



INSTITUTO TECNOLÓGICO DE IZTAPALAPA
INGENIERÍA EN SISTEMAS COMPUTACIONALES

TEMA:

IMPLEMENTACION DE UN MODULO DE IA EN EL SISTEMA TO DO LIST
PARA GINP DEL STC

LUGAR DE REALIZACIÓN:

INSTITUTO TECNOLÓGICO DE IZTAPALAPA

ASESOR INTERNO:

M.C. Abiel Tomás Parra Hernández

PRESENTA:

EDUARDO LARA GOMEZ
JUAN PABLO MARTINEZ ALVAREZ

No DE CONTROL:

161080147

161080140

CIUDAD DE MÉXICO

MES / AÑO





RESUMEN

El proyecto titulado 'Implementación de un Módulo de IA en el sistema to do list para la gerencia de GINP' es un sistema de organización de trabajo por medio de un planograma el cual contiene diferentes leyendas para organizar las tareas de acuerdo a la importancia el cual ya está creado y deberá, contener un módulo de ia el cual deberá adaptarse y a su vez ayudará a definir la importancia para que las tareas se puedan colocar en el lugar óptimo La importancia de definir las partes que conforman de manera general al proyecto final recae en la necesidad de entender el funcionamiento del mismo , a lo largo de este documento se definirán las estructuras , necesidades, justificación , funciones de manera textual y gráfica , finalmente la implementación del módulo de IA es un tipo de buscador chatbot con IA que puede ayudar a tomar decisiones sobre la importancia de las tareas basado y construido con brain.js y la filosofía de software libre implementado como un módulo dentro del sistema principal el proyecto parte del módulo de IA que deberá estar adaptado a las tecnologías ya implantadas al tablero GINP como son :javascript , php, .





ÍNDICE

RESUMEN	2
INTRODUCCIÓN	4
JUSTIFICACIÓN	4
OBJETIVO GENERAL	6
CAPÍTULO 1 GENERALIDADES DE LA EMPRESA	6
1.1 ANTECEDENTES HISTÓRICOS DE LA EMPRESA	6
CAPÍTULO 2 FUNDAMENTO TEÓRICO	8
2.1. BRIAN . JS Y PORQUÉ USARLO?	8
2.2 APRENDIZAJE AUTOMÁTICO	9
2.3 SITUACIÓN ACTUAL CON QUE VAMOS A TRABAJAR Y EN DONDE	11
Capítulo 3 Metodología	15
3.1 GENERALIDADES Y DESARROLLO PRIMARIO	15
3.2 CREACIÓN DE INTERFAZ Y CONEXIÓN	23
Capítulo 4 Resultados	30
Anexos	32



INTRODUCCIÓN

El presente documento define la implementación de un módulo de IA para el tablero TO DO LIST de GINP, el cual ayudará a ubicar, definir e implementar las ideas o objetivos principales para definir una tarea en el tablero

JUSTIFICACIÓN

Dividamos en dos partes el proyecto la primera parte la cual ya está creada es un requerimiento de GINP del stc realizar un sistema tipo trello pero con la filosofía de software libre fue una tarea compleja, se decidió utilizar javascript y ajax así como mysql para los datos, y la segunda parte que es un módulo de IA tipo buscador chatbot para delimitar la importancia de las tareas dentro del sistema TODO LIST de manera general el proyecto tiene un grado de complejidad alto, ya que utiliza tecnologías complejas y el módulo de inteligencia artificial significa un gran reto complejo es decir, primero se tiene que realizar una metodología para el estudio del mismo desde manera particular a general, realizar un análisis teórico a la documentación, y finalmente la implementación, por medio de algunas técnicas de estudio específicas con varias pruebas tanto de funcionalidad como de rendimiento, y así mismo llegar a la conclusión de una BETA Funcional, en cuanto a los motivos de la justificación misma que nos llevaron a la idea de realizar la selección de estas tecnologías las explicaremos a continuación:

1). Necesidad de no usar un servidor

Ya que las mayorías de las tecnologías para IA utilizan un servidor dedicado para su funcionamiento como por ejemplo EXPRESS para el uso de JAVASCRIPT O TYPESCRIPT se planteó una problemática muy grande y claro un reto, así que realizamos pruebas en las tecnologías TENSOR FLOW, BRAIN JS, y solo obtuvimos resultados exitosos en BRAIN.JS aparte de que ya conocíamos NODE js el cual es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y finalmente concluimos que brain.js era lo mejor para nuestro caso pero se planteaba un reto mayor no ocupar un servidor web, así que hicimos pruebas y logramos adecuar la tecnología para que funcionara por medio de vectores o arreglos pero la parte del lenguaje o motor de construcción usarla desde un motor de base de datos y poder usar JS en un entorno menos robusto,

2) Porque Brain Js y no Tensorflow a pesar de su mayor uso y compatibilidad

Tensor Flow fue desarrollado en 2011 en Google como su librería para aplicaciones de Machine Learning (ML) y Deep Learning(DL) en Google. Se convirtió en código abierto en 2015 bajo la licencia de Apache.

Está construido en C ++, lo que permite que el código se ejecute a un nivel muy bajo. Además tiene enlaces con varios lenguajes como Python, R y Java; lo que permite integrarlo en aplicaciones de estas tecnologías.

Convencionalmente, en JavaScript, ML/DL se implementa mediante el uso de una API que consume el modelo en el servidor. El cliente envía una solicitud a una API usando JavaScript para obtener resultados del servidor

Brain.js es una biblioteca acelerada por GPU de redes neuronales escrita en JavaScript para navegadores y Node.js. Es simple, rápido y fácil de usar. Proporciona múltiples implementaciones de redes neuronales, ya que se pueden entrenar diferentes redes neuronales para hacer bien diferentes cosas.

Brain.js no es la primera biblioteca de aprendizaje automático centrada en JavaScript. Que por otro lado a diferencia de tensor flow y la razón principal porque no la elegimos es porque brain js utiliza objetos JSON, lo que elimina la necesidad de que el desarrollador cree tensores y gestione la memoria. a diferencia qué tensor flow no realiza ni ocupa esas funciones así como el rendimiento de memoria y la complejidad del recurso primordial de licenciamiento y optimizacion de codigo

3) Porque una lista to do y un módulo de ia

La razón primordial de la lista fue un requerimiento de la gerencia de GINP , es un diseño que se basa totalmente en la filosofía de desarrollo de software libre y el módulo de ia se obtuvo después de una larga investigación pruebas ,ejemplos , lecturas y se concluyó que este modelo tipo buscador chatbot es lo mejor para este tipo de sistema to do list



OBJETIVO GENERAL

El propósito de este proyecto se divide en dos partes , 1 .- Implementación de un módulo de IA en el proyecto ya creado del tablero tipo trello para GIMP de STC , 2- Crear la tecnología lo suficientemente eficaz, capaz y robusta para generar un módulo que permita definir una IA que ayude a definir ideas, objetivos y plantear tareas dentro del tablero de GIMP, de una manera lógica y organizada. siempre tomando en cuenta el uso de tecnologías existentes como son : JAVASCRIPT PHP MYSQL

CAPÍTULO 1 GENERALIDADES DE LA EMPRESA

1.1 ANTECEDENTES HISTÓRICOS DE LA EMPRESA

El Metro de la Ciudad de México es un sistema de transporte público tipo tren metropolitano³ que sirve a extensas áreas de la Ciudad de México. Su operación y explotación está a cargo del organismo público descentralizado denominado Sistema de Transporte Colectivo (STC),⁴⁵ y su construcción, a cargo de la Secretaría de Obras y Servicios del Distrito Federal.⁶ Hasta el 12 de agosto de 2013 su construcción fue gestionada por el denominado Proyecto Metro del Distrito Federal, un organismo desconcentrado de la citada secretaría.⁷ Se conoce coloquialmente como "Metro", por la contracción del término tren metropolitano.

En el 2006 ocupó el tercer lugar a nivel mundial en captación de usuarios, al transportar a un promedio de 3.9 millones de pasajeros al día (en ocasiones superado por los metros de Nueva York, Moscú y Tokio). También en ese año obtuvo el quinto lugar a nivel mundial por la extensión de su red.⁸⁹¹⁰

El metro de la Ciudad de México cuenta con 12 líneas, cada una con un número, letra y color distintivo. El parque vehicular está formado por trenes de rodadura neumática en diez líneas, y trenes férreos en las líneas A y 12. La longitud total de la red es de 226.49 km. con 195 estaciones. El sistema tiene aproximadamente 3333 vagones con una capacidad estándar de 1530 personas por tren; 15 239 empleados directos; 1684.94 viajes anuales por pasajero; 859.59 millones de kWh en consumo de energía y 114.03 millones de kilómetros por vagón al año.¹¹ El metro cuenta con 115 estaciones construidas de forma subterránea; 54 estaciones de forma superficial y 26 estaciones de manera elevada¹² conformando un total de 184 estaciones en la que atraviesan la Ciudad de México y 11 parte del Estado de México



1.2 MISIÓN Y VISIÓN

MISIÓN

Proveer un servicio de transporte público masivo, seguro, confiable y tecnológicamente limpio. Con una tarifa accesible, que satisfaga las expectativas de calidad, accesibilidad, frecuencia y cobertura de los usuarios y se desempeñe con transparencia, equidad y eficiencia logrando niveles competitivos a nivel mundial.

VISIÓN

Lograr un servicio de transporte de excelencia, que coadyuve al logro de los objetivos de transporte sustentable en la Zona Metropolitana del Valle de México, con un alto grado de avance tecnológico nacional, con cultura, vocación industrial y de servicio a favor del interés general y el mejoramiento de la calidad de vida de los ciudadanos.

1.3 UBICACIÓN

Calz. Ignacio Zaragoza 614, 4 Árboles, Venustiano Carranza, 15730 Ciudad de México, CDMX



1.4 GIRO

Transportación

CAPÍTULO 2 FUNDAMENTO TEÓRICO

2.1. BRIAN . JS Y PORQUÉ USARLO?

Podemos definir el uso de Brian Js como motor principal del funcionamiento de nuestro proyecto dado que :

Brain es una biblioteca de JavaScript creada para redes neuronales fáciles y de alto nivel. Brain maneja casi toda la configuración por usted, lo que le permite preocuparse solo por decisiones de alto nivel.

Función de escala : establece la función para determinar el valor de activación de las neuronas.

Número de capas ocultas : el número de capas adicionales entre las capas de entrada y salida. Casi no hay razón para usar más de dos capas para cualquier proyecto. Aumentar el número de capas aumenta enormemente el tiempo de cálculo.

Iteraciones : la cantidad de veces que la red se ejecuta a través de los datos de entrenamiento antes de que se detenga.

Tasa de aprendizaje: Un escalar global de la cantidad de valores que se pueden modificar. Demasiado bajo, y tomará mucho tiempo converger a la respuesta. Demasiado alto y es posible que se pierda un mínimo local.

```
const network = new brain.NeuralNetwork({
  activation: 'sigmoid', //Sets the function for activation
  hiddenLayers: [2], //Sets the number of hidden layers
  iterations: 20000, //The number of runs before the neural net stops
  training
  learningRate: 0.4 //The multiplier for the backpropagation changes
})
```

Los parámetros anteriores se pasan a la clase Neural Network como un objeto. A continuación, la red se puede entrenar utilizando el método .train. Esto requiere datos de entrenamiento preparados. Los datos de muestra deben estructurarse como una matriz de objetos con valores de entrada y salida. Los valores de entrada y salida deben ser una matriz de números, estos corresponden a los valores de activación de las neuronas en la primera y última capa de la red, respectivamente. Es importante que la cantidad de elementos en las matrices de entrada y salida permanezca constante (internamente, no tienen que ser iguales entre sí) ya que esto determina la cantidad de nodos que existirán en las capas frontal y posterior de la red.


```
let trainingSample1 = {  
  input: [ 5.3, 6 , 1 , -4 ]  
  output: [ 0 , 1 ]  
}  
  
let trainingSample2 = {  
  input: [ 1 , -14 , 0.2 , 4.4 ]  
  output: [ 1 , 1 ]  
}  
  
trainingData.push( trainingSample1 )  
trainingData.push( trainingSample2 )  
network.train(trainingData)
```

Y ahora la red ha hecho todo lo posible para entrenarse a sí misma con las configuraciones y muestras elegidas. Ahora puede usar el comando `.run` para examinar el resultado de una muestra determinada. Y listo, su red podrá hacer aproximaciones basadas en cualquier entrada dada. Diría que es como magia si no hubieras leído 1000 palabras explicando cómo funciona.

```
let sample = [20, -3, -5, 13]  
let result = network.run(sample)
```

2.2 APRENDIZAJE AUTOMÁTICO

Hay un puñado de librerías en JavaScript con algoritmos de Machine Learning pre-diseñados, tales como Regresión Lineal, SVMs, Naive-Bayes, etcétera. He aquí algunos de ellos,

- **brain.js (Redes neuronales)**
- **Sináptico (Redes neuronales)**
- **Natural (Procesamiento del lenguaje natural)**
- **ConvNetJS (redes neuronales convolucionales)**
- **mljs (Un conjunto de sub bibliotecas con una variedad de funciones)**
- **Neatáptico (Redes neuronales)**
- **Webdnn (Deep Learning)**

Según Arthur Samuel, el aprendizaje automático proporciona a los ordenadores la capacidad de aprender sin estar programados explícitamente. En otras palabras, le da a la computadora la habilidad de aprender por sí misma y ejecutar las instrucciones correctas, sin que tú le digas instrucciones.

El resurgimiento del interés en el aprendizaje basado en máquina se debe a los mismos factores que han hecho la minería de datos y el análisis Bayesiano más populares que nunca. Cosas como los volúmenes y variedades crecientes de datos disponibles, procesamiento computacional más económico y poderoso, y almacenamiento de datos asequible.

Una definición relativamente general de aprendizaje dentro del contexto humano podría ser la siguiente: proceso a través del cual se adquieren o modifican habilidades, destrezas, conocimientos, conductas o valores como resultado del estudio, la experiencia, la instrucción, el razonamiento y la observación. De esta definición es importante hacer notar que el aprendizaje debe producirse a partir de la experiencia con el entorno, no se considera aprendizaje toda aquella habilidad o conocimiento que sean innatos en el individuo o que se adquieran como resultado del crecimiento natural de éste. Siguiendo un esquema similar, en el AA vamos a considerar aprendizaje a aquello que la máquina pueda aprender a partir de la experiencia, no a partir del reconocimiento de patrones programados a priori. Por tanto, una tarea central de cómo aplicar esta definición al contexto de la computación va a consistir en alimentar la experiencia de la máquina por medio de objetos con los que entrenarse (ejemplos) para, posteriormente, aplicar los patrones que haya reconocido sobre otros objetos distintos (en un sistema de recomendación de productos, un ejemplo sería un par particular cliente/producto, junto con la información acerca de la valoración que aquel haya hecho de éste).

Los algoritmos de aprendizaje supervisado son entrenados utilizando ejemplos etiquetados, como una entrada donde se conoce el resultado deseado. Por ejemplo, una pieza de equipo podría tener puntos de datos etiquetados como "F" (fallidos) o "R" (corridos). El algoritmo de aprendizaje recibe un conjunto de entradas junto con los resultados correctos correspondientes, y el algoritmo aprende comparando su resultado real con resultados correctos para encontrar errores. Luego modifica el modelo en consecuencia. A través de métodos como la clasificación, regresión, predicción y aumento de gradientes, el aprendizaje supervisado utiliza patrones para predecir los valores de la etiqueta en datos no etiquetados adicionales. El aprendizaje supervisado se utiliza comúnmente en aplicaciones donde datos históricos predicen eventos futuros probables. Por ejemplo, puede anticipar cuándo es probable que transacciones con tarjetas de crédito sean fraudulentas o qué cliente de una aseguradora tiene la probabilidad de iniciar un reclamo.

El aprendizaje no supervisado se utiliza contra datos que no tienen etiquetas históricas. No se da la "respuesta correcta" al sistema. El algoritmo debe descubrir lo que se muestra. El objetivo es explorar los datos y encontrar alguna estructura en su interior. El aprendizaje no supervisado funciona bien con datos de transacciones. Por ejemplo, puede identificar segmentos de clientes con atributos similares que después puedan ser tratados de manera semejante en campañas de marketing. O bien puede encontrar los atributos principales que separan los segmentos de clientes. Algunas técnicas populares incluyen mapas con organización automática, mapping del vecino más cercano, k-means clustering y descomposición de valores singulares. Estos algoritmos se pueden utilizar también para segmentar temas de texto, recomendar elementos e identificar valores atípicos de datos.

El aprendizaje semi supervisado se utiliza para las mismas aplicaciones que el aprendizaje supervisado. Sin embargo, utiliza datos etiquetados y no etiquetados para entrenamiento – por lo general una pequeña cantidad de datos etiquetados con una gran cantidad de datos no etiquetados (porque los datos no etiquetados son menos costosos y se requiere menos esfuerzo en su obtención). Este tipo de aprendizaje se puede utilizar con métodos como la clasificación,

regresión y predicción. El aprendizaje semi supervisado es de utilidad cuando el costo asociado con el etiquetado es demasiado alto para permitir un proceso de entrenamiento completamente etiquetado. Algunos ejemplos iniciales de este tipo de aprendizaje incluyen la identificación del rostro de una persona en una cámara Web.

El aprendizaje con refuerzo se utiliza a menudo para robótica, juegos y navegación. Con el aprendizaje con refuerzo, el algoritmo descubre a través de ensayo y error qué acciones producen las mayores recompensas. Este tipo de aprendizaje tiene tres componentes principales: el agente (el que aprende o toma decisiones), el entorno (todo con lo que interactúa el agente) y acciones (lo que el agente puede hacer). El objetivo es que el agente elija acciones que maximicen la recompensa esperada en cierta cantidad de tiempo. El agente logrará la meta mucho más rápido si aplica una buena política. De modo que el objetivo en el aprendizaje con refuerzo es aprender la mejor política.

2.3 SITUACIÓN ACTUAL CON QUE VAMOS A TRABAJAR Y EN DONDE

Hasta ahora solamente se tiene el tablero tipo trello el cual está desarrollado en JavaScript ,PHP, HTML y MySQL , y ayuda a definir de manera estructurada algunas tareas , como lo menciona el objetivo , es crear un módulo que ayude a definir y crear ideas lo suficientemente estructuradas y organizadas para definir una tarea en este tablero, finalmente se tiene que generar una adaptación a estas tecnologías y se pueda usar de la mejor manera lo más correcta y organizada posible

2020



GOBIERNO DE LA
CIUDAD DE MÉXICO

GERENCIA DE INGENIERÍA Y NUEVOS PROYECTOS



LISTA DE PROYECTOS

Bienvenido : GERENCIA [\[SALIR\]](#)

[CREAR UN PROYECTO](#)

NOMBRE PROYECTO/ACTIVIDAD	TIPO DE PROYECTO O ID	TAREAS RELACIONADAS ACTIVAS	ACCIÓN
IMPRESION 3D	UNIC	6	IR A SU TABLERO
IMPLEMENTACION 5S	RE16	2	IR A SU TABLERO
SALUDAR	UN12	0	IR A SU TABLERO
1 EJEMPLO	1	0	IR A SU TABLERO

2020



GOBIERNO DE LA
CIUDAD DE MÉXICO

GERENCIA DE INGENIERÍA Y NUEVOS PROYECTOS



NUEVO PROYECTO

MENSAJE IMPORTANTE

NOMBRE DEL PROYECTO
NUEVO

TIPO DE PROYECTO / ID

AÑADIR PROYECTO

2020



GOBIERNO DE LA
CIUDAD DE MÉXICO

GERENCIA DE INGENIERÍA Y NUEVOS PROYECTOS



LISTA DE TAREAS

PROYECTO ACTUAL: IMPRESION 3D


CREAR UNA TAREA

[<<- REGRESAR A PROYECTOS](#)

CARDS CON NOMBRE DE LA TAREA Y CÓDIGO , ID DEL PROYECTO ASI COMO ACCIÓN DE MOVIMIENTO


RECURSOS	HACER	HACIENDO	HECHO/TERMINADO
444 ACCIÓN UNIC-9	COMER ACCIÓN UNIC-5	3 ACCIÓN UNIC-3	1 ACCIÓN UNIC-1
GOOGLE ACCIÓN UNIC-10	11111111111111 ACCIÓN UNIC-7	HABALT ACCIÓN UNIC-6	2 ACCIÓN UNIC-2
			4 ACCIÓN UNIC-4

2020



GOBIERNO DE LA
CIUDAD DE MÉXICO

GERENCIA DE INGENIERÍA Y NUEVOS PROYECTOS





NUEVA TAREA (UNIC)

TITULO:



DESCRIPCION:

2020



GOBIERNO DE LA
CIUDAD DE MÉXICO

GERENCIA DE INGENIERÍA Y NUEVOS PROYECTOS



UNIC-11

[<--- REGRESAR AL TABLERO](#)

EJEMPLO

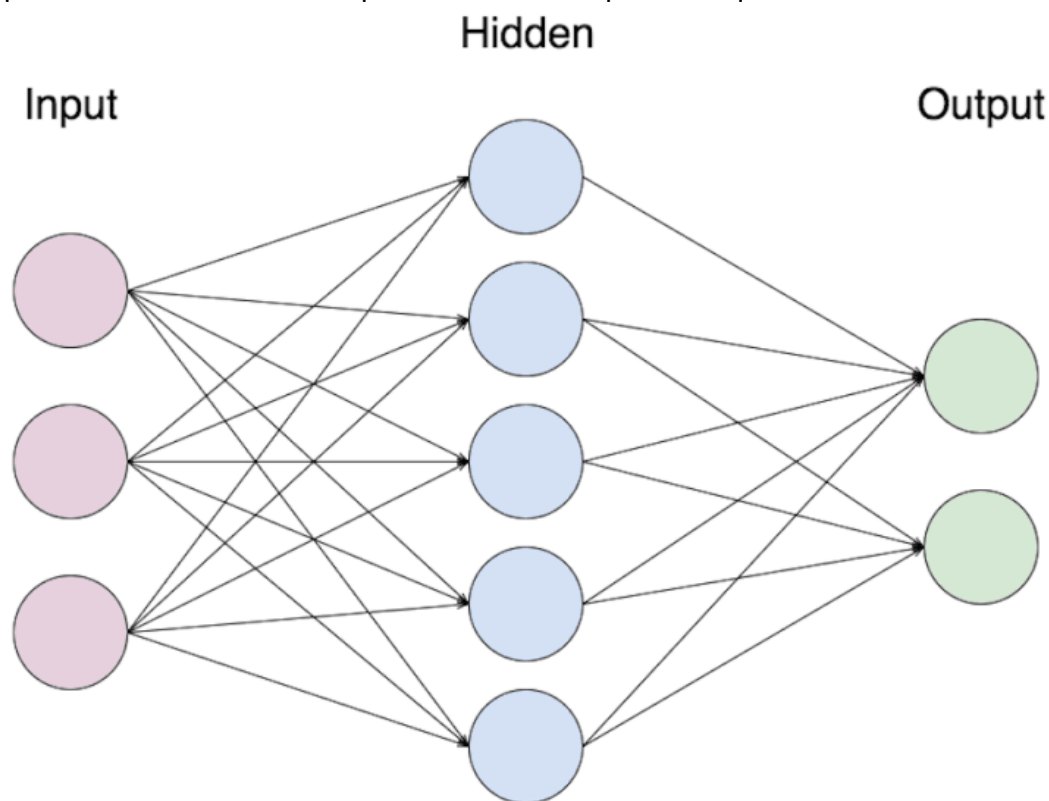
EJEMPLO

Capítulo 3 Metodología

3.1 GENERALIDADES Y DESARROLLO PRIMARIO

Principalmente partimos de la idea de una red neuronal a continuación definiremos que es :

Las redes neuronales son estructuras informáticas increíblemente útiles que permiten a las computadoras procesar entradas complejas y aprender a clasificarlas. La funcionalidad de una red neuronal proviene de su estructura, que se basa en los patrones que se encuentran en el cerebro.



Las redes neuronales, también conocidas como redes neuronales artificiales (ANN) o redes neuronales simuladas (SNN), son un subconjunto del aprendizaje automático y están en el corazón de los algoritmos de aprendizaje profundo . Su nombre y estructura están inspirados en el cerebro humano, imitando la forma en que las neuronas biológicas se transmiten entre sí.

Las redes neuronales artificiales (ANN) se componen de capas de un nodo, que contiene una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo, o neurona artificial, se conecta a otro y tiene un peso y un umbral asociados. Si la salida de cualquier nodo individual está por encima del valor de umbral especificado, ese nodo se activa y envía datos a la siguiente capa de la red. De lo contrario, no se transmiten datos a la siguiente capa de la red.

Teniendo en cuenta lo anterior como podemos generar algo capaz de emular una red neuronal seguir una serie de pasos , en base a un modelo y que a medida de cada iteración como en una red pueda aprender y crear o replicar un cuestionamiento , a lo largo de mucha búsqueda buscamos a :

brain.js la cual es una biblioteca acelerada por GPU para redes neuronales escrita en JavaScript, capaz de emular y entender estos cuestionamientos en base a la documentación se encontró un ejemplo de un identificador de letras el cual es el siguiente

```
const a = character(
  '#####.' +
  '#.....#' +
  '#.....#' +
  '#####' +
  '#.....#' +
  '#.....#' +
  '#.....#'
);
const b = character(
  '#####.' +
  '#.....#' +
  '#.....#' +
  '#####.' +
  '#.....#' +
  '#.....#' +
  '#####.'
);
const c = character(
  '#####' +
  '#.....' +
  '#.....' +
  '#.....' +
  '#.....' +
  '#.....' +
  '#####'
);
```

```
/**
 * Learn the letters A through C.
 */
const net = new brain.NeuralNetwork();
net.train([
  {
    input: a,
    output: {
      a: 1
    }
  },
  {
    input: b,
    output: {
      b: 1
    }
  },
  {
    input: c,
    output: {
      c: 1
    }
  }
], {
  log: detail => console.log(detail)
});

/**
 * Predict the letter A, even with a pixel off.
 */
const result = brain.likely(character(
  '#####' +
  '#.....#' +
  '#.....#' +
  '###.###' +
  '#.....#' +
  '#.....#' +
  '#.....#'
```

```
, net);
console.log(result); // 'a'
/**
 * Turn the # into 1s and . into 0s. for whole string
 * @param string
 * @returns {Array}
 */
function character(string) {
  return string
    .trim()
    .split('')
    .map(integer);
}
/**
 * Return 0 or 1 for '#'
 * @param character
 * @returns {number}
 */
function integer(character) {
  if ('#' === character) return 1;
  return 0;
}
```

y como salida :

```
}
[object Object] {
  error: 0.00940861254376458,
  iterations: 120
}
[object Object] {
  error: 0.007960980481362697,
  iterations: 130
}
[object Object] {
  error: 0.0068612688800259595,
  iterations: 140
}
[object Object] {
  error: 0.006003298390086114,
  iterations: 150
}
[object Object] {
  error: 0.005318737495408648,
  iterations: 160
}
"a"
```

En IA existen diferentes formas de aplicarla una de ellas es la asociación la cual se define como :

El aprendizaje con reglas de asociación lo vemos aplicado principalmente en los sistemas de recomendación, como en el caso donde se nos muestra que las personas que compraron este producto, también compraron este otro .. o quienes vieron tal película también recomiendan estas otras, etc.

Para ello, el algoritmo a priori es uno de los más utilizados en este tema y permite encontrar de forma eficiente conjuntos de ítems frecuentes, los cuales sirven de base para generar reglas de asociación entre los ítems.

Primero identifica los ítems individuales frecuentes dentro del conjunto de datos para luego extenderlo a un conjunto de mayor tamaño siempre y cuando esos conjuntos de datos aparezcan constantemente y de manera frecuente de acuerdo con un umbral establecido.

El algoritmo se aplica principalmente en el análisis de transacciones comerciales y en los problemas de predicción. Es por ello que el algoritmo está diseñado para trabajar con bases de datos que contienen transacciones como los productos o artículos comprados por consumidores, o detalles sobre las visitas a un sitio web, etc.

La forma de generar las reglas de asociación consta de dos pasos:

Generación de combinaciones frecuentes: cuyo objetivo es encontrar aquellos conjuntos que sean frecuentes en la base de datos. Para determinar la frecuencia se establece un umbral.

Generación de reglas: A partir de los conjuntos frecuentes se crean las reglas en base al ordenamiento de un índice que establece los grupos de ítems o productos frecuentes.

Definimos que un sistema tipo chatbot funciona bajo el siguiente esquema o algoritmo

El índice para la generación de combinaciones se llama soporte y el índice para la generación de reglas se llama confianza.

Algoritmo

Paso 1. Se establecen los valores mínimos para el soporte y la confianza

Paso 2. Se toman todos los subconjuntos de transacciones que tienen un soporte mayor al valor del soporte mínimo.

Paso 3. Tomar todas las reglas de estos subconjuntos que tengan una confianza mayor al valor de la confianza mínima.

Paso 4. Ordenar las reglas de forma decreciente en base al valor

Como ya sabemos se trata de una implementación por eso mismo se tiene que trabajar bajo las tecnologías existentes que son javascript y php

Existen muchas tecnologías que pueden hacer esto pero elegimos BRAIN JS por las siguientes razones . aparte de las mencionadas anteriormente en la situación actual :

1. Seguridad

A diferencia de Python, JavaScript se creó para la seguridad. Después de un lo, que fue diseñado para permitir que alguien que no confía, como Amazon o Google, para poder ejecutar scripts en su ordenador sin acceso a todos sus archivos y secretos. Por diseño, JavaScript no puede acceder a archivos o incluso al sistema operativo. Esto lo convierte en un excelente marco de implementación para la IA.

2. Desempeño

Se ha realizado un gran esfuerzo para que JavaScript se ejecute realmente rápido. Y me refiero a mucho. Uno de los efectos secundarios del panorama competitivo de los navegadores ha sido una fuerte evaluación comparativa, con el rendimiento de JavaScript como una métrica clave en esa competencia. Esto significa que empresas como Apple, Google y Microsoft han invertido muchos millones para hacer que JavaScript se ejecute de forma rápida y pequeña en sus respectivos navegadores. Atrás quedaron los días en que JavaScript era un lenguaje interpretado: el JavaScript moderno se traduce a código de máquina de la misma manera que Java, y en las evaluaciones de rendimiento pueden ser muy competitivos, especialmente para la manipulación de texto.

3. Tiempo de desarrollo

JavaScript es muy rápido de desarrollar, de la misma manera que Python, pero quizás incluso más. Ambos proporcionan el bucle interactivo que hace que la depuración sea agradable y fácil, ambos tienen excelentes marcos de desarrollo. Hay un gran soporte de linting para ambos, las herramientas de desarrollo estándar como VSCode los soportan bien. No soy un experto en Python, pero mi experiencia ha sido que JavaScript es, en todo caso, más rápido de desarrollar que Perl y Python.

Donde Python tiene una ventaja son las bibliotecas, scikit-learn, etc. No hay nada en el mundo de JavaScript que abra el nivel de rendimiento de la GPU para el aprendizaje profundo. Hay bibliotecas decentes (como Synaptic) que pueden entrenar bien redes neuronales moderadas. Synaptic tiene una característica deliciosa en la que puede tomar una red neuronal entrenada y exportarla como una función JavaScript sin procesar como código fuente.

Podría haber sido más negativo sobre JavaScript antes de que ES6 tuviera un soporte tan amplio. Lo admito, solía usar CoffeeScript, porque sus funciones y clases de flecha hicieron que el código fuera mucho más legible; ahora son estándar en ES6, por lo que el 90% de los beneficios de CoffeeScript son estándar en todas las plataformas JavaScript.

Con todo, JavaScript tiene las características de un gran lenguaje de IA:

- Está construido para la seguridad
- Es rápido de correr
- Es rápido de desarrollar
- Y tiene una gran reserva de talento asequible.

Brain.js es una biblioteca Javascript para redes neuronales que reemplaza a la biblioteca " cerebro " (ahora en desuso) , que se puede usar con Node.js o en el navegador (cálculo de nota) y proporciona diferentes tipos de redes para diferentes tareas.

Brain.js es excelente para crear rápidamente una NN simple en un lenguaje de alto nivel donde puede aprovechar la gran cantidad de bibliotecas de código abierto. Con un buen conjunto de datos y algunas líneas de código, puede crear algunas funciones realmente interesantes.

Las bibliotecas altamente científicas / computacionales escritas en JavaScript como esta tienden a ser muy criticadas , Brain.js tiene su lugar en JS siempre que tenga las expectativas y la aplicación adecuadas. Por ejemplo, JS es el lenguaje dominante (¿único?) Que se ejecuta en el lado del cliente en el navegador, así que ¿por qué no aprovechar esta biblioteca para cosas como juegos en el navegador, colocación de anuncios o reconocimiento de personajes? o hasta nuestro sistema

Desventajas

Si bien definitivamente podemos obtener algo de valor de una biblioteca como esta, no es perfecta. Como mencioné, la biblioteca limita la arquitectura de su red a un punto en el que solo puede hacer aplicaciones simples. No hay muchas posibilidades para capas Softmax u otra estructura. Sería bueno tener al menos la opción de personalizar más la arquitectura en lugar de ocultarle todo.

Finalmente

Brain.js . Brain.js hace un gran trabajo simplificando el proceso de creación y entrenamiento de una NN utilizando la facilidad de uso de JavaScript y limitando la API a unas pocas llamadas y opciones de métodos por eso no se decidió usar a **TENSOR FLOW**, aparte de que **Brain js es más fácil de usar y tiene un mejor desempeño.**

Como primer paso se realizaron pruebas

```
const brain = require('brain.js');
const trainingData = [
  'I am the best person',
  'Generar archivo para el area',
  'Gerencia',
  'Realizar la tarea',
  'Mandar informe',
  'Solicitar informacion',
  'CDT'
]

const network = new brain.recurrent.LSTM();

network.train(trainingData, {
  iterations: 100,
  log: stats => {
    console.log(stats)
  }
})

const queryString = 'Generar';

console.log(queryString + network.run(queryString));
```

Como podemos ver anteriormente se creó un modelo y este modelo genera un arreglo que contiene las posibles coincidencias , a su vez se genera una función que relaciona una parte del arreglo con una cadena de palabra , si ponemos la palabra generar como posible búsqueda , el sistema deberá arrojar generar informe para el área como posible coincidencia

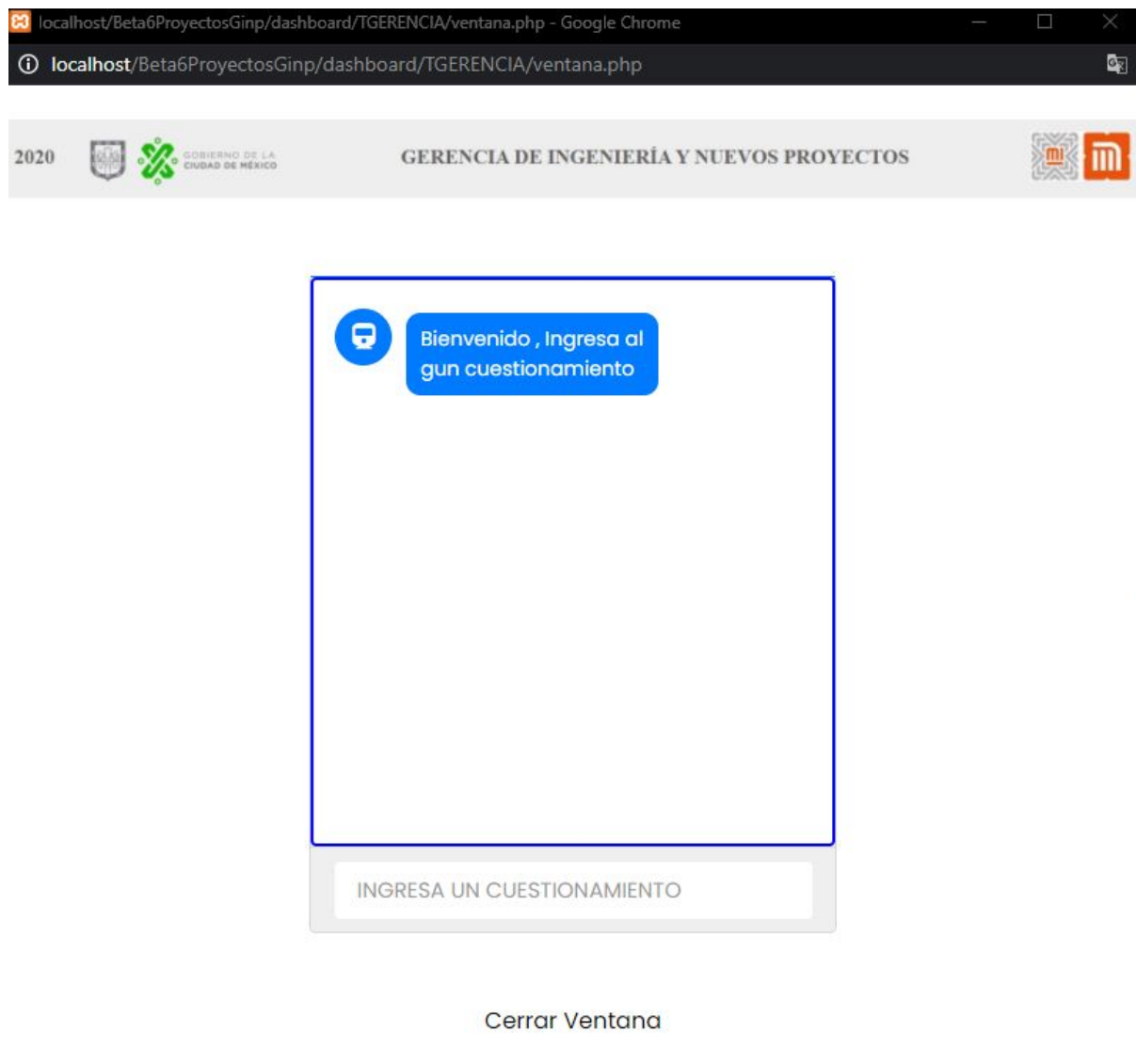
3.2 CREACIÓN DE INTERFAZ Y CONEXIÓN

Como se mencionaba en el objetivo dado que esta es una adaptación o implementación de un módulo de ia se tiene que usar php ,javascript etc para su correcto funcionamiento
Como primer paso se desarrolló el botón que permite abrir una nueva ventana



RECURSOS	HACER	HACIENDO	HECHO/TERMINADO
444 ACCIÓN UNIC-9	COMER ACCIÓN UNIC-5	3 ACCIÓN UNIC-3	1 ACCIÓN UNIC-1
GOOGLE ACCIÓN UNIC-10	11111111111111 ACCIÓN UNIC-7	HABALT ACCIÓN UNIC-6	2 ACCIÓN UNIC-2
			4 ACCIÓN UNIC-4

Ahora creamos el archivo y la interfaz de ese botón mediante css y html



Podemos ver eso en este código :

PRIMERO IMPORTAMOS BRAIN JS Y EL CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <link rel="stylesheet" href="style.css">
  <script src="https://kit.fontawesome.com/a076d05399.js"></script>
  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <script const brain = require('brain.js'); ></script>
</head>
<body>
```

LUEGO CREAMOS LA CAJA Y LOS BOTONES ASI COMO EL MENSAJE

```
<div class="wrapper">
  <!-- <div class="title">Simple Online Chatbot</div-->
  <div class="form">
    <div class="bot-inbox inbox">
      <div class="icon">
        <i class="fas fa-train"></i>
      </div>
      <div class="msg-header">
        <p>Bienvenido , Ingresa algun cuestionamiento</p>
      </div>
    </div>
  </div>
  <div class="typing-field">
    <div class="input-data">
      <input id="data" type="text" placeholder="INGRESA UN CUESTIONAMIENTO " required>
      <button id="send-btn">ENVIAR</button>
    </div>
  </div>
```

```
</div>
</div>

<script>

$(document).ready(function() {
    $("#send-btn").on("click", function() {
        $value = $("#data").val();

        $msg = '<div class="user-inbox inbox"><div
class="msg-header"><p>'+ $value +'</p></div></div>';

        $(".form").append($msg);
        $("#data").val('');

        // AJAX para el motor de vista del mensaje de php
        $.ajax({
            url: 'message.php',
            type: 'POST',
            data: 'text='+$value,
            success: function(result) {
                $replay = '<div class="bot-inbox inbox"><div
class="icon"><i class="fas fa-train"></i></div><div class="msg-header"><p>'+
result +'</p></div></div>';

                $(".form").append($replay);
                // cuando escribes se activa el boton
                $(".form").scrollTop($(".form")[0].scrollHeight);
            }
        });
    });
});

$( document ).ready( function() {
    $("a[rel='pop-up']").click(function () {

        var      características =
"height=700,width=800,scrollTo,resizable=1,scrollbars=1,location=0";
        nueva=window.open(this.href, 'Popup', características);
        return false;
    });
});
```

```
</script>
<a href="ventana.php" rel="pop-up">Abrir </a>

</body>
</html>
```

AHORA GENERAMOS LA FUNCIÓN QUE MANDARA EL MENSAJE DE BIENVENIDA Y OBTENDRÁ LOS DATOS

```
// AJAX para el motor de vista del mensaje de php
$.ajax({
  url: 'message.php',
  type: 'POST',
  data: 'text='+$value,
  success: function(result){
    $replay = '<div class="bot-inbox inbox"><div
class="icon"><i class="fas fa-train"></i></div><div class="msg-header"><p>'+
result + '</p></div></div>';
    $(".form").append($replay);
    // cuando escribes se activa el boton
    $(".form").scrollTop($(".form")[0].scrollHeight);
  }
});
});
});
```


Generamos el arreglo que manda la información pertinente y genera la comparación con un modelo en este caso se eligió la generación de una tabla en MYSQL para que no consuma muchos recursos

```
<?php
// connecting to database
$conn = mysqli_connect("localhost", "root", "", "bot") or die("Database
Error");

// getting user message through ajax
$getMesg = mysqli_real_escape_string($conn, $_POST['text']);

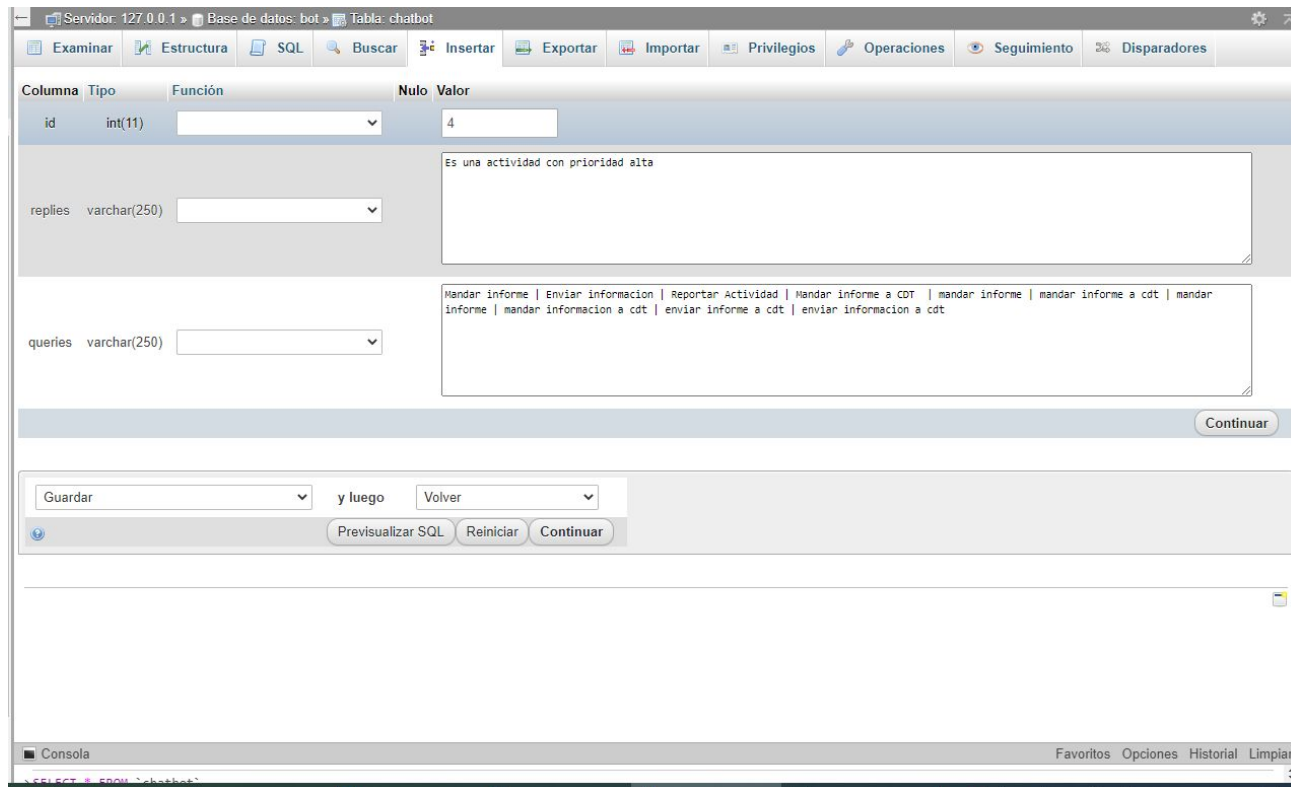
//checking user query to database query
$check_data = "SELECT replies FROM chatbot WHERE queries LIKE '%$getMesg%'";
$run_query = mysqli_query($conn, $check_data) or die("Error");

// if user query matched to database query we'll show the reply otherwise it
go to else statement
if(mysqli_num_rows($run_query) > 0){
    //fetching replay from the database according to the user query
    $fetch_data = mysqli_fetch_assoc($run_query);
    //storing replay to a variable which we'll send to ajax
    $replay = $fetch_data['replies'];
    echo $replay;
}else{
    echo "Disculpe el sistema aun no procesa el cuestinamiento ingresado
corrijalo y ingreselo de nuevo ";
}

?>
```

Como se puede ver en el texto anterior solamente es un arreglo que hace la comparación con el modelo en base de datos y genera una cadena de palabras , al completar o hacer las iteraciones correspondientes se genera la cadena final y manda un mensaje o salda con la coincidencia encontrada

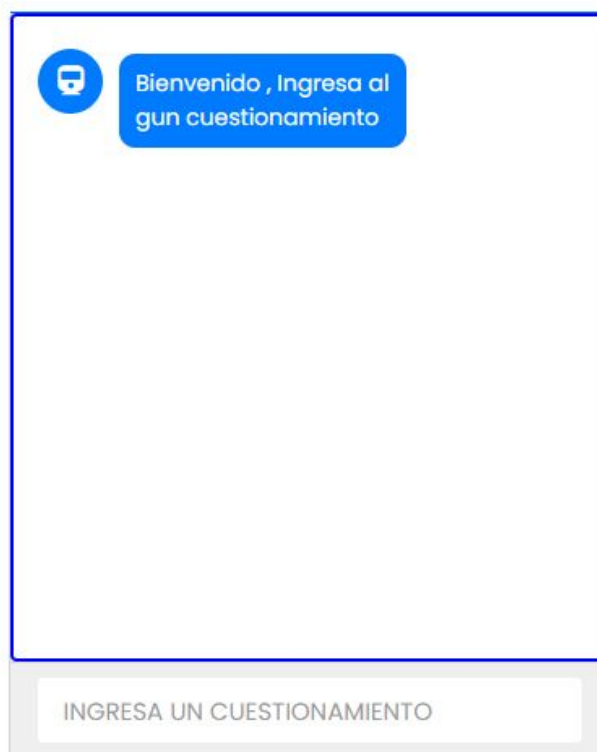
Como se puede observar en la siguiente pantalla



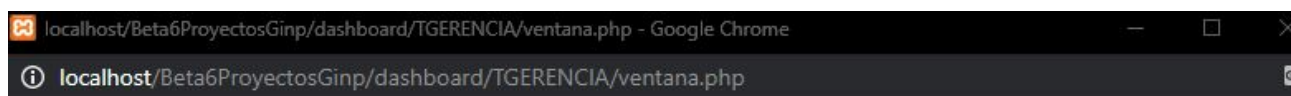
FUNCIONAMIENTO GENERAL

Como se trato anteriormente existen dos partes del código , la ventana que emerge y muestra una caja de texto para ingresar un cuestionamiento y el trasfondo que es la parte principal , en el momento que un usuario ingresa un cuestionamiento se genera una llamada y esa llamada se ingresa dentro de un arreglo , ese arreglo hace una comparación con la base de datos o modelo y compara cada palabra y si se tiene alguna coincidencia va a completar la cadena de texto y mandar el mensaje completo , en dado caso que no exista ninguna coincidencia mandará un mensaje de que no existe como tal respuesta a ese cuestionamiento o un modelo que la pueda interpretar así funciona una red neuronal

Capítulo 4 Resultados



Cerrar Ventana





Es una actividad con prioridad alta



Es una actividad con prioridad alta



Disculpe el sistema a un no procesa el cuestionamiento ingresado corrija y ingreselo de nuevo

mandar

hola

INGRESA UN CUESTIONAMIENTO

Cerrar Ventana



Anexos



