INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Trabajo Terminal I.

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

TT2021-AXXX.

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

Directores:
Kolesnikova Olga
López Rojas Ariel

Índice.

Índice de figuras.

Índice de cuadros.

Parte I

Trabajo Terminal I

Capítulo 1

Introducción

En la actualidad 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. La generación de texto es una de las tareas más populares y desafiantes en el área de procesamiento del lenguaje natural. Recientemente, hay gran cantidad de trabajos (Generating Text with Recurrent Neural Networks [4], Convolutional Neural Networks for Sentence Classification[5]) los cuales propusieron generar texto utilizando redes neuronales recurrentes y redes neuronales 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 hacen que generar letras musicales sea mucho más difícil que generar 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 un tema es un asunto poco trabajado. Por

lo tanto, planeamos centrarnos en este nuevo problema. Estamos interesados en ver si el modelo propuesto puede aprender diferentes características en 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 iría 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 donde se 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 es 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 de manera eficiente colaborativamente, 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.

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, de herramientas y servicios así como investigación realizada para llevar a cabo en la implementación dentro del trabajo.

1.4.2. Capítulo 3. Análisis.

Dentro de este capitulo 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 3 componente 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 codificación. 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 hemos desarrollado para este TT1 (Componente I). Se muestra que el prototipo cumpla 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. Redes neuronales

El aprendizaje profundo (deep learning) es, de hecho, un nuevo nombre o enfoque de la inteligencia artificial llamada redes neuronales. Las redes neuronales son una forma de hacer que las computadoras aprendan, en 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 delante" (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 que de entrenamiento pasa a alimentar las capas inferiores (capas de entrada) la cual será procesada y pasada a posteriores capas donde será transformada hasta llegar a las capas superiores (capas de salida).[14]

2.1.1. Recurrent Neural Network – Long Short Term Memory

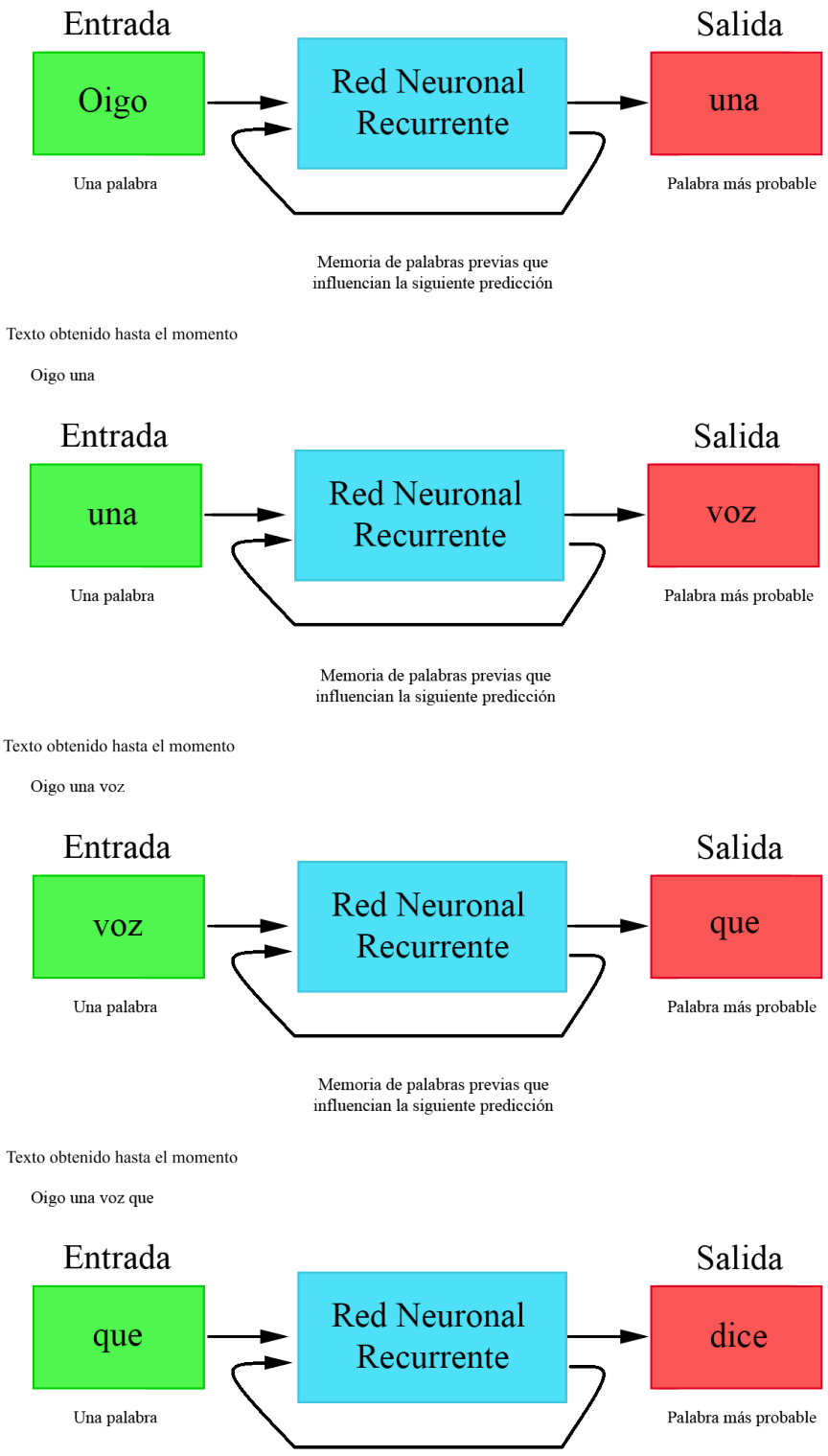
Una red neuronal Recurrente es un tipo de red neuronal artificial donde la salida de alguna capa en particular es salvada y sirve para retroalimentar la entrada de esta capa, lo cual ayuda a predecir futuras salidas de esta.

La primera capa esta forma 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 esta capa comenzara 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.[15]

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.[18]

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 2.1: Diagrama Red Neuronal Recurrente



2.2. 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 alguna consulta futura, ingreso de nuevos datos, actualización o eliminación de estos.

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

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.[19]

2.2.1. 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.

2.3. ¿Qué es una canción?

De manera resumida podemos decir que una canción es una composición literaria, generalmente en verso, a la que se le puede acompañar con música para poder ser cantada.[20]

2.3.1. Elementos de una canción

Los elementos que conforman una canción son los siguietes:

2.3.2. 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.[21]

2.3.3. Verso

En este se comienza a desarrollar la idea a transmitir, trata de contarnos de que va a ser la canción, y ya cuenta con una Armonía bien establecida.

2.3.4. 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.

2.3.5. Estribillo

Un 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 canción.

2.3.6. 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.

2.3.7. 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.

2.3.8. 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

2.4. Cadenas de Markov

Las cadenas de Markov son un Sistema matemático la cuales experimenta con las transiciones de un Estado a otro de acuerdo con ciertas reglas probabilísticas. La característica que definen 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, pero 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 mas inmediato es el de una cadena de Markov homogénea en el tiempo, en la que la probabilidad de que cualquier transición de estado es independiente del tiempo.

En el lenguaje de probabilidad condicional y variables aleatorias, una cadena de Markov es una secuencia X0, X1, X2, ... 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. [22]

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

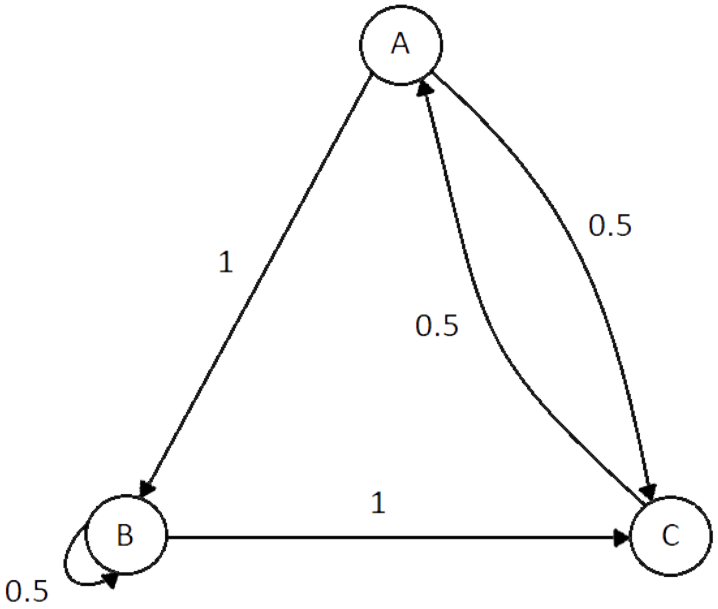


Figura 2.2: Diagrama De Estados finitos De Una Cadena De Markov

2.5. Flask

Flask es un marco (framework) web, el cual es un modulo de Python el cual permite desarrollar aplicaciones web. No cuenta con un Manejador de Objetos Realacionales u ORM por sus siglas en inglés (Object Relational Manager), pero si cuenta con características como el enrutamiento de URLS y un motor de plantillas. En general es un marco de aplicación web WSGI.[25]

WGSI son las siglas de Web Server Gateway Interface (Interfaz de Puerta de enlace del Servidor Web) la cual 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.[24]

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

Para poder hacer esto, en la terminal escribimos algo como:

virtualenv –no-site-packages myapp

Al usar ese comando creara 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 debemos acceder a la carpeta bin o Scripts y activar este ambiente de la siguiente manera:

source myapp/bin/actívate
source myapp/Scripts/activate

Luego se procede a escribir el código de una aplicación, por ejemplo, el siguiente extraído de la documentación de flask.[23]

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

Se guarda como cualquier otra aplicación de Python y se ejecuta de la misma manera que una aplicación de Python. Una vez ejecutada te aparecerá los siguiente en terminal:

* Running on http://127.0.0.1:5000/

Si copiamos esa URL y la abrimos en algún navegador podremos observar nuestra aplicación web en ejecución, en este caso solo se mostrará en pantalla "Hello, World!"

Como se había mencionado al inicio, una de las características de Flask es que podemos manejar las rutas de la aplicación web usando el método route(), siempre que un usuario visita esa ruta se ejecuta 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 requerimos manejar solicitudes, Flask tiene soporte para solicitudes de tipo GET y POST, por defecto Flask solo admite peticiones de tipo GET, si necesitamos hacer uso de peticiones tipo POST estas debe de estar especificada como parámetro de la función route(), por ejemplo:

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

Aun me falta mas por escribir sobre Flask...

2.6. Apache

Apache es un servidor web gratuito y de código abierto que permite que los dueños de sitios web mostrar 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 regreso 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. Palabra1.

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

Cuadro 2.1: Tabla comparativa de los distintos servidores web contemplados

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.	

2.7. 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. Jala contenido del servidor en cada petición del 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 escuchas la palabra 'servidor web', piensa en eso como la herramienta responsable por la correcta comunicación entre el cliente y el servidor.

2.8. Certificado SSL

Conocido como (Secure Sockets Layer) o (Capa de Conexión Segura) es un estándar de seguridad global que fue originalmente creado por Netscape in los 1990. SSL crea una conexión encriptada entre tu servidor web y el navegador web de tu visitante permitiendo que la información privada sea transmitida sin los problemas de espionaje, manipulación de la información, y falsificación del mensaje. Básicamente, la capa SSL permite que dos partes tengan una çonversación"privada.
Para establecer esta conexión segura, se instala en un servidor web un certificado SSL (también llamado çertificado digital") que cumple dos funciones:

- Autentificar 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.

2.9. Plataforma en la Nube de Aprendizaje Automático

Cuadro 2.2: Tabla comparativa de las diversas plataformas contempladas

Característica	GCP	AWS	Azure
Hosted and/or local notebook	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

Iteración 1.

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 él 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 la infraestructura, es decir, los equipos, el software, el conocimiento, la experiencia, etc. que se poseen son los necesarios para efectuar las actividades 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 del trabajo terminal.

3.1.1. Factibilidad Técnica

Dentro de este apartado se explica detalladamente las tecnologías que se utilizarán. La elección de estas herramientas estuvo basada en las tecnologías que más se usan en la actualidad, así como 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 Rx58
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 trabajo terminal y por eso mismo generan un costo, dicho costo se menciona en la Factibilidad Económica.

3.1.2. Factibilidad Operativa

Los recursos operativos de este trabajo terminal se calcularon con base en los recursos humanos con los que se cuenta actualmente y un análisis de las horas que el personal estará en operación trabajando sobre éste, el cual se muestra a continuación:

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 cuan 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 cuentan con las viviendas de cada uno de los integrantes y principalmente los equipos de cómputo correspondientes.

En cuanto a los costos monetarios de todo el trabajo terminal se cuenta con el siguiente equipo:

- **Servicios**
  En cuando a los servicios se considera un gasto mensual aproximado de $1,600.00 que al multiplicarlo por todo el tiempo de elaboración tenemos $ 14,400.00.
- **Software**
  En este caso durante algunos periodos se usarán herramientas gratuitas y de software libre, en cuanto al servicio en la de la nube de AWS, se planea trabajar inicialmente con alguno de los paquetes gratuitos, y si se llegaran a consumir los recursos de ese paquete, se procederá a unirse a un plan donde se cobran 0.15 centavos de dólar cada hora de uso.
- **Hardware**
  Se utilizarán los equipos de cómputo personal de cada integrante lo que daría un costo aproximado total de $ 24,012.00 aplicando los parámetros de vida útil de un equipo de cómputo y su depreciación anual.
- **Recursos Humanos**
  Se estima un gasto aproximado de $15,000.00 por cada integrante dell equipo para la elaboración del proyecto por lo que se generará un gasto total de $45,000.00

Por lo que el costo final del desarrollo de este trabajo terminal es:

$83,420.00

Conclusión Tras realizar el análisis el estudio de factibilidad de este proyectp 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 Palabra1; 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 carcterísticas 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 proveóan 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 Palabra1, 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 [**?**].

Apache Tomcat.

Apache Tomcat (también llamado Jakarta Tomcat o simplemente Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Oracle Corporation (aunque creado por Sun Microsystems).

Tomcat es un contenedor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador

Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache. Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java [39].

En este trabajo terminal, estaremos usando Apache Tomcat 9.

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 [36].

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 [40].

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.

3.2.2. Hardware.

En el ámbito del hardware, utilizaremos los equipos de cómputo con los cuales contamos actualmente los integrantes de este equipo, los cuales se especificarán a continuación:

Equipo de hardware utilizado. [1]	
Procesador	Ryzen 5 3600
Tarjeta de video	Amd Radeon Rx58
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

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] 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].

[15] 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].

[16] 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].

[17] 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].

[18] 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].

[19] 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].

[20] Escribir Canciones (2008), Estructura y elementos de una canción, [En línea]. Disponible: https://dle.rae.es [Último acceso: 2 de diciembre del 2020].

[21] 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].

[22] 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].

[23] 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].

[24] 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].

[25] 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].

[26] https://httpd.apache.org/

[27] https://www.hostinger.com/tutorials/what-is-apache

[28] https://developer.mozilla.org/en-US/docs/Learn/Common$_{q}uestions/What_{i}s_{aw}eb_{s}erverhttp$ $//httpd.apache.org/$

[29] https://letsencrypt.org/es/

[30] https://www.sslshopper.com/what-is-ssl.html

[31] https://www.verisign.com/es$_{L}A/website - presence/online/ssl - certificates/index.xhtmlhttps : //towardsdatascience.com/data - science - in - the - cloud - 239b795a5792(September2020)$

[32] https://medium.com/georgian-impact-blog/comparing-google-cloud-platform-aws-and-azure-d4a52a3adbd2 (May 2020)

[33] https://determined.ai/blog/cloud-v-onprem/ (July, 2020)

[34] https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what$_i s_j upyter.html https : //kinsta.com/blog/google - cloud - vs - aws/(March, 2021)

[36] MongoDB. (2019), ¿Qué es MongoDB?, [En línea]. Disponible: https://www.mongodb.com/es. [Último acceso: 26 Marzo 2021].

[37] Universitat de Valéncia (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]

[38] Noteworthy Programming Masterpiece (2019), openssl-nodejs [En línea]. Disponible: https://www.npmjs.com/package/openssl-nodejs [Último acceso: 26 Marzo 2021]

[39] The Apache Software Foundation (2019), Apache Tomcat [En línea]. Disponible: http://tomcat.apache.org/index.html [Último acceso: 26 Marzo 2021]

[40] Oracle (2018), MySQL [En línea]. Disponible: https://www.oracle.com/mysql/ [Último acceso: 26 Marzo 2021]