



MANUAL DE PRÁCTICAS PARA

INTELIGENCIA ARTIFICIAL



INDICE

| | |
|---|----|
| TITULO DE LA PRÁCTICA:..... | 5 |
| IDENTIFICADOR DE LA PRÁCTICA | 5 |
| PRESENTACIÓN..... | 5 |
| FUNDAMENTACION TEÓRICA | 6 |
| ¿Qué es el procesamiento del lenguaje natural?..... | 6 |
| Componentes del procesamiento del lenguaje natural..... | 7 |
| Niveles y bloques en la PLN..... | 7 |
| Ejemplos del PLN | 9 |
| DESCRIPCIÓN DE LA PRÁCTICA..... | 11 |
| Objetivo:..... | 11 |
| Competencias desarrolladas: | 11 |
| Requisitos de software:..... | 11 |
| Requisitos de hardware: | 11 |
| Desarrollo del código | 11 |
| Resultados | 15 |
| Diagramas del proyecto | 16 |
| Recomendaciones | 17 |
| TITULO DE LA PRÁCTICA..... | 19 |
| IDENTIFICADOR DE LA PRÁCTICA | 19 |
| PRESENTACIÓN..... | 19 |
| FUNDAMENTACIÓN TEÓRICA | 20 |
| Evolución de las RN | 20 |
| Funcionamiento de las redes neuronales | 21 |
| Tipos de redes neuronales | 21 |
| Arquitectura de una red neuronal | 22 |
| Aplicaciones de las redes neuronales | 23 |
| DESCRIPCIÓN DE LA PRÁCTICA..... | 24 |
| Objetivo:..... | 24 |
| Competencias Desarrolladas..... | 24 |
| Requisitos de Software | 24 |
| Requisitos de Hardware | 24 |
| Diseño de la red neuronal | 25 |

| | |
|--|----|
| Desarrollo del código | 30 |
| Resultados | 33 |
| Diagramas del proyecto | 34 |
| RECOMENDACIONES | 35 |
| | 35 |
| | 35 |
| TITULO DE LA PRÁCTICA..... | 36 |
| IDENTIFICADOR DE LA PRÁCTICA | 36 |
| PRESENTACIÓN..... | 36 |
| FUNDAMENTACION TEÓRICA | 37 |
| Origen de la visión artificial..... | 37 |
| Beneficios como parte de la Inteligencia Artificial..... | 38 |
| Aplicaciones de la visión artificial..... | 38 |
| DESCRIPCIÓN DE LA PRÁCTICA..... | 40 |
| Objetivo | 40 |
| Competencias desarrolladas | 40 |
| Requisitos de Software | 40 |
| Requisitos de Hardware | 40 |
| Desarrollo del código | 40 |
| Resultados | 43 |
| Recomendaciones | 44 |
| TITULO DE LA PRÁCTICA..... | 45 |
| IDENTIFICADOR DE LA PRÁCTICA | 45 |
| PRESENTACIÓN..... | 45 |
| FUNDAMENTACIÓN TEÓRICA | 46 |
| Como funciona | 46 |
| Elementos de la lógica difusa | 47 |
| Aplicaciones de la lógica difusa | 47 |
| DESCRIPCION DE LA PRÁCTICA..... | 49 |
| Objetivo | 49 |
| Competencias Desarrolladas..... | 49 |
| Requisitos de Software | 49 |
| Requisitos de Hardware | 49 |

| | |
|---------------------------------|----|
| Sistema de control difuso | 49 |
| Resultados | 54 |
| RECOMENDACIONES | 57 |
| Bibliografías | 58 |

INDICE DE ILUSTRACIONES

| | |
|--|----|
| Ilustración 1 Diagrama de proceso de flujo del dialogo del chatbot Albot | 16 |
| Ilustración 2 Comparación temperatura Celsius-Kelvin | 25 |
| Ilustración 3 Formula de conversión Celsius-Kelvin | 25 |
| Ilustración 4 Capa de entrada de una RN | 26 |
| Ilustración 5 Capa oculta de una RN | 27 |
| Ilustración 6 Capa de salida de una RN | 28 |
| Ilustración 7 Representación de la RN Gkelvin | 29 |
| Ilustración 8 Velocidades de la caja manual | 50 |
| Ilustración 9 Elementos del universo | 52 |

TITULO DE LA PRÁCTICA:

MANUAL PROCEDIMENTAL PARA EL DESARROLLO DE LA PRÁCTICA
“CHATBOT FUNCIONAL CON PYTHON”

IDENTIFICADOR DE LA PRÁCTICA

El identificador de la práctica es una clave formulada para hacer referencia exacta a la práctica que nos estamos refiriendo, esta sirve también para ser identificada con mayor facilidad en el repositorio digital donde se almacena.

Clave:

Practica-IA-01

PRESENTACIÓN

Cuando hablamos de inteligencia artificial en la mayoría de los casos la primera aplicación de esta en la que pensamos es en los asistentes de voz, los chatbot o simplemente los bot y es que son uno de los ejemplos más representativos de la inteligencia artificial. El procesamiento del lenguaje natural es una de las tantas técnicas que se aplican en el área de investigación y desarrollo de la inteligencia artificial entre sus aplicaciones más significativas encontramos los chatbot, los asistentes de voz, aplicaciones con reconocimiento de voz y traductores cada uno de estos ejemplos emplea un nivel distinto de PLN siendo los más básicos los traductores y los más complejos los asistentes de voz.

El PLN está presente en la mayoría de los dispositivos tal es el caso de los portátiles los teléfonos inteligentes y de los dispositivos de asistencia inteligente para el hogar a modo de ejemplo y con fines educativos se ha seleccionado como practica la creación de un bot sencillo pero que cubra el tema de procesamiento de lenguaje natural.

En el presente manual de procedimientos se documenta el desarrollo de la práctica con identificador Pactica-IA-01 así como el análisis y la lógica fundamentada en el código para que sea de fácil comprensión además se incluyen recomendaciones y consideraciones para futuras replicas en prácticas similares.

FUNDAMENTACION TEÓRICA

El uso de los chatbot es cada vez más común en los comercios, empresas y en áreas donde la atención al cliente debe agilizarse este es también uno de los usos más relevantes del procesamiento del lenguaje natural. Actualmente el procesamiento de lenguaje natural (PLN) consiste en transformar el lenguaje natural en un lenguaje formal como el de programación que los ordenadores puedan procesar por lo que si nos vamos a una definición más técnica IBM la define como la rama de la informática más concretamente, de la inteligencia artificial que se ocupa de dotar a los ordenadores de la capacidad de entender lenguaje hablado y escrito del mismo modo que los seres humanos.

¿Qué es el procesamiento del lenguaje natural?

El procesamiento de lenguaje natural es la rama de la ciencia informática especializada en las interacciones entre humanos y ordenadores a través del lenguaje hablado este tiene sus raíces en la década de 1950, cuando Alan Turing publicó un artículo (Máquinas computacionales e inteligencia) en el que proponía lo que hoy se conoce como el Test de Turing. La prueba examinaba la capacidad de una máquina para exhibir un comportamiento inteligente similar al de un ser humano. través del PLN se consigue combinar la lingüística computacional (modelado del lenguaje humano basado en reglas) y el análisis estadístico, basado en los modelos de aprendizaje automático y profundo. Estos modelos estadísticos emplean supuestos estadísticos que proporcionan una aproximación más precisa al verdadero significado, intención y sentimiento de quien habla o escribe.

El aprendizaje automático se basa en dichos datos para poder hacer predicciones. En ausencia de datos, es imposible que un sistema de inteligencia artificial sea capaz de aprender. Así pues, es necesario un corpus de texto o de lenguaje hablado para entrenar el algoritmo de PLN. El Procesamiento de Lenguaje Natural no es una técnica independiente, sino que se apoya en otros procesos como la Compresión del Lenguaje Natural (CLN) y la Generación del Lenguaje Natural (GLN).

la Comprensión del Lenguaje Natural (CLN) está orientada a la interpretación y la comprensión del lenguaje. Para ello tiene en cuenta la gramática y el contexto, de tal modo que la intención del hablante quede clara. Para conseguirlo, la CLN emplea algoritmos de inteligencia artificial. Estos algoritmos son capaces de realizar análisis estadísticos y, con posterioridad, identificar similitudes en textos que no han sido analizados. Con la CLN las aplicaciones informáticas tienen la posibilidad de deducir cuál es el significado del lenguaje escrito o hablado, incluso cuando estos presentan algún defecto. En esencia, el PLN se fija en lo que se dice (contenido en inglés) mientras que la CLN presta más atención a lo que se quiere decir.

El PLN la Generación de Lenguaje Natural es la generación de texto a partir de datos estructurados. Esto implica que el mismo proceso se puede realizar en sentido inverso, poniendo en el punto de mira la comunicación con los ordenadores y no con los humanos. Por ejemplo, el PNL puede crear informes de contenido e indicar

cuáles deben abordarse al tratar un tema determinado. Esta clasificación puede hacerse incluso en diferentes niveles de especialización y resultar de utilidad en las distintas fases del embudo de ventas.

Componentes del procesamiento del lenguaje natural

Análisis morfológico o léxico. Consiste en el análisis interno de las palabras que forman oraciones para extraer lemas, rasgos flexivos, unidades léxicas compuestas. Es esencial para la información básica: categoría sintáctica y significado léxico.

Análisis sintáctico. Consiste en el análisis de la estructura de las oraciones de acuerdo con el modelo gramatical empleado (lógico o estadístico).

Análisis semántico. Proporciona la interpretación de las oraciones, una vez eliminadas las ambigüedades morfosintácticas.

Análisis pragmático. Incorpora el análisis del contexto de uso a la interpretación final. Aquí se incluye el tratamiento del lenguaje figurado (metáfora e ironía) como el conocimiento del mundo específico necesario para entender un texto especializado.

Niveles y bloques en la PLN

El primer nivel lo constituye lo que denominamos la PNL básica. La PNL se constituye en 6 bloques temáticos y experienciales (ya que la PNL se aprende de forma experiencial)

- VAC: descubrir los elementos básicos de la experiencia humana y poder operar con ellos.
- Creencias básicas de la PNL: las presuposiciones básicas de la PNL son creencias útiles para una vida más feliz.
- Definición de objetivos: el paso de deseo a objetivo alcanzable. Filtros de definición de un objetivo y cómo facilitar el logro.
- Empatía: el valor de la empatía en las relaciones y cómo conseguir establecerla y sostenerla (acompañamiento verbal y no verbal)
- Calibración: aprender a estar en la experiencia con todos los canales perceptivos abiertos.
- Metamodelo del lenguaje: cómo obtener información de calidad sobre la experiencia aprendiendo a preguntar para
- Especificar con precisión las declaraciones de una persona
- Facilitar el cambio reconectando lo que decimos (lenguaje) con la experiencia (vivencia sensorial)

El segundo nivel descansa sobre el primero. Una vez asentadas las bases de la intervención con PNL (primer nivel), podremos elegir cómo continuar trabajando contando con diferentes opciones. Las técnicas del segundo nivel que constituyen la PNL específica nos dan una gran riqueza de opciones. Dependiendo del objetivo

o la intención de la intervención, podremos elegir cuál o cuáles de ellas aplicar. 6 nuevos bloques temáticos y experienciales constituyen la PNL específica:

- Anclajes: cómo generar asociaciones de estímulo respuesta para

Disparar una conducta deseada

Generar una conducta nueva

Acceder a un estado positivo

Neutralizar un anclaje negativo – cambiar el estado

Mejorar una experiencia pasada

- Submodalidades: cómo trabajar con las características de las experiencias internas para

Eliminar imágenes y sonidos internos

Mejorar una sensación dolorosa

Cambiar un estado

Desactivar una conducta automática

Mejorar una relación

Desactivar una fobia

- Neutralizar el impacto emocional de experiencias pasadas

Hipnosis ericksoniana (Modelo Milton): cómo facilitar el acceso a los procesos inconscientes de la mente mediante el modelo Milton (patrones de comunicación hipnótica ericksoniana) para

Inducir una conducta o un estado (hipnosis)

Autoinducir una conducta o estado (autohipnosis)

- Metáforas y comunicación persuasiva: utilizar la comunicación metafórica favoreciendo el trabajo a diferentes niveles de conciencia para

Acompañar al otro a un nivel profundo

Movilizar hacia el cambio desde el inconsciente

- Resolución de conflictos internos e interpersonales: cómo favorecer la resolución de conflictos internos y externos aprendiendo a

Detectar y acompañar incongruencias

Reencuadrar experiencias con el lenguaje

Integrar polaridades (conductas enfrentadas)

Cambiar conductas indeseadas persistentes con el inconsciente

Llegar a un acuerdo condicional en los conflictos con otras personas

- Estrategias de excelencia: cómo utilizar estrategias de excelencia con diferentes objetivos

Estrategias de excelencia de motivación para dejar de procrastinar

Estrategias de excelencia de creatividad Disney para hacer realidad los sueños

Estrategias de excelencia de ecología para asegurar la ecología sistémica de un objetivo deseado

Los aprendizajes del Máster en PNL (o tercer nivel de nuestra Torre de la PNL) nos enseñan a adentrarnos y trabajar en los niveles más profundos de la personalidad: creencias, valores, identidad y sistémico-transpersonal.

Para ello, este tercer nivel descansa y se asienta sobre los dos anteriores que aprendemos en el Practitioner.

Las personas vivimos las experiencias a diferentes niveles. Robert Dilts identificó 6:

- Ambiente
- Conducta
- Capacidades
- Creencias y Valores
- Identidad
- Sistémico – Transpersonal

Ejemplos del PLN

Clasificación de documentos

La tarea de clasificar grandes cantidades de documentos según su temática o estilo puede agilizarse con sistemas de PLN.

Análisis del sentimiento y de la opinión

Los comentarios en redes sociales sobre productos o servicios son muy relevantes para las empresas y los sistemas de PLN permiten extraer información relevante.

Comparación de textos

Los sistemas de PLN permiten hallar patrones en textos y detectar coincidencias entre ellos, lo que facilita la detección de plagios y el control de calidad.

Amonificación de documentos

A través de sistemas de PLN pueden procesarse documentos para identificar y eliminar las menciones a datos personales, asegurando así la privacidad de personas e instituciones.

Programas informáticos

Los programas de traducción Google Translate o DeepL, los asistentes de voz como Siri, Alexa o Google Assistant, o chatbots son programas informáticos que emplean el PLN.

búsqueda y análisis

Interpretación de las consultas de búsqueda y análisis de contenido: determinación de las necesidades del usuario cuando interactúa con un dispositivo (chatbot, motores de búsqueda, asistentes de voz).

Detección de sentimientos y emociones

Para comprender ciertos mensajes con su intencionalidad, el procesamiento de lenguaje natural ya incursiona en el análisis de las emociones que se expresan a través de frases que aparecen en opiniones.

DESCRIPCIÓN DE LA PRÁCTICA

Objetivo:

Crear un chatbot mediante programación que implemente el procesamiento de lenguaje natural

Competencias desarrolladas:

La presente practica abarca el tema 4 aplicaciones con técnicas de la IA del temario de la asignatura de la materia de Inteligencia Artificial con clave SCC – 1012 de la carrera de Sistemas Computacionales, permitiendo que el alumno implemente los conocimientos previos en programación y las bases teóricas de del tema 4.5 Procesamiento del lenguaje Natural (PLN)

Requisitos de software:



Lenguaje de programación de alto nivel Python



Entorno de desarrollo (IDE) Visual Studio Code

Requisitos de hardware:



Portátil u ordenador de escritorio

Desarrollo del código

La práctica se desarrolla con el lenguaje de programación Python mediante el entorno de desarrollo grafico visual studio code, como todo proyecto nuevo crearemos una carpeta nueva o new file ahí es donde escribiremos todas las líneas de código.

A continuación se le pondrá un nombre que identificara el área de trabajo en caso de tener más de un archivo ejecutable en nuestro código fuente es importe ponerle un nombre que sea claro a la parte de código que en él se escribe, para esta práctica solo tendremos un archivo ejecutable con el nombre “ chatbot”.

El siguiente código se compone de dos partes estructurales, las importaciones de librerías y bibliotecas y la definición del cuerpo donde se definen las funciones que van a contener las instrucciones para que nuestro bot funcione.

Para poder utilizar las librerías que python nos proporciona debemos de seguir la siguiente estructura

Palabra reservada <<import>> seguida del <<nombre de la librería>>

```
Get Started chatbot X Release Notes: 1.74.0
C: > Users > hp > Documents > Codigos python > chatbot > probabilidad_respuesta
1 import re
2 import random
3
```

La primera función que encontramos es llamada `get_response`, es una función que recibe como parámetro los datos ingresados por usuario mediante la terminal. Esta función es definida para obtener los datos que los usuarios introduzcan, haciendo conversiones para nuestro código pueda utilizar estas entradas y procesarlas devolviendo con el método. `split` las oraciones en palabras separadas.

```
4 def get_response(user_input):
5     split_message = re.split(r'\s|[,;.:?!-_\s*]', user_input.lower())
6     response = intenciones(split_message)
7     return response
8
```

`Probabilidad respuesta` es una función que recibe cuatro parámetros, como su nombre lo indica aquí se calculara una probabilidad de que las respuestas que tiene nuestro bot guardado puedan ser mostradas al usuario como una respuesta a su petición.

Inicialmente cada mensaje tiene una certeza de 0 % ya que ninguna debe salir antes de esperar una pregunta. Esta variable sirve de contador ya que cuando crece la posibilidad de ser un mensaje correcto este se agregará un +1 de forma iterativa.

El primer `for` verifica que la entrada dada por el usuario contenga palabras claves que también tengan nuestro bot dentro de su base de conocimiento.

El segundo `for` verifica que en la lista de palabras almacenada en `required_word` se encuentre alguna coincidencia con las palabras que usuario ingresó de no ser así no existen palabras requeridas y entonces el bot procede a dar una respuesta simple.

```

9 def probabilidad_respuesta(user_message, recognized_words, single_response=False, required_word=[]):
10     message_certainty = 0
11     has_required_words = True
12
13     for word in user_message:
14         if word in recognized_words:
15             message_certainty +=1
16
17     percentage = float(message_certainty) / float (len(recognized_words))
18
19     for word in required_word:
20         if word not in user_message:
21             has_required_words = False
22             break
23     if has_required_words or single_response:
24         return int((percentage * 100))
25     else:
26         return 0

```

Las líneas 28 y 29 son el inicio de la siguiente función, **intenciones** va contener dentro del mismo una nueva función para la selección de la respuesta a mostrar.

Respuestas recibe la actualización de cuatro parámetros que trabajan en conjunto a otras funciones ya que estos son variables de tipo global.

La siguiente línea de código es donde pasamos la probabilidad para obtener la que tenga mayor porcentaje.

```

28 def intenciones(message):
29     highest_prob = {}
30
31     def respuestas(bot_response, list_of_words, single_response = False, required_words = []):
32         nonlocal highest_prob
33         highest_prob[bot_response] = probabilidad_respuesta(message, list_of_words, single_response, required_words)
34

```

La implementación del PLN en este chatbot se define principalmente en la posibilidad de dar una respuesta a una pregunta, es necesario crear una serie de patrones que sirven de consultas para las diversas situaciones.

Al conjunto de patrones en IA se le conoce como intents o intenciones, las líneas 35 a 48 conforman el conjunto de intenciones con la que funciona el chatbot.

La estructura de estos patrones es simple cada respuesta que pueda ser una posible respuesta optima está acompañada de una palabra o serie de palabras que pueden ser dadas como entradas en el chat.

<<Salida>> – <<Entrada >> – <<Requerimiento>>

entendemos como salida el mensaje que mostrara el bot como mensaje de salida, la entrada es la o las palabras que puedan ingresar, ya sea una pregunta o un simple saludo y el requerimiento es para diferenciar entre una respuesta que tenga palabras requeridas es decir que solo si se encuentra en la entrada la palabra clave

ese mensaje se mostrara. `Single _response` es un mensaje de respuesta simple que se mostrara como opción cuando la entra coincida.

Tantas más respuestas haya más profundo es el aprendizaje y mejores respuestas pueden ser dadas.

```
35 respuestas('Hola,¿En que puedo ayudarte?', ['hola'], single_response = True)
36 respuestas('La materia es impartida actualmente por la M.I. Sonia Martínez Guzman',
37 ['quien','que','da','ia','inteligencia','artificial'],
38 required_words=['quien'])
39 respuestas('Cuatro temas conforman la materia', ['cuantos','temas','temario','asignatura','temarios'],
40 required_words= ['cuantos'])
41 respuestas('Siempre a la orden', ['gracias', 'te lo agradezco', 'thanks'], single_response=True)
42 respuestas('Estoy para ayudarte', ['como','estas'],single_response=True)
43 respuestas('SCC-1012',['clave', 'asignatura'],single_response=True)
44 respuestas('Puedes encontrar información en la web, biblioteca o consultar con un docente',
45 ['donde','encontrar','información'],required_words=['donde'])
46 respuestas('Fui nombrado Aibot',['como','llamas'],single_response=True)
47 respuestas('Aibot',['cual','nombre'],single_response=True)
48 respuestas('Puedo proporcionar informacion relacionada a la asignatura de IA',['que','informacion','das'],required_words=['informacion'])
49
```

Intenciones finaliza con el cálculo de la mejor respuesta basándose en la probabilidad que tiene cada palabra con las coincidencias.

Existe también una respuesta alternativa que puede ser devuelta en caso de que la probabilidad no haya aumentado en las iteraciones de `probabilidad_respuesta` al no aumentar esta sigue teniendo el valor inicial de 0.

```
50 mejor_respuesta = max(highest_prob, key=highest_prob.get)
51 #print(highest_prob)
52
53 return respuesta_desconocida() if highest_prob[mejor_respuesta] < 1 else mejor_respuesta
54
```

`Respuesta_desconocida` es la última función que compone el código fuente como su nombre lo indica aquí se define una respuesta alternativa que funciona como comodín es decir que si al hacer un cálculo de probabilidad en las intenciones no hay respuesta que satisfaga la pregunta del usuario esta respuesta se activará y será mostrada. La idea en general no es que sea una respuesta global si no una alternativa sin embargo entre mayores patrones tenga el bot de donde alimentarse menor será el uso de este mensaje.

```
55 def respuesta_desconocida():
56     response = ['puedes decirlo de nuevo?', 'No estoy seguro de lo quieres','No puedo responder'][random.randrange(3)]
57     return response
58
59 while True:
60     print("AIBot: " + get_response(input('Tu: ')))
```

Resultados

Para ver el funcionamiento de chatbot comenzamos a ejecutar el código.

El primer escenario muestra una pequeña conversación en donde se pide información sobre la materia de inteligencia artificial.

```
Tu: hola
AIBot: Hola,¿En que puedo ayudarte?
Tu: quiero informacion
AIBot: Puedo proporcionar informacion relacionada a la asignatura de IA
Tu: quien es el encargado de la materia?
AIBot: La materia es impartida actualmente por la M.I. Sonia Martinez Guzman
Tu: y cuantos temas son en la materia?
AIBot: Cuatro temas conforman la materia
Tu: gracias
AIBot: Siempre a la orden
```

Escenario dos, conversación informal con el Albot.

```
Tu: hola
AIBot: Hola,¿En que puedo ayudarte?
Tu: como te llamas?
AIBot: Fui nombrado AIbot
Tu: donde vender libros de IA?
AIBot: Puedes encontrar información en la web, biblioteca o consultar con un docente
```

Escenario tres: en este caso podemos observar una conversación que no tiene un flujo direccional ni un tema en concreto por lo que podemos observar la respuesta a un caso desconocido que definimos en la última función.

```
Tu: azul
AIBot: No estoy seguro de lo quieres
Tu: sabes sumar?
AIBot: No estoy seguro de lo quieres
Tu: []
```

```
Tu: ok
AIBot: No estoy seguro de lo quieres
Tu: ok
AIBot: No puedo responder
```

Diagramas del proyecto

Flujo del dialogo en Albot

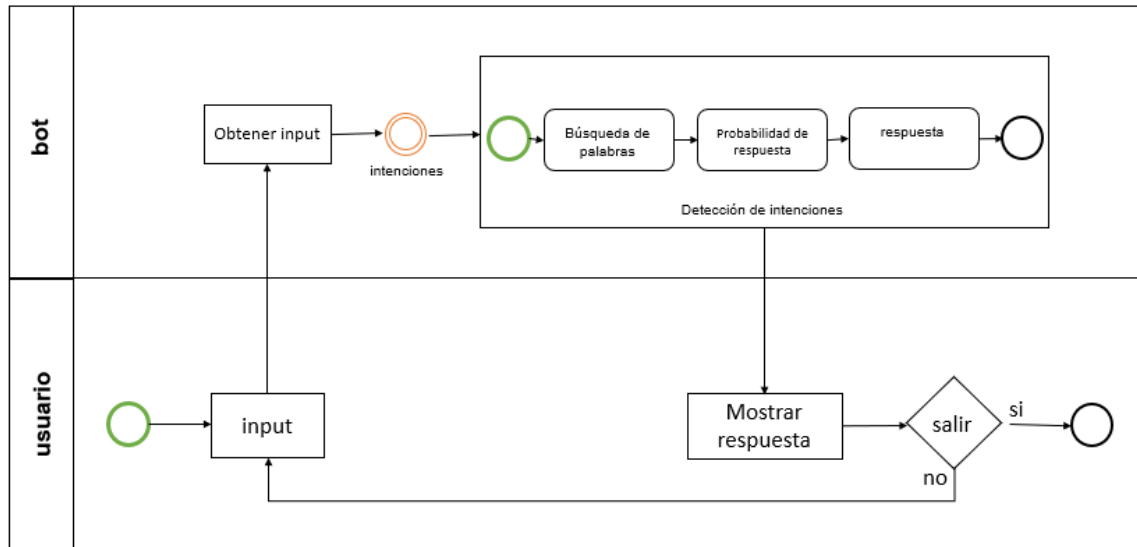


Ilustración 1 Diagrama de proceso de flujo del dialogo del chatbot Albot

Recomendaciones



Para trabajar con Python:

- Instalar el lenguaje Python desde la página oficial <https://www.python.org/>
- Antes de descargar los instaladores de Python verificar la información del sistema y el sistema operativo de la maquina donde se prevé instalar Python ya que estos pueden llegar a presentar problemas.
- Aunque se recomienda descargar la última versión estable si considera futuros proyectos utilizando aprendizaje profundo se recomienda descargar la versión 3.6.8
- Utilizar algún IDE de su preferencia para facilitar las importaciones de librerías, ya que el IDLE de Python solo es editor de texto.



Para trabajar con visual studio code:

- Descargar visual studio code de la página oficial <https://code.visualstudio.com/>
- Antes de descargar el instalador de visual studio code revise las especificaciones de su computador esto evitara que ocurran fallos en la instalación.
- Para trabajar con el lenguaje Python ir a extensión y seleccionar la primera opción que aparece importante verificar que sea publicado de la empresa Microsoft.
- Visual estudio code es un editor de código fuente de gran utilidad sin embargo para proyectos más extensos con uso de bibliotecas para inteligencia artificial están deben ser instaladas desde la línea de comandos.



Para escribir código

una vez iniciado el proyecto asignarle un nombre corto y simple pero que a su vez sea preciso para identificar de que se trata el proyecto.

Las buenas prácticas entre los programadores recomiendan utilizar la estructura camelCase al momento de escribir el nombre del proyecto.

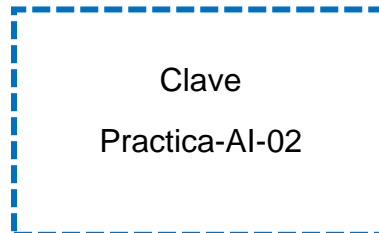
Al igual que el nombre del proyecto el nombre que le asigne a sus funciones y sus variables tienen que ser precisas y de fácil identificación a modo que su código sea entendible con una vista rápida incluso si este no será leído por terceros.

TITULO DE LA PRÁCTICA

MANUAL PROCEDIMENTAL PARA EL DESARROLLO DE LA PRÁCTICA
“DISEÑO DE UNA RED NEURONAL MULTICAPA”

IDENTIFICADOR DE LA PRÁCTICA

El identificador de la práctica es una clave formulada para hacer referencia exacta a la práctica que nos estamos refiriendo, esta sirve también para ser identificada con mayor facilidad en el repositorio digital donde se almacena.



PRESENTACIÓN

Con la evolución de la inteligencia artificial podemos ver ejemplos de su aplicación a grande y pequeña escala, muchos de estos proyectos son impresionantes por el resultado que son capaces de entregar alguna vez todos nos preguntamos si los robots igualaran al ser humano y conforme va avanzando la tecnología y las investigaciones vemos más cerca la respuesta. En el año de 1992 se plantearon manejar un vehículo no tripulado los investigadores de la universidad de Carnegie trabajaron en este proyecto para finalmente demostrar que precisamente ese hallazgo se podía cumplir, pero ¿cómo paso esto? A base de una red neuronal que se alimentaba de una cámara fotográfica con una única tarea establecida, dirigir el volante dentro del camino. Las redes neuronales artificiales están presentes en muchas áreas gracias a la función que cumplen dándole honor a su nombre ya que estas toman el principio de las neuronas de cerebro humano. Al igual que el cerebro le permite tomar decisiones y procesar grandes cantidades de información al hombre las redes neuronales artificiales tienen el objetivo de entender, procesar y clasificar los datos del mundo para crear patrones que pueden ser aplicados y optimizar el procesamiento de la información.

La clasificación de una red neuronal es variada y puede estar dada por sus números de capas, tipos de conexiones y otros factores más. Lo que ayuda a que el entrenamiento de la red neuronal sea cada vez más eficiente y disminuya la magnitud de la pérdida de la información. A continuación, se documenta el proceso que se lleva a cabo al diseñar una red neuronal artificial.

FUNDAMENTACIÓN TEÓRICA

Una de las herramientas más potente de la inteligencia artificial son las redes neuronales artificiales debido a las tareas que con ellas se pueden hacer por ejemplo la red neuronal más conocida es la del algoritmo de búsqueda de Google todos conocemos el motor de búsqueda tan impresionante y que día a día miles de personas ocupan. La inspiración de las redes neuronales nació de un trabajo científico donde se creó el perceptrón, este tomaba valores binarios y tenía como salida un único nodo un modelo sumamente pequeño a comparación de las redes que hoy en día se manejan que tienen miles y miles de nodos por los que se procesa toda la información.

Evolución de las RN

1958 el científico Frank Rosenblatt, inspirado en el trabajo de Warren McCulloch y Walter Pitts creó el Perceptrón, la unidad desde donde nacería y se potenciarían las redes neuronales artificiales.

1965 el perceptrón evoluciona a multilayer perceptrón que es una “ampliación” de la percepción de una única neurona a más de una apareciendo también el concepto de capas de entrada, oculta y salida.

En 1980 avance del aprendizaje automático gracias a un nuevo modelo de neurona llamadas Neuronas Sigmoides que son similares al perceptrón, pero permiten que las entradas, en vez de ser ceros o unos, puedan tener valores reales como 0,5 ó 0,377 ó lo que sea. Además, existe el concepto de “fully connected Feedforward Networks” y se refiere a que todas las neuronas de entrada, están conectadas con todas las neuronas de la siguiente capa.

1986 Gracias al algoritmo de backpropagation se hizo posible entrenar redes neuronales de múltiples capas de manera supervisada. Al calcular el error obtenido en la salida e ir propagando hacia las capas anteriores se van haciendo ajustes pequeños (minimizando costo) en cada iteración para lograr que la red aprenda consiguiendo que la red pueda -por ejemplo- clasificar las entradas correctamente.

1989 la primera Convolutional Neural Networks CNN fue creada por Yann LeCun que son redes multilayered que toman su inspiración del cortex visual de los animales. Esta arquitectura es útil en varias aplicaciones, principalmente procesamiento de imágenes

En 1997 se crearon las LSTM que consisten en unas celdas de memoria que permiten a la red recordar valores por períodos cortos o largos, esta arquitectura permite conexiones “hacia atrás” entre las capas una celda de memoria contiene compuertas que administran como la información fluye dentro o fuera.

2006 la creación de una DBN que logro obtener un mejor resultado en el MNIST, se devolvió el entusiasmo en poder lograr el aprendizaje profundo en redes neuronales. Hoy en día las DBN no se utilizan demasiado, pero fueron un gran hito en la historia

en el desarrollo del deep learning y permitieron seguir la exploración para mejorar las redes existentes CNN, LSTM, etc.

2014 las redes tienen dos modelos de redes neuronales compitiendo estas dos redes juegan una partida continua donde el Generador aprende a producir muestras más realistas y el Discriminador aprende a distinguir entre datos reales y muestras artificiales. Estas redes son entrenadas simultáneamente para finalmente lograr que los datos generados no puedan detectarse de datos reales.

Funcionamiento de las redes neuronales

Una red neuronal enseña a las computadoras a procesar datos de una manera que está inspirada en la forma en que lo hace el cerebro humano, este proceso es conocido como aprendizaje profundo, que utiliza los nodos o las neuronas interconectados en una estructura de capas que se parece al cerebro humano. Crea un sistema adaptable que las computadoras utilizan para aprender de sus errores y mejorar continuamente. De esta forma, las redes neuronales artificiales intentan resolver problemas complicados, como la realización de resúmenes de documentos o el reconocimiento de rostros, con mayor precisión.

Las células del cerebro humano, llamadas neuronas, forman una red compleja y con un alto nivel de interconexión y se envían señales eléctricas entre sí para ayudar a los humanos a procesar la información. De manera similar, una red neuronal artificial está formada por neuronas artificiales que trabajan juntas para resolver un problema. Las neuronas artificiales son módulos de software, llamados nodos, y las redes neuronales artificiales son programas de software o algoritmos que, en esencia, utilizan sistemas informáticos para resolver cálculos matemáticos.

Tipos de redes neuronales

Perceptrón multicapa

La red neuronal multicapa es una generalización de la red neuronal monocapa, la diferencia reside en que mientras la red neuronal monocapa está compuesta por una capa de neuronas de entrada y una capa de neuronas de salida, esta dispone de un conjunto de capas intermedias (capas ocultas) entre la capa de entrada y la de salida.

Dependiendo del número de conexiones que presente la red esta puede estar total o parcialmente conectada.

Perceptrón simple

La red neuronal monocapa se corresponde con la red neuronal más simple, está compuesta por una capa de neuronas que proyectan las entradas a una capa de neuronas de salida donde se realizan los diferentes cálculos.

Red neuronal Convolutiva (CNN)

La principal diferencia de la red neuronal convolucional con el perceptrón multicapa viene en que cada neurona no se une con todas y cada una de las capas siguientes, sino que solo con un subgrupo de ellas (se especializa), con esto se consigue reducir el número de neuronas necesarias y la complejidad computacional necesaria para su ejecución.

Red neuronal recurrente (RNN)

Las redes neuronales recurrentes no tienen una estructura de capas, sino que permiten conexiones arbitrarias entre las neuronas, incluso pudiendo crear ciclos, con esto se consigue crear la temporalidad, permitiendo que la red tenga memoria.

Los datos introducidos en el momento t en la entrada, son transformados y van circulando por la red incluso en los instantes de tiempo siguientes $t + 1$, $t + 2$

Redes de base radial (RBF)

Las redes de base radial calculan la salida de la función en función de la distancia a un punto denominado centro. La salida es una combinación lineal de las funciones de activación radiales utilizadas por las neuronas individuales.

Las redes de base radial tienen la ventaja de que no presentan mínimos locales donde la retro propagación pueda quedarse bloqueada.

Arquitectura de una red neuronal

Una red neuronal básica tiene neuronas artificiales interconectadas en tres capas:

- Capa de entrada

La información del mundo exterior entra en la red neuronal artificial desde la capa de entrada. Los nodos de entrada procesan los datos, los analizan o los clasifican y los pasan a la siguiente capa.

- Capa oculta

Las capas ocultas toman su entrada de la capa de entrada o de otras capas ocultas. Las redes neuronales artificiales pueden tener una gran cantidad de capas ocultas. Cada capa oculta analiza la salida de la capa anterior, la procesa aún más y la pasa a la siguiente capa.

- Capa de salida

La capa de salida proporciona el resultado final de todo el procesamiento de datos que realiza la red neuronal artificial. Puede tener uno o varios nodos. Por ejemplo, si tenemos un problema de clasificación binaria (sí/no), la capa de salida tendrá un nodo de salida que dará como resultado 1 o 0. Sin embargo, si tenemos un problema de clasificación multiclase, la capa de salida puede estar formada por más de un nodo de salida.

Aplicaciones de las redes neuronales

Son muchas las funciones que puede llevar a cabo una red neuronal, sobre todo, porque gracias a su funcionamiento son capaces de realizar multitud de tareas dependiendo de su entrenamiento:

- Reconocimiento. Una de las funciones más recurrentes de esta tecnología es su uso para el reconocimiento. Gracias a su capacidad de aprendizaje y de entrenamiento, una red neuronal puede diferenciar entre diferentes elementos.
- Clasificación. De una forma muy similar a su función de reconocimiento, una red neuronal puede usarse para clasificar diferentes elementos.
- Predicción y diagnóstico. Más allá de sus capacidades para diferenciar un elemento, una red neuronal puede predecir eventos futuros y es por ello por lo que se utiliza esta tecnología para temas de predicciones económicas. Por otro lado, en medicina, este tipo de función puede utilizarse para diagnosticar posibles problemas de salud.
- Predicción de sucesos y simulaciones: Producción de los valores de salida esperados en función de los datos entrantes.
- Reconocimiento y clasificación: Asociación de patrones y organización de conjuntos de datos en clases predefinidas. Incluso identificando características únicas sin datos previos.
- Procesamiento de datos y modelización: Validación, agregación y análisis de datos. Diseño y búsqueda de fallos en sistemas de software complejos.
- Ingeniería de control: Monitorización de sistemas informáticos y manipulación de robots. Incluida la creación de sistemas y robots autónomos.
- Inteligencia Artificial: Formando parte de las tecnologías de deep learning y machine learning que son partes fundamentales de la inteligencia artificial

DESCRIPCIÓN DE LA PRÁCTICA

Objetivo:

Diseñar una red neuronal artificial multicapa capaz de predecir la temperatura en kelvin

Competencias Desarrolladas

La presente practica abarca el tema 4 aplicaciones con técnicas de la IA del temario de la asignatura de la materia de Inteligencia Artificial con clave SCC – 1012 de la carrera de Sistemas Computacionales, permitiendo que el alumno implemente los conocimientos previos en modelos matemáticos y compresión de las funciones lineales y no lineales de materias como algebra lineal y las bases teóricas de del tema 4.2 de la materia de inteligencia artificial Redes Neuronales (RN)

Requisitos de Software



Navegador web



Servicio de Notebook en la nube



Cuenta de Google con servicio de Gmail



Framework TensorFlow

Requisitos de Hardware



Portátil o computador de escritorio



Procesador polivalente

Diseño de la red neuronal

En esta práctica se realizará el diseño de una red neuronal artificial multicapa que sea capaz de entrenarse con un conjunto de datos de tal forma que cuando se le indique un valor en términos de grados Celsius este realice el procesamiento de la información y retorne la conversión a grados kelvin. Nótese que no hay necesidad de indicarle una formula establecida a la red neuronal ya que este hará los cálculos con las funciones de activación indicadas en la programación de la misma.

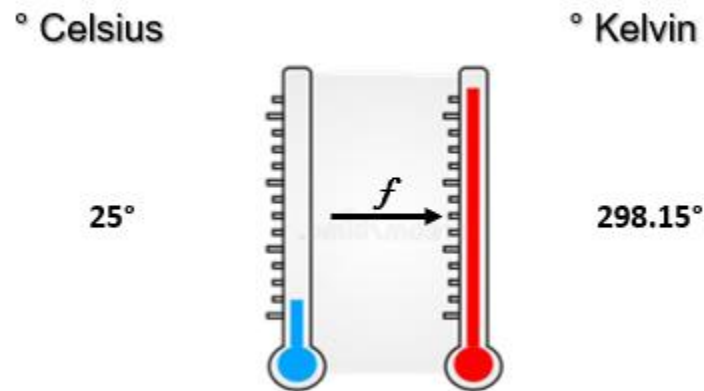


Ilustración 2 Comparación temperatura Celsius-Kelvin

$$C = K - 273.15$$

$$K = C + 273.15$$

Ilustración 3 Formula de conversión Celsius-Kelvin

Datos de entrenamiento

Información para alimentar el entrenamiento del modelo de red neuronal, donde los grados Celsius son los datos de entrada y los grados kelvin los datos de salida.

| ° Celsius | ° kelvin |
|-----------|----------|
| 36 | 309.15 |
| 40 | 313.15 |
| -6 | 267.15 |
| 17 | 290.15 |
| 20 | 293.15 |
| 66 | 339.15 |
| -1 | 272.15 |
| 10 | 283.15 |

Arquitectura de la red neuronal

La RNA diseñada consta de los siguientes elementos:

Capa de entrada: es la primera capa que constituye nuestra red neuronal por ella entran los datos del mundo exterior para ser enviados a la siguiente capa, aquí se realiza una primera clasificación en nuestro ejemplo de la temperatura kelvin la capa de entrada solo consta de una neurona para representarla en nuestro editor colocamos un círculo, dividiremos las capas por tipo dentro del recuadró a modo de tener una mejor visualización.

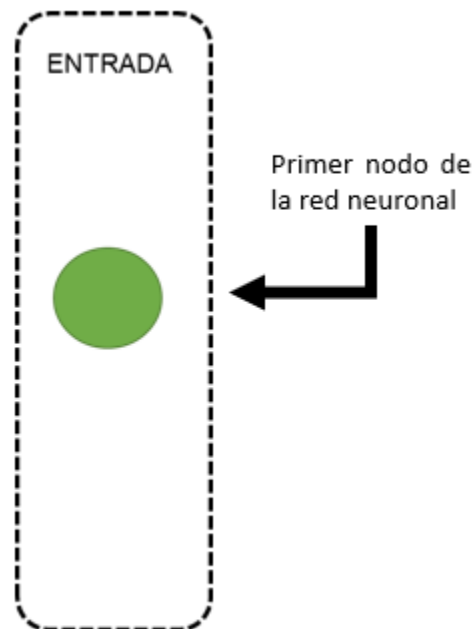


Ilustración 4 Capa de entrada de una RN

Capa oculta: la capa oculta puede estar constituida por n capas de n números de neuronas, toman los datos de entrada de la salida de la capa de entrada cada una de ellas procesa los datos que recibe con la función de activación una vez procesada la información se envía a la siguiente capa oculta, la RNA tiene dos capas ocultas con 3 neuronas cada una, el aprendizaje automático se irá haciendo en esta parte del modelo como no hemos dado ninguna fórmula a seguir para dar con el resultado cada capa se encarga de analizar su entrada de tal forma que entre más capas pasen más procesados están los datos y por ende más cerca se está de la respuesta correcta sin embargo así como en cada capa se encuentra una función de activación, en las capas ocultas podemos encontrar el sesgo este ayudará a el modelo a controlar la predisposición de la probabilidad de salida.

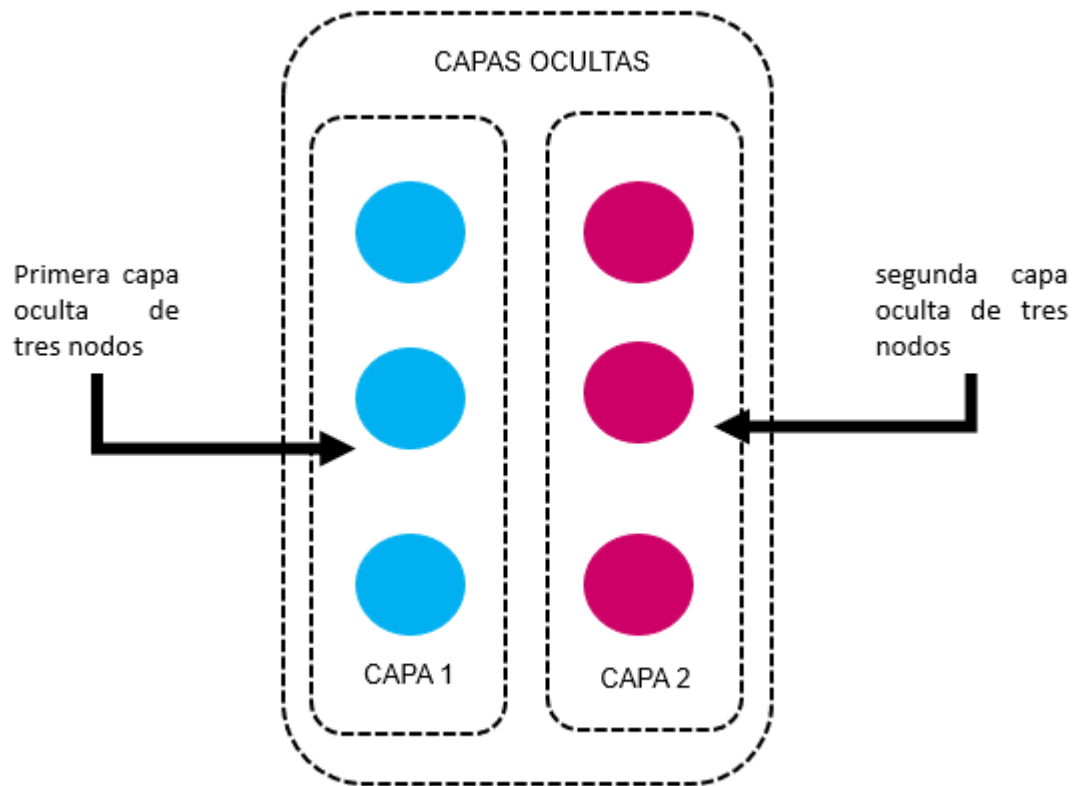


Ilustración 5 Capa oculta de una RN

Capa de salida: esta será nuestra última capa es por donde se proporcionará el resultado de todo el proceso de datos realizados en los nodos anteriores, este resultado será nuestra predicción, en nuestro modelo solo tenemos un nodo de salida puesto que solo esperamos un resultado correcto.

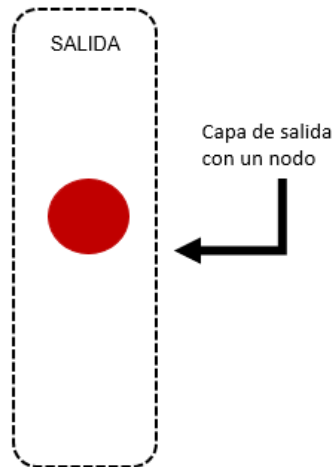


Ilustración 6 Capa de salida de una RN

Conexiones: las conexiones de nodo a nodo las pondremos de entrada a la primer capa oculta que aparezca, cada dato que entre debe pasar a cada nodo siguiente para ser procesado estas a su vez dan como salida la información y pasa a convertirse a la entrada de otro nodo en la siguiente capa oculta, de haber más capas ocultas este proceso se repite sin embargo aquí solo tenemos dos capas por lo tanto la segunda capa procesara los datos recibidos en su entrada y arrojará la información a la capa de salida ahí es donde nos arrojará el resultado o la predicción de la red neuronal.

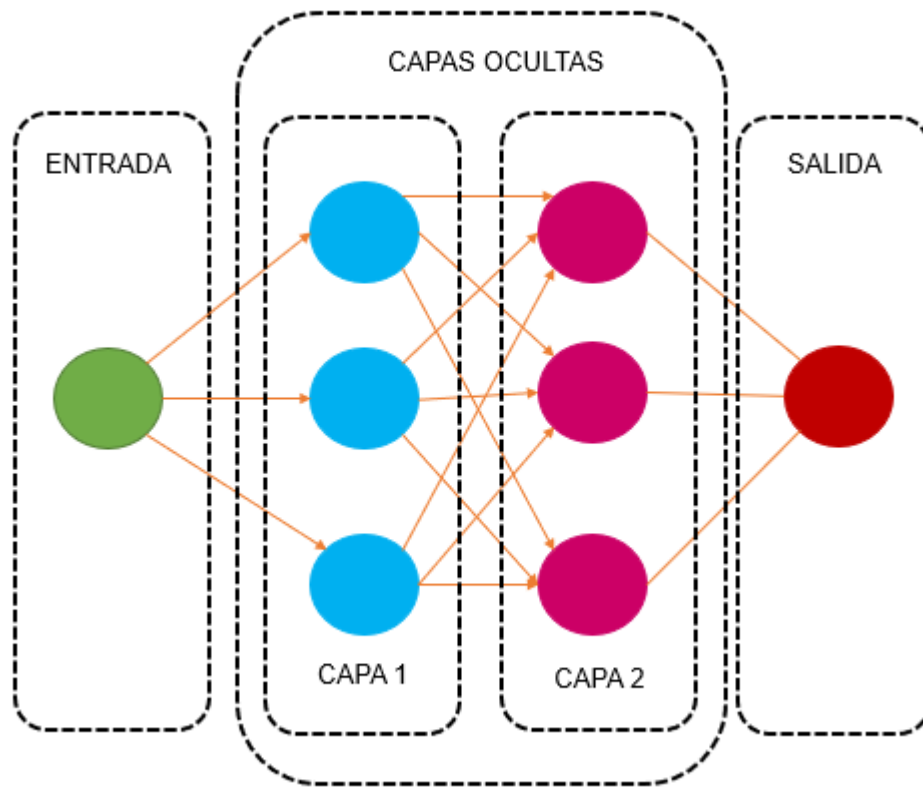


Ilustración 7 Representación de la RN Gkelvin

Desarrollo del código

Para escribir el siguiente código estamos utilizando un servicio de notebook en la nube que nos proporciona la integración del lenguaje de programación Python. Empezamos creando un nuevo cuaderno.

Ocuparemos la biblioteca TensorFlow para el aprendizaje automático de la red neuronal, así como la librería numpy que permite el trabajar con arrays y matrices multidimensionales. Para hacer uso de ellas escribimos las siguientes líneas al principio de nuestro código

```
import tensorflow as tf  
import numpy as np
```

Para alimentar nuestra red neuronal necesitamos un conjunto de datos con los que se pueda establecer una relación y con ella se cree el aprendizaje automático. Vamos a utilizar dos arrays el primero representa los datos bases mismos que funcionan como datos de entrada para la red neuronal y el segundo array representa el conjunto de datos de salida, importante que ambos tipos de datos sean float y que por cada dato de entrada exista un dato de salida por lo que si nuestro array **Mcelsius** tiene 7 registros el segundo array **Mkelvin** también deberá contener 7 registros.

```
Mcelsius = np.array([36,40,-6,17,20,66,-1,10], dtype=float)  
Mkelvin = np.array([309.15,313.15,267.15,290.15,293.15,339.15,272.15,283.15], dtype=float)
```

Una vez tenemos los datos, vamos a crear la estructura de la red neuronal empezamos creando las capas ocultas procedemos con la capa de salida y después definimos la secuencia que llevara nuestro modelo.

Para la primera capa llamada **oculta1** le vamos a indicar cuantas neuronas la componen con la instrucción **units = 3** seguido los datos de entrada ya que es la primera capa por la que van a pasar los datos sin procesar.

Ocultas2 sigue la misma estructura a excepción que no se le indica la capa de entrada.

Salida es nuestra última capa por donde se muestra la salida de información ya procesada esta solo tiene una única neurona artificial.

Decimos que modelo es toda la estructura de la red neuronal por lo que aquí indicamos cual va primero y cual va después, utilizamos **Sequential** ya que nuestra red tiene una entrada de datos y una salida de datos si nuestro modelo tuviera múltiples entradas y salidas nuestra modelo seria incorrecto.

```

oculta1 = tf.keras.layers.Dense(units=3, input_shape=[1])
oculta2 = tf.keras.layers.Dense(units=3)
salida = tf.keras.layers.Dense(units=1)
modelo = tf.keras.Sequential([oculta1, oculta2, salida])

modelo.compile(
    optimizer=tf.keras.optimizers.Adam(0.1),
    loss='mean_squared_error'
)

```

El fragmento marcado con rojo es la declaración que hacemos para la optimización del método de adam y obtener la gradiente de descenso, la pérdida o “los” es el error medio cuadrático para obtener que tan cerca esta la línea de regresión de nuestro punto de datos

```

oculta1 = tf.keras.layers.Dense(units=3, input_shape=[1])
oculta2 = tf.keras.layers.Dense(units=3)
salida = tf.keras.layers.Dense(units=1)
modelo = tf.keras.Sequential([oculta1, oculta2, salida])

modelo.compile(
    optimizer=tf.keras.optimizers.Adam(0.1),
    loss='mean_squared_error'
)

```

Las 5 primeras líneas son para imprimir y las empleamos para darle una apariencia más clara y limpia a la salida de datos tomando en consideración que estos se mostraran en la terminar del notebook

Historial representa la bitácora de aprendizaje que está trabajando la red neuronal vemos también los valores con los que se alimenta y las veces en las que esta se entrena epochs = 500 el ultimo parámetro verbose= falso es para mostrar a detalle los errores que llegaron a tener durante la compilación del aprendizaje.

```

print("Comenzando entrenamiento...")
print("|")
print("|  |")
print("|  |  |")
historial = modelo.fit(Mcelsius, Mkelvin, epochs=500, verbose=False)
print("|  Modelo entrenado  |")

```

La importación de matplotlib nos permite usar una gráfica en donde podemos ver la gradiente del ciclo de entrenamientos que realizo el modelo y el nivel de perdida que se llega a tener.

```
import matplotlib.pyplot as plt
plt.xlabel("ciclo de entrenamiento")
plt.ylabel("Nivel de error porcentual")
plt.plot(historial.history["loss"])
```

Lo último por hacer es probar si a la red neuronal, para ello ingresaremos un dato que no está en los datos iniciales, el modelo empezara a trabajar de modo que llegue al resultado más próximo. La predicción de la red es la respuesta que esta pueda llegar a dar con un modelo matemático que ella misma fórmula y que satisfaga en la mejor medida la relación entre los patrones que la alimentan.

```
print("Predicción")
data = 22
resultado = modelo.predict([data])
print("El resultado de la predicción es: ")
print("_____")
print("Grados centigrados" + "|" + "Grados Kelvin")
print("|" + str(data) + "      |" + str(resultado) + "|")
print("_____")
```

Las 5 líneas de código mostradas están comentadas ya que son líneas para ver de una forma un tanto más claro el funcionamiento que realizan las capas ocultas del modelo de la red neuronal.

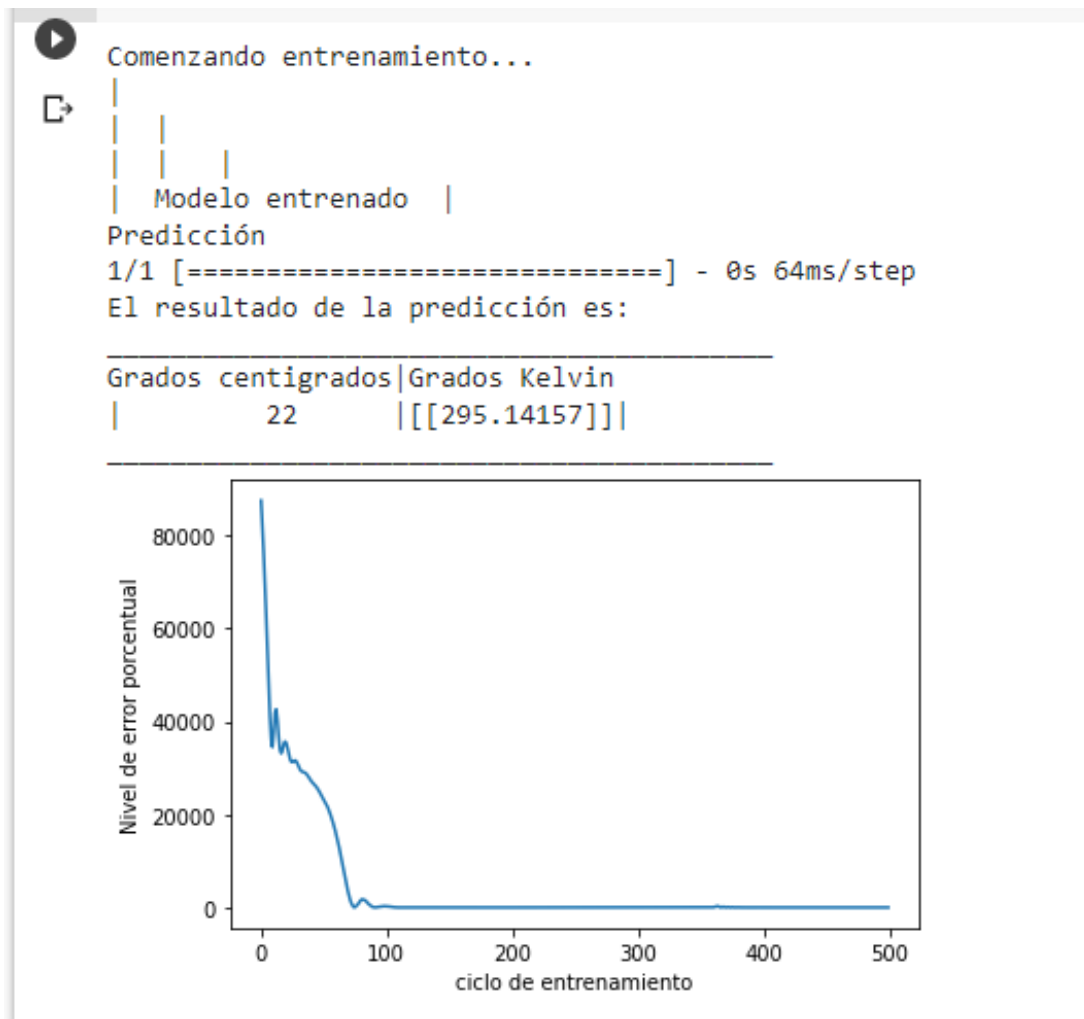
```
#print("Calculos internos de las capas")
#print(capa.get_weights())
#print(oculta1.get_weights())
#print(oculta2.get_weights())
#print(salida.get_weights())
```


Resultados

En la siguiente imagen se muestran los resultados arrojados por la red neuronal artificial que creamos para el calculo y prediccion de la temperatura en grados Kelvin

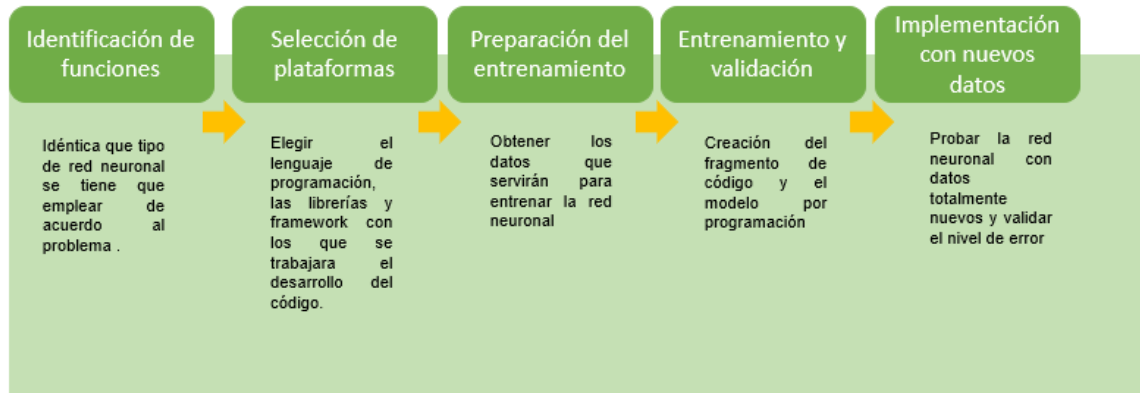
Al comparar los datos con datos veridicos de conversion externos podemos observar que la red neural tiene un pequeño margen de error sin embargo para ser de dos capas los resultados pueden ser tomados como correctos. De igual forma podemos ver la relacion que existe entre mas capas tenga el modelo de red neuronal mayor aprendizaje hay y mas proximos son los calculos para llegar al original.

Al observar la grafica podemos ver que el aprendizaje se establecio pero al cabo de 100 a 200 epocas de ciclo el error no disminuye solo se mantiene llegando al final del entrenamiento y obteniendo un patron.



Diagramas del proyecto

Pasos para el Diseño e implementación de una red neuronal



RECOMENDACIONES



Para utilizar colab

- Utilizar colab en una computadora con procesador core i5 o superior ya que estos al tener dos núcleos permiten correr aplicaciones web de una manera eficiente.
- Colab se compone de celdas, puede ejecutar fragmentos de códigos en cada celda lo que es más recomendable para códigos pequeños pero que requieran seguir un paso a paso en el resultado.
- Al igual que un documento de drive el código creado en colab puede ser compartido o puede mantenerse en privado si no quiere que sea público su código configure la opción *compartir*.
- Los navegadores principales para utilizar colab son Google Chrome y Mozilla Firefox.



Para utilizar TensorFlow

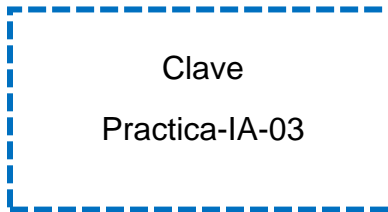
- La biblioteca de TensorFlow ya se encuentra integrada en el cuadernillo de colab por lo que no se requiere instalación de las extensiones
- Si desea utilizar el lenguaje Python con un entorno de desarrollo completamente diferente se recomienda la versión de Python 3.6.8 ya que en otras no existe soporte en la instalación
- Para realizar la instalación es mejor utilizar miniconda o anaconda ya que proporciona las bibliotecas completas para el uso de TensorFlow y más bibliotecas de inteligencia artificial.
- TensorFlow es una biblioteca muy poderosa y extensa porque su instalación también requiere de hardware específico para saber cuál es la mejor instalación consultar <https://www.tensorflow.org/install/pip>

TITULO DE LA PRÁCTICA

MANUAL PROCEDIMENTAL PARA EL DESARROLLO DE LA PRACTICA
“LECTOR DIGITAL DE IMAGEN EMPLEANDO VISION ARTIFICIAL”

IDENTIFICADOR DE LA PRÁCTICA

El identificador de la práctica es una clave formulada para hacer referencia exacta a la práctica que nos estamos refiriendo, esta sirve también para ser identificada con mayor facilidad en el repositorio digital donde se almacena.



PRESENTACIÓN

La visión artificial es una de las técnicas de la inteligencia artificial con mayor visión hacia proyectos del futuro ya que la búsqueda constante de automatización y optimización de trabajo en tareas realizadas por obra humana es uno de los puntos clave para seguir desarrollando en este campo. La presencia de la visión artificial es muy extensa y cubre casi todas las ramas disciplinarias, pero en esencia ¿qué es la visión artificial? En pocas palabras comprendemos este término como la especialización de un sistema que es capaz de adquirir, procesar y analizar imágenes reales para generar información que pueda asimilar una máquina de manera paradójica se puede comparar al sentido de la vista humana así como una persona es capaz de visualizar su entorno y todo lo que nos rodea y es capaz de comprender y obtener información mediante la vista la visión artificial también toma esos principios para procesar las imágenes y generar información.

Para el procesamiento de imágenes se requiere un conjunto de herramientas específicas para trabajar con visión artificial, estas herramientas pueden ser tanto herramientas de tipo hardware como herramientas de software entre ellas podemos encontrar sensores, detectores de color, software para procesamiento óptico entre otros más la gran variedad de estos ayudan a la implementación de esta técnica de inteligencia artificial en diferentes sistemas.

Basándose en la teoría y para comprender mejor el concepto y función que desempeña la visión artificial se efectúa la siguiente práctica en donde se desarrollará el código utilizando el lenguaje Python para el desarrollo de un lector digital de imagen en cual es capaz de procesar las imágenes dadas y obtener el texto que en ellas se pudiera encontrar.

FUNDAMENTACION TEÓRICA

La visión artificial funciona de manera muy similar a la visión humana donde las personas usan el sentido de la vista para comprender el mundo que las rodea de igual forma la IA permite que las computadoras y los sistemas obtengan información significativa de imágenes digitales, videos y otras entradas visuales, y tomen acciones o hagan recomendaciones basadas en esa información en el sentido más simple podemos decir que la visión artificial es una disciplina científica que incluye métodos para adquirir, procesar y analizar imágenes del mundo real con el fin de producir información que pueda ser tratada por una máquina.

Origen de la visión artificial

Para conseguir llegar a realizar tareas como detectar objetos, categorizarlos, segmentarlos o procesarlos, a partir de imágenes la inteligencia artificial debió adaptarse en un área específica que tuviera como objetivo conseguir estos funcionamientos es entonces que la Visión Artificial nace de la necesidad de poder imitar el comportamiento inteligente que muestran los seres vivos en el tratamiento de imágenes. En 1959 se llevó a cabo una experimentación donde los neurofisiólogos le mostraron a un gato una serie de imágenes, intentando correlacionar una respuesta en su cerebro. Descubrieron que respondía primero a bordes o líneas sólidas y, científicamente, esto significaba que el procesamiento de imágenes inicia con formas simples, como los bordes rectos ese mismo año se desarrolló la primera tecnología de escaneo artificial de imágenes, que permite a las computadoras digitalizar y adquirir imágenes.

En la década de 1960, la IA surgió como un campo de estudio académico y también marcó el comienzo de la búsqueda de la IA para resolver el problema de la visión humana.

En 1974 salió a la luz la tecnología de reconocimiento óptico de caracteres (OCR), que podía reconocer el texto impreso en cualquier fuente o tipo de letra. De manera similar, el reconocimiento inteligente de caracteres (ICR) podría descifrar el texto escrito a mano utilizando redes neuronales.

El científico informático Kunihiko Fukushima en el año 1982 desarrolló una red de células capaces de reconocer patrones. La red, llamada Neocognitron, incluía capas convolucionales en una red neuronal.

A lo largo de la década de 2000 surgió la estandarización de cómo se etiquetan y anotan los conjuntos de datos visuales.

En 2012, un equipo de la Universidad de Toronto inscribió a las CNN en un concurso de reconocimiento de imágenes. El modelo, llamado AlexNet, redujo significativamente la tasa de error para el reconocimiento de imágenes. Después de este gran avance, las tasas de error se han reducido a solo un pequeño porcentaje.

En los próximos años la visión artificial a crecido junto a proyectos de diversas áreas en donde requiere la obtención de información para su procesamiento significativo.

Beneficios como parte de la Inteligencia Artificial

Los beneficios que puede ofrecer la visión artificial van a depender en gran medida del área en donde se esté aplicando la técnica puesto que en algunas su presencia aumenta de valor. Podemos mencionar los siguientes beneficios en el área industrial:

- ✚ La visión artificial ayuda a cumplir metas específicas
- ✚ elimina el tiempo y los costos de mantenimientos vinculados al desgaste de componentes mecánicos
- ✚ brinda beneficios operativos
- ✚ seguridades adicionales al disminuir la participación humana en el proceso de fabricación
- ✚ Mayor velocidad en la medición cuantitativa de escenas estructuradas
- ✚ Precisión en la medición cuantitativa de escenas estructuradas
- ✚ Mejor replicabilidad en la medición cuantitativa de escenas estructuradas
- ✚ La visión artificial mejora la calidad y la producción
- ✚ Mejora la calidad del resultado.
- ✚ Optimiza el rendimiento, los costes y la eficiencia.
- ✚ Aumenta la flexibilidad y la seguridad de la producción.
- ✚ Reduce la tasa de desperdicio y los tiempos de inactividad de las máquinas.
- ✚ Incrementa el nivel de detalle de la información y de control de los procesos.
- ✚ Disminuye el espacio dedicado en la planta.
- ✚ Facilita el diagnóstico de problemas y su resolución.

Los beneficios en el área de la medicina y áreas farmacéuticas son:

Las aplicaciones de la visión artificial en el sector sanitario abren un mundo de posibilidades que favorecen a muchos sectores relacionados con este ámbito, como la biomedicina, la ciencia, la biología, la farmacia o la veterinaria. Las técnicas de visión artificial para aplicaciones como la cirugía guiada por imagen permiten personalizar las distintas fases del tratamiento quirúrgico de un paciente, de forma que se mejore el entrenamiento, la planificación y la ejecución del tratamiento personalizado.

Aplicaciones de la visión artificial

Medicina

- Termografía medica
- Cirugía guiada por imagen
- Diagnóstico de enfermedades
- Análisis forense
- Detección de equipos de protección individual (EPIs)
- Identificación de principios activos en productos

- Detección de defectos en envase y blíster
- Presencia y cierre de tapones
- Presencia, aplicación e integridad de etiquetas
- Lectura de códigos para trazabilidad de productos

General

- Localización 3D en tareas de “pick & place”.
- Localización de piezas en lotes mezclados.
- Asistencia para evitar accidentes en desplazamientos marcha atrás.
- Inspección óptica de defectos, errores o irregularidades.
- Marcado directo de piezas.
- Identificación y reconocimiento de etiquetas, envases o componentes.
- Calcular distancias entre varios puntos de un mismo objeto o su ubicación geométrica.
- Detección de objetos.
- Reconocimiento óptico de caracteres (OCR).
- Localización para robots con visión artificial.

Sector Industrial

- Visión artificial para la detección de defectos y control de calidad
- Metrología
- Detección de intrusos
- Lectura de códigos
- Verificación de montajes

DESCRIPCIÓN DE LA PRÁCTICA

Objetivo

Implementar un lector de imagen digital mediante herramientas de visión artificial como practica de la materia de inteligencia artificial.

Competencias desarrolladas

La presente practica abarca el tema 4 aplicaciones con técnicas de la IA del temario de la asignatura de la materia de Inteligencia Artificial con clave SCC – 1012 de la carrera de Sistemas Computacionales, permitiendo que el alumno implemente los conocimientos previos adquiridos en los temas de la misma materia y las bases teóricas de del tema 4.3 de la materia de inteligencia artificial visión artificial.

Requisitos de Software



Navegador web



Servicio de Notebook en la nube



Cuenta de Google con servicio de Gmail



Software de reconocimiento óptico Tesseract OCR

Requisitos de Hardware



Portátil o computador de escritorio



Procesador polivalente

Desarrollo del código

El siguiente código se ejecuta con ayuda de Colab, ya que es una notebook web es necesario hacer las instalaciones de algunas herramientas adicionales dentro del mismo código que escribimos. De manera general podemos estructurar el siguiente código en cuatro partes **#instalaciones** **#importar bibliotecas/librerías** **#desarrollo** **#mostrar resultados**

Para la primera parte vamos a comenzar instalando Tesseract OCR, este motor de reconocimiento óptico es empleado en inteligencia artificial y es el que nos ayudara a hacer el procesamiento y extracción de los caracteres en las imágenes.



```
!sudo apt-get install tesseract-ocr  
!pip install pytesseract==0.3.9
```

Una vez realizada la instalación vamos a importar las librerías que vamos a estar ocupando

- importamos todas las posibles librerías para no hacer el código mas extenso y hacer un mal consumo de memoria
- es necesario incluir files para trabajar con imágenes externas y de entorno local ya que son el principal componente con el trabajara nuestro lector digital no olvidar la línea Google.colab ya que este nos permite trabajar en el entorno de colab específicamente
- cv2 es parte de la librería openCV y con ella podemos hacer manipulación de las imágenes dentro de nuestro código
- adicionalmente importaremos un parche de cv2 para un correcto funcionamiento ya que como las versiones de escritorio de Python tienen diferente soporte colab requiere de versiones especificadas y extensiones para su marco de trabajo



```
from pytesseract import *  
from google.colab import files  
import cv2  
from google.colab.patches import cv2_imshow
```

La esencia del lector es trabajo y manejo de imágenes por lo tanto el primer paso es preparar el lector para que cargue una imagen esto lo logramos con la línea `files.upload()` diciéndole que en los próximos pasos debe haber un proceso de carga de imagen.

ImagenSubida es la variable donde vamos a guardar el retorno de la matriz que genera la lectura del método `cv2.imread`

la siguiente línea de código es muy importante y es la que hace la extracción del texto de las imágenes. Con el método `pytesseract.image_to_string` convertimos el retorno de **imagenSubida** a cadena una vez que se hace esta extracción es muy fácil trabajar ya que nuestro resultado es un string

```
files.upload()  
imagenSubida = cv2.imread('runBTS.png', cv2.IMREAD_UNCHANGED)  
extraer = pytesseract.image_to_string(imagenSubida, lang = 'eng')
```

Por último, debemos crear el espacio donde vamos a mostrar los resultados, ocuparemos la terminal. Como decíamos antes trabajar con un string es fácil por lo tanto para mostrar el texto que se encontraba en la imagen debemos imprimir en pantalla con la instrucción `print()` también verificamos que el texto que se extrajo coincida con el de la imagen original entonces vamos a mostrar la imagen que cargamos en el lector esto lo hacemos con `cv2_imshow()`

```
print('\n')  
print('TEXTO EXTRAIDO:')  
print(extraer)  
print('\n')  
print('IMAGEN ORIGINAL')  
cv2_imshow(imagenSubida)
```

`Files.upload()`

Permite cargar un archivo imagen local o externo al entorno de trabajo, al ejecutar se nos permite seleccionar la imagen desde una ventana emergente como se muestra a continuación.



Resultados

Resultado de la instalación de Tesseract OCR

```
↳ Reading package lists... Done
Building dependency tree
Reading state information... Done
tesseract-ocr is already the newest version (4.00~git2288-10f4998a-2).
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 20 not upgraded.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pytesseract==0.3.9 in /usr/local/lib/python3.8/dist-packages (0.3.9)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.8/dist-packages (from pytes
Requirement already satisfied: Pillow>=8.0.0 in /usr/local/lib/python3.8/dist-packages (from pytesse
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.8/dist-packages (f
```

Muestra del funcionamiento del lector digital construido en esta práctica.

```
↳ Elegir archivos runBTS.png
• runBTS.png(image/png) - 3570 bytes, last modified: 18/12/2022 - 100% done
Saving runBTS.png to runBTS (6).png
```

TEXTO EXTRAIDO:

```
Get ready, get ready, get ready,
Go get it, go get it, go get it, go.
(if we live fast, let us die young)
```

IMAGEN ORIGINAL

```
Get ready, get ready, get ready,
Go get it, go get it, go get it, go !
(If we live fast, let us die young)
```

+ Código

+ Texto

Recomendaciones



Para trabajar en Colab

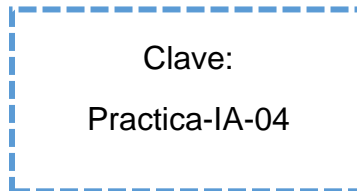
- Aprender a utilizar las celdas en colab le ayudaran a tener una mejor estructura en su código y con una apariencia más profesional
- Para instalar cualquier biblioteca con la que deseamos trabajar hacer la descarga e instalación en una celda previa al código que ejecutaremos.
- Cada vez que hagamos una instalación reiniciar tiempo de ejecución para que las configuraciones sean correctamente leídas
- Colab trabaja con la versión 3 en adelante de Python por lo tanto se recomienda revisar la documentación de cada dependencia que se desee instalar y utilizar en el notebook
- Las funciones incorporadas en colab facilitan mucho el trabajo al momento de escribir código, una de ellas es la función cargar imagen ya que se puede hacer de formo manual, importar desde drive o con líneas de código se recomienda utilizar esta última mencionada para ir tomando practica al escribir programas.

TITULO DE LA PRÁCTICA

MANUAL PROCEDIMENTAL PARA EL DESARROLLO DE LA PRÁCTICA
“SISTEMA DE INFERENCIA DUFUSA PARA EL CONTROL DE VELOCIDADES”

IDENTIFICADOR DE LA PRÁCTICA

El identificador de la práctica es una clave formulada para hacer referencia exacta a la práctica que nos estamos refiriendo, esta sirve también para ser identificada con mayor facilidad en el repositorio digital donde se almacena.



PRESENTACIÓN

Como anteriormente se abordaron temas que describen las técnicas que se utilizan en la inteligencia artificial para poder comprender y crear formas de que la computadora los programas informáticos y softwares puedan en gran medida igual actividades humanas ahora el tema correspondiente es sobre lógica difusa. La lógica clásica clasifica solo dos tipos de valores que normalmente se representan entre cero y uno estos valores son valores booleanos y solo dan opción de tener dos posibilidades en un rango de valores de un todo sin embargo al analizar casos de la vida real o casos cotidianos nos damos cuenta que no es una de las mejores alternativas no es una lógica incorrecta pero tampoco esta adecuada para abordar cualquier caso práctico.

Es así que la lógica difusa tiene un paradigma diferente en los que las variables de entrada pueden tomar valores de cualquier tipo dentro de un universo dado proporcionando un porcentaje a cada valor, al implementarse con la IA su objetivo es de imitar el razonamiento y la cognición humana. La lógica difusa tiene varias aplicaciones en la ingeniería, así como en la minería de datos.

En la presente practica se desarrolla el análisis de un sistema de inferencia difusa que toma como ejemplo de la vida real las velocidades que se maneja en un velocímetro de un automóvil, al desarrollar esta práctica ayudara a comprender mejor los elementos que integran la lógica difusa y que signica cada uno de ellos.

FUNDAMENTACIÓN TEÓRICA

La idea de la lógica difusa fue propuesta por primera vez por Lotfi Zadeh de la Universidad de California en Berkeley en la década de 1960, Zadeh presentó la fuzzy logic como una forma de procesamiento de la información en la que los datos podrían tener asociados un grado de pertenencia parcial a conjuntos. La lógica difusa es un enfoque de la informática basado en "grados de verdad" en lugar de la lógica booleana habitual "verdadero o falso" (1 o 0) en la que se basa la computadora moderna. En los sistemas de inteligencia artificial (IA), la lógica difusa se utiliza para imitar el razonamiento y la cognición humanos. En lugar de casos estrictamente binarios de verdad, la lógica difusa incluye 0 y 1 como casos extremos de verdad, pero con varios grados intermedios de verdad.

Como resultado, la lógica difusa es adecuada para lo siguiente:

- ✚ Ingeniería para decisiones sin certezas e incertidumbres claras, o con datos imprecisos, como con tecnologías de procesamiento de lenguaje natural; y
- ✚ Regular y controlar las salidas de la máquina, de acuerdo con múltiples entradas/variables de entrada, como con los sistemas de control de temperatura.

Como funciona

la lógica difusa se ocupa de conjuntos con definiciones subjetivas o relativas, como "alto", "grande" o "hermoso". Esto intenta imitar la forma en que los humanos analizan los problemas y toman decisiones, de una manera que se basa en valores vagos o imprecisos en lugar de la verdad o falsedad absoluta.

La lógica difusa básicamente funciona en los conjuntos. Hay diferentes tipos de suposiciones en los conjuntos: la salida se obtendrá con precisión a partir de los valores supuestos de los conjuntos. La configuración básica utilizada en la lógica difusa es conjunta if - else. La respuesta precisa se obtendrá de las variaciones de entrada, las consideraciones if - else nos darán el valor exacto como salida.

Las entradas se clasifican en diferentes membresías para obtener las salidas. A las entradas se les asignará la función de pertenencia utilizando la lógica if - else. En la lógica difusa en el sistema de entrada múltiple, hay diferentes tipos de variables de entrada, por lo que la salida del sistema se dará como la operación AND de las variables de entrada dadas.

Esta lógica se usa como un controlador estándar en algunas plataformas. El control de la temperatura del sistema de enfriamiento automático dependerá de la temperatura ambiente, por lo que aquí se considera como entrada una temperatura ambiente diferente y la salida se obtiene utilizando la lógica difusa que produce la salida controlada. Este es el funcionamiento básico de la lógica difusa.

Elementos de la lógica difusa

Reglas

Contiene todas las reglas y las condiciones si-entonces que ofrecen los expertos para controlar el sistema de toma de decisiones. La reciente actualización de la teoría difusa proporciona diferentes métodos efectivos para el diseño y ajuste de controladores difusos. Por lo general, estos desarrollos reducen el número de reglas difusas.

Fuzzificación

Este paso convierte las entradas o los números nítidos en conjuntos borrosos. Puede medir las entradas crujientes mediante sensores y pasarlas al sistema de control para su posterior procesamiento. Divide la señal de entrada en cinco pasos tales como:

LP: X is Large Positive

MP: X is Medium Positive

S: Small

MN: X is Medium Negative

LN: X is Large Negative

Motor de inferencia




Determina el grado de coincidencia entre la entrada difusa y las reglas. Según el campo de entrada, decidirá las reglas que se van a disparar. Combinando las reglas disparadas, forman las acciones de control.

Defuzzificación

El proceso de defuzzificación convierte los conjuntos borrosos en un valor nítido. Hay diferentes tipos de técnicas disponibles, y debe seleccionar la más adecuada con un sistema experto.

Aplicaciones de la lógica difusa

La lógica difusa se utiliza en diversos campos, como sistemas de automoción, bienes domésticos, control ambiental, etc. Algunas de las aplicaciones comunes son:

-  Campo aeroespacial para el control de altitud de naves espaciales y satélites.
-  Controla la velocidad y el tráfico en los sistemas automotrices.
-  Sistemas de apoyo a la toma de decisiones y evaluación personal en el negocio de las grandes empresas.

- ✚ Controla el proceso de pH, secado, destilación química en la industria química.
- ✚ Procesamiento del lenguaje natural y varias aplicaciones intensivas en inteligencia artificial.
- ✚ Sistemas de control modernos, como los sistemas expertos.
- ✚ Eficiencia de combustible en motores.
- ✚ Nevera, lavadora.
- ✚ El piloto automático se basa en Fuzzy_logic. Esto hace que las aplicaciones en el sistema de transporte. Los vehículos autónomos submarinos se trabajan sobre la base de la lógica difusa.
- ✚ También es aplicable en Defensa, como la localización de objetivos, la generación de imágenes infrarrojas para el reconocimiento de objetivos.
- ✚ Principalmente apoya los sistemas de defensa naval.
- ✚ La purificación del agua y el control de calidad se realizan mediante fuzzy_logic. El manejo de datos estructurales también se realiza utilizando esta lógica en los sectores industriales.
- ✚ En el sector médico, se utiliza sobre todo en el diagnóstico de pacientes con cáncer y diabéticos.
- ✚ En muchos sectores de automatización, la lógica difusa se utiliza con fines de simplificación.

DESCRIPCION DE LA PRÁCTICA

Objetivo

Desarrollar el sistema de inferencia difusa para el control de velocidades para la materia de inteligencia artificial.

Competencias Desarrolladas

La presente practica abarca el tema 4 aplicaciones con técnicas de la IA del temario de la asignatura de la materia de Inteligencia Artificial con clave SCC – 1012 de la carrera de Sistemas Computacionales, permitiendo que el alumno identifique los elementos presentes en un sistema de control difuso de acuerdo al tema 4.4 Lógica difusa (Fuzzy Logic) de la materia de inteligencia artificial.

Requisitos de Software



Navegador web



Servicio de Notebook en la nube



Cuenta de Google con servicio de Gmail

Requisitos de Hardware



Portátil o computador de escritorio

Sistema de control difuso

Normalmente cuando hablamos en términos de velocidad podemos clasificar estos en dos “velocidad lenta” y “velocidad rápida” aun que no esta definido concretamente que rangos numéricos son velocidad lenta o rápida cuando hablamos de coches empleamos otros términos, estos van de la mano con la marcha de un vehículo.

En el cambio de marchas manual, debemos escoger nosotros el momento más recomendable en el que cambiar a una marcha más alta o más baja por el contrario el coche automático consigue aprovechar todo el potencial del vehículo en este sentido aun que en la práctica, cada modelo de coche relaciona el motor, la aceleración y el cambio de marchas, para una conducción perfecta. El cambio de marcha y la velocidad en la que esta debe hacerse depende del coche o vehículo sin embargo puede tomarse como guía de medición las siguientes relaciones con respecto a la velocidad en cada marcha.

| CAMBIO DE CAJA MANUAL (5 VELOCIDADES) | | | | |
|---------------------------------------|------------|-----------|-----------|------------|
| RELACIÓN | 1.000 RPM | 1.200 RPM | 2.000 RPM | 4.000 RPM |
| 1 ^a | 8,2 km/h | 9,8 km/h | 16,4 km/h | 32,8 km/h |
| 2 ^a | 16 km/h | 19,2 km/h | 32 km/h | 64 km/h |
| 3 ^a | 25 km/h | 30 km/h | 50 km/h | 100 km/h |
| 4 ^a | 34,8 km/h | 41,2 km/h | 69,6 km/h | 139,2 km/h |
| 5 ^a | 44, 5 km/h | 53,2 km/h | 89 km/h | 178 km/h |



Ilustración 8 Velocidades de la caja manual

Elementos de un sistema difuso

Fuzzificación

Compuesto por valores numéricos de entrada para pasar por la interfaz de Fuzzificación y ser valores lingüísticos.

Sistema de inferencia

Está compuesto por las reglas de inferencia y las funciones de membresía, adjunto a ella está la base de conocimiento.

Base de conocimiento

Contiene la información de la Fuzzificación y la defuzzificación del sistema de inferencia.

defuzzificación

Tiene por salida los valores lingüísticos a valores numéricos

Construcción de valores

Las variables en nuestro ejemplo son las velocidades, tenemos como estándar 5 tipos de cambios en una caja manual por lo que podemos definir:

-  1^{ra} marcha
-  2^{da} marcha
-  3^{ra} marcha
-  4^{ta} marcha
-  5^{ta} marcha

Revoluciones por minuto (RPM) en curvas de par y potencia y potencia del motor con distribución de fase variable, tienen como clasificación RPM bajo – RPM medio – RPM alto.

Tomando estos datos podemos definir las siguientes variables numéricas

Variable a ---> 1 marcha

Variable b ---> 2 marcha

Variable c ---> 3 marcha

Variable d ---> 4 marcha

Variable e ---> 5 marcha

Variables lingüísticas

Velocidad baja ---> 1 marcha

Velocidad normal ---> 2 marcha

Velocidad media ---> 3 marcha

Velocidad alta ---> 4 marcha

Velocidad muy alta ---> marcha

Funciones de membresía

El velocímetro del vehículo alcanza una máxima de 180 km/h lo cual esa será nuestro límite de velocidad por lo tanto nuestro universo será limitado entre todos

los valores encontrados en 0 y 180. En la imagen siguiente se ilustran las velocidades.

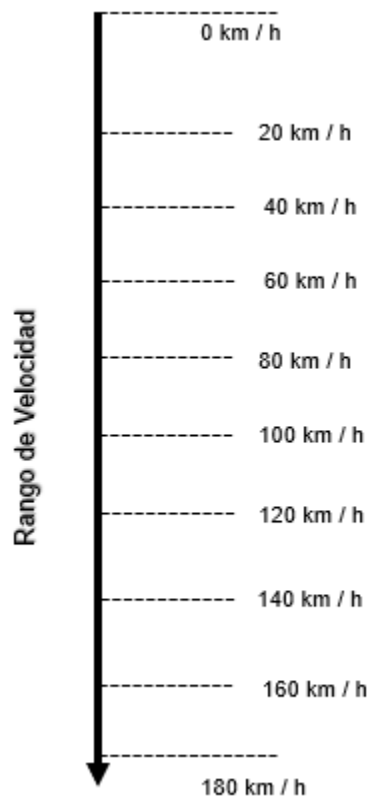


Ilustración 9 Elementos del universo

Ahora definido el universo debemos definir los subconjuntos para cada función de pertenencia.

Subconjunto a: compuesta por todos los valores en los rangos 0 a 32.8 que conforman la primera marcha, a su vez estos pueden clasificarse en RPM bajo, medio y alto.

Subconjunto b: compuesta por todos los valores en los rangos 0 a 64 que conforman la primera marcha, a su vez estos pueden clasificarse en RPM bajo, medio y alto.

Subconjunto c: compuesta por todos los valores en los rangos 0 a 100 que conforman la primera marcha, a su vez estos pueden clasificarse en RPM bajo, medio y alto.

Subconjunto d: compuesta por todos los valores en los rangos 0 a 139.2 que conforman la primera marcha, a su vez estos pueden clasificarse en RPM bajo, medio y alto.

Subconjunto e: compuesta por todos los valores en los rangos 0 a 178 que conforman la primera marcha, a su vez estos pueden clasificarse en RPM bajo, medio y alto.

Las graficaremos con ayuda de Python para tener una mejor vista de los subconjuntos que se definieron anteriormente. Para ello hacemos uso de la librería matplotlib que permite crear los gráficos en Python, así como la librería numpy que ayuda a el manejo numérico de las funciones. La misma biblioteca tiene sus propias propiedades que permiten asignar las etiquetas que queramos y el título de nuestro gráfico.

```
import matplotlib.pyplot as plt
import numpy as np

x = [0,1,8.2,8.2]
y = [0,1,1,0]

x2 = [0,8.3,9.8,9.8]
y2 = [0,1,1,0]

x3 = [0,9.9,16.4,16.4]
y3 = [0,1,1,0]

x4 = [0,16.5,32.8,32.8]
y4 = [0,1,1,0]

plt.plot(x,y)
plt.plot(x2,y2)
plt.plot(x3,y3)
plt.plot(x4,y4)
plt.title("Funcion de membresia - 1ra marcha")
plt.xlabel("% de pertenencia")
plt.ylabel("valores de velocidad")
plt.show()
```

Reglas de inferencia

Las expresiones condicionales para formular las reglas difusas siguen la siguiente estructura valores lingüísticos de entrada – premisa – consecuencia

$$A - B = \text{IF } x \text{ is } A \text{ then } y \text{ is } B$$

R1: si la velocidad alcanzada por el vehículo es baja entonces el vehículo está avanzando en primera marcha

R2: si la velocidad alcanzada por el vehículo es normal entonces el vehículo está avanzando en segunda marcha

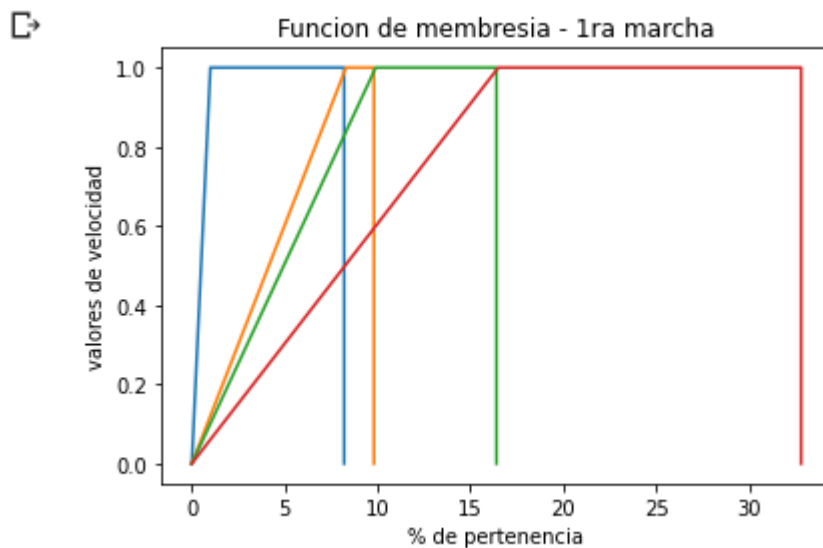
R3: si la velocidad alcanzada por el vehículo es media entonces el vehículo está avanzando en tercera marcha

R4: si la velocidad alcanzada por el vehículo es alta entonces el vehículo está avanzando en cuarta marcha

R5: si la velocidad alcanzada por el vehículo es muy alta entonces el vehículo está avanzando en quinta marcha

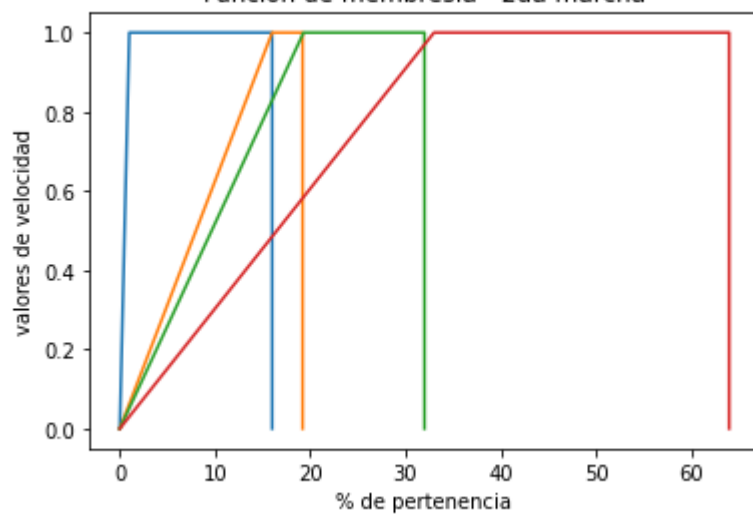
Resultados

Graficamos cada función de pertenencia de cada subconjunto creado en los pasos anteriores de esta manera podemos observar mejor cual es el grado de pertenecía de cada velocidad que pueda existir entre el rango del universo delimitado.

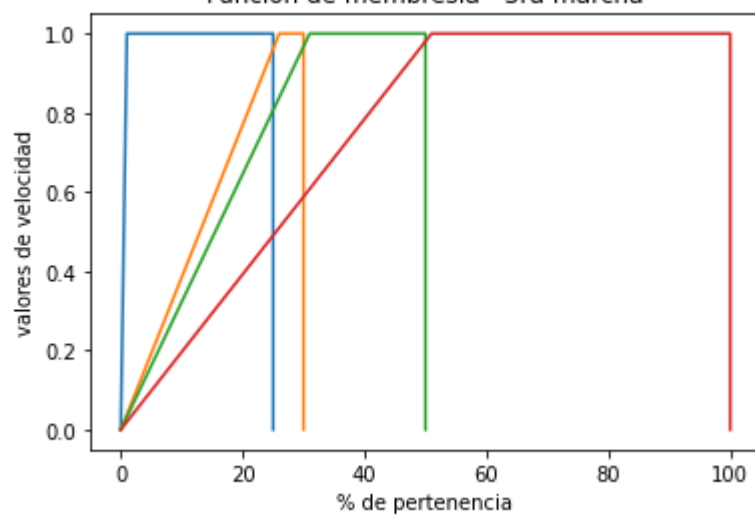


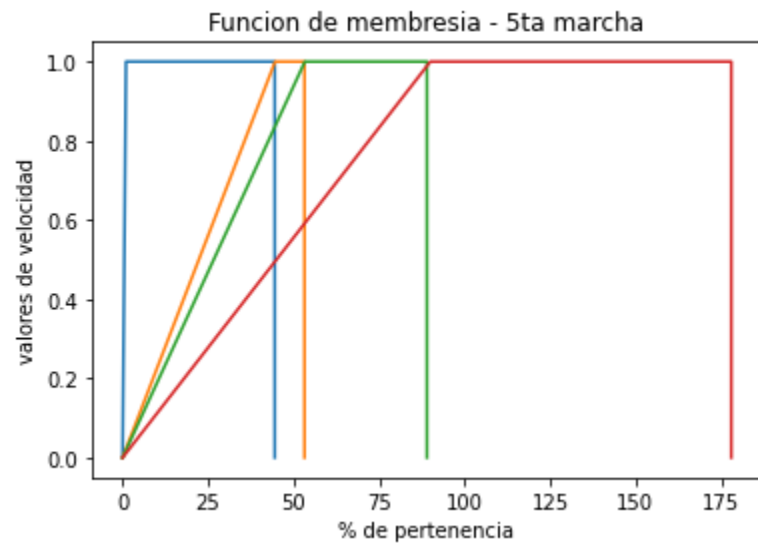
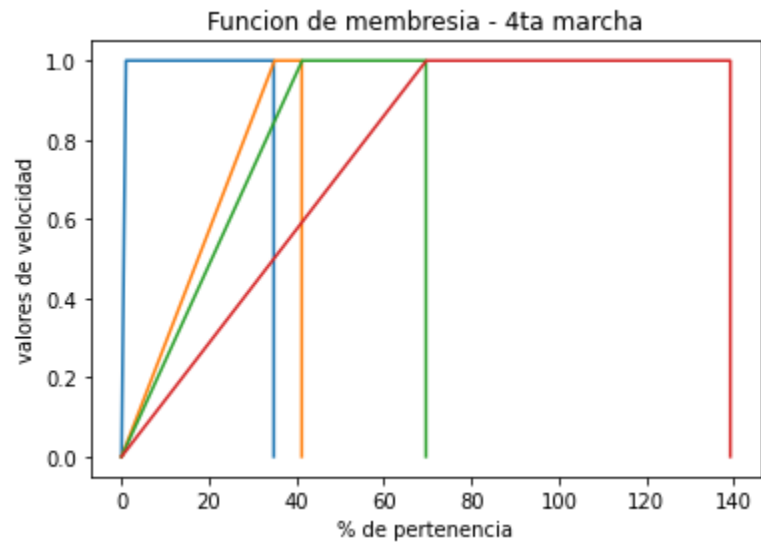


Funcion de membresia - 2da marcha



Funcion de membresia - 3ra marcha





RECOMENDACIONES



Para el análisis de un sistema difuso

- Obtener la información de entrada para definir las variables de un sitio oficial de información o con ayuda de un experto en el tema ya que estos deben ser los mas exactos para continuar con el desarrollo y que los resultados en la defuzzificación sea lo más exacta posible.
- Definir el universo con los rangos de valores que se utilizan en la practica ya que si se agregan valores que no se utilicen pueden llegar a desviar los porcentajes de pertenencias de los valores numéricos reales.

Bibliografías

AWS (2022) ¿Qué es una red neuronal? AWS. <https://aws.amazon.com/es/what-is/neural-network/#:~:text=Las%20c%C3%A9lulas%20del%20cerebro%20humano,humanos%20a%20procesar%20la%20informaci%C3%B3n.>

ATRIA INNOVATION (2021) Aplicaciones de la visión artificial en la industria. ATRIA INNOVATION <https://www.atriainnovation.com/aplicaciones-de-la-vision-artificial-en-la-industria/>

Bello E. (2021) Lógica Difusa o Fuzzy Logic: Qué es y cómo funciona + Ejemplos. IEBS <https://www.iebschool.com/blog/fuzzy-logic-que-es-big-data/#:~:text=la%20Fuzzy%20Logic-,%C2%BFQu%C3%A9%20es%20la%20L%C3%B3gica%20Difusa%20o%20Fuzzy%20Logic%3F,un%20mayor%20n%C3%BAmero%20de%20seguidores.>

Leeuwen Q.S.V & Peláez B. (2022) ¿Qué es el procesamiento de lenguaje natural? Cómo las empresas pueden beneficiarse del PLN. Getapp. <https://www.getapp.es/blog/2403/que-es-pln-procesamiento-de-lenguaje-natural>

Cognex (2022) BENEFICIOS DE LA VISIÓN ARTIFICIAL. Cognex <https://www.cognex.com/es-mx/what-is/machine-vision/benefits#:~:text=La%20visi%C3%B3n%20artificial%20brinda%20beneficios,los%20trabajadores%20de%20ambientes%20peligrosos.>

Cognizant (2022) Procesamiento de lenguaje natural. Cognizant. <https://www.cognizant.com/es/es/glossary/natural-language-processing>

Calvo D. (2017) Clasificación de redes neuronales artificiales. <https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/>

Chai W. (2018) Fuzzy Logic. techTarget. <https://www.techtarget.com/searchenterpriseai/definition/fuzzy-logic>

Deyde DataCentric (2021) Qué es y qué aplicaciones tiene una red neuronal artificial. Deyde DataCentric. <https://www.datacentric.es/blog/insight/red-neuronal-artificial-aplicaciones/#:~:text=Para%20qu%C3%A9%20se%20usa%20una%20red%20neuronal&text=Reconocimiento%20y%20clasificaci%C3%B3n%3A%20Asociaci%C3%B3n%20de,agregaci%C3%B3n%20y%20an%C3%A1lisis%20de%20datos.>

IBERDROLA (2022) ¿Qué es el procesamiento de lenguaje natural y qué aplicaciones tiene?. <https://www.iberdrola.com/innovacion/procesamiento-lenguaje-natural-pln>

IBM CLOUD EDUCATION. (2020) Redes neuronales. <https://www.ibm.com/mx-es/cloud/learn/neural-networks>

INFAIMON (2020) Visión artificial aplicada a la medicina y la industria farmacéutica. INFAIMON. <https://infaimon.com/vision-artificial-medicina-industria-farmaceutica/>

INESDI (2022) ¿Qué son las redes neuronales? Tipos y funciones. INESDI. <https://www.inesdi.com/blog/que-son-las-redes-neuronales/>

INSTITUTO DE INGENIERÍA DEL CONOCIMIENTO (2022) Procesamiento del lenguaje natural ¿qué es? INSTITUTO DE INGENIERÍA DEL CONOCIMIENTO. <https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/>

NA8 (2018) Breve Historia de las Redes Neuronales Artificiales. <https://www.aprendemachinelearning.com/breve-historia-de-las-redes-neuronales-artificiales/>

SICK (2022) ¿QUÉ ES LA VISIÓN ARTIFICIAL? SICK. <https://s.sick.com/es-es-soluciones-de-vision-artificial>

Scott G. (2022) Fuzzy Logic: Definition, Meaning, Examples, and History. Investopedia. <https://www.investopedia.com/terms/f/fuzzy-logic.asp>

Xeridia (2019) La Visión Artificial y el procesamiento de imágenes. Xeridia. <https://www.xeridia.com/blog/la-vision-artificial-y-el-procesamiento-de-imagenes#:~:text=%C2%BFC%C3%B3mo%20nace%20la%20Visi%C3%B3n%20Artificial,en%20el%20tratamiento%20de%20im%C3%A1genes.>