se describe cómo alguien toma frutas del frutero sin permiso, y otra donde Gregorio, abrumado por su nueva sensibilidad, siente repulsión por ciertos olores y se esconde debajo del canapé al escuchar el giro de la llave en la cerradura, un reflejo de su creciente vulnerabilidad. Aunque su cuerpo está cada vez más deteriorado, Gregorio encuentra cierto consuelo en observar a su familia desde su habitación, especialmente al anochecer, cuando la puerta del cuarto de estar queda entreabierta.

Sin embargo, la existencia de Gregorio se vuelve aún más precaria cuando su padre, enfurecido, lo persigue con la intención de herirlo. Gregorio intenta escapar, pero su falta de habilidad para moverse rápidamente lo pone en una situación de peligro inminente, temiendo recibir un golpe mortal con el bastón de su padre. En otra escena tensa, la hermana de Gregorio se niega a limpiar su habitación, lo que desata una acalorada discusión familiar. Gregorio, lleno de frustración, silba con rabia al ver cómo nadie toma la iniciativa de cerrar la puerta para evitar el escándalo.

En general, el artículo describe la lucha interna de Gregorio por adaptarse a su nueva forma de vida como insecto y el impacto devastador que esta transformación tiene en la dinámica de su familia. A pesar de los esfuerzos de Gregorio por encontrar algún consuelo en su trágica situación, se siente cada vez más aislado y amenazado en su propio hogar.

Prueba N8:Estrategia e Informática, Introducción

El artículo aborda varios aspectos clave de la gestión estratégica en las organizaciones. Comienza destacando la importancia del "modelo de cinco fuerzas" de Michael Porter para analizar la industria en la que opera una organización. Este modelo, que considera la rivalidad entre competidores, la amenaza de nuevos competidores, el poder de negociación de proveedores y clientes, y la amenaza de productos o servicios sustitutos, permite a las empresas comprender su entorno competitivo y evaluar las oportunidades y amenazas del mercado. El artículo también discute la intensa competencia entre las empresas en el mercado actual. Las empresas deben buscar constantemente formas de diferenciarse y destacar en el mercado, y deben estar preparadas para la amenaza que representan las nuevas empresas emergentes con propuestas innovadoras. Además, las empresas deben tener en cuenta la posibilidad de que los consumidores opten por productos o servicios sustitutos de otras marcas o empresas. Se destaca la importancia de tener una visión a largo plazo en una empresa, estableciendo objetivos claros y definidos, y asignando los recursos necesarios para alcanzar estos objetivos. Las políticas, que son reglas y directrices que guían las acciones y decisiones de la empresa, son fundamentales para este proceso. El artículo también menciona la importancia de las reglas en los programas, que establecen las pautas y criterios que deben seguirse para alcanzar un objetivo específico. Estas reglas pueden abarcar desde requisitos técnicos y de seguridad hasta políticas y procedimientos operativos. Los fundadores de Google, Larry Page y Sergey Brin, son citados como ejemplos de líderes que tienen como objetivo principal organizar y hacer accesible la información del mundo de manera útil. Han desarrollado tecnologías y herramientas para organizar y clasificar la información de manera eficiente, y han priorizado la relevancia y la calidad de la información en sus algoritmos de

búsqueda. El artículo también discute la importancia de la declaración de misión de la organización como un marco para formular estrategias. Esta declaración proporciona una base sólida para la toma de decisiones y la planificación estratégica, permitiendo a la organización evaluar su situación actual y determinar las acciones necesarias para alcanzar sus objetivos. Se destaca la importancia de implementar estrategias para obtener una ventaja competitiva en un entorno empresarial cada vez más competitivo. Estas estrategias deben basarse en un análisis del entorno y el mercado, y deben ser implementadas de manera efectiva, considerando los recursos y capacidades de la empresa. El proceso de planificación estratégica es fundamental para el éxito de cualquier organización. Este proceso implica establecer los objetivos y metas a largo plazo de la empresa, analizar el entorno externo e interno de la organización, identificar las oportunidades y amenazas, así como las fortalezas y debilidades, y desarrollar estrategias para aprovechar las oportunidades y superar las amenazas. El artículo también menciona la estrategia de integración vertical, que implica expandirse hacia atrás en la producción de insumos o hacia adelante en la cadena de suministro. Esta estrategia puede tener varios beneficios, como reducir costos, mejorar la calidad y tener un mayor control sobre la cadena de suministro. Finalmente, el Lic. Jorge A. Perdiguero habla sobre el proceso de planificación estratégica y la importancia de las estrategias en este proceso. Las estrategias son fundamentales en la planificación estratégica, ya que permiten a las organizaciones diferenciarse de sus competidores y alcanzar una ventaja competitiva. Además, las estrategias deben ser flexibles y adaptarse a los cambios del entorno.

Prueba N9: Educating to Compete, Trabajo de fin de Grado

The article discusses the significance of educational resilience during crises, particularly focusing on the recent pandemic. It explores how the crisis tested the digital literacy and infrastructure of different countries, as well as their ability to adapt and implement effective strategies for remote teaching and learning. The article also delves into the relationship between Information and Communication Technology (ICT) incorporation and perceived educational achievement, noting that empirical studies have presented ambiguous outcomes. It provides a contextual background of the patterns of ICT incorporation in Latin America before the pandemic. The author conducts a multivariate regression analysis in Chapter 4 to determine the statistical nature of the relationship between perceived education resilience during the pandemic and ICT. The aim is to understand how the use of ICT in education during the pandemic has affected the perceived resilience of the education system. The article emphasizes the importance of ICT incorporation in education and the need for further research to understand its impact on educational achievement. It also highlights the relevance of studying the patterns of ICT

incorporation in Latin America and how it has been affected by the pandemic. The article also discusses the importance of global competitiveness in the knowledge economy and the role of international organizations (IOs) in shaping this perception. It emphasizes that IOs have a significant influence on global competitiveness and that their perception of it is crucial. The World Bank has identified four fundamental requisites for a state to fully participate in the knowledge economy, one of which is the need for education and training of the population. This is seen as essential in creating a workforce that is equipped with the necessary knowledge and skills to thrive in the knowledge economy. The relationship between Information Communication Technologies (ICT) and global competitiveness is also discussed. The literature on this topic has been extensive over the past thirty years, examining both the impact of ICT on individual firms and on countries as a whole. The studies in this field recognize the importance of ICT in achieving success within the knowledge economy. By leveraging ICT, firms and countries can gain a competitive edge in the global market. The author discusses the data and variables used in their analysis, including a graph that shows the top 10 countries that have been most impacted by COVID-19 in 2020. The data for this graph is sourced from OXCGRT human capital. Additional variables such as GDP per capita and the literacy rate for adults above 15 years old are used as control variables to account for the potential influence of a country's economic and educational factors on the impact of COVID-19. The number of hospital beds, physicians, and nurses are also included as variables in their analysis to assess the healthcare capacity and resources available in each country. The author discusses the trade-off between sample size and information availability when it comes to analyzing data from different sources. They mention that the independent variable of interest, which is related to the level of ICT, was coded based on the answers from the PISA school questionnaire. The author notes that this questionnaire provided the most comprehensive information on the level of ICT. The article discusses the importance of understanding the impact of public policies on the level of success in incorporating ICT in education, particularly in Uruguay and Peru. The level of involvement in implementing these policies played a significant role in determining the success of ICT incorporation. In Uruguay, the government replicated the One Laptop per Child (OLPC) initiative, which aimed to provide every child with a laptop for educational purposes. This initiative was implemented as a national education policy called "Plan Ceibal" or Digital Education Program at National Level. The article discusses the advantages that education can bring to Argentina and Uruguay in terms of integrating into the global economy. While both countries have similar educational approaches, there is a key difference in how they are perceived globally. Uruguay is seen as more competitive in terms of education compared to Argentina, according to international organizations. This difference in perception could have implications for how each country is

able to participate in the global economy. The article discusses a study that found that the use of technology in mathematics education has a positive impact on students' educational resilience in the subject. The study, conducted by Lugalía et al. in 2013, concluded that the implementation of pedagogical uses of ICT in mathematics education is beneficial for students. This suggests that incorporating technology into math lessons can help students overcome challenges and persist in their learning. The author discusses the use of test scores as a way to measure learning outcomes in education. Specifically, the PISA and TIMSS tests are highlighted as important indicators of a country's international standing in terms of education. These tests are often used for national and regional comparisons. The author also mentions a study conducted by Delen et al. in 2011, which used hierarchical linear modeling to analyze test scores. The purpose of this study was to understand the relationship between test scores and learning outcomes.

Prueba N10:Autoencoders, paper científico

El artículo aborda varios aspectos del aprendizaje automático y la inteligencia artificial, centrándose en particular en los autoencoders y sus aplicaciones. Se presentan diferentes tipos de cabello en una serie de columnas, que representan cabello negro, castaño, rubio, negro y ondulado, rubio con flequillo y rubio con una línea de cabello retrocedida. Sin embargo, no se proporciona información adicional sobre la columna (h). Se introduce una nueva ecuación para calcular la log-verosimilitud utilizando probabilidades condicionales y distribuciones latentes. Esta ecuación establece un límite inferior en la log-verosimilitud y utiliza una suma de logaritmos y probabilidades condicionales para calcular la log-verosimilitud de un conjunto de datos dado un conjunto de parámetros y una distribución latente. El artículo también destaca la importancia del error de reconstrucción y el factor de regularización en el proceso de reconstrucción de datos. El error de reconstrucción se refiere a la diferencia entre los datos originales y los datos reconstruidos a partir de una representación latente, mientras que el factor de regularización garantiza que la representación latente se asemeje a una distribución previa. Se introduce el concepto de la distancia de Wasserstein, una medida utilizada para cuantificar la diferencia entre dos distribuciones de probabilidad. Esta medida es útil en problemas de transporte óptimo y tiene aplicaciones en diversos campos, como la economía, la física y la estadística. El uso de autoencoders en sistemas de recomendación se menciona, con el modelo AutoRec como un ejemplo básico. Los autoencoders son una técnica de aprendizaje automático que se utiliza para aprender las preferencias de los usuarios y generar recomendaciones personalizadas. Las aplicaciones más comunes de los autoencoders incluyen la reducción de dimensionalidad, la eliminación de ruido en imágenes y la generación de nuevas muestras de datos similares a las de entrenamiento. Se proporciona una breve discusión

sobre la comparación entre los autoencoders y las redes generativas adversarias (GANs), destacando las diferencias en sus enfoques y objetivos. Se menciona el uso de PixelCNN, una técnica utilizada para generar imágenes de alta calidad. Esta técnica ordena los píxeles de una imagen de manera arbitraria y utiliza este orden para generar la salida del modelo, capturando las dependencias espaciales entre los píxeles. Se introduce un codificador probabilístico llamado "is equivalent" que se utiliza para representar información latente en un modelo. Se asume la existencia de una distribución previa para las variables latentes. El artículo también destaca la capacidad de los modelos de autoencoder para extraer características relevantes de las imágenes de entrada, siempre y cuando la distribución de los datos en los que se entrenó el modelo sea similar a la distribución de los datos en los que se entrenó el autoencoder. Finalmente, se menciona que el objetivo del decodificador en un autoencoder es minimizar la pérdida cuadrada, lo que implica que el decodificador intenta reconstruir los datos de entrada lo más fielmente posible. Esto es útil en aplicaciones de compresión de datos.

Referencias

- Abhigyan. (2020, julio 19). Understanding The Linear Regression!!!! *Analytics Vidhya*.
<https://medium.com/analytics-vidhya/understanding-the-linear-regression-808c1f6941c0>
- Alammar, J. (2020). *The Illustrated Transformer*.
<https://jalammar.github.io/illustrated-transformer/>
- Álvarez Díaz, A. (2023). *Procesamiento de lenguaje natural (NLP)*.
- Asaxena. (2018, junio 1). Introduction to Deep Learning. *GeeksforGeeks*.
<https://www.geeksforgeeks.org/introduction-deep-learning/>
- AWS. (s. f.). ¿Qué es una API? - Explicación de interfaz de programación de aplicaciones - AWS. Amazon Web Services, Inc. Recuperado 5 de mayo de 2024, de
<https://aws.amazon.com/es/what-is/api/>
- Bag, S. (2023, enero 2). Activation Functions—All You Need To Know! *Analytics Vidhya*.
<https://medium.com/analytics-vidhya/activation-functions-all-you-need-to-know-355a850d025e>
- Bank, D., Koenigstein, N., & Giryas, R. (2021). *Autoencoders* (arXiv:2003.05991). arXiv.
<http://arxiv.org/abs/2003.05991>
- BitBoss (Director). (2020, mayo 27). *Redes Neuronales—Backpropagation* [Video recording]. <https://www.youtube.com/watch?v=boP3O89rErA>
- Brownlee, J. (2017, diciembre 19). A Gentle Introduction to Transfer Learning for Deep Learning. *MachineLearningMastery.Com*.
<https://machinelearningmastery.com/transfer-learning-for-deep-learning/>
- Chomsky, N. (1955). Logical Syntax and Semantics: Their Linguistic Relevance. *Language*, 31(1), 36-45. <https://doi.org/10.2307/410891>
- Díaz Álvarez, A. (2023, marzo 28). *Procesamiento de lenguaje natural*. Departamento de Sistemas Informáticos - Universidad Politécnica de Madrid.
- Díaz Álvarez, A. D., Talavera Muñoz, E., & Iglesias, G. (2023, abril 17). *Autoencoders, Aprendizaje Profundo*.
- Frost, J. (2021, noviembre 12). *Mean Squared Error (MSE)*. Statistics By Jim.
<https://statisticsbyjim.com/regression/mean-squared-error-mse/>
- García Serradilla, F. (2023, marzo 1). *Deep Learning aplicado a sistemas inteligentes*. Departamento de Inteligencia Artificial de la UPM.
- Gobierno de España. (s. f.). *Qué es la Inteligencia Artificial*. Recuperado 17 de agosto de 2023, de
<https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr>
- Gonzalez, J. L. (2020, julio 13). Tipos de aprendizaje automático. *SoldAI*.

<https://medium.com/soldai/tipos-de-aprendizaje-autom%C3%A1tico-6413e3c615e2>
 Granieri, M. (2023, marzo 5). *¿Qué es la Inteligencia Artificial Generativa?* OBS Business School.
<https://www.obsbusiness.school/blog/que-es-la-inteligencia-artificial-generativa>
 Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735-1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
 Huanca, L. A. R. (2021, febrero 16). CLUSTERING. *Aprendizaje No Supervisado*.
<https://medium.com/aprendizaje-no-supervisado/clustering-cee49ad0061f>
 Karpathy, A. (Director). (2022, agosto 17). *The spelled-out intro to neural networks and backpropagation: Building micrograd* [Video recording].
<https://www.youtube.com/watch?v=VMj-3S1tku0>
 Keim, R. (2020, febrero 6). *Understanding Local Minima in Neural-Network Training—Technical Articles*. Understanding Local Minima in Neural-Network Training.
<https://www.allaboutcircuits.com/technical-articles/understanding-local-minima-in-neural-network-training/>
 Law, M. (2022, noviembre 29). *Top 10: Promising AI startups in 2022*.
<https://aimagazine.com/articles/top-10-promising-ai-startups-in-2022>
 Merritt, R. (2022, abril 19). *¿Qué Es un Modelo Transformer?* | *Blog de NVIDIA*. Blog oficial de NVIDIA Latino América.
<https://la.blogs.nvidia.com/2022/04/19/que-es-un-modelo-transformer/>
 Nvidia. (2023). *What are Large Language Models?* | *NVIDIA Glossary*. NVIDIA.
<https://www.nvidia.com/en-us/glossary/large-language-models/>
 Patel, A. A. (2022, abril 30). *GPT-4, GPT-3, and GPT-3.5 Turbo: A Review Of OpenAI's Large Language Models*.
<https://www.ankursnewsletter.com/p/gpt-4-gpt-3-and-gpt-35-turbo-a-review>
 Penumudy, T. (2021, enero 29). Everything you need to know about K-Means Clustering. *Analytics Vidhya*.
<https://medium.com/analytics-vidhya/everything-you-need-to-know-about-k-means-clustering-88ad4058cce0>
 Pykes, K. (2020, mayo 17). *The Vanishing/Exploding Gradient Problem in Deep Neural Networks*. Medium.
<https://towardsdatascience.com/the-vanishing-exploding-gradient-problem-in-deep-neural-networks-191358470c11>
 Sáenz de Cabezón, E. (Director). (2021, agosto 4). *¿Qué es y cómo funciona la INTELIGENCIA ARTIFICIAL?* [Video recording].
https://www.youtube.com/watch?v=_tA5cinv0U8

- Santana Vega, C. (Director). (2020, junio 30). *Introduccion al Natural Language Processing (NLP) #1—¡De PALABRAS a VECTORES!* [Video recording].
<https://www.youtube.com/watch?v=Tg1MjMIVArc>
- Silverio, M. (s. f.). *Bing: Usuarios y estadísticas clave (2024)*. Recuperado 22 de abril de 2024, de <https://www.primeweb.com.mx/bing-estadisticas>
- Starmer, J. (Director). (2019, febrero 5). *Gradient Descent, Step-by-Step* [Video recording].
<https://www.youtube.com/watch?v=sDv4f4s2SB8>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention Is All You Need* (arXiv:1706.03762). arXiv.
<http://arxiv.org/abs/1706.03762>