



INSTITUTO POLITÉCNICO NACIONAL
Centro de Innovación y Desarrollo Tecnológico en Cómputo



**Sistema de aprendizaje no supervisado para la detección y
automatización de tareas repetitivas en el entorno de una
computadora**

T E S I S

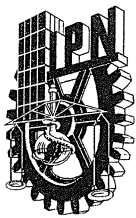
Que para obtener el grado de:
Maestro en Tecnología de Cómputo presenta el

Presenta:
Ing. Ricardo González Tello

Director: Dr. Mauricio Olguin Carbajal
Director: Dr. José Félix Sérrano Talamantes

México, CDMX,

Diciembre de 2018



INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 11:00 horas del día 13 del mes de diciembre del 2018 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del CIDETEC para examinar la tesis titulada:

Sistema de aprendizaje no supervisado para la detección y automatización de tareas repetitivas en el entorno de una computadora"

Presentada por el alumno:

GONZÁLEZ
Apellido paterno

TELLO
Apellido materno

RICARDO
Nombre(s)

Con registro:

A	1	7	0	6	6	5
---	---	---	---	---	---	---


aspirante de:

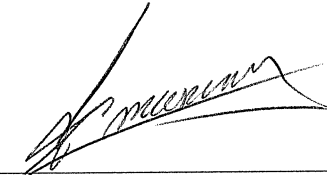
Maestría en Tecnología de Cómputo

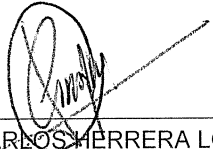
Después de intercambiar opiniones los miembros de la Comisión manifestaron **APROBAR LA TESIS**, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis


DR. JOSÉ FELIX SERRANO TALAMANTES
Primer Vocal


DR. MAURICIO OLGUÍN CARBAJAL
Segundo Vocal


DR. JUAN CARLOS HERRERA LOZADA
Presidente



M. EN C. ISRAEL RIVERA ZÁRATE
Secretario

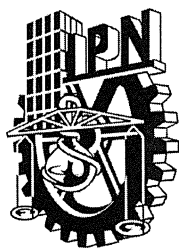

M. EN C. PATRICIA PÉREZ ROMERO
Tercer Vocal


M. EN C. MIGUEL HERNÁNDEZ BOLAÑOS
Suplente

PRESIDENTE DEL COLEGIO DE PROFESORES


DR. ITZAMÁ LÓPEZ YÁÑEZ


S. E. P
INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INNOVACIÓN Y DESARROLLO
TECNOLÓGICO EN COMPUTO

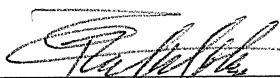


INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 10 del mes Enero del año 2019, el que suscribe Ricardo González Tello alumno del Programa de Maestría en Tecnología de Cómputo con número de registro A170665, adscrito al Centro de Innovación y Desarrollo Tecnológico en Cómputo, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. José Félix Serrano Talamantes y el Dr. Mauricio Olguín Carbajal, y cede los derechos del trabajo titulado *Sistema de aprendizaje no supervisado para la detección y automatización de tareas repetitivas en el entorno de una computadora*, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección rgonzaleztello@live.com.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.


Ricardo González Tello

Resumen

En la presente tesis se da a conocer una metodología de aprendizaje no supervisado, que permite identificar secuencias repetitivas, de una lista de datos nominales en la que los elementos están por orden aparición. Esta herramienta, fue utilizada para identificar las tareas que realiza una persona con movilidad reducida en brazos y manos en una computadora y automatizandolas lo más posible para que la persona pueda aumentar su desempeño. El software fue diseñado como una interfaz gráfica básica, sin embargo, es suficiente, ya que la intención es proveer de un sistema de fácil uso y reducir el número de pasos necesarios para realizar una tarea en el equipo de cómputo.

La metodología mencionada, hace uso de un grafo dirigido el cual se va construyendo mientras el usuario hace uso de la computadora y en el momento se le muestran las secuencias encontradas, para que él identifique si le es útil o no. Las acciones definidas como útiles, son registradas para su posterior uso, permitiendo que el usuario haga uso de estas, mejorando su rendimiento y las sugerencias de tareas que se le muestran él.

Considerando el objetivo de la metodología en este trabajo, se clasifica como una variante de Robotic Process Automation (RPA).

Abstract

In the present thesis an unsupervised learning methodology is presented that allows to identify repetitive sequences of a list of nominal data in which the elements are in order of appearance. This tool was used to identify the tasks performed by a person with reduced mobility in their arms and hands on a computer and automate these tasks as much as possible so that the person can increase their performance. The software was designed as a basic graphic interface, however, it is sufficient, since the intention is to provide an easy-to-use system and reduce the number of steps necessary to perform a task on the computer equipment.

The aforementioned methodology makes use of a directed graph which is constructed while the user makes use of the computer and at the moment the sequences found are shown, so that he can identify if it is useful or not. The actions defined as useful, are registered for their later use, allowing the user to make use of them, improving their performance and the task suggestions that are shown to him.

Considering the objective of the methodology in this work, it is classified as a variant of Robotic Process Automation (RPA).

Agradecimientos técnicos

Principalmente mi familia y amigos, que me han proporcionado un hombro cada que se requirió y una guía para resolver los problemas que se han presentado, así como una inspiración para marcar el camino que he seguido.

Al Instituto Politécnico Nacional y al Centro de Innovación y Desarrollo Tecnológico en Cómputo, le agradezco por haberme permitido llegar a donde estoy, por la formación que me ha proporcionado, mencionaré a cada profesor que se ha convertido en un maestro, pero la verdad son muchos, sobre todo, aquellos que se han ganado mi respeto y admiración. Les puedo decir, que soy el resultado de su dedicación y espero que estén orgullosos mí, así como estoy orgulloso de ustedes y de esta institución.

Al CONACyT, por el apoyo recibido durante mis estudios de maestría y sin el cual no hubiera podido terminar exitosamente mi trabajo.

Índice general

Resumen	VII
Abstract	IX
Agradecimientos técnicos	XI
Índice general	I
Índice de figuras	III
Índice de tablas	IV
1 Introducción	1
1.1. Justificación	2
1.2. Planteamiento del problema	4
1.3. Hipótesis	5
1.4. Propuesta de trabajo	5
1.5. Objetivo general	6
1.6. Objetivos específicos	6
1.7. Metodología	6
1.8. Estructura de la tesis	7
2 Trabajos relacionados	9
2.1. Antecedentes	9
2.2. Estado actual de la ciencia	11
3 Marco teórico	19
3.1. Aprendizaje Máquina	19
3.1.1. RPA	20
3.2. Tipos de programación	21
3.3. Grafos	21

3.3.1. Grafo Dirigido	22
3.3.2. Teoría de árboles	22
3.4. Tipos de datos	24
4 Desarrollo de la investigación	27
4.1. Metodología	28
4.2. Pseudo-código	29
4.3. Desarrollo	35
5 Experimentos y resultados	39
5.1. Resultados de experimentación	39
5.2. Discusión de resultados	44
6 Conclusiones y resultados	47
6.1. Conclusiones	47
6.2. Trabajos a futuro	48
6.3. Productos de la investigación	48
Bibliografía	51

Índice de figuras

1.1. Ejemplo del árbol ideal generado al pasar 20 días de uso de una PC.	5
2.1. Ejecución de un archivo por lotes en Linux.	10
2.2. Interfaz de usuario de PuloVers Macro Creator con una macro de ejemplo proporcionada por el desarrollador[1].	11
2.3. Interfaz de usuario de UIPath[2].	12
2.4. Ambiente del juego de acción[3].	12
2.5. Sistema experimental de la soldadora[4].	13
2.6. Robot humanoide actor en escenario real[5].	13
2.7. Robot virtual en “sophie is kitchen” (La cocina de Sofia) [6].	14
3.1. Ejemplo de un grafo.	22
3.2. Ejemplo de un grafo dirigido.	23
3.3. Ejemplo de un árbol.	24
4.1. Ejemplo del algoritmo con una máquina de Turing multicinta.	28
4.2. Ejemplo del algoritmo con un grafo dirigido.	28
4.3. Diagrama de flujo del algoritmo general.	30
4.4. Diagrama de flujo del bloque “identificar secuencia” (ver figura 4.3).	31
4.5. Ventana principal con el asistente y una lista de tareas guardadas.	37
4.6. Ventana mostrando una tarea encontrada.	37
5.1. Grafo generado por el sujeto <i>número 3</i> solo con acciones del teclado.	41
5.2. Ampliación al grafo generado por el sujeto <i>número 3</i> solo con acciones del teclado.	42
5.3. Ampliación para la correcta visualización de los nodos en el grafo generado por el sujeto <i>número 3</i> solo con acciones del teclado.	42

Índice de tablas

1.1. Porcentaje de la población con discapacidad, por grupo de edad según tipo de discapacidad 1ra parte[7].	3
1.2. Porcentaje de la población con discapacidad, por grupo de edad según tipo de discapacidad 2da parte[7].	3
1.3. Distribución porcentual de las discapacidades, por tipo según causa de la discapacidad[7].	4
2.1. Tabla de requisitos y análisis comparativo	15
5.1. Información de los datos recabados.	40
5.2. Tabla de resultados con secuencias de una longitud mínima de 1 acción.	43
5.3. Tabla de resultados con secuencias de una longitud mínima de 2 acciones.	43
5.4. Análisis comparativo del software con un generador de macros.	45
5.5. Análisis comparativo de la propuesta con Robotic Process Automation.	46

Capítulo 1

Introducción

Como nos ha marcado la experiencia, la computadora al igual que cualquier otra máquina fue diseñada para facilitar la vida de las personas con tareas repetitivas, ya sea acelerando o automatizandolas, así mismo se ha llegado a un punto en la operación de la computadora en la que se realizan actividades de forma mecanizada, ya que no hay variantes en estas.

Los desarrolladores de software han contribuido a la automatización de estas tareas, sin embargo, cuando el software no es específico para una persona sino para un sector de la población, las necesidades llegan a ser variadas de un usuario a otro lo que genera un software con múltiples funciones de las cuales cada usuario usa un conjunto diferente, por lo tanto, mientras más genérico se quiere hacer un software, más complicado será su uso.

Algunos de los desarrollos enfocados a la automatización de acciones humanas con las variantes involucradas en el mundo real, principalmente, son aplicaciones de la robótica, sin embargo, las soluciones propuestas también pueden ser enfocadas a un ambiente virtual.

La propuesta presentada va enfocada al apoyo de las personas con acceso a la computadora, pero que debido a sus capacidades físicas no pueden usar el equipo con la misma agilidad que una persona con todas sus facultades. Estas son las Personas con Movilidad Reducida(PMR), que principalmente, por cuestiones laborales tienen que trabajar con una computadora y que por su discapacidad se les dificulta el uso de la misma.

Por lo tanto, en este trabajo de investigación se propone desarrollar un sistema que permita el monitoreo de las acciones del usuario realizadas en una computadora personal (PC, Personal Computer) y obtener la secuencia de acciones frecuentes para su posterior reproducción.

1.1. Justificación

“Siguiendo el criterio del grupo de Washington, se identifica a la población con discapacidad como aquella que declara no poder hacer, o tener dificultades graves para realizar actividades consideradas básicas” [7]

La Organización Mundial de la Salud (OMS) con el objetivo de establecer un “lenguaje unificado y estandarizado” [8] entre otros, categorizó las discapacidades en el documento CIF (clasificación internacional del funcionamiento, la discapacidad y la salud), en tres aspectos principales [8]:

- Funciones y estructuras corporales
- Actividades
- Participación

A partir de las cuales se obtuvieron los siguientes tipos y porcentajes de discapacidad en México en el año 2014 [7]:

- Caminar, subir o bajar usando sus piernas: 64,1 %
- Ver (aunque use lentes): 58,4 %
- Aprender, recordar o concentrarse: 38,8 %
- Escuchar (aunque use aparato auditivo): 33,5 %
- Mover o usar sus brazos o manos: 33,0 %
- Bañarse, vestirse o comer: 23,7 %
- Problemas emocionales o mentales: 19,6 %
- Hablar o comunicarse: 18,0 %

“En 2014, residían en el país aproximadamente 120 millones de personas . . . La prevalencia de la discapacidad en México para 2014 es de 6 %, según los datos de la ENADID 2014. Esto significa que 7.1 millones de habitantes del país no pueden o tienen mucha dificultad para hacer alguna de las ocho actividades evaluadas” [7]

Con los datos obtenidos en 2014 por la ENADID (Encuesta Nacional de la Dinámica Demográfica) se concluye que “. . . la población con discapacidad muestra la estrecha relación de esta condición con el proceso de envejecimiento demográfico” [7], esto se puede observar en las tablas 1.1 y 1.2 con el porcentaje de cada grupo de personas con discapacidad según el tipo de discapacidad.

Entre causales cuantificadas se encuentran las siguientes con sus respectivos porcentajes:

Tabla 1.1: Porcentaje de la población con discapacidad, por grupo de edad según tipo de discapacidad 1ra parte[7].

Grupo de edad	Tipo de discapacidad			
	Caminar, subir o bajar usando sus piernas	Ver (aunque use lentes)	Mover o usar sus brazos o manos	Aprender, recordar o concentrarse
Niños (0 a 14 años)	36,2 %	26,9 %	14,1 %	40,8 %
Jóvenes (15 a 29 años)	32,1 %	44,6 %	18,2 %	31,5 %
Adultos (30 a 59 años)	56,2 %	58,2 %	28,5 %	32,1 %
Adultos mayores (60 años y mas)	81,3 %	67,2 %	42,7 %	44,6 %
Total	64,1 %	58,4 %	33,0 %	38,8 %

Tabla 1.2: Porcentaje de la población con discapacidad, por grupo de edad según tipo de discapacidad 2da parte[7].

Grupo de edad	Tipo de discapacidad			
	Escuchar(aunque use aparato auditivo)	Bañarse, vestirse o comer	Hablar o comunicarse	Problemas emocionales o mentales
Niños (0 a 14 años)	13,4 %	37,4 %	45,6 %	26,6 %
Jóvenes (15 a 29 años)	18,5 %	16,4 %	28,5 %	28,0 %
Adultos (30 a 59 años)	24,2 %	14,5 %	13,4 %	20,1 %
Adultos mayores (60 años y mas)	46,9 %	29,3 %	14,0 %	16,3 %
Total	33,5 %	23,7 %	18,0 %	19,6 %

- Enfermedad: 41,3 %
- Edad avanzada: 33,1 %
- Nacimiento: 10,7 %
- Accidente: 8,8 %
- Violencia: 0,6 %
- Otra causa: 5,5 %

El apartado de *Otra causa* es para cuando la causa es por factores ambientales y contextuales. En la tabla 1.3 se puede apreciar cual es la causa mas probable de alguna discapacidad en específico, por ejemplo, para *escuchar (aunque use aparato auditivo)* la causa mas probable es por *edad avanzada*.

Con referencia a los datos obtenidos del Censo de Población y Vivienda de 2010 era poco más del 5 % de la Población de México la que presentaba algún

Tabla 1.3: Distribución porcentual de las discapacidades, por tipo según causa de la discapacidad[7].

Tipo de discapacidad	Causa de la discapacidad					
	Enfermedad	Edad Avanzada	Nacimiento	Accidente	Violencia	Otra causa
Caminar, subir o bajar usando sus piernas	49,0 %	25,1 %	5,8 %	16,2 %	0,3 %	3,0 %
Ver (aunque use lentes)	44,3 %	36,7 %	9,1 %	5,6 %	0,2 %	4,1 %
Mover o usar sus brazos o manos	47,8 %	29,2 %	6,1 %	14,1 %	0,5 %	2,3 %
Aprender, recordar o concentrarse	27,5 %	48,7 %	13,2 %	3,3 %	1,0 %	6,3 %
Escuchar (aunque use aparato auditivo)	28,9 %	49,6 %	9,3 %	6,3 %	0,8 %	5,1 %
Bañarse, vestirse o comer	45,6 %	25,9 %	10,1 %	9,5 %	0,4 %	8,5 %
Hablar o comunicarse	34,6 %	19,9 %	31,8 %	3,6 %	0,6 %	9,5 %
Problemas emocionales o mentales	45,5 %	16,9 %	18,1 %	4,2 %	2,4 %	12,9 %
Total	41,3 %	33,1 %	10,7 %	8,8 %	0,6 %	5,5 %

tipo de discapacidad, pero se puede apreciar que esta cifra va en aumento ya que en la Encuesta Nacional de Ingresos y Gasto de los Hogares (ENIGH) de 2012 fue el 6.6 % de la población la que tenía alguna discapacidad[9].

A nivel mundial, la discapacidad va en aumento dado que la población está envejeciendo y son pocos los programas privados y gubernamentales que apoyan a este grupo de personas[10], la Organización Mundial de la Salud y el Banco Mundial propusieron en 2011 [10] una estrategia de colaboración entre el sector privado y gubernamental para rehabilitar e incorporar a la sociedad a las personas discapacitadas y así poder aprovechar el potencial de toda esta gente. Algunas de estas propuestas tienen por objetivo proporcionar accesibilidad en los servicios convencionales, por ejemplo transporte y educación, así como adiestrar a los servidores públicos, para que las personas sean tratadas con los cuidados necesarios.

1.2. Planteamiento del problema

La población de Personas con Movilidad Reducida (PRM) va en aumento en México y con el uso de la computadora como algo imprescindible en la actualidad, la discapacidad de estas personas puede representar un obstáculo en su desarrollo laboral.

1.3. Hipótesis

El ser humano es un ser de costumbres, por lo que hay tareas repetitivas que son automatizables, por tal, el uso del software a desarrollar permitirá al usuario de la PC agilizar este tipo de tareas, ayudando con un manejo más ágil del equipo.

1.4. Propuesta de trabajo

Como un apoyo a las personas cuya discapacidad representa un obstáculo para interactuar de manera ágil y precisa con las computadoras de escritorio, se plantea desarrollar un sistema de reconocimiento de patrones, con la capacidad de reproducir las acciones que tengan mayor incidencia de uso.

La propuesta presentada se enfoca en utilizar una estructura de datos en árbol, en la cual se van a almacenar las acciones que el usuario realice, para posteriormente cuantificar las incidencias de cada rama, esto entregará las secuencias útiles. Una vez detectada, se le solicitará al usuario un nombre para esta secuencia por el cual pueda ser invocada posteriormente.

En la figura 1.1 se muestra un ejemplo ideal del árbol esperado después de 20 días de uso de una PC por la misma persona, al realizar sus actividades cotidianas, donde cada camino, desde el nodo raíz hasta el nodo hoja representa las acciones realizadas por el usuario durante un día.

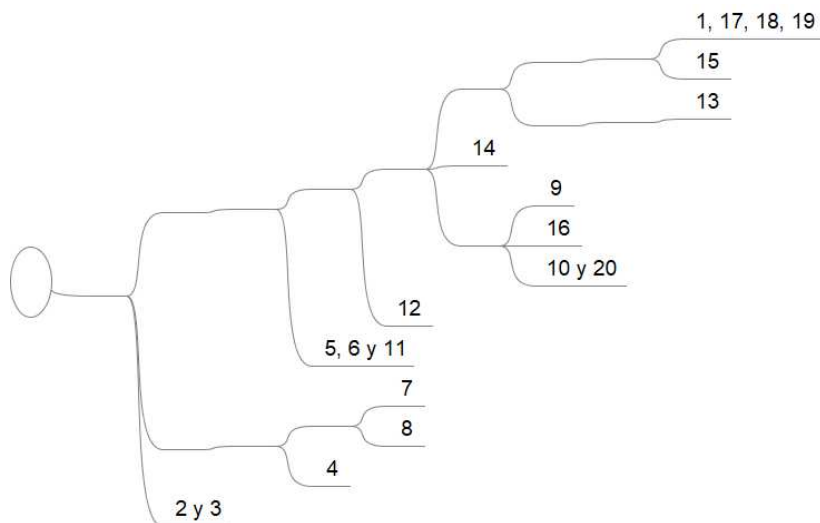


Figura 1.1: Ejemplo del árbol ideal generado al pasar 20 días de uso de una PC.

1.5. Objetivo general

Diseñar y desarrollar un software que defina a partir de un periodo de tiempo determinado el conjunto de acciones con mayor incidencia de uso por un usuario realizadas en una computadora por un usuario, para su uso posterior.

1.6. Objetivos específicos

- Desarrollar un sistema para la captura de acciones, tanto del ratón como del teclado.
- Crear e implementar un árbol para resolver el problema.
- Obtener una muestra de las acciones realizadas con el teclado y el ratón por un usuario en una computadora.
- Diseñar y desarrollar el algoritmo para la determinación de tareas repetitivas.

1.7. Metodología

Con el propósito de desarrollar y lograr los objetivos propuestos en el presente trabajo de investigación, se plantearon las siguientes metas:

Metas

- Investigación de sistemas similares.
- Recopilación de resultados de sistemas similares.
- Desarrollo del sistema de captura de datos.
- Obtención de los datos de ejemplo.
- Desarrollo del sistema de procesamiento de datos.
- Realizar pruebas del sistema.
- Realizar un análisis comparativo de los resultados.

1.8. Estructura de la tesis

La tesis presentada está dividida en 6 capítulos, los cuales se describen brevemente a continuación:

- El **capítulo 1** es la presente introducción, exponiendo, las bases en las que esta cimentada la tesis, incluyendo desde el problema hasta el primer acercamiento a la solución.
- En el **capítulo 2**, se presentan aquellos trabajos que tienen similitud con el desarrollo presentado y se analiza la similitud entre ellos con la propuesta.
- El **capítulo 3**, presenta la teoría necesaria para diseñar y desarrollar el software.
- En el **capítulo 4**, se explica desde una analogía con la máquina de Turing hasta el desarrollo de la aplicación en Python.
- El **capítulo 5**, presenta las pruebas y los resultados, el análisis de los datos obtenidos de personas que utilizan sus propios equipos de cómputo para realizar sus tareas cotidianas, así como la discusión de dichos resultados.
- Finalmente, en el **capítulo 6**, se exponen las conclusiones respecto al trabajo y las posibles mejoras y desarrollos futuros.

Capítulo 2

Trabajos relacionados

En este capítulo se muestran los trabajos relacionados al proyecto propuesto en este documento y una breve descripción de estos, se podrá observar una diferencia entre la temática de los trabajos mostrados; empezando por aquellos que tienen por objetivo ayudar a los discapacitados, posteriormente los trabajos cuyo objetivo es que un robot mecánico o virtual realice tareas de manera similar a como las haría un ser humano.

2.1. Antecedentes

Desde el Microsoft Windows 3,11 [11] hasta la actualidad las Opciones de accesibilidad de Microsoft Windows han permitido que el sistema operativo sea posible usarlo sin importar las condiciones físicas del usuario [12]. A continuación, se menciona una breve descripción de estas opciones.

- Lupa: Aumenta el tamaño del contenido de la pantalla para que sea más fácil leerlo [13].
- Narrador: Esta función es una ayuda auditiva que ayuda a saber cuáles son las ventanas abiertas y su contenido, por medio de una voz sintetizada [13].
- Teclado en pantalla: Como su nombre lo indica es un teclado virtual con el cual podemos escribir presionando la pantalla, en caso de ser un dispositivo táctil, o presionando los botones con el ratón [13].
- Contraste alto: Otra ayuda visual para poder distinguir mejor los elementos en pantalla modificando el contraste de Windows [13].

- Reconocimiento de voz: Es una herramienta de dictado con la cual se puede escribir y manipular aplicaciones e incluso el mismo sistema operativo por medio de comandos de voz [14].

Aunque no está considerado dentro de las opciones de accesibilidad de Windows, el asistente *Microsoft Cortana* facilita la realización de algunas tareas por medio de comandos de voz, por ejemplo, apertura de programas, la manipulación de recordatorios, mensajes de texto y correo electrónico [15].

Los archivos por lotes (Batch o Script Shell) [16] son aquellos que contienen instrucciones para el sistema operativo en formato ASCII por lo que son dependientes de él, por lo general tienen la extensión `.bat` o `.sh`, sin embargo, para el caso de UNIX esto no es obligatorio. En la figura 2.1 se muestra el código de un archivo por lotes y la ejecución en consola.

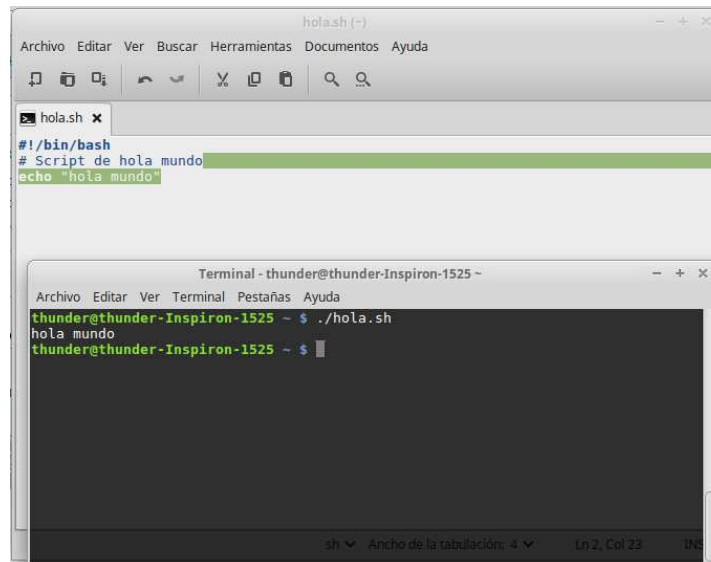


Figura 2.1: Ejecución de un archivo por lotes en Linux.

Pulover's Macro Creator[1], desarrollado y mantenido principalmente por Rodolfo U. Batista, es una herramienta de automatización y creación de scripts basada en el lenguaje “AutoHotKey”. Este creador de macros facilita la tarea de la creación del script por medio de su interfaz gráfica ó con la grabadora de macros que proporciona. Entre sus características destaca el proporcionar control de ventanas en segundo plano y sentencias de control(ciclos y condicionales). En la figura 2.2 se puede observar la interfaz de usuario del software.

Automatización de Procesos Robóticos (Robotic Process Automation, RPA) [17] es una metodología que tiene por objetivo la automatización de procesos repetitivos usando robots de software, haciendo uso de procesamiento de imágenes, se identifican los elementos de la interfaz gráfica con los que interactúa el usua-

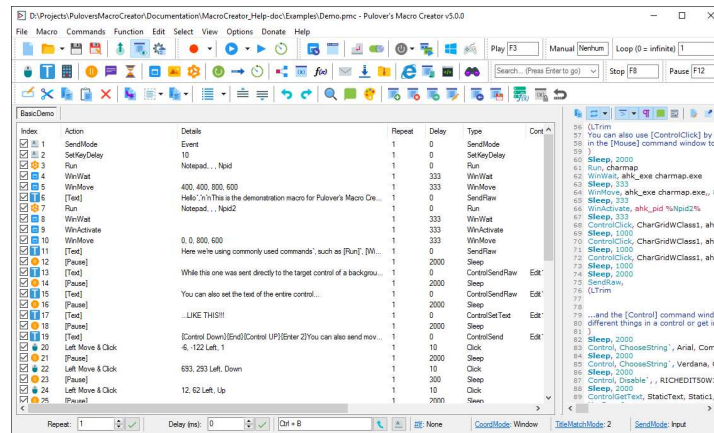


Figura 2.2: Interfaz de usuario de Pulovers Macro Creator con una macro de ejemplo proporcionada por el desarrollador[1].

rio, y un monitor de los dispositivos de entrada y salida para identificar la acción realizada por él[17]. Sin embargo, al haber un cambio en la interfaz gráfica de usuario, del software utilizado, el RPA no lo puede procesar, por lo que tampoco puede realizar la tarea programada, considerando este problema, una posible solución es la mezcla de RPA con técnicas de inteligencia artificial y aprendizaje máquina, permitiendo una mejora y adaptación a los cambios en el proceso ya establecido[18, 19].

El principal uso para el RPA son los procesos empresariales de fianzas y contabilidad, banca, mercados capitales y seguros[19] ya que se realizan en computadoras, sin embargo, no se ve limitado a esto, ya que también se están explorando otras aplicaciones, por ejemplo, supervisión de campos de agricultura con drones[20].

El software UIPath Studio es un software que ofrece una interfaz gráfica para automatizar las tareas con RPA facilitando la implementación de este[2]. En [21] muestran una comparativa entre UIPath Studio, Automation Anywhere y Blue Prism, desarrollos de software que permiten la implementación de RPA, con la intención de que el lector pueda elegir la mejor herramienta dependiendo de sus necesidades. En la figura 2.3 se muestra la interfaz gráfica de UIPath.

2.2. Estado actual de la ciencia

En [22] se presenta el proyecto de la automatización de una silla de ruedas VAHM3, en la que se instaló un sistema con 3 agentes cuyo objetivo es aprender las rutas por las que pasa el usuario de la silla y con apoyo de un mapa topológico ubicar la silla y ayudar a realizar la ruta que el usuario desee, esto usando un algoritmo de condensación.

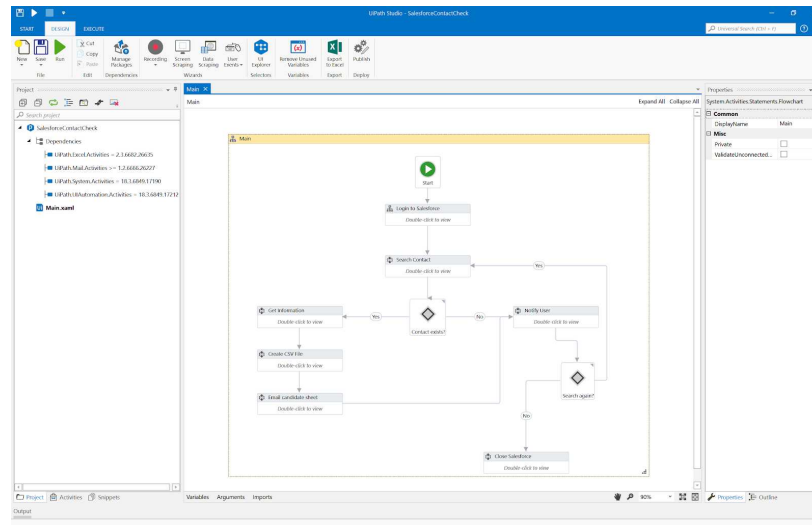


Figura 2.3: Interfaz de usuario de UiPath[2].

Un trabajo desarrollado por la Universidad de Tsukuba en Japón[3], tiene por objetivo el crear un oponente virtual al nivel de un oponente humano que represente un reto para el jugador. Para lograr esto se crearon perfiles con las estrategias de los jugadores y posteriormente se reproducen en otra partida, en la figura 2.4 se muestra el entorno del juego.

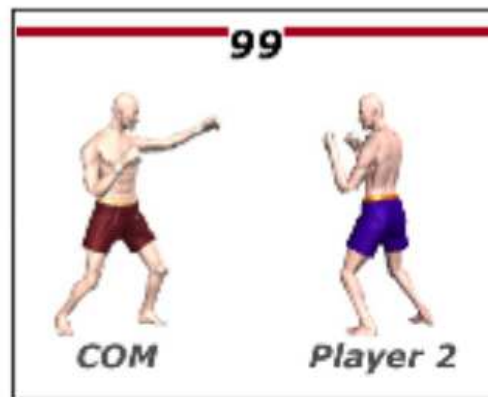


Figura 2.4: Ambiente del juego de acción[3].

En el trabajo desarrollado en la Universidad Tecnológica de Lanzhou en China [4], hacen un análisis del comportamiento del soldador experto, utilizando un sistema de inferencia neurodifuso adaptativo (ANFIS, por sus siglas en inglés) para su automatización, considerando las variables de los materiales usados y caracterizando la tarea del soldador humano, En la figura 2.5 se puede apreciar el sistema experimental resultante del proyecto mencionado.

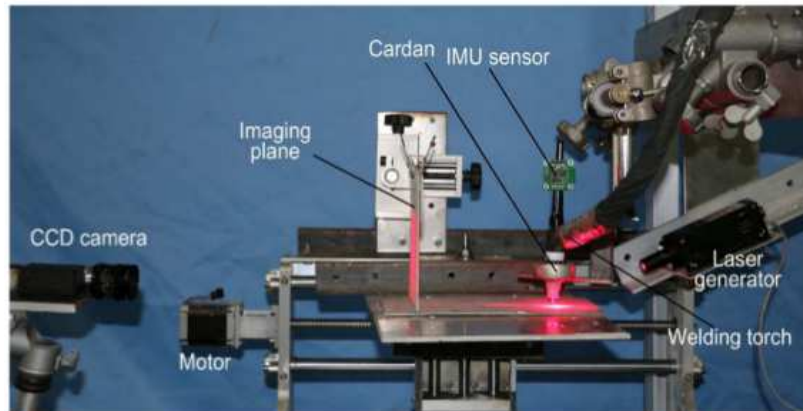


Figura 2.5: Sistema experimental de la soldadora[4].

En el ambiente artístico [5], el trabajo desarrollado en Japón por la Universidad de artes de Tokyo y la Universidad de Osaka, cuyo objetivo era brindar un comportamiento natural humano a un Robot Humanoide. Para cumplir esto utilizaron el conocimiento del director de escena Hirata, que dada la precisión en sus instrucciones a los actores, facilita la traducción de esas órdenes a las reglas para el robot humanoide, además, desarrollaron una interfaz de usuario para que el manejo del robot sea más sencillo, ayudando a los principiantes, ya que proporcionan menos datos se puede obtener buenos resultados. Como se puede observar en la figura 2.6 el robot llegó a desempeñar un amigo del personaje principal en la obra “Night on the milky way train” (El tren nocturno de la vía láctea) en un escenario real.



Figura 2.6: Robot humanoide actor en escenario real[5].

El trabajo desarrollado por la Universidad de Plymouth, Universidad de Lincoln ambas en el Reino Unido y la Universidad de Gante en Bélgica [6], realiza un análisis comparativo de su método SPARC (Supervised Progressively Autonomous Robot Competencies) con el IRL (Interactive Reinforcement Learning), estos dos métodos se basan en el aprendizaje automático de un robot, haciendo que un ser humano con conocimiento del tema apruebe la actividad que está realizando o que va a realizar el robot. Ambos sistemas fueron probados en un ambiente virtual nombrado “sophie is kitchen” (La cocina de Sofia) cuyo objetivo es hornear un pastel, la figura 2.7 muestra al robot en la cocina con los materiales necesarios para realizar la tarea.



Figura 2.7: Robot virtual en “sophie is kitchen” (La cocina de Sofia) [6].

En [23], L. Qiao et al. presenta una revisión de metodologías cuyo objetivo es analizar las propiedades valiosas o minar patrones informativos en grafos, ya que en los problemas con espacios de alta dimensionalidad; con diferentes distribuciones, con ruido en los datos o incertidumbre, se necesitan construir grafos antes de cualquier análisis o tarea de aprendizaje.

Para la construcción de un grafo a partir de un conjunto de vértices o nodos [23], primero se determina la estructura topológica del grafo, para lo cual se pueden hacer uso de las siguientes metodologías que se dividen en tres grupos; el primero son los algoritmos libres de parámetros, los cuales son, árbol de expansión mínimo (minimum spanning tree, MST), triangulación de Delaunay (Delaunay triangulation, DT), grafo de Gabriel (Gabriel graph, GG) y gráfico de vecindad relativa (Relative neighborhood graph, RNG). El segundo grupo incluye esqueleto beta (Beta-skeleton), K vecinos más cercanos (K nearest neighbors, KNN), vecindario de la bola- ϵ (ϵ -ball neighborhood, ϵ -N) y concordancia B (B-matching) y en el tercer grupo se incluyen aquellos con parámetros flexibles.

Después considerando el conjunto de bordes [23] o aristas obtenido se determina la matriz de pesos por una metodología de las mencionadas en las siguientes dos categorías. La primera es por definición directa entre las cuales están; peso binario (Binary weight), Kernel Gaussiano (Gaussian kernel) y sus variantes, inversa de distancia (Inverse of distance), correlación completa (Full correlation) y correlación parcial (Partial correlation).

La otra categoría es, aprendizaje automático [23] en la cual se encuentran englobadas las siguientes siete metodologías; aprendizaje de grafos paramétricos (por ejemplo, ponderación de borde adaptativa (Adaptive Edge Weighting, AEW)), aprendizaje de grafos induciendo la localidad (por ejemplo, Reconstrucción lineal local (Local linear reconstruction, LLR)), aprendizaje de grafos basado en reconstrucción global (por ejemplo, ajuste de gráfico a vector (Fitting a Graph to Vector, FGV)), aprendizaje de grafos induciendo la escasez (por ejemplo, grafo L_1), Aprendizaje de grafos de bajo rango (por ejemplo, Markov random walk (MRW)), Aprendizaje multígrafo y finalmente el aprendizaje de grafos en un espacio transformado adaptable (por ejemplo, Modelo de conservación de la localidad optimizado para grafos (graph- optimized locality preserving model, GoLP)).

El software que ayude a solucionar el problema planteado; debe de tener interacción con todos los programas del usuario, sin importar cual sea, dado que se desconoce el software que él pueda llegar a ocupar. Debe ser de fácil uso, considerando el grupo de personas a las que va dirigido, además, la intención es agilizar las tareas realizadas, no el darle mas trabajo al usuario. El software debe de tener la capacidad de automatizar varias tareas, mientras más se automaticen, mejor. Tomando en consideración estos requisitos, se realizó la tabla 2.1 en la que se someten a evaluación los proyectos ya presentados en este capítulo.

Tabla 2.1: Tabla de requisitos y análisis comparativo

Nombre del autor o del proyecto	interacción con todos los programas del usuario	Facilidad de uso	Metodología empleada	Automatizar acciones	Múltiples tareas objetivo
Microsoft Cortana	Solo de Microsoft	Si	Patentado	Si	Si
Archivo por lotes	Solo el Sistema Operativo	Si	Scripts	Si	Si
Pullover's Macro Creator	Si	Si	Scripts	Si	Si
A. Nakano, et al.	No	Si	No especifica	Si	No
ANFIS	No	Si	Sistema de Inferencia Neurodifuso Adaptativo	Si	No
Shogo Nishiguchi, et al.	No	Si	Interprete	No	Si
SPARC	No	Si	Aprendizaje por refuerzo	Si	No
VAHM3	No	Si	algoritmo de condensación	Si	No
UIPath	Si	Si	Robotic Process Automation	Si	Si
Propuesta	Si	Si	Grafo	Si	Si

Empezando por Microsoft Cortana, su uso se expande a oraciones habladas o escritas aceptando lenguaje coloquial, pero con ordenes limitadas a la interacción con el software de la misma compañía, sin embargo, se le pueden dar órdenes para agendar reuniones o eventos, e incluso enviar mensajes de correo electrónico o texto con comandos de voz.

Un archivo por lotes permite la automatización de rutinas repetitivas que el usuario realiza en el sistema operativo, por ejemplo, revisar si hay actualizaciones disponibles y abrir algún archivo en específico, pero para esto es necesario que inicialmente el usuario o algún experto programe el script, posteriormente el usuario solo debe ejecutar dicho script y se realizaran las tareas programadas.

En el creador de macros, como Pulover's Macro Creator, se tiene una interfaz y una grabadora de macros que facilitan la creación de un script, incluso este es posible que manipule aplicaciones de distintitos desarrolladores, así como la operación de ventanas en segundo plano, pero el desarrollo del script más allá de lo entregado por la grabadora, requiere un amplio conocimiento por parte del usuario o incluso de un experto, claro está, una vez desarrollado el script el uso de este, es fácil.

Otro software que permite la automatización de tareas repetitivas es UIPath, el cual utiliza la metodología de RPA, para automatizar procesos realizados sobre cualquier software, permitiendo la manipulación de aplicaciones sin importar el desarrollador, sin embargo, al igual que el creador de macros, se requiere de un experto en el software para la automatización de procesos completos.

Nuestra propuesta, a diferencia de las anteriores; obtiene múltiples tareas que se podrían considerar tareas objetivo, la metodología implementada es relativamente simple ya que en esencia es un grafo dirigido, y provee de interacción con todos los programas que el usuario requiera sin realizar cambio al software. En la tabla 2.1, se observa que los proyectos que tienen mayor similitud son, *Microsoft Cortana*, un *archivo por lotes*, *UIPath (o RPA)* y el *Pulover's Macro Creator*, sin embargo, considerando estos requisitos, se tienen que considerar las siguientes diferencias respecto a la propuesta planteada:

- La propuesta, va a encontrar las tareas objetivo, mientras que a las demás metodologías y sistemas, se les debe indicar cual es la tarea objetivo.
- La propuesta va a ser genérica, ya que esta va enfocada a un sector de la población donde se puede hacer uso de cualquier tipo de software, sin embargo, para la mayoría de los trabajos mencionados, se debe conocer de antemano el software con el cual va a interactuar el usuario, a excepción de UIPath y Pulover's Macro Creator, ya que son sistemas genéricos que permiten ser configurados.

- La propuesta debe ser de fácil uso, la intención es facilitar el uso de la computadora, por lo que la persona debería interactuar lo mínimo posible con el software desarrollado. La mayoría de los trabajos investigados, requieren de un experto en el sistema de automatización para que este sea configurado correctamente, Pullover's Macro Creator, permite grabar la secuencia de acciones para llevar a cabo la tarea y posteriormente reproducirla, sin embargo, esto solo aplica a tareas sencillas, ya que se requiere de un conocimiento más profundo del software para realizar la edición de la secuencia grabada o incluso la creación de una.

Por lo mencionado anteriormente, se determina que el desarrollo expuesto en la presente tesis es una alternativa o una posible mejora del RPA y los creadores de macros.

Capítulo 3

Marco teórico

3.1. Aprendizaje Máquina

Como se menciona en [24] “En el sentido más amplio, cualquier método que incorpore información de ejemplo para el entrenamiento en el diseño de un clasificador emplea aprendizaje”. Lo cual es evidente en el aprendizaje supervisado ya que en este existe un maestro explícito que proporciona la información etiquetada [24], para que sirva de ejemplo al algoritmo de los casos desconocidos que se le presentaran más adelante. Pero en el aprendizaje no supervisado, no requiere un maestro explícito [24], sin embargo, se le proporciona información sobre la tarea, por ejemplo, el número de grupos en los que hay que separar la información proporcionada.

En los textos [24, 25] se menciona el aprendizaje por refuerzo (RL, por sus siglas en inglés) como otra división, ya que a éste se le proporciona un conjunto de políticas de recompensa para cumplir con un objetivo específico, en lugar de ejemplos etiquetados como en el caso del aprendizaje supervisado, estas reglas indicaran de forma binaria cuales acciones le permite maximizar la recompensa por prueba y error.

El aprendizaje por demostración (LfD, por sus siglas en inglés) [26] es una metodología de aprendizaje máquina, que a partir del ejemplo de la tarea objetivo (proporcionado por un maestro o experto) se generan las políticas necesarias, para que por medio del aprendizaje por refuerzo, el aprendiz realice la tarea.

Una de las principales características de esta última metodología mencionada, es la forma de adquisición de datos; tele-operación o sombreado. Para la tele-operación[26], el aprendiz va a grabar desde sus propios sensores mientras que el maestro va a manipularlo desde un controlador o con sus propias manos, esto proporciona un mapeo directo entre la acción grabada y la acción a realizar. Por otro lado, el sombreado[26] requiere un algoritmo adicional de mapeo, ya que esté

seguirá las acciones realizadas por el maestro por medios visuales y marcadores en el maestro, mientras graba de sus propios sensores.

La forma de procesar los datos recabados en la adquisición de datos para obtener las políticas de recompensa para el RL puede ser por: función de mapeo, plan o modelo del sistema. El usar una función de mapeo [26] puede ser en un enfoque continuo (Regresión) o discreto (Clasificadores); los algoritmos de regresión pueden ser a modo de ejemplo, aprendizaje lento (lazy learning, en inglés) o regresión ponderada localmente (locally weighted regression, en inglés); los clasificadores usados para el enfoque discreto pueden ser KNN (k-Nearest Neighbors, en inglés), árboles de decisión, redes neuronales, etc.

Para obtener las reglas a través de un modelo de sistema[26] se realiza un mapeo del mundo en el que se va a desenvolver el autómata y a partir de este con los datos recabados se generan las políticas de recompensa. Mientras que para usar un plan[26], solo se considera la secuencia de acciones desde un estado inicial a un estado final objetivo con estados condicionales previos y resultantes.

3.1.1. RPA

En la Automatización de Procesos Robóticos, se tiene por objetivo automatizar las acciones repetitivas en una computadora usando robots de software. RPA es una metodología en la que, por medio de imágenes o video, tomadas por un dispositivo externo o en la misma computadora mientras se lleva a cabo el proceso a automatizar, se analizan los cambios ocurridos entre una imagen y otra utilizando técnicas de reconocimiento de imágenes, adicionalmente, se tiene en cuenta la acción realizada por el usuario con los dispositivos de entrada (teclado, ratón o entrada táctil) y el tiempo que se tarda en sufrir un cambio entre las imágenes. Para la ejecución de las tareas se verifica por medios visuales que el software en la computadora se encuentre en el estado correcto para la ejecución del proceso deseado, por ejemplo, la presencia de una ventana específica.

De acuerdo con la clasificación realizada por Francesco Corea, RPA está definido como una “tecnología que extrae la lista de reglas y acciones a realizar observando al usuario que realiza una determinada tarea” [27] por tal esta categorizado como una herramienta que se basa en el conocimiento y la lógica.

La implementación de RPA en una empresa proporciona la agilización de los procesos, el aumento en la precisión, por tal, la reducción de errores y reducción de costos a corto plazo, sin embargo, se requiere de mucho análisis y conocimiento del proceso a automatizar, por lo que también se han propuesto metodologías para análisis del proceso[28, 29], herramientas de procesamiento del lenguaje, que a partir de una descripción textual del proceso pueden indicar las partes de este y quien las desarrolla[30] e incluso aplicar técnicas de inteligencia artificial y aprendizaje máquina para que la implementación de RPA sea más fácil, principalmente para personas sin conocimiento profundo del software [19].

3.2. Tipos de programación

Para programación orientada a objetos (POO) se tiene la siguiente definición:

“La programación orientada a objetos es un método de implementación en el que los programas se organizan como colecciones de objetos cooperativos, donde cada uno representa una instancia de alguna clase y éstas clases son todas miembros de una jerarquía de clases unidas por herencia” [31].

Algunas de las ventajas de la POO sobre la programación procedimental son la modularidad y la reutilización de código, facilitando el desarrollo de programas, como si solo se unieran “bloques de construcción” [31], también, existe el encapsulamiento de los datos, lo cual permite controlar el acceso a la información almacenada [32]. Además, el tipo de abstracción es diferente en un lenguaje procedimental a uno que es orientado a objetos, ya que se trata de modelar objetos del mundo real en el programa y en lugar de empezar por la definición de las estructuras de datos y procedimientos a usar, se empieza el desarrollo pensando en el objeto y cuales son las tareas a desempeñar de ese objeto, lo que facilita la implementación de una idea [32].

“En un lenguaje orientado a objetos verdadero, toda entidad en el dominio del problema se expresa a través del concepto de objetos” [32], entre los lenguajes orientados a objetos Python [33] y C# [32], mientras que C++ [32] no es un lenguaje orientados a objetos y “...java, tan bueno como es, tiene algunas limitantes como lenguaje orientado a objetos” [32].

“Se dice que un sistema de software está basado en eventos si sus partes interactúan principalmente usando notificaciones de eventos” [34]. Los sistemas Operativos con interfaz gráfica están basados en eventos, ya que la interfaz gráfica de usuario (GUI, por sus siglas en inglés) espera de forma pasiva a que el usuario realice alguna acción sobre los controles [34] (botones, cuadros de texto, etc). La programación basada en eventos, empezó a tener su relevancia a principios de los años 90 con el surgimiento de Microsoft Visual Basic [34].

Considerando que el paso de mensajes entre las diferentes partes de un sistema para poder sincronizar éstas [34] es una característica que distingue a este paradigma de la programación, aquel lenguaje de programación que permita el uso de hilos y sus métodos de sincronización, se puede decir que permite la programación basada en eventos, esto sin mencionar los que son compatibles con algún framework para desarrollar GUI similares y/o compatibles con Microsoft Windows.

3.3. Grafos

Un grafo consiste en dos conjuntos finitos: un conjunto no vacío de vértices y un conjunto de aristas, donde cada arista esta asociada a uno o dos vértices llamados puntos extremos[35]. Se dice que una arista incide sobre cada uno de sus

puntos extremos, dos aristas que inciden en el mismo punto se llaman adyacentes y el vértice en el que no incide ninguna arista se llama aislado[35], En la figura 3.1 se presenta un grafo que muestra las acciones disponibles con el teclado y ratón.

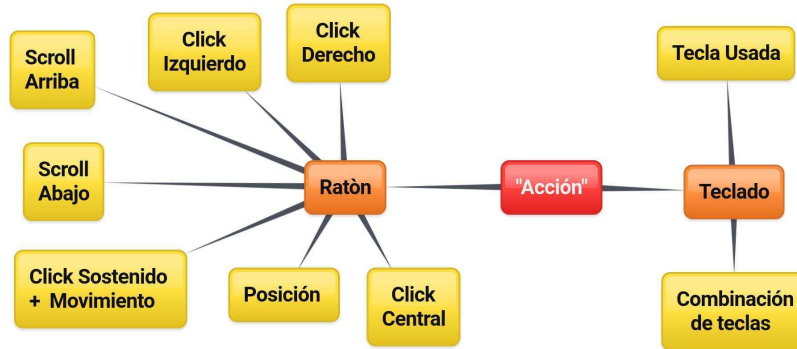


Figura 3.1: Ejemplo de un grafo.

3.3.1. Grafo Dirigido

Un grafo dirigido es aquel donde cada arista, es un par ordenado asignando la dirección del primer elemento mencionado al segundo[35]. éste tipo de grafo se puede representar, en términos de programación, como una lista enlazada, la cual es “una colección lineal de objetos de una clase autoreferenciada, conocidos como nodos” [36], a éstas solo se les puede referenciar por el primer nodo, sin embargo, como cada elemento tiene la referencia al siguiente se puede acceder a todos los elementos de la lista, la ventaja ofrecida es la posibilidad de crear una lista de dimensión variable, por lo que es usada cuando se desconoce la cantidad total de elementos a introducir[36], otro ejemplo sería un autómata finito determinista como el presentado en la imagen 3.2, en el cual se puede apreciar la diferencia con el grafo por mostrar la dirección de las aristas con una flecha, en lugar de usar solo una recta.

3.3.2. Teoría de árboles

En términos matemáticos un árbol es un grafo T el cual se puede definir de la siguiente forma[35]:

- “Un grafo se llama árbol si y solo si, está libre de circuitos y es conexo.”
- “Un vértice de grado 1 en T se denomina un vértice terminal (o una hoja).”
- “Un vértice de grado superior a 1 en T es un vértice interno (o un vértice de rama).”

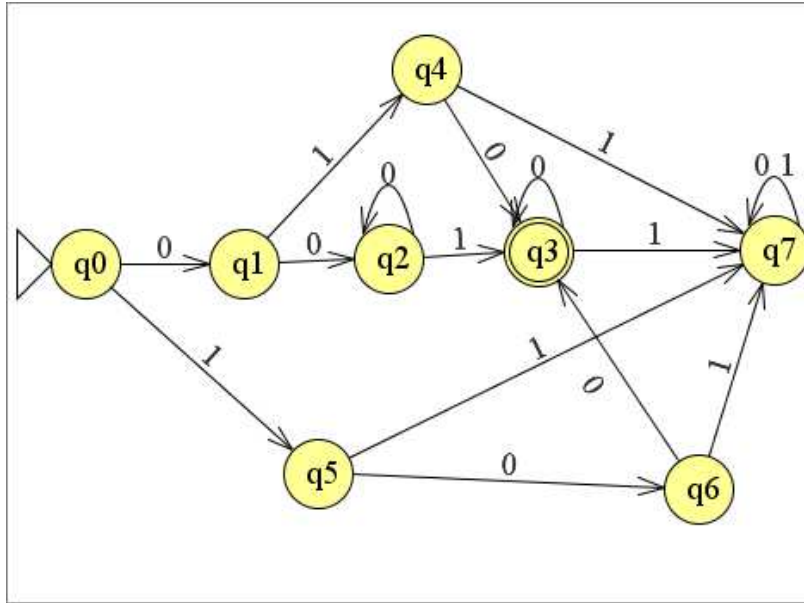


Figura 3.2: Ejemplo de un grafo dirigido.

Los árboles son muy usados en la actualidad ya que permiten darle sentido a la información contenida, gracias a la asociatividad, parentización y prioridad que este permite de manera implícita. Entre sus usos múltiples en el área de la informática se pueden destacar los siguientes; relaciones entre módulos de programación, árboles de decisión en inteligencia artificial y representaciones gramáticas [37].

Una forma de ver a un árbol puede ser como un solo nodo, esté recibe el nombre de nodo raíz, al cual se le puede enraizar un sinfín de árboles lo que da origen a un otro con otras características, dependiendo del resultado se le puede clasificar en alguno de los modelos existentes, por ejemplo; árbol general o árbol binario[37], el mostrado en la figura 3.3, es un árbol binario que representa la jerarquía e interconexión entre varios procesadores.

El árbol general es un modelo con una cantidad indeterminada de nodos hijos, mientras que el árbol binario es un caso particular del árbol general ya que este tiene la característica de tener siempre en cada nodo 2 nodos hijos como máximo, cabe mencionar que este es uno de los modelos mas usados, así como el ultimo caso mencionado, hay árboles que tienen una cantidad fija de nodos hijos, de manera general estos son llamados árboles n-arios [37].

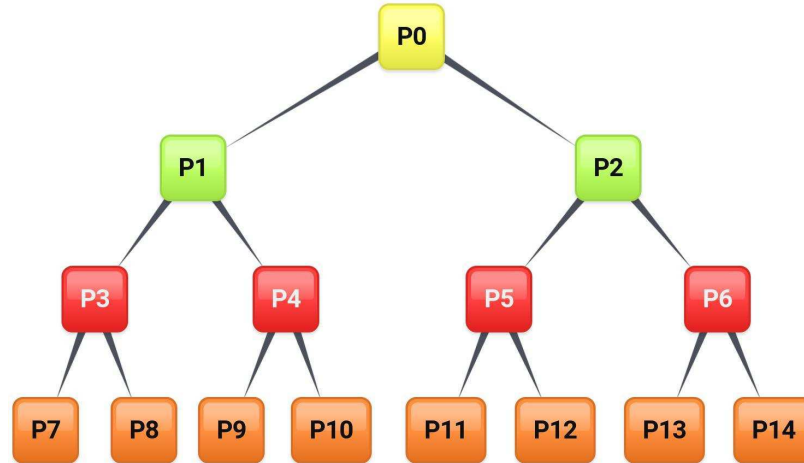


Figura 3.3: Ejemplo de un árbol.

3.4. Tipos de datos

En minería de datos, existen 2 clasificaciones para los tipos de datos [38], los cualitativos (por ejemplo: tipo de sangre, opinión personal, código postal, teléfono) y numéricos. A continuación se presentan sus características.

- Cualitativos[38]
 - Son los que se pueden ubicar en distintas categorías.
 - Algunos se pueden colocar en orden significativo
 - No se les pueden aplicar operaciones matemáticas
- Cuantitativos[38]
 - Son de naturaleza numérica.
 - Se pueden clasificar en orden.
 - Admiten operaciones aritméticas significativas.
 - Pueden ser discretos o continuos.

Los datos también se pueden clasificar por la forma en que se categorizan, cuentan o miden. Este tipo de clasificación utiliza escalas de medición, y cuatro niveles comunes de escalas son:

- Nominal[38]:
 - Se clasifican los datos en categorías mutuamente excluyentes (no superpuestas) y exhaustivas en las que no se puede imponer un orden o clasificación significativa en los datos.

- Ordinal[38]:
 - Se clasifican los datos en categorías que se pueden clasificar.
 - Las diferencias entre los rangos no pueden calcularse mediante aritmética.
 - Se pueden ordenar
- Intervalo[38]:
 - Se clasifican los datos y las diferencias entre las unidades de medida se pueden calcular mediante aritmética.
 - El cero en el nivel de intervalo de medición no significa *null* o *nada* como cero en aritmética.
- Relación[38]:
 - Se clasifican los datos y las diferencias entre las unidades de medida se pueden calcular mediante aritmética.
 - El cero en el nivel de intervalo de medición significa *null* o *nada* como cero en aritmética.

En este capítulo se presentó un breve marco de lo que es el aprendizaje máquina y la teoría necesaria para el desarrollo de un software capaz de identificar secuencias de acciones a partir de una lista de acciones, en resumen, se desarrollará un sistema, con el paradigma de la POO, que genere un grafo dirigido y en cada vértice se almacenara la información de cada elemento de la lista de acciones, considerándolo como un dato nominal, es decir, un dato que carece de orden o valor inherente; dado que no hay relación numérica o de precedencia entre los datos, por lo que, con lo único que se puede trabajar es el orden de aparición, relación representada por el grafo dirigido, además el uso de un contador de incidencias en cada vértice. De este modo es posible identificar el camino con los vértices más usados, esto es, las secuencias que se han repetido mas veces, lo cual es el objetivo de esta tesis.

Capítulo 4

Desarrollo de la investigación

Un primer acercamiento a esta metodología sería con una máquina de Turing, que tiene por configuración inicial una cinta infinita con la entrada, e infinitas cintas auxiliares vacías, al leer un carácter en la cinta de entrada, se copia dicho carácter a la primer cinta auxiliar, avanzando de posición tanto en la cinta de entrada como en la cinta auxiliar, posteriormente se lee el siguiente carácter y nuevamente se copia en esta primer cinta; se prosigue de la misma forma hasta encontrar algún carácter existente en la auxiliar, en este caso, se verifica si este carácter es el primero de dicha cinta, en caso de no serlo, se copia a la siguiente cinta auxiliar, repitiendo el proceso de lectura de la entrada y copia en esta. Para el caso de que sea el mismo carácter de la primer cinta auxiliar, se continúa la lectura de la de entrada verificando que cada carácter sea igual tanto en la cinta de entrada como en las otras cintas; si algún carácter llega a ser diferente se copiará desde el inicio hasta el actual en la siguiente cinta vacía.

En la figura 4.1 se puede apreciar un ejemplo de la simulación de la máquina de Turing explicada anteriormente. Considerando como cadena de ejemplo “ABCDEABDEABXGRWACE”, “ABCDE” se escriben en la primer cinta auxiliar, al leer el carácter “A”, se identifica que ya existe en esta cinta y es el primer carácter, por lo que se empieza a verificar que sean los mismos, los de la entrada que los de la auxiliar, al llegar al carácter “D” de la cinta de entrada, se observa que el de la auxiliar es una “C”, dado que son diferentes, se copia el contenido anterior a esta diferencia, de la primer cinta auxiliar a la siguiente, continuando con la lectura de la entrada, se termina de escribir en la segunda cinta auxiliar la cadena “ABDE”, ya que el siguiente carácter es otra “A” se vuelve a verificar desde el inicio de las cintas auxiliares. Repitiendo el proceso se obtiene una tercer cadena (“ABXGRW”) y una cuarta (“ACE”).



Figura 4.1: Ejemplo del algoritmo con una máquina de Turing multicinta.

4.1. Metodología

Otra forma de visualizarlo es con un grafo dirigido, en la figura 4.2 se aprecia el correspondiente a la cadena de entrada del ejemplo anterior (“ABCDEAB-DEABXGRWACE”), el cual tiene un nodo inicial sin información, posteriormente, cada acción realizada por la persona con los dispositivos de entrada (teclado y ratón) es un nodo único, es decir, a cada acción corresponde un nodo. El cual contiene la tupla de información (dispositivo, acción y colocación) de la acción realizada, el tiempo de espera para ejecutar dicha acción, y un contador de incidencias para el nodo y la referencia a los siguientes.

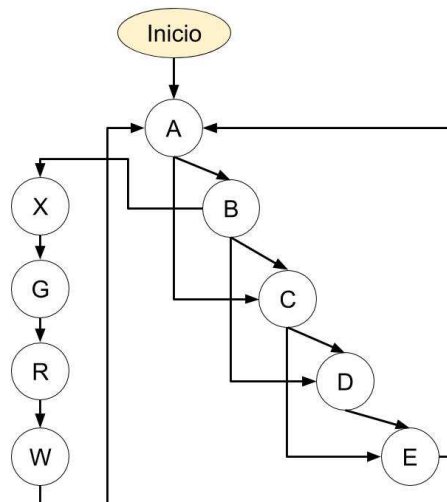


Figura 4.2: Ejemplo del algoritmo con un grafo dirigido.

En la figura 4.2, se muestran las mismas cadenas generadas en el ejemplo de la figura 4.1 (“ABCDE”, “ABDE”, “ABXGRW” y “ACE”), la principal diferencia radica, en la reutilización de los nodos, ya que cada vez que se ingrese el carácter “A”, se recurre al nodo generado anteriormente en lugar de generar uno nuevo, esto permite el ahorro de memoria durante la ejecución y la extracción de la secuencias repetitivas.

Con la anterior definición para los nodos se crea el grafo dirigido, siguiendo el algoritmo descrito gráficamente por el diagrama de flujo de la figura 4.3; cuando se lee una acción se crea el nodo (en caso de no existir) y el enlace a esté desde el anterior (el nodo Inicio, para la primer acción leída), y se sigue así durante la sesión del usuario. Cuando una acción se repite, es decir, que ya existe el nodo, se busca el nodo ya creado correspondiente a dicha acción, se crea el enlace a este y el contador de incidencias se incrementa por la unidad.

El contador de los nodos es utilizado para conocer la frecuencia de cada acción y definir cuáles son relevantes, para este caso de estudio se definió empíricamente que la frecuencia del nodo debe ser 70 incidencias, además, como se puede ver en la figura 4.4, para obtener una secuencia de acciones se genera una lista con las acciones que se están realizando con el numero de incidencias y de forma similar a la analogía con la máquina de Turing (ver figura 4.1), al momento de encontrar una acción que se repite en la secuencia y que esta se haya repetido por lo menos 5 veces, se muestra como sugerencia de secuencia útil y se reinicia la lista; en caso de ser de utilidad a la persona, este se guardará para su posterior uso con un nombre significativo para la persona, de lo contrario, la secuencia se guarda para evitar que se muestre nuevamente.

4.2. Pseudo-código

A continuación se presenta el pseudo-código del programa desarrollado y su explicación.

```

1  Procedimiento Monitoreo
2  Mientras Verdad Hacer
3      Nodo = Capturar accion de teclado o raton
4      Si Nodo existe en Grafo Entonces
5          Incrementar contador_Nodo
6          Si contador_Nodo > 70 Entonces
7              Si Nodo existe en Secuencia Entonces
8                  Ejecutar Seleccion_Tarea
9                  Borrar Secuencia
10             Si no Entonces
11                 Agregar nodo a Secuencia
12             Si no Entonces
13                 Ejecutar Seleccion_Tarea

```

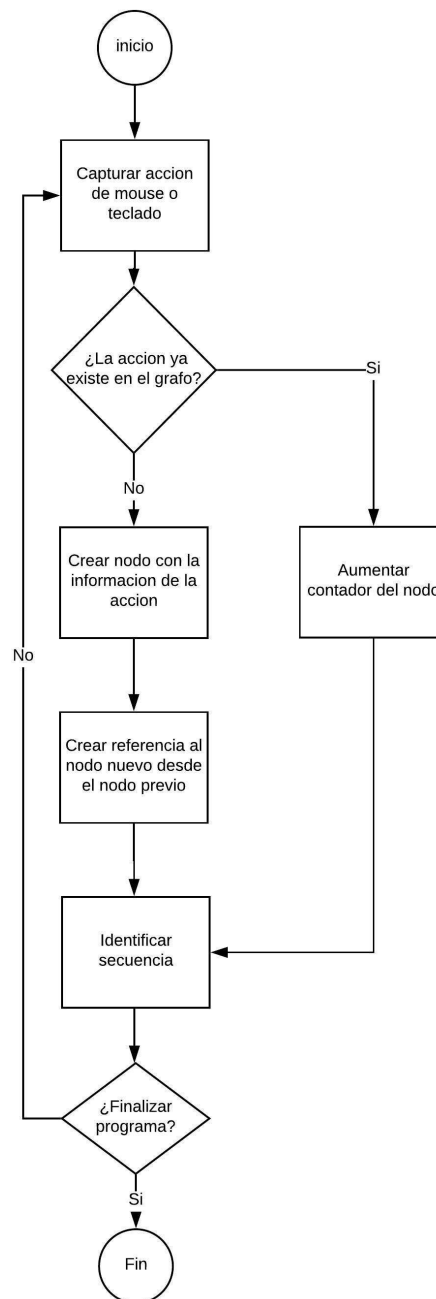


Figura 4.3: Diagrama de flujo del algoritmo general.

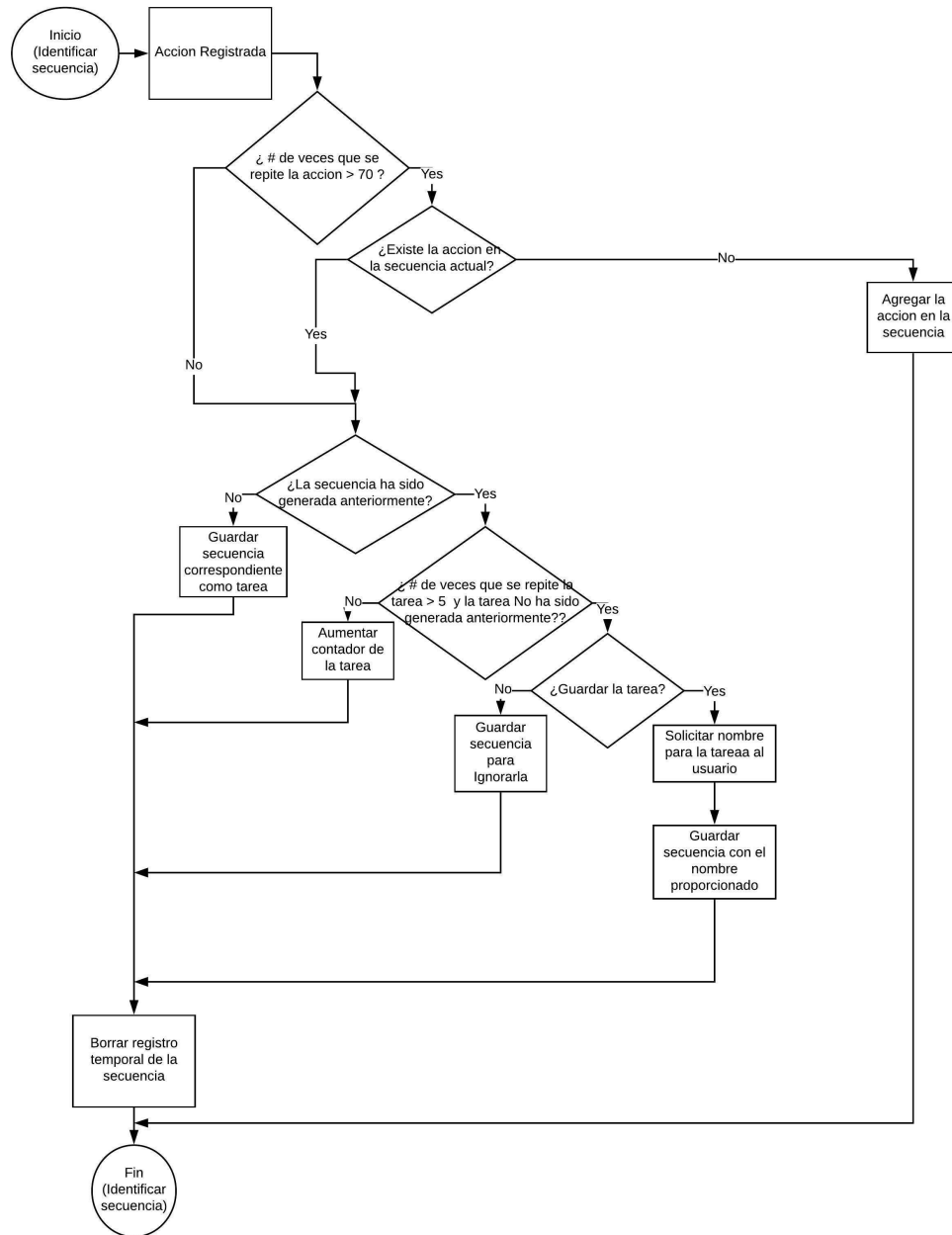


Figura 4.4: Diagrama de flujo del bloque “identificar secuencia” (ver figura 4.3).

```

14      Borrar Secuencia
15      Si no Entonces
16          Agregar Nodo en Grafo
17  Fin Mientras
18  Fin Procedimiento

1  Procedimiento Seleccion_Tarea
2  Si Secuencia existe en lista_Secuencias Entonces
3      Si contador_Secuencia > 5 Y Secuencia No existe en
        lista_Tareas 0 en lista_Ignoradas Entonces
4          Escribir ‘‘Si desea Guardar la tarea [Secuencia],
            escriba un nombre’’
5          Leer Respuesta
6          Si Respuesta es nombre Entonces
7              Agregar Secuencia a lista_Tareas
8          Si no Entonces
9              Agregar Secuencia a lista_Ignoradas
10         Si no Entonces
11             Incrementar contador_Secuencia
12 Si no Entonces
13     Agregar Secuencia a Lista_Secuencias
14 Fin Procedimiento

```

De la línea 1 a la 17 se define el inicio del *Procedimiento de monitoreo*, en este se leen los dispositivos de entrada, se genera el grafo y se buscan las secuencias. Este proceso, es necesario que permanezca activo desde que se enciende la computadora hasta que se apague el equipo, para capturar cada acción que realice el usuario, por lo que en la línea 2 se inicia un ciclo *Mientras* o *While* que no tiene condición de paro y abarca hasta la línea 16.

```

1  Procedimiento monitoreo
2  Mientras Verdad Hacer

```

En la línea 3, se lee la acción del dispositivo de entrada y esta es almacenada en forma de *Nodo*.

```

3      Nodo = Capturar accion de teclado o raton

```

En la línea 4, se verifica que exista un *Nodo* con la misma información que el *Nodo* recién creado en el *Grafo*, en caso de ser así, al *Nodo* existente se le aumenta 1 al *contador_Nodo*, operación realizada en la línea 5, adicionalmente, se va a empezar con una secuencia de validaciones que servirán para identificar una secuencia y posteriormente una tarea.


```

4      Si Nodo existe en Grafo Entonces
5          Incrementar contador_Nodo

```

Primero, se verifica que el *Nodo* se haya repetido mas de 70 veces.

```

6      Si contador_Nodo > 70 Entonces

```

En el caso de ser verdadera la verificación de la línea 6, se verifica que el *Nodo* exista en una lista de nodos llamada *Secuencia*, esta es una tarea que ha realizado el usuario.

```

7      Si Nodo existe en Secuencia Entonces

```

En el caso de ser verdadera la verificación de la línea 7, en la línea 8, se ejecuta el procedimiento *Seleccion_Tarea*, en el cual se va identificar si la *Secuencia* actual es una *Tarea* que le pueda ser de utilidad al usuario.

```

8      Ejecutar Seleccion_Tarea

```

Después de verificar si la secuencia es una *Tarea* o no, la línea *Borra* la *Secuencia* dejándola sin *Nodos*, para empezar a procesar con otra *Secuencia*.

```

9      Borrar Secuencia

```

La instrucción 10 es el caso contrario de la verificación de la línea 7, por tal, en caso de que el *Nodo* no se encuentre en la *Secuencia*, este se agregará.

```

10     Si no Entonces
11         Agregar nodo a Secuencia

```

La línea 13 es para el caso falso de la verificación de la línea 6, como el *Nodo* no cumple la condición de tener mas de 70 repeticiones, por tal, se procesa la *Secuencia* hasta el momento en busca de *Tareas* (línea 13)y posteriormente, se *Borra* la *Secuencia* (línea 14).

```

12     Si no Entonces
13         Ejecutar Seleccion_Tarea
14         Borrar Secuencia

```

En la línea 15, se presenta el caso contrario de la (línea 4), que el *Nodo* no exista en el grafo, este se agrega (línea 16) en el *Grafo*, enlazándolo desde el *Nodo* anterior.

```

15     Si no Entonces

```

```
16      Agregar Nodo en Grafo
```

Por ultimo; en la línea 17 se encuentra el final del *Mientras* que se inició en la línea 2, y en la línea 18 está el final del *Procedimiento monitoreo* que se empezó en la línea 1.

```
17      Fin Mientras
18      Fin Procedimiento
```

En las líneas siguientes, se presenta el pseudo-código del procedimiento para la identificación de tareas. Empezando nuevamente por la línea 1, se presenta la el inicio del mismo, el cual comprende de 14 líneas. Al ser un módulo del procedimiento anterior, este se trabaja con la misma *Secuencia* que el anterior, por tal, en la línea 2, se procede a verificar que la *Secuencia* exista en la *lista_Secuencias*

```
1      Procedimiento Seleccion_Tarea
2      Si Secuencia existe en lista_Secuencias Entonces
```

Si la verificación de la línea 2 es verdadera, se verifica que la *Secuencia* se haya repetido mas de 5 veces, para validar que le pueda ser de utilidad al usuario, además se verifica que la *secuencia* no exista en la *lista_Tareas* o en la *lista_Ignoradas*, ya que esto significaría que ha sido encontrada anteriormente. En caso de que esta validación de la línea 3 resulte verdadera y que el usuario considere que sí le es útil, en la línea 4, se le solicita que le asigne un nombre a la *Secuencia* mostrada. Posteriormente en la línea 5 se lee la *Respuesta* introducida por el usuario.

```
3          Si contador_Secuencia > 5 Y Secuencia No existe en
              lista_Tareas 0 en lista_Ignoradas Entonces
4              Escribir ‘‘Si desea Guardar la tarea [Secuencia],
                  escriba un nombre’’
5              Leer Respuesta
```

En caso de que la *respuesta* sea un nombre para la acción se ejecuta la línea 7, en la que se agrega la *Secuencia* a una *lista_Tareas*, que son las *Tareas* a las que tiene acceso el usuario, para poder ejecutarlas. En caso contrario se prosigue con la línea 4, agregando la *Secuencia* a la *lista_Ignoradas*, estas son las tareas que el usuario ha decidido que no le son de utilidad y por tal no se le volverán a mostrar.

```
6          Si Respuesta es nombre Entonces
7              Agregar Secuencia a lista_Tareas
8          Si no Entonces
9              Agregar Secuencia a lista_Ignoradas
```

En caso contrario de que la verificación de la línea 3 no sea verdadera, en la línea 11, se incrementa en 1 el *contador_Secuencia*, para indicar que se ha usado una vez más la *Secuencia*.

```

10     Si no Entonces
11         Incrementar contador_Secuencia

```

Para el caso de que la verificación de la línea 2 sea falsa, se prosigue con la línea 13, agregando la *Secuencia* a la *lista_Secuencias*.

```

12     Si no Entonces
13         Agregar Secuencia a Lista_Secuencias

```

Finalmente, se presenta en la línea 14, el fin del procedimiento *Seleccion_Tarea*, en este punto se tienen tres posibilidades para la *Secuencia* actual; primero, la *Secuencia* puede haber sido de utilidad para el usuario, por lo que se guardo en *lista_Tareas* con el nombre asignado como referencia, el segundo caso es, que la *Secuencia* no le fue de utilidad al usuario y fue almacenada en *lista_Ignoradas* y en el último caso la *Secuencia* no cumplía la característica de las 5 repeticiones y solo se queda almacenada en *lista_Secuencias*

```

14     Fin Procedimiento

```

4.3. Desarrollo

El algoritmo mostrado en la sección anterior fue implementado en el lenguaje de programación *Python* versión 3.6.3, aprovechando la versatilidad que este proporciona con las variables, ya que no es necesario especificar un tipo, pues este se asigna al momento de crearla. Tomando en consideración esto, solo se definió una clase *nodo*, donde una variable miembro es la que almacena la información del nodo independientemente del contenido de éste, también, se almacena el tiempo de retraso de espera para realizar la acción y un contador del número de veces que se ha usado ese nodo. Y como se mencionó en la subsección 4.1, se utiliza un grafo dirigido y en ningún momento es necesario leer el grafo en sentido contrario, por lo que dentro de la clase solo se almacena una lista de nodos a los que apunta.

Para realizar la búsqueda de un nodo en específico en el grafo, se utiliza un diccionario con los nodos existentes en él, de este modo, no importa el tamaño o complejidad del grafo, no es necesario realizar un recorrido para localizar un nodo en específico.

El software fue dividido en 4 partes; *monitoreo*, *respaldo*, *carga* y *ejecución*:

- **Monitoreo**, se hace uso de la biblioteca *pyinput* en su versión 1.3.9 la cual por medio de 2 hilos; uno para el monitor del teclado(**A**) y otro para el monitor del ratón(**B**), se capturan las acciones que el usuario realice en la computadora con estos dos dispositivos. En **A** se capturan las acciones de presionar (pressed) o liberar (released) una tecla, mientras que en **B** se tienen más tipos de acciones (presionar, liberar, mover (move) y rueda (scroll)), entre las cuales, mover, captura coordenadas bidimensionales y la rueda, se mueve arriba o abajo.

Al detectar alguna de las acciones mencionadas se registra la información correspondiente (Tiempo de espera, dispositivo, acción, [tecla, botón, coordenadas o dirección del scroll]) en una lista en memoria, posteriormente un hilo de procesamiento lee la lista e intenta crear el nodo o en caso de que este ya exista se debe buscar en el grafo y verificar la relación con el nodo anterior. Además, se evalúa si la secuencia es candidato para formar parte de una secuencia, en caso dado se almacenará en una lista temporal y posteriormente si esta es una tarea repetitiva, se almacenara en otra lista para poder contabilizar las repeticiones de esta y mostrársela al usuario; si él decide que es una tarea útil, esta se almacenará en otro diccionario con el comando que decida el usuario como llave, en caso contrario, se mete a una lista de secuencias ignoradas para evitar mostrarlas de nuevo.

- **Respaldo**, lee la lista de acciones, el diccionario de tareas y la lista de secuencias ignoradas anteriormente mencionadas, en la explicación del monitoreo y las almacena en archivos de texto. Para realizar esta tarea fue necesario asignarla a un hilo de ejecución que realizará la ejecución cada 120 segundos.
- **Carga**, se ejecuta una sola vez al iniciar el programa leyendo los archivos mencionados (en caso de existir), empezando con la lista de acciones realizadas por el usuario, para reconstruir el grafo, pero sin realizar la búsqueda de tareas, posteriormente, las que han sido guardadas se vuelven a colocar en el diccionario de tareas y finalmente se restaura la lista de las secuencias ignoradas con los datos respaldados, restaurando el sistema tal cual estaba antes de finalizar el programa.
- **Ejecución**, se desarrolló una interfaz gráfica simple con *TkInter* para que el usuario interactúe con el software, ver figura 4.5, en la cual se observa la ventana principal y desde la parte superior se aprecia al asistente amarillo, creado para esta aplicación, sobre una lista con algunas tareas identificables por números y en la parte inferior un botón para que la computadora realice

la tarea seleccionada en la lista. Para lo cual se definió un diccionario con los comandos posibles para que se traduzcan del texto capturado al comando ejecutable.



Figura 4.5: Ventana principal con el asistente y una lista de tareas guardadas.

Al momento de detectarse una tarea repetitiva, se le muestra al usuario una ventana de captura como se observa en la figura 4.6, en la cual se aprecia al mismo asistente amarillo con otro letrero y la secuencia obtenida, también se observa un cuadro de texto para introducir el nombre solicitado, mas abajo se localizan los botones para guardar la tarea y agregarla a la lista de la figura 4.5 o ignorarla.



Figura 4.6: Ventana mostrando una tarea encontrada.

Capítulo 5

Experimentos y resultados

En el presente capítulo, se describe el conjunto de experimentos realizados, para evaluar la propuesta planteada, ante la problemática de la discapacidad, que presentan las personas con problemas de movilidad en los brazos o manos.

Los algoritmos mostrados en el capítulo 2, proponen formas de automatizar actividades realizadas por una persona, siendo posible implementar ese tipo de solución al problema en cuestión, pero esto, representaría un problema aún mayor, dado que se tendría que conocer las actividades que realiza el usuario del equipo, para proponer la automatización de éstas.

La propuesta planteada en este escrito, es un aprendizaje progresivo, discriminando las acciones menos frecuentes realizadas por el usuario y descartando por completo aquellas que no se realizan.

5.1. Resultados de experimentación

El uso del software no se ve limitado a las personas con movilidad reducida, este le puede servir incluso a las personas que no tengan discapacidad alguna, ya que no va por una tarea objetivo específica, sino por las actividades que realice la persona de forma frecuente sin importar cuales sean. Con esto en mente, se lograron obtener los archivos con la lista de acciones de 7 personas (1 con discapacidad en brazos y manos y 6 sin discapacidades), que realizaban sus actividades cotidianas en sus propios equipos de cómputo, esto durante diferentes periodos de tiempo; los primeros cuatro, fueron monitoreados en un plazo de 4 meses; el quinto, 7 días y los últimos dos, durante 1 mes.

Durante los periodos mencionados las personas tuvieron encendida su computadora diferente cantidad de tiempo el cual se muestra en la segunda columna de la tabla 5.1. Como ya se explicó anteriormente en el capítulo 4 una acción

Tabla 5.1: Información de los datos recabados.

No. de Sujeto	Tiempo de Uso (Hr:Min)	Número de Nodos	Repeticiones
1	166:23	1,494,792	46,036
2	490:24	1,333,016	116,001
3	1060:48	1,448,016	378,541
4	148:23	972,828	56,606
5	17:56	281,794	8,945
6	285:45	1,570,951	130,220
7	37:40	418,966	23,660

realizada por el usuario genera un nodo, esta cantidad es mostrada en la columna 3, mientras que la columna 4 es el número máximo de veces que se repitió un nodo y para mantener el anonimato de las personas participantes en esta prueba, se les hará referencia por un número como se muestra en la primer columna.

Se observa en la tabla 5.1 que el sujeto **número 3** es el que tiene el mayor tiempo de uso con mas de 1060 horas, con 1,448,016 nodos generados y 378,541 veces que se repitió un nodo como máximo; mientras que el sujeto **número 6** con 1,570,951 nodos, es el que obtuvo mayor cantidad de nodos reportados en 285 horas y 45 minutos con 130,220 repeticiones de un nodo. Esta diferencia en el número de nodos, se puede deber a la resolución de la pantalla usado, ya que se monitorea el desplazamiento del ratón por pixel, por lo tanto, a mayor resolución mayor cantidad de pixeles y más nodos por movimientos del ratón.

A cada lista de acciones se le aplicó el algoritmo explicado en el capítulo 4, utilizando un factor de 70 incidencias, para que el nodo sea candidato a formar una secuencia; ya que al utilizar un número menor, se estarían aceptando la mayoría de las acciones, lo cual representa un menor tiempo para mostrar una tarea, pero más secuencias basura que tendría que depurar el usuario manualmente y por el contrario si el factor es mayor, la identificación de secuencias sería lenta y con algunas secuencias basura, y un mínimo de 5 repeticiones por secuencia para declarar que la tarea es útil.

Las secuencias de acciones que empieza con la acción *Release* fueron descartadas, ya que esta acción implica que se quedó presionada una tecla o botón específico; también se rechazaron las que terminen con la acción *Pressed* ya que esto dejaría presionada la tecla o botón hasta que se presione físicamente o se mande a llamar la acción *Release*, por ejemplo, una de la tareas que se verían afectadas por las limitantes mencionadas, podría ser el seleccionar varios iconos no consecutivos en Windows, ya que una forma de hacerlo es mantener presionada(Pressed) la tecla CTRL y seleccionar los iconos con el ratón uno a uno dando clic sobre ellos y posteriormente soltar(Release) la tecla CTRL, principalmente por la falta de precisión al mover el ratón, ya que estas variaciones evitan que se genere una secuencia completa de la tarea, dejando la tecla presionada.

En la figura 5.1, se presenta el grafo generado con las primeras 150 acciones del teclado del sujeto **número 3**, mientras que en la figura 5.2 se hace la ampliación a un conjunto de nodos en los que se muestra la relación de ('Keyboard','Pressed ','Key.shift_r') en la parte inferior central y el nodo ('Keyboard', 'Release', 'Key.shift_r ') ubicado en la parte superior izquierda. En la figura 5.3 se muestra el acercamiento al nodo ('Keyboard','Pressed ','Key.shift_r'), en esta se aprecia la relación con otros nodos, lo cual implica que este nodo, es uno de los que mas se repite, dado que es utilizado en varias secuencias usadas por este usuario.

En la siguiente dirección web es posible encontrar el grafo en formato PDF, para su mejor observación, así como el generado con las primeras 600 acciones del ratón por el mismo sujeto.

<https://1drv.ms/f/s!Ap61Bj4GJQX2i0ppX2n1jChCHyxqdQ>

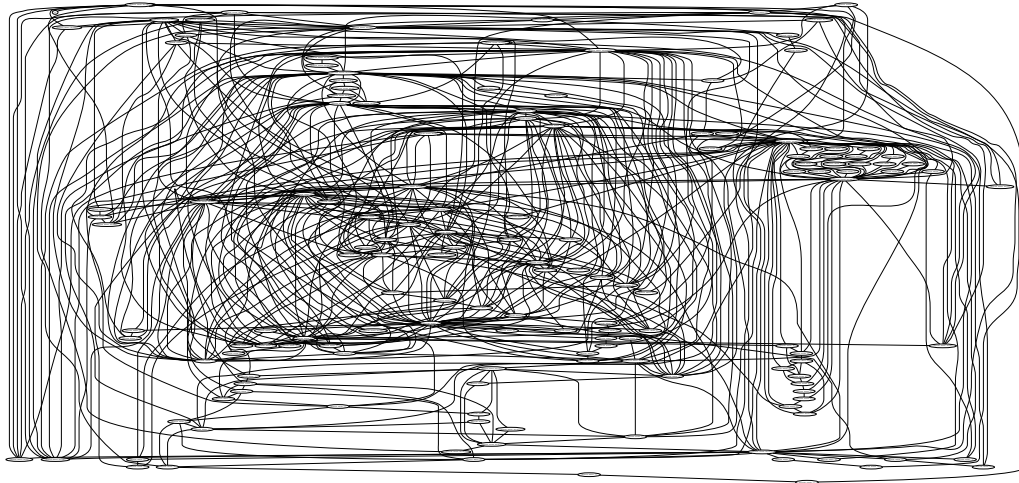


Figura 5.1: Grafo generado por el sujeto *número 3* solo con acciones del teclado.

Como filtro adicional, se limitó a obtener secuencias de acciones con longitudes en múltiplos de 2, considerando que una tecla o botón presionado debe de ser liberado para concluir la acción, sin embargo, pese a esta limitante, las secuencias a obtener naturalmente son; desde una acción (tabla 5.2) y desde dos acciones (tabla 5.3).

El sujeto **número 1** es el que obtuvo mayor número de *Secuencias Aceptables*, 189 secuencias lo cual corresponde a un 36% de *Porcentaje de Precisión* (P_P), porcentaje obtenido de la ecuación 5.1, en la que se realiza la división de las *Secuencias Aceptables* (S_A) multiplicadas por 100 entre el *Total de Secuencias Encontradas* (S_T).

$$P_P = \frac{S_A \cdot 100}{S_T} \quad (5.1)$$



Figura 5.2: Ampliación al grafo generado por el sujeto *número 3* solo con acciones del teclado.

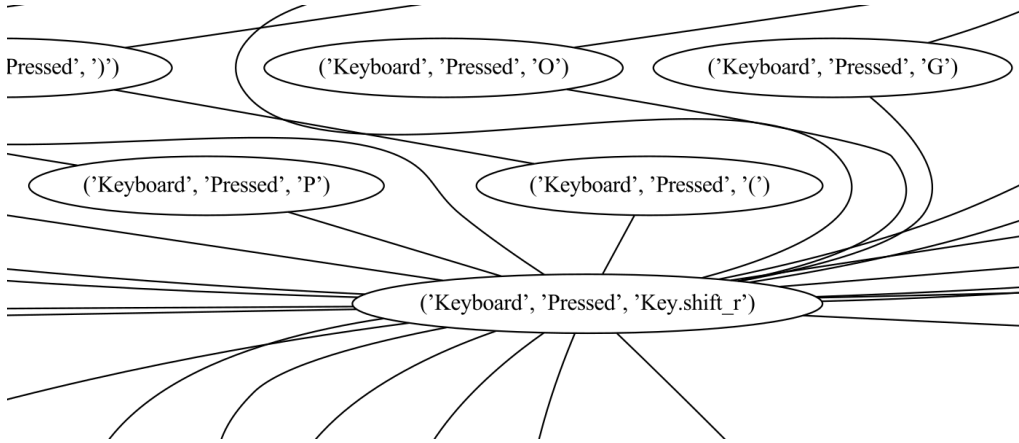


Figura 5.3: Ampliación para la correcta visualización de los nodos en el grafo generado por el sujeto *número 3* solo con acciones del teclado.

El caso contrario es el sujeto **número 6**, el cual presenta 43 *Secuencias Aceptables* y 28.47% como *Porcentaje de Precisión*, lo cual se podría deber a que este uso mayormente el ratón, a diferencia del sujeto **número 5** (la persona con discapacidad), que obtuvo un *Porcentaje de Precisión* de 35.48%, pese a ser el usuario que interactuó menos tiempo con la computadora (17 horas y 56 minutos), también, fue el que obtuvo la menor cantidad de nodos (281,794) y el número más bajo de repeticiones (8,945 repeticiones), ver tabla 5.1.

En la tabla 5.3, se aprecia un incremento en el *Porcentaje de Precisión* con respecto a la tabla 5.2, esto es debido, a la eliminación previa de las acciones unitarias, que por las restricciones establecidas eran rechazadas, lo cual lleva a un incremento considerable en los *Porcentajes de Precisión* de los individuos, principalmente, el sujeto **número 6**, ya que se incrementó del 28.47% al 48.71%.

Tabla 5.2: Tabla de resultados con secuencias de una longitud mínima de 1 acción.

No. de Sujeto	Secuencias Aceptables	Secuencias Totales	Porcentaje de Precisión
1	189	525	36.00 %
2	165	487	33.88 %
3	151	467	32.33 %
4	56	180	31.11 %
5	55	155	35.48 %
6	43	151	28.47 %
7	66	205	32.19 %

Tabla 5.3: Tabla de resultados con secuencias de una longitud mínima de 2 acciones.

No. de Sujeto	Secuencias Aceptables	Secuencias Totales	Porcentaje de Precisión
1	179	410	43.65 %
2	170	377	45.09 %
3	154	346	44.50 %
4	52	119	43.69 %
5	49	108	45.37 %
6	38	78	48.71 %
7	60	145	41.37 %

Por el hecho de que este es un método de aprendizaje acumulativo, así sea que la primer secuencia sea del agrado del usuario o no, la siguiente secuencia puede contenerla con alguna acción adicional y es mas probable que esta variante, sea de utilidad al usuario.

Recapitulando, para que una tarea sea identificable; los nodos que forman la secuencia deben de tener un mínimo de 75 repeticiones cada uno y cumplir con las restricciones planteadas; no empezar con la acción *Release*, no terminar con la acción *Pressed* y la longitud mínima de 2 acciones, por lo tanto, las secuencias mostradas a continuación cumplen con estas características:

Sujeto: 1

Descripción: La tarea es utilizada en el software Blender para girar el objeto en el eje Y.

Secuencia obtenida:

Keyboard,Pressed,G

Keyboard,Release,G

Keyboard,Pressed,Y

Keyboard,Release,Y

Sujeto: 2

Descripción: En Windows es utilizada esta combinación de teclas para cambiar entre las ventanas abiertas.

Secuencia obtenida:

Keyboard,Pressed,Key.alt_l

Keyboard,Pressed,Key.tab

Keyboard,Release,Key.tab

Keyboard,Release,Key.alt_l

Sujeto: 3

Descripción: es la palabra “el”.

Secuencia obtenida:

Keyboard,Pressed,e

Keyboard,Release,e

Keyboard,Pressed,l

Keyboard,Release,l

Sujeto: 4

Descripción: En Windows, selecciona el objeto señalado por el puntero y obtiene el menú contextual de ese objeto.

Secuencia obtenida:

Mouse,Pressed,Button.left

Mouse,Released,Button.left

Mouse,Pressed,Button.right

Mouse,Released,Button.right

5.2. Discusión de resultados

Lo más destacable de las pruebas realizadas son los resultados de los sujetos **número 5** y **número 7** (ver tabla 5.1, 5.2 y 5.3) ya que pese a ser los que menos tiempo de uso y *Secuencias Totales* obtuvieron, su *Porcentaje de Precisión* no varía tanto con respecto al de los demás sujetos, lo cual demuestra una estabilidad en el algoritmo mostrado, considerando que es probable que estos usuarios utilizaron el teclado más que el ratón, a diferencia del sujeto **número 6**, y también es probable que realizaron acciones similares cada vez que usaba la computadora, mientras que los demás sujetos tuvieron más diversidad en las acciones realizadas.

Considerando que los criterios mencionados son para darle un uso específico al grafo, es destacable que pese al entorno variable en el cual fue evaluado el software, solo fue necesario asignar unas pocas condiciones generales para obtener las secuencias coherentes. Entre las secuencias rechazadas se pueden localizar tareas

que tienen la longitud de una acción, las cuales es posible que para algún usuario sean de utilidad, se presenta la siguiente lista a modo de ejemplo.

Secuencia 1:

Mouse,Scrolled,Down

Secuencia 2:

Mouse,Scrolled,Up

Cabe recordar que la principal intención de este software es el apoyar a las personas con discapacidad en brazos y manos, teniendo en cuenta esto, las secuencias de longitud uno, pueden ser de utilidad a alguna persona, sin embargo, considerando que la secuencia más útil es la que contiene mas elementos, estas acciones unitarias fueron descartadas. Adicionalmente, si se piensa en algún otro uso para el algoritmo, aparte de la automatización de las tareas realizadas en una computadora, es posible que las acciones de longitud unitaria tengan importancia, por lo que el *Porcentaje de Precisión* mostradas en las tablas 5.2 y 5.3, puede variar dependiendo del caso de uso.

La lista de tareas obtenida se puede ver incrementada con el uso de estas, por el hecho de que, al hacer uso de las tareas guardadas, el usuario le indica a la maquina que tarea realizar, en lugar de hacerla él mismo, esté es otro aspecto para destacar ya que solo se crean las secuencias, no se ejecutan automáticamente, si se desea ejecutar alguna tarea guardada, el usuario es el que debe ejecutarla manualmente.

Considerando el objetivo a cumplir, el software desarrollado tiene mucha similitud con un creador de macros, por ejemplo, Pulovers Macro Creator, las diferencias que hay que destacar se mencionan en la tabla 5.4, en la que se realiza un análisis comparativo general entre ambos desarrollos. También, existe mucha similitud con el objetivo de la metodología de RPA, por lo que en el análisis comparativo entre las metodologías, presentado en la tabla 5.5, se presentan diferencias destacables entre ambas.

Tabla 5.4: Análisis comparativo del software con un generador de macros.

Creador de macros	Software desarrollado
Hay que indicar manualmente cuando empieza y termina la acción deseada	Se monitorea cada acción realizada por el usuario.
El usuario graba manualmente la tarea que desea automatizar	Se muestra al usuario las acciones que realiza con mayor frecuencia para que él decida cual guardar
El usuario requiere conocimiento del software para crear tareas complejas	El usuario no requiere editar las tareas

Tabla 5.5: Análisis comparativo de la propuesta con Robotic Process Automation.

Robotic Process Automation	Software desarrollado
Hay que indicar manualmente cuando empieza y termina la acción deseada.	Se monitorea cada acción realizada por el usuario.
Por medio de técnicas de reconocimiento de imágenes y monitoreo a los dispositivos de E/S, se determina la acción realizada y el momento de ejecución.	Por medio del análisis en tiempo ejecución de un grafo dirigido se obtienen las tareas realizadas.
Se automatiza un proceso en específico.	Se automatiza la tarea que más realice el usuario.

Finalmente, la metodología desarrollada es una forma automática de obtener secuencias, se ha observado que, para esta aplicación, los resultados son prometedores y subjetivos, ya que depende del usuario la definición de las tareas que le puedan ser de utilidad, sin embargo, es posible presentar el proyecto como una forma automática de realizar la automatización de procesos con RPA o de un creador de macros, por medio de aprendizaje automático.

Capítulo 6

Conclusiones y resultados

6.1. Conclusiones

Se implementó un sistema de captura para el teclado y ratón, con el cual se obtuvo la información de 7 sujetos para realizar las pruebas mencionadas.

Se diseñó un algoritmo de aprendizaje no supervisado que no tiene un tiempo de finalización determinado, el aprendizaje es continuo, durante la ejecución obtiene secuencias de acciones realizadas por un usuario, sin datos de ejemplo. Los resultados experimentales demuestran que el software desarrollado es capaz de proporcionar tareas útiles para la automatización de las mismas, independientemente del software que este usando la persona, de forma que en una lista pseudo-infinita de datos ordenados por momento de aparición, es posible encontrar secuencias de información coherente para un usuario.

La propuesta de la implementación de una estructura arborescente fue cambiada a un grafo dirigido, ya que al diseñar el sistema se observó que no se cumple con la característica de no tener circuitos y esto fue necesario para ahorrar espacio en memoria y cumplir con el objetivo principal.

Se planteó la búsqueda de sistemas similares en la cual no se logró el éxito esperado, dejando como sistemas similares a los creadores de macros y RPA, por lo que no se encontraron los elementos necesarios para realizar un análisis comparativo de resultados.

En la hipótesis propuesta se menciona que los humanos son seres de costumbres por lo que el software propuesto debe de reconocer esos hábitos, pero considerando que en las secuencias obtenidas solo hay registro de teclas y botones del teclado y ratón respectivamente, lo cual implica que los movimientos con el ratón no son tan mecánicos como se esperaba.

6.2. Trabajos a futuro

Se pueden desarrollar trabajos posteriores basados en el presente desarrollo mejorando la interacción del usuario para que en lugar de tener que hacer clic en una ventana para realizar la tarea, se realice por medio de otro dispositivo de entrada, por ejemplo, un micrófono, lo que requiere la implementación de un software de conversión de voz a texto, permitiendo que se guarden y ejecuten las secuencias por medio de comandos de voz.

Se propone como mejora implementar un filtro dinámico adicional que tome en consideración las acciones elegidas por el usuario y así aumentar el porcentaje de asertividad, lo cual implica reducir las secuencias poco útiles para él mismo, otra alternativa, es implementar un sistema de reconocimiento de imágenes como se hace en RPA, para dar contexto a las acciones capturadas.

En el caso dado de que se desee implementar una técnica de inteligencia artificial, para que la ejecución de estas tareas se realice de forma automática, sería posible, o incluso, en otra vertiente, utilizar esta metodología para la obtención de políticas para RL o aprendizaje por demostración.

El software identifica secuencias que son repetitivas, por lo que se plantean las siguientes interrogantes ¿Qué pasaría si se le proporciona el software a una persona con Parkinson? ¿Es posible identificar un patrón entre sus movimientos? ¿Es posible proporcionarle asistencia con este software?.

6.3. Productos de la investigación

A continuación se presentan los productos generados durante el desarrollo de la presente investigación: entre los que se destacan

- González Tello; R., Chan Alejandro; E. A., Serrano Talamantes; J. F., & Carbajal, Olguín; M. (2016, May). COMPUTACIÓN INTELIGENTE: UN ESTUDIO COMPARATIVO DE METAHEURÍSTICAS. Boletín UPIITA No. 66. Retrieved from <http://www.boletin.upiita.ipn.mx/index.php/ciencia/762-cyt-numero-66/1519-computacion-inteligente-un-estudio-comparativo-de-metaheurísticas>
- Chan Alejandro, E. A., Rivera Zárate, I., Olguín Carbajal, M., & González Tello, R. (2017, Sep). Electronic impact system for a football player's helmet. In 17th International Congress on Computer Science (p. 8). México: Centro de Investigación en Computación.

- Chan Alejandro, E. A., Rivera Zárate, I., Olguín Carbajal, M., & González Tello, R. (2017, Sep). SISTEMA MEDIDOR ELECTRÓNICO DE IMPACTOS PARA CASCO DE FUTBOL AMERICANO POR MEDIO DE ACCELEROMETROS. In XVIII Simposium Internacional : “Aportaciones de la universidades a la docencia, la investigación, la tecnología y el desarrollo” (p. 5). México: Escuela Superior de Ingeniería Química e Industrias Extractivas.

Bibliografía

- [1] R. U. Batista, “Pullover’s Macro Creator - The Complete Automation Tool,” accedido 2017-05-18. [Online]. Available: <http://www.macrocreator.com/>
- [2] D. Dines and M. Tirca, “UIPath,” 2014, accedido 2018-11-26. [Online]. Available: [/www.uipath.com](http://www.uipath.com)
- [3] A. Nakano, A. Tanaka, and J. Hoshino, “Imitating the behavior of human players in action games,” *Entertainment Computing-ICEC 2006*, pp. 1–4, 2006. [Online]. Available: <http://www.springerlink.com/index/g271642r6722687m.pdf>
- [4] G. Zhang, Y. Shi, Y. F. Gu, and D. Fan, “Welding torch attitude-based study of human welder interactive behavior with weld pool in GTAW,” *Robotics and Computer-Integrated Manufacturing*, vol. 48, no. August 2016, pp. 145–156, 2017. [Online]. Available: <http://dx.doi.org/10.1016/j.rcim.2017.03.009>
- [5] S. Nishiguchi, K. Ogawa, Y. Yoshikawa, T. Chikaraishi, O. Hirata, and H. Ishiguro, “Theatrical approach: Designing human-like behaviour in humanoid robots,” *Robotics and Autonomous Systems*, vol. 89, pp. 158–166, 2017. [Online]. Available: <http://dx.doi.org/10.1016/j.robot.2016.11.017>
- [6] E. Senft, P. Baxter, J. Kennedy, S. Lemaignan, and T. Belpaeme, “Supervised autonomy for online learning in human-robot interaction,” *Pattern Recognition Letters*, vol. 0, pp. 1–10, 2016.
- [7] INEGI, “La discapacidad en México , datos al 2014,” INEGI, Tech. Rep., 2014.
- [8] W. H. Organization, *International Classification of Functioning, Disability and Health*. World Health Organization, 2001.
- [9] V. Milosavljevic and D. Alméras, “INFORME REGIONAL SOBRE LA MEDICIÓN DE LA DISCAPACIDAD,” Organización de las Naciones

- Unidas, Santiago, Tech. Rep., 2014. [Online]. Available: http://repositorio.cepal.org/bitstream/handle/11362/36906/1/S1420251{_-}es.pdf
- [10] Organización Mundial de la Salud, “Informe mundial sobre la discapacidad,” Tech. Rep., 2011. [Online]. Available: http://who.int/disabilities/world{_-}report/2011/summary{_-}es.pdf
- [11] R. Romero Zunica, F. Alcantud Marin, A. M. Ferrer Manchon, and Universidad de Valencia., *Estudio de accesibilidad a la red*. Universitat de Valencia, Servei de publicacions, 1998. [Online]. Available: https://books.google.com.mx/books?id=FG0BiHJ0izsC{&}printsec=frontcover{&}hl=es{&}source=gb{_-}vpt{_-}buy{#}v=onepage{&}q{&}f=false
- [12] Daniel Hubbell, “Making progress on accessibility with the Windows 10 Anniversary Update – Microsoft Accessibility Blog,” 2016, accedido 2017-06-07. [Online]. Available: <https://blogs.msdn.microsoft.com/accessibility/2016/07/01/making-progress-on-accessibility-with-the-windows-10-anniversary-update/>
- [13] “Windows accesible para todos: así es el centro de accesibilidad,” 2014, accedido 2017-06-07. [Online]. Available: <https://www.xatakawindows.com/bienvenidoawindows8/windows-accesible-para-todos-asi-es-el-centro-de-accesibilidad>
- [14] “Cómo usar el reconocimiento de voz - Ayuda de Windows,” 2016, accedido 2017-06-07. [Online]. Available: <https://support.microsoft.com/es-mx/help/14213/windows-how-to-use-speech-recognition>
- [15] “¿Qué es Cortana?” 2017, accedido 2017-06-07. [Online]. Available: <https://support.microsoft.com/es-mx/help/17214/windows-10-what-is>
- [16] A. Silberschatz, *Sistemas operativos*, quinta edi ed., P. E. R. Vázquez, Ed. Mexico: Addison Wesley Longman de México, S.A. de C.V., 1999.
- [17] C. Bataller, A. Jacquot, and S. R. Torres, “Robotic process automation,” p. 15, 2017. [Online]. Available: <https://patents.google.com/patent/US9555544B2/en>
- [18] W. M. P. van der Aalst, M. Bichler, and A. Heinzl, “Robotic Process Automation,” *Business & Information Systems Engineering*, vol. 60, no. 4, pp. 269–272, 2018. [Online]. Available: <https://doi.org/10.1007/s12599-018-0542-4>
- [19] S. Mohanty and S. Vyas, *Intelligent Process Automation = RPA + AI*. Berkeley, CA: Apress, 2018, pp. 125–141. [Online]. Available: https://doi.org/10.1007/978-1-4842-3808-0{_-}5

- [20] M. Kulbacki, J. Segen, W. Knieć, R. Klempous, K. Kluwak, J. Nikodem, J. Kulbacka, and A. Serester, *Survey of Drones for Agriculture Automation from Planting to Harvest*, 2018, pp. 353–358.
- [21] R. Issac, R. Muni, and K. Desai, “Delineated Analysis of Robotic Process Automation Tools,” in *2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, feb 2018, pp. 1–5.
- [22] R. Grasse, Y. Morère, and A. Pruski, “Assisted Navigation for Persons with Reduced Mobility: Path Recognition Through Particle Filtering (Condensation Algorithm),” *Journal of Intelligent & Robotic Systems*, vol. 60, no. 1, pp. 19–57, oct 2010. [Online]. Available: <https://doi.org/10.1007/s10846-010-9406-y>
- [23] L. Qiao, L. Zhang, S. Chen, and D. Shen, “Data-driven graph construction and graph learning: A review,” *Neurocomputing*, vol. 312, pp. 336–351, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0925231218306696>
- [24] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (Pt.1)*. Wiley-Interscience, 2000.
- [25] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2011.
- [26] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0921889008001772>
- [27] F. Corea, *AI Knowledge Map: How to Classify AI Technologies*. Cham: Springer International Publishing, 2019, pp. 25–29. [Online]. Available: https://doi.org/10.1007/978-3-030-04468-8_{_}4
- [28] A. Leshob, A. Bourguoin, and L. Renard, “Towards a Process Analysis Approach to Adopt Robotic Process Automation,” in *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*, oct 2018, pp. 46–53.
- [29] C. Cewe, D. Koch, and R. Mertens, “Minimal Effort Requirements Engineering for Robotic Process Automation with Test Driven Development and Screen Recording,” in *Business Process Management Workshops*, E. Teniente and M. Weidlich, Eds. Cham: Springer International Publishing, 2018, pp. 642–648.

- [30] H. Leopold, H. van der Aa, and H. A. Reijers, “Identifying Candidate Tasks for Robotic Process Automation in Textual Process Descriptions,” in *Enterprise, Business-Process and Information Systems Modeling*, J. Gulden, I. Reinhartz-Berger, R. Schmidt, S. Guerreiro, W. Guédria, and P. Bera, Eds. Cham: Springer International Publishing, 2018, pp. 67–81.
- [31] G. Booch, R. A. Maksimchuk, M. W. Engle, B. J. Young, J. Conallen, and K. A. Houston, *Object-Oriented Analysis and Design with Applications*. Addison-Wesley Professional, 2007. [Online]. Available: <https://www.amazon.com/Object-Oriented-Analysis-Design-Applications-3rd/dp/020189551X?SubscriptionId=AKIAIOBINVZYXZQZ2U3A{%&}tag=chimbori05-20{%&}linkCode=xm2{%&}camp=2025{%&}creative=165953{%&}creativeASIN=020189551X>
- [32] T. Archer, *A Fondo C#*. MC Graw Hill, 2002. [Online]. Available: <https://www.amazon.com/Fondo-CD-ROM-Spanish/dp/8448132467?SubscriptionId=AKIAIOBINVZYXZQZ2U3A{%&}tag=chimbori05-20{%&}linkCode=xm2{%&}camp=2025{%&}creative=165953{%&}creativeASIN=8448132467>
- [33] A. M. Varó, I. G. Luengo, and P. G. Sevilla, *Introducción a la programación con Python 3*. Universitat Jaume I, 2014. [Online]. Available: <https://doi.org/10.6035/sapientia93>
- [34] T. Faison, *Event-Based Programming*. Apress, 2006. [Online]. Available: <https://doi.org/10.1007/978-1-4302-0156-4>
- [35] SUSANNA S. EPP, *MATEMATICAS DISCRETAS CON APLICACIONES*, 4th ed., CENGAGE LEARNING, Ed., 2012.
- [36] P. Deitel, *Java : como programar*. Mexico, D.F: Pearson educacion, 2008.
- [37] X. F. Gutiérrez, *Estructuras de datos: especificación, diseño e implementación*, ser. Politecnos: Computación y control. Universitat Politècnica de Catalunya, 1999. [Online]. Available: <https://books.google.com.mx/books?id=ySST2zEUbD4C>
- [38] Y. Yang, G. I. Webb, and X. Wu, *Discretization Methods*. Boston, MA: Springer US, 2010, pp. 101–116. [Online]. Available: <https://doi.org/10.1007/978-0-387-09823-4{-}6>