



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Trabajo Terminal.

**Generador de versos musicales en el idioma
inglés por medio de procesamiento de
lenguaje natural y redes neuronales**

TT2020-B002.

Presentan:

Espinosa de los Monteros Lechuga Jaime Daniel
Nava Romo Edgar Adrián
Salgado Gómez Alfredo Emilio

Directores:

Olga Kolesnikova
Ariel López Rojas

Anexos

Índice.

Anexos	1
I Trabajo Terminal I	9
1. Introducción	10
1.1. Objetivos.	13
1.1.1. Objetivo general.	13
1.1.2. Objetivos particulares.	13
1.2. Justificación.	13
1.3. Metodología.	15
1.4. Organización del documento	17
1.4.1. Capítulo 2. Marco teórico.	17
1.4.2. Capítulo 3. Análisis.	17
1.4.3. Capítulo 4. Diseño.	18
1.4.4. Capítulo 5. Desarrollo.	18
1.4.5. Capítulo 6. Avances y trabajo por hacer.	18
2. Marco teórico	19
2.1. Inteligencia Artificial	19
2.2. Redes neuronales	19
2.2.1. Neuronas	20
2.2.2. Aprendizaje de las Redes Neuronales	21
2.2.3. Recurrent Neural Network – Long Short Term Memory	23
2.3. Procesamiento de Lenguaje Natural	25
2.3.1. Tareas del Procesamiento de Lenguaje Natural	25
2.3.2. Usos del Procesamiento de Lenguaje Natural	26
2.4. Transformers	26
2.4.1. Arquitectura de un transformer	27
2.5. BERT	28
2.5.1. ¿Qué es Bert?	28

2.5.2.	¿Cómo funciona BERT?	28
2.5.3.	BERT en la actualidad	29
2.6.	GPT-2	29
2.6.1.	¿Qué es GPT-2?	29
2.6.2.	Arquitectura GPT	31
2.7.	Base de datos	32
2.7.1.	Base de datos NoSQL o no relacionales	33
2.8.	Canción	34
2.8.1.	Elementos de una canción	34
2.8.2.	Estructura de una canción	35
2.9.	Cadenas de Markov	36
2.10.	Flask	37
2.11.	Apache	39
2.12.	Servidor Web	40
2.13.	Certificado SSL	41
2.14.	Plataforma en la Nube de Aprendizaje Automático	42
2.14.1.	Amazon	42
3.	Análisis.	45
3.1.	Estudio de Factibilidad.	45
3.1.1.	Factibilidad Técnica	45
3.1.2.	Factibilidad Operativa	47
3.1.3.	Factibilidad Económica	48
3.2.	Herramientas a usar.	49
3.2.1.	Software.	49
3.2.2.	Hardware.	54
3.3.	Arquitectura del sistema.	55
3.3.1.	Descripción de la arquitectura del sistema.	55
3.3.2.	Aspectos Económicos	55
3.3.3.	Aspectos Legales	55
3.4.	Diagrama de casos de uso general.	55
4.	Iteración 1.	56
4.1.	Descripción.	56
4.2.	Base de Datos	56
4.3.	Género Musical	56
4.4.	Resultados	56
5.	Iteración 2.	57
5.1.	Descripción.	57
5.2.	Limpieza	57

5.3. Regular Expressions	57
5.4. Técnicas de Scrapping	57
5.5. Implementación	57
5.6. Resultados	57

Índice de figuras.

1.1. Funcionamiento de la planificación por cubetas	17
2.1. Estructura de una neurona	21
2.2. Neurona Artificial	21
2.3. Diagrama Red Neuronal Recurrente	24
2.4. Arquitectura general de un Transformer [25]	27
2.5. Representación de entrada de BERT y la separación de la oración por palabras y la asignación de valores.	29
2.6. Arquitectura del modelo de GPT-2 [31]	31
2.7. Ejemplo análisis de una oración por GPT-2	32
2.8. Diagrama De Estados finitos De Una Cadena De Markov . . .	37

Índice de cuadros.

1.1. Resumen de productos similares comparados con nuestra propuesta	12
1.2. Tabla comparativa de los distintos servidores web contemplados	15
2.1. Tabla comparativa de los distintos servidores web contemplados para nuestro proyecto	40
2.2. Tabla comparativa de las diversas plataformas contempladas .	43
2.3. Tabla comparativa GCP vs AWS (en dólares)	44
3.1. Herramientas de Software	46
3.2. Horas de trabajo	47

Glosario.

- Acordes** Conjunto de notas musicales sonando al mismo tiempo.[1]. 27
- Aprende** Adquirir el conocimiento de algo por medio del estudio, análisis o de la experiencia.[1]. 23
- Armonía** Unión y combinación de sonidos simultáneos y diferentes, pero acordes.[1]. 26
- Estado** Situación en que se encuentra alguien o algo, y en especial cada uno de sus sucesivos modos de ser o estar.[1]. 28
- Estocástico** Que está sometido al azar y que es un objeto de análisis estadístico.[1]. 28
- Estrofa** Conjunto de versos que generalmente se ajustan a una medida y a un ritmo determinado y constante.[1]. 26
- Melodía** Es una sucesión lineal ordenada y coherente de sonidos musicales los cuales forman una unidad estructurada con un sentido musical, independiente del acompañamiento.
Sucesión de sonidos que por su manera de combinarse resulta agradable de oír.[2]. 26
- Máquinas de Estados Finitos** Conocidas también como Finite State Machines por su traducción en inglés, nos sirven para realizar procesos bien definidos en un tiempo discreto. Reciben una entrada, realizan un proceso y nos entregan una salida.[13]. 28
- Recurrente** Algo que vuelve a ocurrir o a aparecer después de un intervalo.[1]. 23
- Sistema** Conjunto de elementos con relaciones de interacción e interdependencia las cuales le confieren entidad propia al formar un todo unificado.[12]. 23, 28

Acrónimos.

DBMS Sistema de gestión de base de datos. 25

ORM Object Relational Manager. 29

WSGI Web Server Gateway Interface. 29

Parte I

Trabajo Terminal I

Capítulo 1

Introducción

La industria musical obtiene ganancias a través de la creación y divulgación de la música de manera física y digital (Bourreau and Gensollen 2006 [3]), dejando que aficionados y emprendedores de la música no tengan oportunidad de avanzar en su carrera por falta de creatividad, tiempo y/o recursos, haciendo que la creación de nuevas letras para sus canciones sea un gran obstáculo, nuestra propuesta implica la utilización de nuevas tecnologías que permitan la generación de letras para integrar con sus canciones. Esta es una de las tareas más populares y desafiantes en el área de procesamiento del lenguaje natural. Hay una gran cantidad de trabajos (Generating Text with Recurrent Neural Networks [4], Convolutional Neural Networks for Sentence Classification[5]) que proponen generar texto utilizando redes neuronales recurrentes y/o convolucionales. Sin embargo, la mayoría de los trabajos actuales solo se enfocan en generar una o varias oraciones, ni siquiera un párrafo largo y mucho menos una canción completa.

Las letras de canciones, como un tipo de texto, tienen algunas características propias, estas se constituyen de rimas, versos, coros y en algunos casos, patrones de repetición. Coro se refiere a la parte de una canción que se repite sin modificaciones dentro de la misma después de un verso, mientras que en el verso suelen cambiar una o varias líneas que lo componen. Estas características particulares hacen que generar letras musicales sea mucho más difícil que textos normales.

La mayoría de las investigaciones actuales sobre generación de letras vienen con muchas condiciones, como dar una pieza de melodía (Automatic Generation of Melodic Accompaniments for Lyrics [6]), o solo generar un tipo específico de letra (Conditional Rap Lyrics Generation with Denoising Autoencoders [7]). Sin embargo, la generación de letras por medio de inteligencia artificial dado un estilo y tema en particular, ha sido muy poco trabajado y debido a esto planeamos centrarnos en este nuevo problema. Es-

tamos interesados en ver si el modelo propuesto puede aprender diferentes características de un género musical y generar letras que sean acorde a este. Actualmente, en el mercado se encuentran cuatro aplicaciones web que tienen una funcionalidad similar a la propuesta en este Trabajo Terminal:

- These lyrics do not exist.
- Bored humans - lyrics_generator.
- DeepBeat.
- Premium Lyrics.

En el cuadro 1 que se presenta a continuación, se muestran las características de aplicaciones web similares y comparándolas con nuestra propuesta:

Cuadro 1.1: Resumen de productos similares comparados con nuestra propuesta

Software	Características	Precio en el mercado
These Lyrics do not Exist	Aplicación web que genera letras completamente originales de varios temas, hace uso de IA para generar coros y versos originales; se puede escoger el tema principal de la letra, género musical e incluso el estado de ánimo al que irá dirigido.	Gratuito (Contiene Anuncios)
Boredhumans lyrics_generator	Aplicación web en el que la IA fue entrenada con una base de datos con miles de letras para generar letras de canciones totalmente nuevas. La letra que crea es única y no una copia de alguna que exista actualmente, sin embargo, no permite customizar la letra.	Gratuito
DeepBeat	Aplicación web que por medio de IA genera letras de música enfocada principalmente en el género rap. Si una línea no es del agrado se puede sustituir por alguna de las otras propuestas de las que ofrece.	Gratuito
Premium Lyrics	Aplicación web que proporciona versos compuestos en distintos idiomas por artistas independientes que se escogen manualmente de acuerdo a su originalidad y calidad.	3\$ a 75\$ Dólares por letra musical
Nuestra propuesta	Aplicación web que haciendo uso de una IA va a generar letras musicales originales a partir de un género musical en exclusivo, lo que asegurará un resultado nal con coros y versos distintos cada vez que se utilice.	Gratuito

1.1. Objetivos.

1.1.1. Objetivo general.

Crear una herramienta de apoyo para estudiantes o aficionados interesados en este rubro que se les dificulte componer nuevas letras musicales de un solo género musical debido a la carencia de creatividad, la falta de conocimientos en la estructura del género o que no tengan inspiración suficiente para poder crear nuevas canciones, esto con el fin de impulsar la carrera de futuros artistas en la industria musical que no tengan los suficientes recursos para poder contratar servicios particulares de compositores.

1.1.2. Objetivos particulares.

- Generar un conjunto de datos (dataset) con letras musicales de un género musical para efecto de entrenamiento de la red semántica.
- Hacer uso de alguna herramienta de aprendizaje automático (machine learning) e implementar su uso en la nube para ayudar a procesar las letras musicales de un género en específico.
- Implementar un módulo analizador de semántica para entrenar redes neuronales.
- Desarrollar una interfaz web intuitiva en versión prototipo que utilice una aplicación web alojada en la nube para la visualización del verso musical generado a partir de un género.

1.2. Justificación.

El crear nuevas composiciones musicales puede llegar a ser muy difícil, estresante e incluso agotador para cualquier aficionado o incluso algunos expertos en este medio, esto se debe a la falta de creatividad y/o tiempo de quien lo quiera realizar [8]. En ocasiones se pueden contratar servicios particulares para la producción de la letra de una canción, sin embargo, puede ser muy costoso y en ocasiones el resultado final no alcanza a llenar las expectativas de la inversión que se hace; por ende, se pretende crear una herramienta para estudiantes, aficionados o cualquier persona interesada en este rubro que se les dificulte componer nuevas letras musicales.

Normalmente las letras musicales creadas por el humano, tienden a estar compuestas por patrones de acuerdo al género musical [9]. Algunos ejemplos

de estos patrones pueden ser las rimas, enunciados, frases cortas y que tengan una semántica correcta, estos pueden ser encontrados por medio de procesamiento de lenguaje natural y una investigación profunda en la composición de letras de estos géneros.

Se eligió el idioma inglés debido a que existe una gran cantidad de bases de datos para procesar, al igual que herramientas y documentación para este idioma.

Nos proponemos orientar esta solución en un entorno de nube, donde la información de configuración, servicios y datos necesarios pueden mantenerse de forma independiente a la implementación, facilitando la adaptación y flexibilidad de la plataforma.

Nuestro proyecto ayudará al usuario utilizando herramientas como el procesamiento de lenguaje natural, redes neuronales, aprendizaje de máquina (machine learning) y servidores en la nube. Se hará uso de un conjunto de datos (dataset) y herramientas alojadas en la nube (Google Cloud Platform o Amazon Web Services) para procesar estos datos; se pretende utilizar un módulo que encuentre patrones por medio de redes neuronales para analizar la semántica mediante técnicas y herramientas ya existentes de procesamiento de lenguaje natural. Se van a realizar pruebas y experimentos con estas herramientas antes de la implementación (Bert [10], spaCy[11]) para poder generar versos. A su vez se va a desarrollar una interfaz web intuitiva en versión prototipo donde el usuario va a poder utilizar esta herramienta la cual le va a mostrar la letra musical que se va a generar en ese momento.

A diferencia de los proyectos señalados en la Tabla 1 nuestra propuesta se va a centrar en generar letras musicales con métodos y tecnologías distintas a los que se usaron, esto es, aunque se utilicen los mismos géneros musicales se tendrán resultados completamente diferentes con propuestas distintas.

En el desarrollo de este proyecto haremos uso de los conocimientos adquiridos durante el transcurso de la carrera. Se van a utilizar técnicas de diseño de proyectos aprendidas en el curso de Ingeniería de Software, se van a aplicar los conocimientos de programación adquiridos en unidades de aprendizaje como Inteligencia Artificial, Procesamiento de Lenguaje Natural, Web Application Development, Programación Orientada a Objetos, Análisis de Algoritmos, así como técnicas de construcción de documentos y análisis de semántica vistas en Análisis y Diseño orientado a Objetos y Comunicación Oral y Escrita.

1.3. Metodología.

Para el desarrollo de este trabajo terminal se utilizará la metodología ágil Scrumban, que combina algunas partes de la metodología Scrum y Kanban, debido a que este es un proceso de gestión el cual reduce la complejidad en el desarrollo de productos para satisfacer las necesidades de los clientes. Además, permite trabajar colaborativamente de manera eficiente, es decir, en equipo, para obtener el mejor resultado posible.

Scrumban combina la estructura que utiliza Scrum con los métodos basados en flujo junto con la visualización de Kanban. Es decir, se le permite a los equipos tener la agilidad de Scrum y la simplicidad de Kanban sin tener que utilizar las actualizaciones de roles y es más sencillo de adoptar.

En la siguiente tabla se pueden observar las principales diferencias entre las tres metodologías:

Cuadro 1.2: Tabla comparativa de los distintos servidores web contemplados

	Scrum	Kanban	Scrumban
Procesos	Iterativo e incremental desarrollando sprints	Continuo	Iterativo e incremental de forma continua desarrollando iteraciones
Personas	Las personas son el centro	Las personas son el pilar	Equipo motivado con personas como pilar y en el centro
Producto	Foco en la efectividad	Foco en la eficiencia	Balance entre efectividad y eficiencia
Organización	Mejora continua del producto	Mejora continua del proceso	Mejora continua del producto y del proceso
Equipo	De 3 a 9 personas	No hay limitaciones	El equipo no requiere de un número específico de integrantes
Roles	Scrum master Product owner Scrum team	Pueden incluir especialistas o integrantes generalizados	No requiere un rol específico

Scrumban utiliza iteraciones y se monitorea con la ayuda de un tablero visual. Las reuniones para planificar se llevan a cabo cuando son necesarias para determinar las tareas a implementar hasta la próxima iteración. Para que estas iteraciones se mantengan cortas, se utiliza un límite de trabajo en progreso (WIP por sus siglas en inglés Work in Progress). Cuando WIP cae por debajo del nivel predeterminado, se establece un activador de planificación bajo demanda para que el equipo sepa cuándo planificar a continuación.

Iteración

En Scrumban, las iteraciones son cortas para garantizar que el equipo pueda adaptarse al entorno cambiante durante el proyecto. La duración de estas iteraciones en este proyecto se medirán como máximo en lapsos de dos semanas.

Priorización

La priorización se da de tal forma que las tareas más importantes se colocan en la parte superior de la tabla de planificación seguidas por las tareas menos importantes.

Antes de llegar al tablero las tareas deben pasar por 3 etapas donde se van depurando las tareas a realizar para largo plazo (1 año), medio plazo (6 meses) y corto plazo (3 meses) siendo de esta última de donde salen las tareas más claras que se pueden completar y que ganan mayor prioridad para la próxima iteración.

Principio de Elección

Cada miembro del equipo elige solo qué tarea de la sección “Tareas Pendientes” va a completar a continuación.

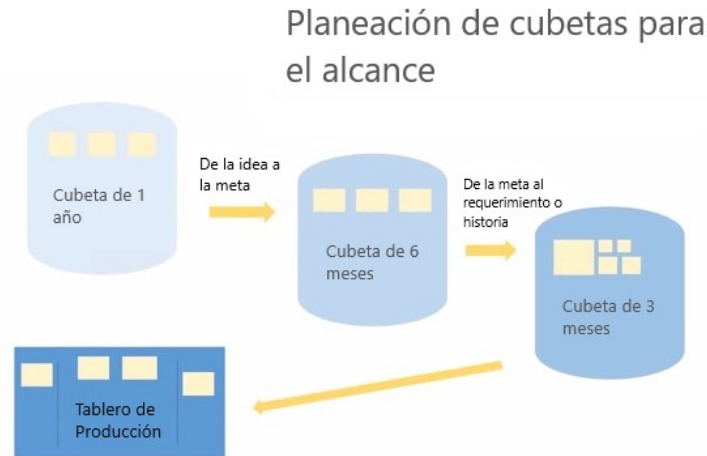
Congelación de Funciones

Se utiliza en Scrumban cuando se acerca la fecha límite del proyecto, significando que sólo pueden trabajar sobre las tareas previamente pensadas sin cabida para implementar nuevas características.

Triaje

Ocurre después de la congelación de funciones, y es el punto donde el gerente del proyecto decide cuáles de las características en desarrollo se completarán y cuáles quedarán sin terminar.

Figura 1.1: Funcionamiento de la planificación por cubetas



1.4. Organización del documento

Para dar inicio a este trabajo terminal, presentamos de manera breve la estructura de este reporte, con el objetivo de que el lector pueda tener un mejor entendimiento del trabajo.

1.4.1. Capítulo 2. Marco teórico.

En esta parte del documento, se describen puntos esenciales de nuestro trabajo como definiciones, técnicas, herramientas, servicios, así como investigación realizada para llevar a cabo en la implementación dentro del trabajo.

1.4.2. Capítulo 3. Análisis.

Dentro de este capítulo se analiza el estudio de factibilidad tanto técnico como operativo y económico con la finalidad de conocer los recursos necesarios para la elaboración de este trabajo terminal. Se mencionan resumidamente las herramientas a utilizar y se explica de manera general la arquitectura

del sistema y el diagrama de casos de uso general. Finalmente se muestra el análisis de los tres componentes que tenemos en el sistema.

1.4.3. Capítulo 4. Diseño.

En el cuarto capítulo, nos adentraremos en el desarrollo de los prototipos, es decir, se encuentran los diagramas pertinentes para poder modelar nuestro trabajo terminal y proceder a la etapa de implementación. En este capítulo se desarrollan y explican los siguientes diagramas: 'Casos de uso', 'Flujo', 'Flujo de datos', 'Clases', 'Secuencia' y 'Actividades' y se muestra la interfaz de usuario propuesta junto con los requisitos de diseño.

1.4.4. Capítulo 5. Desarrollo.

En este capítulo mostraremos lo que se ha desarrollado para este Trabajo Terminal I (Iteración I e Iteración II). Se muestra que el prototipo cumple los requerimientos que se le impusieron en la sección de análisis.

1.4.5. Capítulo 6. Avances y trabajo por hacer.

En el último capítulo de este reporte, hablaremos de los avances que hemos logrado a lo largo de la asignatura de Trabajo Terminal I y además, exponemos el trabajo esperado para la asignatura de Trabajo Terminal II.

Capítulo 2

Marco teórico

2.1. Inteligencia Artificial

Por Inteligencia Artificial se entiende a una simulación de procesos computacionales cognitivos para que simulen el comportamiento de una mente humana, estos comportamientos abarcan diferentes áreas de investigación, como el razonamiento, aprendizaje, percepción, comunicación y la capacidad de desarrollarse en entornos más complejos. “La inteligencia no es una dimensión única, sino un espacio profusamente estructurado de capacidades diversas para procesar la información. Del mismo modo, la Inteligencia Artificial utiliza muchas técnicas diferentes para resolver una gran variedad de tareas que se encuentran en todas partes.” [?]

En ciencias de la computación, el término de inteligencia artificial se refiere a cualquier inteligencia similar a la humana exhibida por una computadora, robot u otra máquina. En el uso popular, la inteligencia artificial hace referencia a la capacidad de una computadora o máquina para imitar las capacidades del cerebro humano, es decir, aprender de ejemplos y experiencias, reconocer objetos, comprender y responder al lenguaje, tomar decisiones, resolver problemas y combinar estas y otras capacidades para realizar funciones que un ser humano podría realizar. [14]

Como objetivo, la Inteligencia Artificial pretende hacer un uso de los recursos tecnológicos para desarrollar modelos computacionales y poder resolver problemas del mundo real evolucionando constantemente.

2.2. Redes neuronales

Una red neuronal es una herramienta derivada de la inteligencia artificial que utiliza modelos matemáticos para ser utilizada como un mecanismo de

predicción de texto.

Las redes neuronales son una forma de hacer que las computadoras aprendan, donde la computadora aprende a realizar alguna tarea analizando ejemplos de entrenamiento. Por lo general, estos ejemplos han sido etiquetados previamente.

Modelado vagamente en el cerebro humano, una red neuronal consiste en miles de millones de nodos de procesamiento los cuales están densamente interconectados. En la actualidad las redes neuronales están organizadas como capas de nodos, y estas capas están “alimentadas hacia adelante” (feed-forward), es decir, la información que se mueve a través de ellas solo fluye en una sola dirección. Un solo nodo puede estar conectado a muchos nodos en capas inferiores de las cuales recibe información y a su vez, puede estar conectado a nodos en capas superiores a los cuales envía información.

Cuando una red neuronal está siendo entrenada la información de entrenamiento pasa a alimentar las capas inferiores (capas de entrada), donde será procesada y pasada a capas posteriores donde será transformada hasta llegar a las capas superiores (capas de salida).[15]

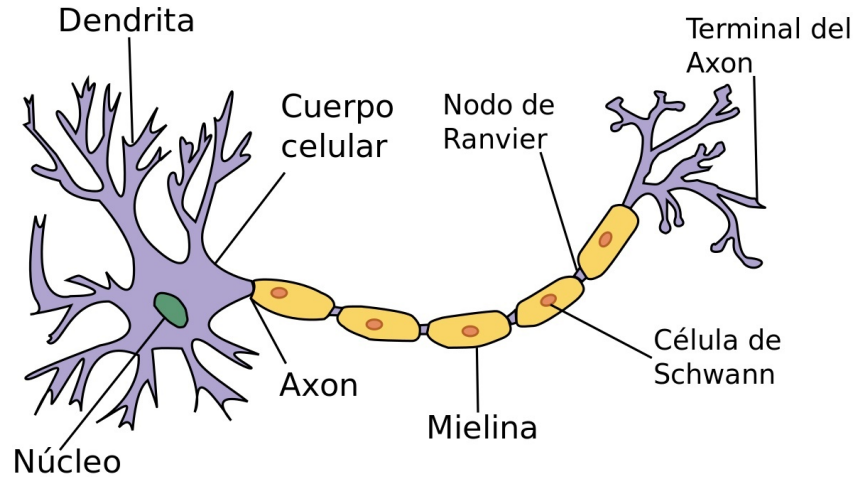
2.2.1. Neuronas

2.2.1.1. Neurona Natural

La neurona es un tipo de célula que representa la unidad estructural y funcional del sistema nervioso. Su función consiste en transmitir información a través de impulsos nerviosos o químicos, desde un lugar del cuerpo hacia otro.

Este impulso viaja siempre en el mismo sentido, es decir, llega a la neurona a través de las dendritas, se procesa en el soma y posteriormente se transmite al axón, las neuronas no están en contacto entre sí, existe un espacio de separación entre ellas denominado sinapsis o espacio sináptico. Cuando el impulso nervioso llega al extremo del axón se liberan neurotransmisores al espacio sináptico transformando la señal eléctrica en otra química que penetra en la dendrita de la neurona contigua, desencadenando un impulso eléctrico en la neurona receptora y se repite el proceso con la contigua.[16]

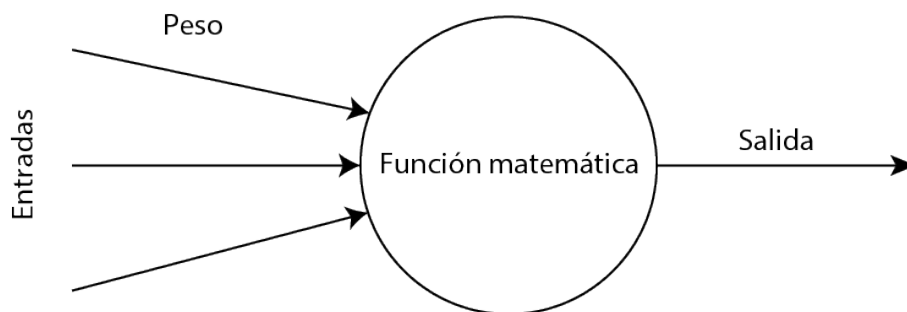
Figura 2.1: Estructura de una neurona



2.2.1.2. Neurona Artificial

La complejidad de las neuronas reales se abstrae mucho cuando se realiza el modelado de neuronas artificiales. Estas consisten básicamente en entradas, que se multiplican por pesos (fuerza de las respectivas señales), y luego calculada por una función matemática que determina la activación de la neurona. Otra función (que puede ser la identidad) calcula la salida de la neurona artificial (a veces en dependencia de un cierto umbral). Las redes neuronales combinan neuronas artificiales para procesar información.[17]

Figura 2.2: Neurona Artificial



2.2.2. Aprendizaje de las Redes Neuronales

Si observamos la naturaleza, podemos ver que los sistemas que pueden aprender son altamente adaptables. En su búsqueda por adquirir conocimientos, estos sistemas utilizan información del mundo exterior y modifican

la información que ya han recopilado o modifican su estructura interna. Eso es exactamente lo que hacen las redes neuronales. Adaptan y modifican su arquitectura para aprender. Para ser más precisos, las redes neuronales cambian los pesos de las conexiones según la entrada y la salida deseada.

¿Por qué tienen un peso?, bueno, si observamos la estructura de las redes neuronales, hay algunos componentes que podríamos cambiar, si queremos modificar su arquitectura. Por ejemplo, podríamos crear nuevas conexiones entre neuronas, o eliminarlas, o agregar y eliminar neuronas. Incluso podríamos modificar la función de entrada o la función de activación. Resulta que cambiar los pesos es el enfoque más práctico. Además, la mayoría de los otros casos podrían cubrirse cambiando los pesos. La eliminación de una conexión, por ejemplo, se puede hacer estableciendo el peso en 0. Y una neurona se puede eliminar si establecemos los pesos de todas sus conexiones en cero.[18]

El entrenamiento es un proceso necesario para toda red neuronal, y es un proceso en el que la red se familiariza con el problema que necesita resolver. En la práctica, generalmente se tienen algunos datos recopilados en función de los cuales necesitamos crear nuestras predicciones, clasificación, o cualquier otro procesamiento. Estos datos se denominan conjunto de entrenamiento. Según el comportamiento durante el entrenamiento y la naturaleza del conjunto de entrenamiento, tenemos algunos tipos de aprendizajes:

- **Aprendizaje no supervisado:** el conjunto de entrenamiento solo contiene entradas. La red intenta identificar entradas similares y clasificarlas en ciertas categorías.
- **Aprendizaje reforzado:** el conjunto de entrenamiento contiene entradas, pero la red también recibe información adicional durante la formación. Lo que sucede es que una vez que la red calcula la salida para una de las entradas, proporcionamos información que indica si el resultado fue correcto o incorrecto y, posiblemente, la naturaleza del error que cometió la red.
- **Aprendizaje supervisado:** el conjunto de entrenamiento contiene entradas y salidas deseadas. De esta manera, la red puede verificar su salida calculada con la salida deseada y tomar las acciones pertinentes para reformular su cálculo.

2.2.3. Recurrent Neural Network – Long Short Term Memory

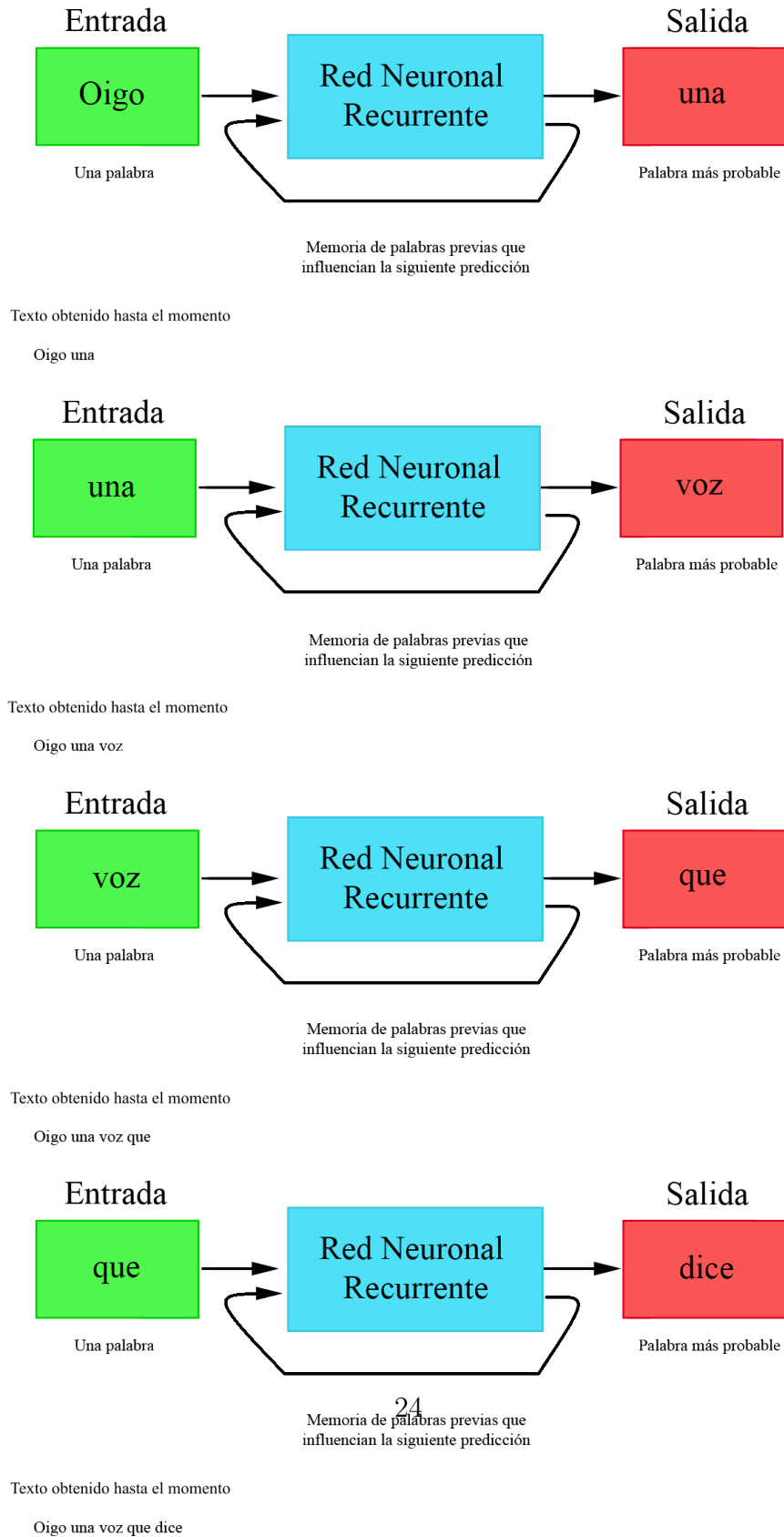
Una red neuronal Recurrente es un tipo de red neuronal artificial donde la salida de alguna capa en particular es salvada y sirve para retroalimentar la entrada de esta capa, lo cual ayuda a predecir futuras salidas de esta.

La primera capa esta formada de la misma manera que la Feedforward Neural Network, es decir, solo pasa la información que entra a la siguiente capa inmediata, posteriormente la siguiente capa con el paso del tiempo comenzará a retroalimentarse, pero manteniendo la propagación frontal. Haciendo uso de esta retroalimentación la capa en futuras operaciones puede realizar predicciones, si estas predicciones no son los resultados esperados, el Sistema Aprende y trabaja para corregir sus futuras predicciones.[19]

Se distinguen por su “memoria”, ya que toman información de entradas anteriores para influir en la entrada y salida actuales. Mientras que las redes neuronales profundas tradicionales asumen que las entradas y salidas son independientes entre sí, la salida de las redes neuronales recurrentes depende de los elementos anteriores dentro de la secuencia. Si bien los eventos futuros también serían útiles para determinar la salida de una secuencia dada, las redes neuronales recurrentes unidireccionales no pueden tener en cuenta estos eventos en sus predicciones.[22]

Estos algoritmos de aprendizaje profundo se utilizan comúnmente para problemas relacionados con la traducción de idiomas, el procesamiento del lenguaje natural, el reconocimiento de voz y los subtítulos de imágenes.

Figura 2.3: Diagrama Red Neuronal Recurrente



2.3. Procesamiento de Lenguaje Natural

El Procesamiento del lenguaje natural (PLN) se refiere a una rama de la informática, específicamente, a una rama de la inteligencia artificial, la cual se ocupa de dar a las computadoras la capacidad de comprender texto y palabras de la misma manera que los seres humanos.

La Procesamiento del lenguaje natural (PLN) combina la lingüística computacional con modelos estadísticos, machine learning y deep learning. Juntas, estas tecnologías permiten a las computadoras procesar el lenguaje humano en forma de texto o datos de voz y comprender su significado completo, es decir, la intención y el sentimiento del hablante o escritor. [23]

2.3.1. Tareas del Procesamiento de Lenguaje Natural

- **Reconocimiento de voz:** La tarea pertinente al reconocimiento de voz es la conversión de la voz a texto, requiere convertir de manera confiable datos de voz en datos de texto. Lo que hace que el reconocimiento de voz sea especialmente desafiante es la forma en que las personas hablan: velocidad, arrastrando las palabras, con diferentes énfasis y entonación, así como con diferentes acentos.
- **Etiquetado de palabras:** Es el proceso de determinar la parte gramatical de una palabra o fragmento de texto en particular, en función de su uso y contexto.
- **Desambiguación de las palabras:** Es la selección del significado de una palabra con múltiples significados a través de un proceso de análisis semántico que determina la palabra que tiene más sentido en el contexto dado.
- **Reconocimiento de entidad nombrada:** Identifica palabras o frases como entidades útiles, es decir, identifica nombres propios.
- **Resolución de correferencia:** Es la tarea de identificar si dos palabras se refieren a la misma entidad y cuándo. El ejemplo más común es determinar la persona u objeto al que se refiere un determinado pronombre.
- **Análisis de sentimientos:** Intenta extraer del texto cualidades subjetivas (actitudes, emociones, sarcasmo, confusión, sospecha).
- **Generación de lenguaje natural:** Es la tarea de producir texto en lenguaje comprensible por un humano.

2.3.2. Usos del Procesamiento de Lenguaje Natural

El procesamiento del lenguaje natural es el cerebro detrás de la inteligencia artificial en muchas aplicaciones del mundo real. Algunos ejemplos son los siguientes:

- **Aplicaciones dedicadas a la traducción:** un ejemplo de estos es Google Translate, el cual hace uso de BERT para el procesamiento del lenguaje natural así como para el etiquetado de las palabras que requiere traducir.
- **Asistentes virtuales y chatbots:** Asistentes como Siri o Alexa requieren de un reconocimiento de voz para distinguir comandos, así como para responder adecuadamente a estos. En el caso de los chatbots realizan un análisis contextual de las preguntas para proveer respuestas relacionadas a las mismas.
- **Análisis de sentimientos en redes sociales:** Consta de analizar el lenguaje utilizado en publicaciones en redes sociales, respuestas, reseñas y más para extraer actitudes y emociones en respuesta a productos, promociones y eventos; información que las empresas pueden usar en diseños de productos, campañas publicitarias y más
- **Resúmenes de textos:** Consta de realizar resúmenes y sinopsis de distintos textos.

2.4. Transformers

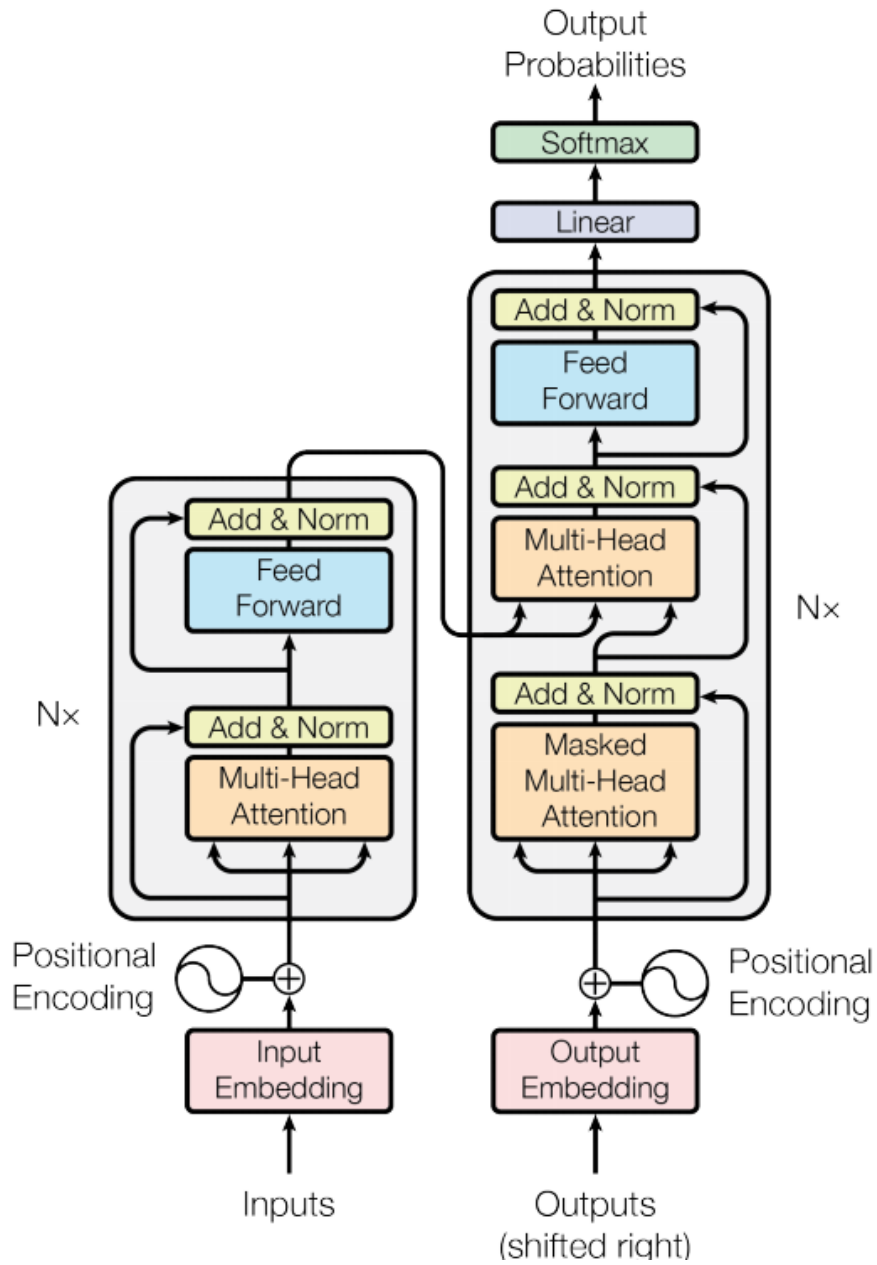
El Transformer en Procesamiento del lenguaje natural (PLN) es una arquitectura que tiene como objetivo codificar cada palabra de una frase en función del resto de la secuencia, permitiendo así introducir el contexto en la representación matemática del texto.

Se suele trabajar con los transformadores en 2 etapas: [24]

- **Pre-entrenamiento:** en esta fase, el modelo aprende cómo se estructura el lenguaje de forma general, además de conseguir un conocimiento genérico del significado de las palabras.
- **Afinado:** se le añaden ciertas capas a la arquitectura para adaptar los modelos a tareas concretas, y se les re-entrena en esas tareas.

2.4.1. Arquitectura de un transformer

Figura 2.4: Arquitectura general de un Transformer [25]



2.5. BERT

2.5.1. ¿Qué es Bert?

Bidirectional Encoder Representations from Transformers (BERT) es un marco de aprendizaje automático de código abierto para el Procesamiento del lenguaje natural (PLN). BERT, significa Representaciones de codificador bidireccional de transformers, basado en transformers, el cual es un modelo de aprendizaje profundo en el que cada elemento de salida está conectado a cada elemento de entrada y las ponderaciones entre ellos se calculan dinámicamente en función de su conexión. [26]

Al contar con la capacidad bidireccional, BERT está previamente entrenado en dos tareas de PLN diferentes, pero relacionadas: el modelado de lenguaje enmascarado y la predicción de la siguiente oración.

El objetivo del entrenamiento del Modelado de lenguaje enmascarado (MLM) es ocultar una palabra en una oración y luego hacer que el programa prediga qué palabra se ha ocultado en función del contexto de la palabra oculta. El objetivo del entrenamiento de predicción de la siguiente oración es que el programa prediga si dos oraciones dadas tienen una conexión lógica secuencial o si su relación es simplemente aleatoria.

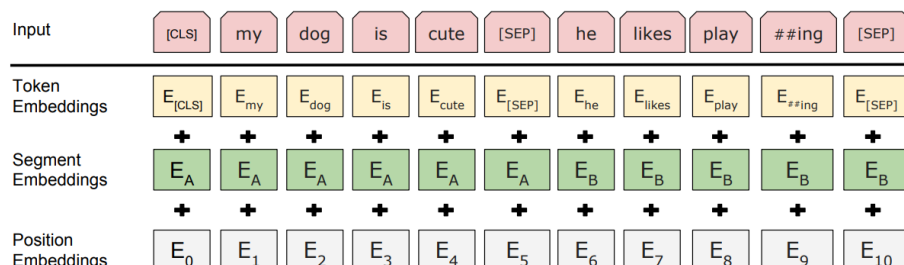
2.5.2. ¿Cómo funciona BERT?

El objetivo de cualquier técnica de PLN es comprender el lenguaje humano tal como se habla de forma natural. En el caso de BERT, esto normalmente significa predecir la siguiente palabra. Para hacer esto, los modelos normalmente necesitan entrenarse usando un gran repositorio de datos de entrenamiento especializados y etiquetados.

BERT, sin embargo, fue entrenado previamente usando solo un corpus de texto plano sin etiquetar (textos de Wikipedia en inglés). Continúa aprendiendo sin supervisión del texto sin etiquetar y mejorando incluso cuando se usa en aplicaciones prácticas. Su entrenamiento previo sirve como una capa base de “conocimiento”. A partir de ahí, BERT puede adaptarse al creciente cuerpo de contenido y consultas para ajustarse a las especificaciones del usuario. Este proceso se conoce como aprendizaje por transferencia.

BERT es posible gracias a la investigación de Google sobre transformers. El transformer es la parte del modelo que le da a BERT su mayor capacidad para comprender el contexto y la ambigüedad en el lenguaje. El transformer hace esto procesando cualquier palabra dada en relación con todas las demás palabras en una oración, en lugar de procesarlas una a la vez.

Figura 2.5: Representación de entrada de BERT y la separación de la oración por palabras y la asignación de valores.



En la figura anterior, lo que se realiza es que se inserta un token [CLS] al principio de la primera oración y un token [SEP] al final de cada oración, cada palabra es separada en una ficha, a cada ficha se agrega una inserción que indica si se trata de la oración A o la oración B, finalmente se agrega una incrustación posicional a cada token para indicar su posición en la secuencia. [27]

Esto contrasta con el método tradicional de procesamiento del lenguaje, conocido como incrustación de palabras, en el que los modelos mapeaban cada palabra en un vector, que representa sólo una dimensión, es decir, el significado de esa palabra.

2.5.3. BERT en la actualidad

BERT se utiliza actualmente en Google para optimizar la interpretación de las búsquedas de los usuarios.

Así como en la realización de tareas tales como: [28]

- Generación de lenguaje basadas en secuencia a secuencia (Respuesta a preguntas, resúmenes de documentos, predicción de siguiente oración, chatbots).
- Comprensión del lenguaje natural (Entendimiento de la polisemia, la coreferencia, la desambiguación de las palabras, clasificación de las palabras por sentimientos).

2.6. GPT-2

2.6.1. ¿Qué es GPT-2?

Generative Pretrained Transformer (GPT) es un transformer aprovechado para realizar tanto aprendizaje supervisado como aprendizaje no supervisado

para el Procesamiento del lenguaje natural (PLN).

GPT-2 es el sucesor de GPT, es un gran modelo de lenguaje basado en transformers con 1500 millones de parámetros, entrenado en un conjunto de datos de 8 millones de páginas web. GPT-2 se entrena con un objetivo simple: predecir la siguiente palabra, dadas todas las palabras anteriores dentro de un texto. [29]

El conjunto de datos no requiere ningún paso de procesamiento previo. En otras palabras, se omiten las mayúsculas y minúsculas, la tokenización y otros pasos, ya que los autores creen que estos pasos de preprocesamiento restringen la capacidad del modelo y así ser capaz de evaluar todos los puntos de referencia del lenguaje.

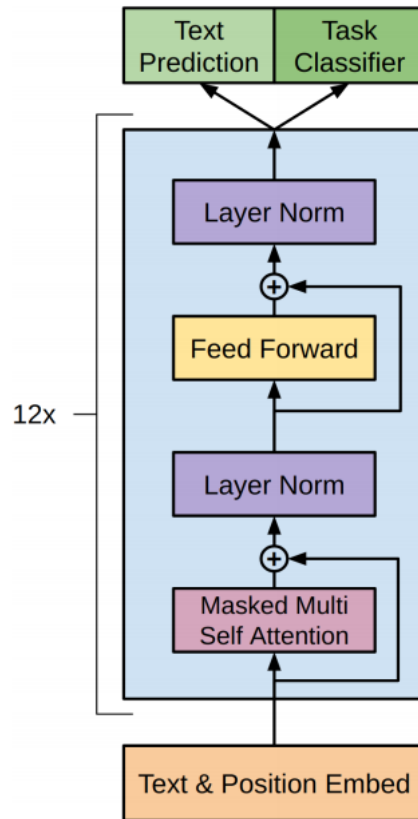
Ya que en GPT no se aplica una representación de las palabras ni a nivel de palabra ni a nivel de carácter. Se elige una opción que se encuentra situada en medio, que es la subpalabra. La subpalabra se puede obtener mediante el algoritmo Byte Pair Encoding (BPE).

BPE es una forma de compresión donde se calculará una lista de subpalabras utilizando el siguiente algoritmo: [30]

- Dividir palabra en secuencia de caracteres.
- Unirse al patrón de frecuencia más alta
- Continuar con el paso anterior hasta alcanzar el número máximo predefinido de subpalabras de iteraciones.

2.6.2. Arquitectura GPT

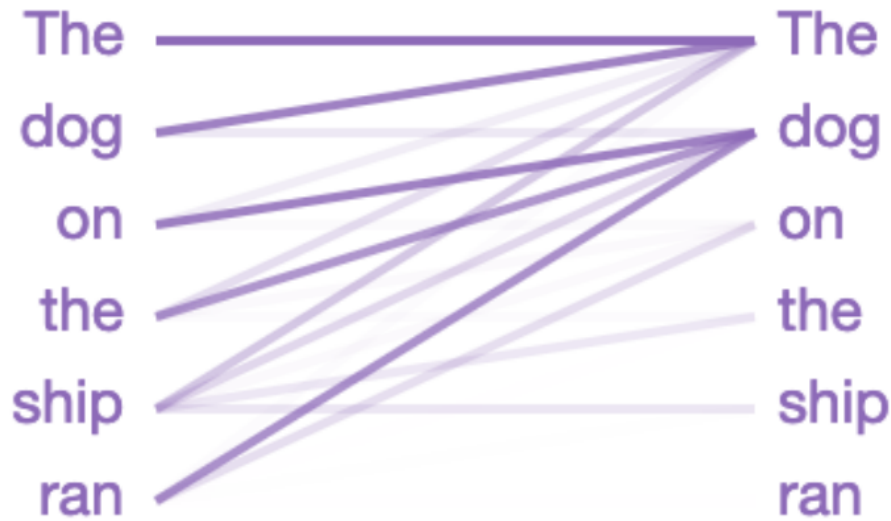
Figura 2.6: Arquitectura del modelo de GPT-2 [31]



El modelo de GPT es un transformer decodificador de 12 capas, cada una con 12 mecanismos de atención independientes, llamados “cabezas”; el resultado es $12 \times 12 = 144$ patrones de atención distintos. Cada uno de estos patrones de atención corresponde a una propiedad lingüística capturada por el modelo. [32]

Lo que le permite identificar la relación que existe entre las palabras, entendiendo la oración y permitiendo predecir la siguiente palabra en relación a la oración.

Figura 2.7: Ejemplo análisis de una oración por GPT-2



Las líneas, leídas de izquierda a derecha, muestran dónde presta atención el modelo prediciendo la siguiente palabra en la oración (la intensidad del color representa la fuerza de la atención). Entonces, al tratar de predecir la siguiente palabra después de correr (ran), el modelo presta mucha atención al perro (dog) en este caso. Esto ya que necesita saber quién o qué está corriendo para predecir qué viene a continuación. [33]

2.7. Base de datos

Una base de datos es una colección organizada de información o datos, los cuales pertenecen a un mismo contexto, se encuentran almacenados de forma física o digital con la finalidad de realizar distintas acciones como consultas futuras, ingreso de nuevos datos, actualización o eliminación de estos.

Las bases de datos se componen de una o más tablas divididas en columnas y filas y son las encargadas de guardar un conjunto de datos.

Una base de datos generalmente es manejada por un Sistema de gestión de base de datos (DBMS). En conjunto, los datos y el DBMS, junto con las aplicaciones asociadas a ellos, se les conoce como un sistema de base de datos.[34]

2.7.1. Base de datos NoSQL o no relacionales

Estas bases permiten que los datos no estructurados y/o semiestructurados se almacenen y manipulen, a diferencia de la base de datos relacional donde se define como deben de componerse todos los datos insertados en esta. Generalmente los registros de este tipo de base de datos suelen almacenarse como un documento de tipo JSON.

2.8. Canción

Una canción es una composición literaria, generalmente en verso, la cual puede ser acompañada con música para poder ser cantada.[35]

2.8.1. Elementos de una canción

Los elementos que conforman una canción son los siguientes:

- **Introducción**

Generalmente es una parte única la cual aparece al inicio de una canción, acompañada de una Armonía o Melodía compuesta solo para este inicio. El objetivo principal de la introducción es captar la atención y generar un ambiente.[36]

- **Verso**

Es la parte encargada de comenzar a desarrollar la idea a transmitir, trata de contarnos el tema de la canción, y ya cuenta con una Armonía bien establecida.

- **Pre-estribillo**

Es un arreglo que permite realizar una transición, su función principal es conectar el verso con el estribillo. También ayuda a evitar que el estribillo se estanque en la monotonía.

- **Estribillo**

Es una Estrofa la cual se repite varias veces dentro de una composición. La función principal del estribillo es destacar la idea de la canción tanto en la letra como en lo musical. El estribillo es considerado la parte más importante de la canción y en algunas ocasiones es repetido al inicio y al final de la misma.

- **Puente**

Es un interludio la cual conecta dos partes de una canción, permitiendo construir una Armonía entre ellas, suele ser usado para llevar a la canción a su clímax, para prepararlo para el desarrollo final de la canción.

- **Cierre**

Hay distintas formas de terminar o concluir una canción, puede ser un quiebre brusco generado por un silencio repentino o por una sucesión de Acordes. Pero la forma más común es haciendo uso de la repetición de un estribillo.

2.8.2. Estructura de una canción

La estructura mínima de una canción está compuesta de:

- Verso
- Estribillo
- Verso
- Estribillo

La estructura más usada en una canción es la siguiente

- Introducción
- Verso
- Pre-estribillo
- Estribillo
- Verso
- Estribillo
- Puente
- Cierre

2.9. Cadenas de Markov

Las cadenas de Markov son un Sistema matemático el cual experimenta con las transiciones de un Estado a otro de acuerdo con ciertas reglas probabilísticas. La característica que define a una cadena de Markov es que no importa cómo llegó el proceso a su estado actual, y sus posibles estados futuros son fijos. En otras palabras, la probabilidad de pasar a cualquier otro estado depende únicamente del estado actual y del tiempo transcurrido. El espacio de estados o conjunto de todos los posibles estados, pueden ser cualquier cosa: letras, números, puntuaciones de un partido, acciones, etc.

Son procesos Estocásticos, con la diferencia de que estos deben ser “sin memoria”, es decir, la probabilidad de las acciones futuras no depende ni se ve afectada de los pasos que la condujeron al estado actual. A esto se le denomina una propiedad de Markov.

En la teoría de probabilidad, el ejemplo más inmediato es el de una cadena de Markov homogénea en el tiempo, en la que la probabilidad de que ocurra cualquier transición de estado es independiente del tiempo.

En el lenguaje de probabilidad condicional y variables aleatorias, una cadena de Markov es una secuencia X_0, X_1, X_2, \dots de variables aleatorias que satisfacen la regla de independencia condicional. En otras palabras, el conocimiento del estado anterior es todo lo que se necesita para determinar la distribución de probabilidad del estado actual. Esta definición es más amplia que la explorada anteriormente, ya que permite probabilidades de transición no estacionarias y, por lo tanto, cadenas de Markov no homogéneas en el tiempo; es decir, a medida que pasa el tiempo los pasos aumentan y la probabilidad de pasar de un estado a otro puede cambiar. [37]

Las cadenas de Markov pueden ser modeladas mediante Máquinas de Estados Finitos.

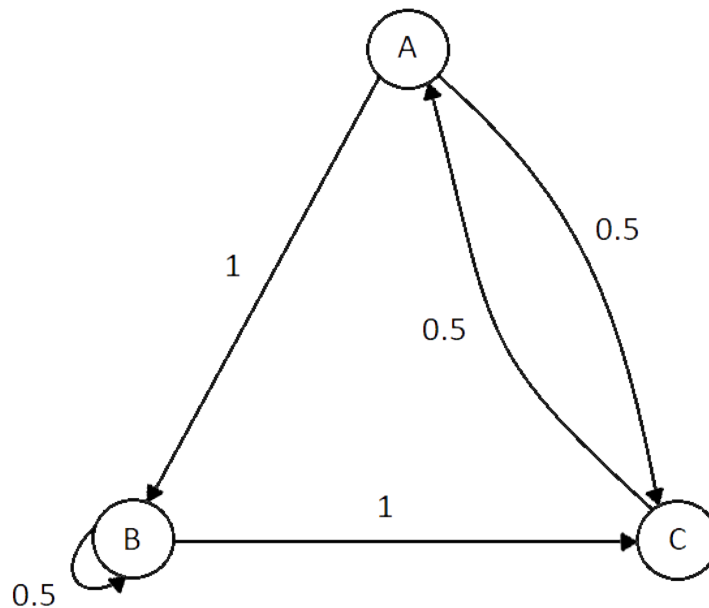


Figura 2.8: Diagrama De Estados finitos De Una Cadena De Markov

2.10. Flask

Flask es un marco (framework) web, esto es, un modulo de Python el cual permite desarrollar aplicaciones web. No cuenta con un Manejador de Objetos Relacionales u ORM por sus siglas en inglés, pero si cuenta con características como el enrutamiento de URLs y un motor de plantillas. En general es un marco de aplicación web WSGI.[40]

La Web Server Gateway Interface (WSGI) es una especificación que describe como se va a comunicar un servidor web con una aplicación web, y como se pueden llegar a enlazar distintas aplicaciones web para procesar una solicitud o una petición.[39]

Para poder empezar a trabajar con un proyecto o aplicación web Flask lo primero que se hace es usar virtualenv para poder manejar un ambiente virtual separado para el proyecto y así evitar conflictos entre dependencias de este. Virtualenv es una herramienta que nos ayuda a crear ambientes Python aislados (en forma de un directorio).

Para poder hacer esto, en la terminal se escribe:

```
virtualenv --no-site-packages myapp
```

Al usar ese comando se va a crear un directorio con la siguiente estructura:

- myapp/
 - bin/
 - include/
 - lib/

Si se trabaja en Windows la estructura es la siguiente:

- myapp/
 - lib/
 - Scripts/

Para poder trabajar con este ambiente, en la terminal se debe acceder a la carpeta bin o Scripts y activar el ambiente de la siguiente manera:

```
source myapp/bin/activate
source myapp/Scripts/activate
```

Posteriormente se procede a escribir el código de una aplicación, por ejemplo, el código mostrado a continuación es extraído de la documentación de flask.[38] Se guarda como cualquier otra aplicación de Python y se ejecuta de la misma manera que una aplicación de Python. Una vez ejecutada aparecerá lo siguiente en terminal:

```
* Running on http://127.0.0.1:5000/
```

Si se copia esa URL y se abre en un navegador se va a poder observar nuestra aplicación web en ejecución, en este caso, y habiendo hecho uso del ejemplo anterior, se mostrará en pantalla “Hello, World!”

Como se hizo mención al inicio, una de las características de Flask es que se pueden manejar las rutas de la aplicación web usando el método `route()`, siempre que un usuario visite esa ruta se va a ejecutar el método adjuntado a este.

Flask soporta diferentes tipos de estructuras de variables. Se pueden incluir cadenas personalizadas, números enteros y otros tipos de caracteres en las URLs.

Si se requiere manejar solicitudes, Flask tiene soporte para solicitudes de tipo GET y POST, por defecto Flask solo admite peticiones de tipo GET, por lo que si se necesita hacer uso de peticiones tipo POST esta debe de estar especificada como parámetro de la función `route()`, por ejemplo:

```
@app.route('/login', methods=['POST'])
```

2.11. Apache

Es un servidor web gratuito y de código abierto que permite que los dueños de sitios web muestren contenido en ellas, no es un servidor físico sino más bien un software que corre en un servidor. Su trabajo es establecer la conexión entre un servidor y los navegadores de los visitantes del sitio web mientras se mandan archivos de ida y vuelta entre ellos (estructura cliente-servidor). El servidor y el cliente se comunican mediante el protocolo HTTP, y el software de Apache es responsable por la comunicación fluida y segura entre las 2 máquinas.

Apache trabaja sin problemas con muchos otros sistemas de gestión de contenido (Joomla, Drupal, etc.), marcos de trabajo web (Django, Laravel, etc.), y lenguajes de programación. Todo esto en conjunto lo vuelve una opción sólida al momento de escoger entre todos los tipos de plataformas de alojamiento web, como serían VPS o alojamiento compartido.

Cuadro 2.1: Tabla comparativa de los distintos servidores web contemplados para nuestro proyecto

Apache	NGINX	Tomcat
Apache viene con ventajas útiles sobre Nginx, como sería su fácil configuración, muchos módulos, y un entorno amigable con los primerizos.	Nginx fue creado para resolver el llamado “problema c10k”, significando que un servidor web que usa hilos para manejar solicitudes del usuario es incapaz de gestionar más de 10,000 conexiones al mismo tiempo.	Tomcat fue creado específicamente para aplicaciones Java, mientras que Apache es un servidor HTTP de propósito general.
De código abierto y gratuito, incluso para su uso comercial. Software estable y confiable. Frecuentemente actualizado incluyendo actualizaciones regulares a los parches de seguridad.	Nginx es uno de los servidores web que soluciona el problema c10k y probablemente el más exitoso al hacerlo, no crea un nuevo proceso por cada solicitud y en su lugar, maneja toda solicitud entrante en un único hilo.	Tomcat es menos configurable comparado a otros servidores web. Por ejemplo, para correr WordPress, la mejor elección es un servidor HTTP de propósito general como Apache o Nginx.
Funciona de forma intuitiva, con sitios web hechos con WordPress. Comunidad grande y posibilidad de contactar con soporte disponible de manera sencilla en caso de cualquier problema.	El modelo basado en eventos de Nginx distribuye solicitudes de usuario entre procesos trabajadores de una manera eficiente, llevando con ello a una mejor escalabilidad.	
Problemas de rendimiento en sitios web con tráfico extremadamente pesado. Tantas opciones de configuración pueden llevar a vulnerabilidades de seguridad.	Si necesitas manejar un sitio web con un alto tráfico, Nginx es una excelente opción, puesto que puede hacerlo con un mínimo uso de los recursos.	

2.12. Servidor Web

El trabajo de un servidor web es mostrar sitios web en internet. Para conseguir este objetivo, actúa como un intermediario entre el servidor y las máquinas del cliente. Obtiene el contenido del servidor en cada petición hecha por el usuario y lo entrega al sitio web.

Los servidores web procesan archivos escritos en diferentes lenguajes de programación como PHP, Python, Java, y otros.

Cuando se escuche la palabra ‘servidor web’, se puede pensar en ello como la herramienta responsable por la correcta comunicación entre el cliente y el servidor.

2.13. Certificado SSL

Conocido como (Secure Sockets Layer) o (Capa de Conexión Segura) es un estándar de seguridad global que fue originalmente creado por Netscape en los 90's. SSL crea una conexión encriptada entre tu servidor web y el navegador web de tu visitante permitiendo que la información privada sea transmitida sin que ocurran problemas como serían espionaje, manipulación de la información, y falsificación de los datos del mensaje. Básicamente, la capa SSL permite que dos partes tengan una “conversación” privada.

Para establecer esta conexión segura, se instala en un servidor web un certificado SSL (también llamado 'certificado digital') que cumple dos funciones:

- Autenticar la identidad del sitio web, garantizando a los visitantes que no están en un sitio falso.
- Cifrar la información transmitida.

Hay varios tipos de certificados SSL según la cantidad de nombres de dominio o subdominios que se tengan, como por ejemplo:

- Único: Asegura un nombre de dominio o subdominio completo (FQDN por sus siglas en inglés).
- Comodín: Cubre un nombre de dominio y un número ilimitado de sus subdominios.
- Multidominio: Asegura varios nombres de dominio.

2.14. Plataforma en la Nube de Aprendizaje Automático

El entrenamiento de aprendizaje automático y de modelos de aprendizaje profundo involucra miles de iteraciones. Se necesita esta gran cantidad de iteraciones para producir el modelo más preciso.

El cómputo en la nube permite modelar capacidad de almacenamiento y manejar cargas a escala, o escalar el procesamiento a través de los nodos. Por ejemplo, AWS ofrece instancias de GPUs con capacidad de memoria que va de los 8Gb's a los 256Gb's, estas instancias son cobradas a ritmos por hora. Las GPUs son procesadores especializados diseñados para procesamiento complejo de imágenes. Azure de Microsoft ofrece GPUs de alto rendimiento de la serie NC para aplicaciones o algoritmos de cómputo de alto rendimiento.

2.14.1. Amazon

Dentro de las bondades que ofrecen los Servicios Web de Amazon (AWS por sus siglas en inglés) y que nos pueden ser de utilidad para las necesidades de nuestro proyecto se pueden encontrar:

- **SageMaker:** Es una plataforma de aprendizaje automático completamente administrado para científicos de datos y desarrolladores. La plataforma corre en Cómputo Elástico en la Nube (EC2 por sus siglas en inglés), y permite construir modelos de aprendizaje automático, organizar la información y escalar sus operaciones. Algunas aplicaciones de aprendizaje automático en SageMaker van desde reconocimiento de voz hasta visión de computadora, e incluso recomendaciones basadas en el comportamiento aprendido del usuario.

El mercado de AWS ofrece modelos que se pueden usar en lugar de empezar desde cero, posterior a eso se puede entonces empezar a entrenar y optimizar el modelo; las elecciones más comunes son frameworks como Keras, TensorFlow, y PyTorch; Sagemaker puede optimizar y configurar estos frameworks automáticamente, o pueden ser entrenadas de manera personal.

Uno mismo puede incluso desarrollar su propio algoritmo construyéndolo en un contenedor de Docker o se puede hacer uso de una “Jupyter notebook” para construir un modelo propio de aprendizaje automático y visualizar su información usada para el entrenamiento del modelo siendo este punto lo que más nos interesa para nuestro proyecto.

Cuadro 2.2: Tabla comparativa de las diversas plataformas contempladas

Característica	GCP	AWS	Azure
Jupyter Notebook alojado de manera local o remota	Plataforma de IA	SageMaker Studio IDE	- Estudio de Notebooks de Azure de Aprendizaje Automático - Databrick de Azure
Entrenamiento Distribuido	Si	Si	Si
Versionado de Modelos	Si	Si	Si
Seguimiento de Experimentos	Si	Si	Si
AutoML			
(UI y API)	Tabla de AutoML	Autopiloto de SageMaker	AutoML
Análisis de Errores	Tabla de AutoML con BigQuery	Debugger de SageMaker	Aprendizaje Profundo de Azure

Los 3 proveedores de la tabla anterior han alojado servicios de “Jupyter Notebook” (contienen tanto código de computadora como elementos ricos en texto como ecuaciones, figuras, etc.), experimentando seguimientos y control de versiones, y métodos de despliegue sencillos.

Dentro de las características únicas de cada una de las plataformas podemos encontrar que la Plataforma de la Nube de Google (GCP) usa un paquete llamado “what if tool” el cual se puede integrar junto a un “Jupyter Notebook” y de esa manera jugar con el modelo cambiando el umbral o un valor característico de un ejemplo dado, esto permite checar como es que ciertos cambios afectan el resultado predicho con anterioridad previo al cambio.

El debugger de SageMaker de AWS permite analizar cómo es que la ingeniería de características y el refinado del modelo son hechos, o de forma más concisa, permite ver qué sucede durante el entrenamiento del modelo.

Azure, por su parte, provee un módulo propio en su SDK el cual parece tener la mejor integración de entre las tres plataformas.

Costo y Rendimiento del Modelo

Azure y GCP puntúan ligeramente mejor que AWS en términos de rendimiento, esto no necesariamente significa que una plataforma es mejor que otra.

El costo, por otro lado, resulta ser más interesante. El costo de AWS fue considerablemente menor que GCP y Azure. Por supuesto, esta no es estrictamente una comparación de manzanas-manzanas.

El diseñador de aprendizaje automático de Azure cuenta con una interfaz de arrastrar y soltar el cual es bastante amigable con aquellos nuevos en la ciencia de datos, con menos codeo ni antecedentes técnicos. AWS y GCP parecen ser más enfocados a desarrolladores. Aunque puede resultar en un poco más de trabajo ensamblar una pipeline (o cadena de procesos), son más personalizables con los diferentes componentes disponibles. Estos componentes y la conexión de la pipeline son usualmente desarrollados usando código y configuraciones, en vez de haciendo uso de una interfaz.

Tanto GCP como AWS ofrecen un modelo de pago “pay-as-you-go”. Este modelo es el mejor para aquellos individuos que puedan llegar a esperar un uso intermitente de la nube, ya que permite un enfoque flexible para añadir y remover servicios cuando se necesite. Por supuesto, este nivel de flexibilidad tiene un costo, haciendo al modelo “pay-as-you-go” el más caro por hora.

Cuadro 2.3: Tabla comparativa GCP vs AWS (en dólares)

Tipo de Instancia	Precio de EC2 (por hora)	Precio de Google (por hora)
Propósito General	\$0.134	\$0.15
Cómputo Optimizado	\$0.136	\$0.188
Optimizado de Memoria	\$0.201	\$0.295
GPU	\$0.526	\$1.4

En conclusión, la opción ideal para las necesidades de nuestro proyecto es AWS debido a que es más económico que las otras dos plataformas y cuenta con herramientas más útiles como SageMaker que nos permitirá saber cómo es que se está entrenando el modelo.

Capítulo 3

Análisis.

3.1. Estudio de Factibilidad.

El estudio de factibilidad es un instrumento que sirve para orientar la toma de decisiones, así como para determinar la posibilidad de desarrollar un negocio o un proyecto; corresponde a la última fase de la etapa pre-operativa del ciclo del proyecto. Se formula con base en información que tiene la menor incertidumbre posible para medir las posibilidades de éxito o fracaso de un proyecto, apoyándose en el resultado se tomará la decisión de proceder o no con su implementación.

Este estudio establecerá la viabilidad, si existe, del trabajo planteado previamente.

- **Factibilidad Técnica:** Este aspecto evalúa que la infraestructura, es decir, los equipos, el software, el conocimiento, la experiencia, etc. Que se poseen son los necesarios para efectuar las actividades requeridas para la realización del trabajo terminal.
- **Factibilidad Operativa:** Analiza si el personal posee las competencias necesarias para el desarrollo del proyecto.
- **Factibilidad Económica:** Consiste en el análisis de los recursos financieros necesarios para llevar a cabo la elaboración de este proyecto.

3.1.1. Factibilidad Técnica

Dentro de este apartado se explican detalladamente las tecnologías que se utilizarán. La elección de estas herramientas estuvo basada tanto en las tecnologías que más se usan en la actualidad como en las que cuentan con

el mayor soporte para su trabajo en la nube, a su vez se detallan las características de los equipos de cómputo con los que se cuenta actualmente.

Factibilidad Técnica	
Sistema Operativo	Linux, Mac, Windows
Navegador Web	Google Chrome
Lenguaje de Programación	Python
Servidor	Apache y Gunicorn
Servicio Nube	Amazon Web Services

Cuadro 3.1: Herramientas de Software a utilizar

Además de las herramientas de software a utilizar, es necesario mencionar el equipo de hardware que se utiliza, tanto para desarrollar, como para probar e implementar cada uno de los prototipos a lo largo de este trabajo terminal.

Equipo de hardware utilizado. [1]	
Procesador	Ryzen 5 3600
Tarjeta de video	Amd Radeon Rx580
Memoria RAM	32 Gb
Disco duro	1Tb HDD y 512Gb SSD

Equipo de hardware utilizado. [2]	
Marca	Apple
Modelo	iMac Late 2012
Procesador	Intel Core i5
Tarjeta de video	NVIDIA GeForce GT 640M 512 Mb
Memoria RAM	8 Gb
Disco duro	1Tb y 256 Gb SSD

Equipo de hardware utilizado. [3]	
Procesador	Amd FX-8350
Tarjeta de video	Nvidia Geforce 1050ti
Memoria RAM	16 Gb
Disco duro	1Tb HDD

Junto con las herramientas de hardware y software a utilizar es necesario mencionar los servicios básicos que son relevantes para el desarrollo de este trabajo terminal como lo son:

- Luz Eléctrica
- Agua Potable
- Internet

Estos servicios forman parte de la factibilidad técnica ya que sin ellos no se podría realizar este proyecto y por eso mismo generan un costo, dicho costo se menciona en la Factibilidad Económica.

3.1.2. Factibilidad Operativa

A continuación se presenta una tabla con los recursos operativos del trabajo terminal que se calcularon con base en los recursos humanos con los que se cuenta actualmente y un análisis de las horas en las que el personal estará en operación:

Horas a trabajar en el desarrollo del trabajo terminal					
Mes	No. de Días	Sábado y Domingo	Días hábiles	Horas de trabajo por día	Horas Totales
Marzo	31	8	22	2	44
Abril	30	8	15	2	30
Mayo	31	10	19	2	38
Junio	30	10	17	2	34
Agosto	31	10	18	2	24
Septiembre	30	8	20	2	40
Octubre	31	10	20	2	40
Noviembre	31	8	18	2	36

Cuadro 3.2: Relación de horas de trabajo estimadas para la realización de este trabajo terminal

Con esto podemos concluir que contamos con 286 horas, suficiente tiempo para el desarrollo de este trabajo terminal, ya que las horas totales de trabajo están contempladas para cada uno de los integrantes del equipo

3.1.3. Factibilidad Económica

Luego de haber realizado el estudio de factibilidad técnica así como el operacional es necesario tomar en cuenta un estudio de factibilidad económica el cual desglosará todo el gasto económico realizado para la elaboración de este trabajo terminal:

- **Capital Humano:** Se tienen contemplados aproximadamente 36 días laborales, es decir, 288 horas para la elaboración de este trabajo terminal en el cual participaremos los tres integrantes
- **Capital Técnico:** Se cuenta con las viviendas y el equipo de cómputo principal de cada uno de los integrantes.

Respecto a los costos monetarios de todo el proyecto se toma a consideración lo siguiente:

- **Servicios**
Se considera un gasto mensual aproximado de \$1,600.00 que al ser multiplicado por todo el tiempo de elaboración contemplado nos da un total de \$14,400.00.
- **Software**
Durante algunos periodos se va a hacer uso principalmente de herramientas gratuitas y de software libre, en cuanto al servicio en la nube se pretende hacer uso de AWS y trabajar inicialmente con los planes gratuitos que ofrece, en caso de que se lleguen a consumir los recursos de este plan, entonces se procederá a cambiarnos a otro plan superior donde el costo actual se cobra en \$ 0.15 centavos de dólar por hora de uso.
- **Hardware**
Se utilizarán los equipos de cómputo personal de cada integrante, lo que da un costo aproximado de \$24,012.00 aplicando los parámetros de vida útil de un equipo de cómputo y su depreciación anual del hardware.
- **Recursos Humanos**
Se estima un gasto aproximado de \$15,000.00 por cada integrante del equipo para la elaboración del proyecto con lo que se generará un gasto total de \$45,000.00

En conclusión, el costo final del desarrollo de este trabajo terminal es de:

\$83,420.00

Conclusión Tras realizar el análisis el estudio de factibilidad de este proyecto es pertinente decir que los integrantes no cuentan con el apoyo financiero y que el hardware mencionado ya es propiedad de los integrantes, por lo que el trabajo terminal se califica como “*Viable*” iniciando de esta manera su implementación acorde con las fechas mencionadas.

3.2. Herramientas a usar.

3.2.1. Software.

Para el desarrollo de software de este prototipo, es necesario hacer mención de algunas de las siguientes herramientas, para tener una idea clara sobre qué herramientas estamos utilizando y porque es que las estamos utilizando:

HTML5. HTML comenzó mucho tiempo atrás con una simple versión propuesta para crear la estructura básica de páginas web, organizar su contenido y compartir información, todo esto tenía la intención de comunicar información por medio de texto. El limitado objetivo de html motivó a varias compañías a desarrollar nuevos lenguajes y programas para agregar características a la web nunca antes implementadas.

Dos de las opciones propuestas fueron Java y Javascript; ambas fueron muy aceptadas y consideradas como el objetivo de la internet, sin embargo, con el crecimiento exponencial del internet, éste dejó de ser únicamente para los aficionados de los computadores y pasó a ser usado como un campo estratégico para los negocios y para la interacción social, ciertas limitaciones presentes en ambas tecnologías probaron ser una sentencia de muerte. Esta falta de integración resultó ser crítica y preparó el camino para la evaluación de un lenguaje del cual hablaremos un poco más a detalle después: JavaScript. Sin embargo, pese a su gran impacto, el mercado no terminó de adoptarlo plenamente y rápidamente su popularidad fue declinando, y el mercado terminó enfocando su atención a Flash. No fue hasta que los navegadores mejoraron su intérprete para JavaScript y la gente se empezaba a dar cuenta de las limitaciones que ofrecía Flash, que JavaScript fue implementado y comenzó a innovar la forma en la que se programaba la web. Al cabo de unos años, JavaScript, html y css eran considerados como la más perfecta combinación para evolucionar la Web.

HTML5 es una mejora de esta combinación, lo que unió todos estos elementos. HTML5 propone estándares para cada aspecto de la Web y también

un propósito claro para cada una de las tecnologías involucradas. A partir de esto, html provee los elementos estructurales, CSS se concentra en como volver esta estructura utilizable y atractiva a la vista, y JavaScript tiene todo lo necesario para brindar dinamismo y construir aplicaciones web completamente funcionales. Cabe mencionar que HTML5 funciona diferente dependiendo del navegador y la versión en la que se esté trabajando, algunos soportan más características o diferentes funcionalidades que otros.

CSS.

HTML5 fue evolucionando a un grado de combinación de estructura y diseño, sin embargo, la web demanda diseño y funcionalidad, no solamente organización estructural o definición de secciones, la función de CSS se concentra en volver la estructura de HTML utilizable y atractivo a la vista.

Oficialmente CSS no tiene nada que ver con HTML4, no es parte de la especificación, es de hecho, un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML. Al principio, atributos dentro de las etiquetas HTML proveían estilos esenciales para cada elemento, pero a medida que HTML evolucionó, la escritura de códigos se volvió más compleja y html por sí mismo no pudo satisfacer más las demandas de los diseñadores.

En consecuencia a esta demanda, CSS fue adoptado como la forma de separar la estructura de la presentación, y ha ido creciendo y ganando importancia, pero siempre desarrollado en paralelo enfocado en las necesidades de los diseñadores y apartado de la estructura de HTML.

La versión 3 de CSS sigue el mismo camino, pero esta vez con un mayor compromiso. La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño, Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web y esta razón por la que cada vez que mencionamos HTML5 también estamos haciendo referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas. Las nuevas características incorporadas en CSS3 están siendo implementadas e incluidas junto al resto de la especificación en navegadores compatibles con HTML5 [?].

JavaScript.

JavaScript es considerado como el lenguaje de programación de html y de la web. Es un lenguaje de programación fácil de usar y muy versátil para el

ámbito de la comunicación en redes. Los programas, llamados “scripts”, se ejecutan en el navegador (Mozilla, Google Chrome, Internet Explorer, etc.) normalmente consisten en unas funciones que son llamadas desde el propio html cuando algún evento sucede.

Su primera aproximación a un uso real, fue en mayor parte para “dar vida a una página web”, como dar animaciones a un botón, interacciones en tiempo real, entre otras más. JavaScript fue desarrollado por Brendan Eich, a partir del lenguaje Java, que en ese momento tenía mucho auge y popularidad, y su principal diferencia es que JavaScript sólo “funciona” dentro de una página html.

JavaScript fue declarado como estándar del European Computer Manufacturers Association (ECMA) en 1997, y poco después, también fue estandarizado por ISO [?].

JavaScript es un lenguaje interpretado, usado mayormente como complemento de ciertos objetivos específicos, sin embargo, uno de las innovaciones que ayudó a JavaScript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento del código. La clave de los motores más exitosos fue transformar el código de Javascript en código máquina para obtener una velocidad de ejecución mejor que antes. Esto a la vez permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje JavaScript como la mejor opción para la Web.

Para aprovechar esta prometedora plataforma de trabajo ofrecida por los nuevos navegadores, JavaScript fue expandido en cuestión de portabilidad e integración, a la vez, interfaces de programación de aplicaciones (APIs) fueron incorporando por defecto con cada navegador para asistir a JavaScript en funciones elementales. El objetivo de esto, fue principalmente hacer disponible funciones a través de técnicas de programación sencillas y estándares, expandiendo el alcance del lenguaje y facilitando la creación de programas útiles para la Web [?].

OpenSSL.

OpenSSL es un proyecto de software libre basado en SSLeay, desarrollado por Eric Young y Tim Hudson.

Consiste en un robusto paquete de herramientas de administración y bibliotecas relacionadas con la criptografía, que suministran funciones criptográficas a otros paquetes como OpenSSH y navegadores web (para acceso seguro a sitios HTTPS). Estas herramientas ayudan al sistema a implementar el Secure Sockets Layer (SSL), así como otros protocolos relacionados

con la seguridad, como el Transport Layer Security (TLS). OpenSSL también permite crear certificados digitales que pueden aplicarse a un servidor, por ejemplo Apache [?].

MongoDB.

MongoDB es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre, esto significa que cada entrada o registro puede tener un esquema de datos diferente, con atributos o "columnas" que no tienen por qué repetirse de un registro a otro.

Las características más destacadas son su velocidad y su sencillo sistema de consulta de los contenidos de la base de datos. Alcanzando así un balance perfecto entre rendimiento y funcionalidad. MongoDB utiliza un modelo NoSQL el cual es un modelo de agregación que se basan en la noción de agregado, entendiendo el agregado como una colección de objetos relacionados que se desean tratar de forma semántica e independiente [52].

Las ventajas que ofrece MongoDB como herramienta de desarrollo de base de datos no relacionales son:

- La base de datos no tiene un esquema de datos predefinido.
- El esquema puede variar para instancias de datos que pertenecen a una misma entidad.
- En ocasiones el gestor de la base de datos no es consciente del esquema de la base de datos.
- Permite reducir los problemas de concordancia entre estructuras de datos usadas por las aplicaciones y la base de datos.
- Frecuentemente se aplican técnicas de desnormalización de los datos.

MySQL.

Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web [56].

El modelo relacional, para el modelado y la gestión de bases de datos, es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. Es

el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente [?].

Las principales ventajas son:

- Provee herramientas que garantizan evitar la duplicidad de registros.
- Garantiza la integridad referencial, así, al eliminar un registro elimina todos los registros relacionados dependientes.
- Favorece la normalización por ser más comprensible y aplicable.

Mientras que las principales desventajas son:

- Presentan deficiencias con datos gráficos, multimedia, CAD y sistemas de información geográfica.
- No se manipulan de forma eficiente los bloques de texto como tipo de dato.

Flask

Gunicorn

Python

Bert

GPT-2

EC2

SageMaker

CLI

SageMaker

S3

Apache

3.2.2. Hardware.

Se usarán los equipos de cómputo con los que los integrantes del equipo contamos actualmente, los cuales se especifican a continuación:

Equipo de hardware utilizado. [1]	
Procesador	Ryzen 5 3600
Tarjeta de video	Amd Radeon Rx580
Memoria RAM	32 Gb
Disco duro	1Tb HDD y 512Gb SSD

Equipo de hardware utilizado. [2]	
Marca	Apple
Modelo	iMac Late 2012
Procesador	Intel Core i5
Tarjeta de video	NVIDIA GeForce GT 640M 512 Mb
Memoria RAM	8 Gb
Disco duro	1Tb y 256 Gb SSD

Equipo de hardware utilizado. [3]	
Procesador	Amd FX-8350
Tarjeta de video	Nvidia Geforce 1050ti
Memoria RAM	16 Gb
Disco duro	1Tb HDD

3.3. Arquitectura del sistema.

3.3.1. Descripción de la arquitectura del sistema.

3.3.2. Aspectos Económicos

Aquí va lo que vienen siendo las licencias, descripción, software a utilizar, etc.

3.3.3. Aspectos Legales

3.4. Diagrama de casos de uso general.

Capítulo 4

Iteración 1.

4.1. Descripción.

4.2. Base de Datos

4.3. Género Musical

4.4. Resultados

Capítulo 5

Iteración 2.

- 5.1. Descripción.
- 5.2. Limpieza
- 5.3. Regular Expressions
- 5.4. Técnicas de Scrapping
- 5.5. Implementación
- 5.6. Resultados

Bibliografía

- [1] Real Academia Española (2020, noviembre), Diccionario de la lengua española, [En línea]. Disponible: <https://dle.rae.es> [Último acceso: 10 de diciembre del 2020].
- [2] Oxford Lexico (2020, noviembre), Definitions, Meanings, Synonyms, and Grammar by Oxford, [En línea]. Disponible: <https://www.lexico.com> [Último acceso: 2 de diciembre del 2020].
- [3] Chaney, D. (2012). The Music Industry in the Digital Age: Consumer Participation in Value Creation. *International Journal of Arts Management*, (1), pp. 15.
- [4] Sutskever, I., Martens, J., Hinton, G. E. (2011, January). Generating text with recurrent neural networks. In *ICML*.
- [5] Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.
- [6] Monteith, K., Martinez, T. R., & Ventura, D. (2012, May). Automatic Generation of Melodic Accompaniments for Lyrics. In *ICCC*, pp. 87-94.
- [7] Nikolov, N. I., Malmi, E., Northcutt, C. G., Parisi, L. (2020). Conditional Rap Lyrics Generation with Denoising Autoencoders. *arXiv preprint arXiv:2004.03965*.
- [8] Baker, F. A. (2015). What about the music? Music therapists' perspectives on the role of music in the therapeutic songwriting process. *Psychology of Music*, 43(1), pp. 122-139.
- [9] Guerrero, J. (2012). El género musical en la música popular: algunos problemas para su caracterización. *Trans. Revista transcultural de música*, (16), pp. 1-22.

- [10] Wang, A., & Cho, K. (2019). Bert has a mouth, and it must speak: Bert as a markov random field language model. arXiv preprint arXiv:1902.04094.
- [11] Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.
- [12] O. Jaramillo, Universidad Nacional Autonoma de México (2007, mayo 03), El concepto de Sistema, [En línea]. Disponible: <https://www.ier.unam.mx/ojs/pub/Termodinamica/node9.html> [Último acceso: 2 de diciembre del 2020].
- [13] J. A. Gutiérrez Orozco, Escuela Superior de Cómputo (2008, septiembre 15), Máquinas de Estados Finitos, [En línea]. Disponible: http://delta.cs.cinvestav.mx/~mcintosh/cellularautomata/Summer_Research_files/maquinasef.pdf [Último acceso: 15 de diciembre del 2020].
- [14] IBM Corporation (2021, marzo 31), Artificial Intelligence (AI), [En línea]. Disponible: <https://www.ibm.com/cloud/learn/what-is-artificial-intelligence> [Último acceso: 6 de abril del 2021].
- [15] L. Hardesty (2017, abril), Explained: Neural networks, [En línea]. Disponible: <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414> [Último acceso: 15 de noviembre del 2020].
- [16] amBientech (2019, julio 30), ¿Qué es la neurona?, [En línea]. Disponible: <https://www.uv.mx/mia/files/2012/10/Artificial-Neural-Networks-for-Beginners.pdf> [Último acceso: 6 de abril del 2021].
- [17] C. Gershenson (2012, octubre), Artificial Neural Networks for Beginners, [En línea]. Disponible: <https://ambientech.org/la-neurona> [Último acceso: 6 de abril del 2021].
- [18] Rubik's Code (2018, febrero), How do Artificial Neural Networks learn?, [En línea]. Disponible: <https://rubikscodene.net/2018/01/15/how-artificial-neural-networks-learn/> [Último acceso: 7 de abril del 2021].
- [19] S. Leijnen, F. van Veen (2020, mayo), The Neural Network Zoo, [En línea]. Disponible: https://www.researchgate.net/publication/341373030_The_Neural_Network_Zoo [Último acceso: 20 de noviembre del 2020].

- [20] A. Mehta (2019, enero 25), A Comprehensive Guide to Types of Neural Networks, [En línea]. Disponible: <https://www.digitalvidya.com/blog/types-of-neural-networks/> [Último acceso: 15 de noviembre del 2020].
- [21] P. Shukla, R. Iriondo (2020, agosto 11), Main Types of Neural Networks and its Applications, [En línea]. Disponible: <https://medium.com/towards-artificial-intelligence/main-types-of-neural-networks-and-its-applications-tutorial-734480d7ec8e> [Último acceso: 20 de noviembre del 2020].
- [22] IBM Cloud Education (2020, septiembre 14), What are Recurrent Neural Networks?, [En línea]. Disponible: <https://www.ibm.com/cloud/learn/recurrent-neural-networks> [Último acceso: 26 de marzo del 2021].
- [23] ref IBM Cloud Education (2020, julio 2), Natural Language Processing [En línea]. Disponible: <https://www.ibm.com/cloud/learn/natural-language-processing> [Último acceso: 20 de abril del 2021].
- [24] ref A. Vaca (2020, mayo), Transformers en Procesamiento del Lenguaje Natural, [En línea]. Disponible: <https://www.iic.uam.es/innovacion/transformers-en-procesamiento-del-lenguaje-natural/> [Último acceso: 15 de abril del 2021].
- [25] ref A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser I. Polosukhin (2017, diciembre), Attention Is All You Need, [En línea]. Disponible: <https://arxiv.org/pdf/1706.03762.pdf> [Último acceso: 15 de abril del 2021].
- [26] ref J. Devlin, M. Chang, K. Lee, K. Toutanova (2019, mayo), BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, [En línea]. Disponible: <https://arxiv.org/pdf/1810.04805.pdf> [Último acceso: 11 de abril del 2021].
- [27] ref R. Horev (2018, noviembre), BERT Explained: State of the art language model for NLP, [En línea]. Disponible: <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270> [Último acceso: 11 de abril del 2021].
- [28] ref B. Lutkevich (2020, enero), BERT language model, [En línea]. Disponible: <https://searchenterpriseai.techtarget.com/definition/BERT-language-model> [Último acceso: 12 de abril del 2021].

- [29] ref A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage I. Sutskever (2020, enero), Better Language Models and Their Implications, [En línea]. Disponible: <https://openai.com/blog/better-language-models/> [Último acceso: 14 de abril del 2021].
- [30] ref M. Edward (2019, febrero), Too powerful NLP model (GPT-2), [En línea]. Disponible: <https://towardsdatascience.com/too-powerful-nlp-model-generative-pre-training-2-4cc6afb6655> [Último acceso: 14 de abril del 2021].
- [31] ref A. Radford, K. Narasimhan, T. Salimans I. Sutskever (2018), Improving Language Understanding by Generative Pre-Training, [En línea]. Disponible: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf [Último acceso: 14 de abril del 2021].
- [32] ref J. Montantes(2019, mayo), Examining the Transformer Architecture, [En línea]. Disponible: <https://towardsdatascience.com/examining-the-transformer-architecture-part-1-the-openai-gpt-2-controversy-feceda4363bb> [Último acceso: 14 de abril del 2021].
- [33] ref J. Vig(2019, marzo), GPT-2: Understanding Language Generation through Visualization, [En línea]. Disponible: <https://towardsdatascience.com/openai-gpt-2-understanding-language-generation-through-visualization-8252f683b2f8> [Último acceso: 14 de abril del 2021].
- [34] Oracle México (2020, noviembre), ¿Qué es una base de datos?, [En línea]. Disponible: <https://www.oracle.com/mx/database/what-is-database/> [Último acceso: 20 de noviembre del 2020].
- [35] Escribir Canciones (2008), Estructura y elementos de una canción, [En línea]. Disponible: <https://dle.rae.es> [Último acceso: 2 de diciembre del 2020].
- [36] Swing this Music (2008), ¿QUÉ SECCIONES PUEDE TENER UNA CANCIÓN?española, [En línea]. Disponible: <https://sites.google.com/site/swingthismusiccast/interpretacio/estructura-cancion/secciones-de-una-cancion> [Último acceso: 2 de diciembre del 2020].
- [37] J. R. Norris (1997), Markov Chains, [En línea]. Disponible: <https://cape.fcfm.buap.mx/jdzf/cursos/procesos/libros/norris.pdf> [Último acceso: 15 de diciembre del 2020].

- [38] Flask (2019, julio 04), Flask's Documentation, [En línea]. Disponible: <https://flask.palletsprojects.com/en/1.0.x/> [Último acceso: 16 de diciembre del 2020].
- [39] Gajesh (2019, julio 17), The complete Flask beginner tutorial, [En línea]. Disponible: <https://dev.to/gajesh/the-complete-flask-beginner-tutorial-124i> [Último acceso: 16 de diciembre del 2020].
- [40] A. Abdelaal (2019), Deploying a Flask Application to Heroku, [En línea]. Disponible: <https://stackabuse.com/deploying-a-flask-application-to-heroku/> [Último acceso: 16 de diciembre del 2020].
- [41] Documentation Group. (n.d.). Welcome! - The Apache HTTP Server Project. Apache. Retrieved April 6, 2021, from <https://httpd.apache.org/>
- [42] G., D. (2021, March 9). What is Apache? An In-Depth Overview of Apache Web Server. Hostinger Tutorials. <https://www.hostinger.com/tutorials/what-is-apache>
- [43] What is a web server? - Learn web development — MDN. (2021, January 27). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server
- [44] <https://letsencrypt.org/es/>
- [45] What is SSL? (2017, May 15). SSLSHOPPER. <https://www.sslshopper.com/what-is-ssl.html>
- [46] Verisign. (2015, August 30). ¿Qué es un certificado SSL? – Verisign. Verisign NameStudio. https://www.verisign.com/es_LA/website-presence/online/ssl-certificates/index.xhtml
- [47] Bavati, I. (2020, September 22). Data Science in the Cloud - Towards Data Science. Medium. <https://towardsdatascience.com/data-science-in-the-cloud-239b795a5792> (September 2020)
- [48] G. (2021, March 5). Comparing Google Cloud Platform, AWS and Azure - Georgian Impact Blog. Medium. <https://medium.com/georgian-impact-blog/comparing-google-cloud-platform-aws-and-azure-d4a52a3adb2>
- [49] J. (2020, July 30). Choosing Your Deep Learning Infrastructure: The Cloud vs. On-Prem Debate. Determined AI. <https://determined.ai/blog/cloud-v-onprem>

- [50] I., A. (2015, October 2). Jupyter/IPython Notebook Quick Start Guide. Jupyter Notebook Beginner Guide. https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html
- [51] Jones, E. (2021, March 25). Google Cloud vs AWS in 2021 (Comparing the Giants). Kinsta. <https://kinsta.com/blog/google-cloud-vs-aws/>
- [52] MongoDB. (2019), ¿Qué es MongoDB?, [En línea]. Disponible: <https://www.mongodb.com/es>. [Último acceso: 26 Marzo 2021].
- [53] Universitat de València (2016), ¿Qué son las cookies? [En línea]. Disponible: <https://www.uv.es/uvweb/universidad/es/politica-privacidad/politica-cookies/son-cookies-1285919089226.html> [Último acceso: 26 Marzo 2021]
- [54] Noteworthy Programming Masterpiece (2019), openssl-nodejs [En línea]. Disponible: <https://www.npmjs.com/package/openssl-nodejs> [Último acceso: 26 Marzo 2021]
- [55] The Apache Software Foundation (2019), Apache Tomcat [En línea]. Disponible: <http://tomcat.apache.org/index.html> [Último acceso: 26 Marzo 2021]
- [56] Oracle (2018), MySQL [En línea]. Disponible: <https://www.oracle.com/mysql/> [Último acceso: 26 Marzo 2021]