UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Taller 2: Lógica Difusa aplicada en sugerencia de géneros musicales utilizando Python

Integrantes: Daniel Frez
Matías Figueroa
Javier Sanhueza
John Serrano
Curso: Teoría de la Computación
Profesor: Daniel Vega
Ayudante: Matias Tobar

30 de Junio de 2022

Tabla de contenidos

1. Introducción

Desde los tiempos de las antiguas civilizaciones, la música ha ido acompañando al ser humano, evolucionando junto a él. Dada la importancia de la música históricamente, y el crecimiento en el consumo de la música, se llegó a tener en la actualidad una gran variedad de canciones y géneros musicales de diversos orígenes y culturas. Esta gran variedad trajo consigo que las personas destinarán tiempo para buscar géneros musicales, canciones y artistas de su gusto. Sin embargo, este proceso puede ser tedioso. Para resolver esto y facilitar la búsqueda, este taller tiene como objetivo principal proponer una aplicación que sugiera géneros musicales y canciones, esto utilizando la base teórica de la **Lógica Difusa** aplicando Python con la librería Scikit-fuzzy. Para lograr sugerir géneros de manera precisa, se tienen como objetivos:

1. Obtener entradas que ayuden a filtrar las canciones y géneros en base a sus características, esto mediante preguntas claves sobre las preferencias del usuario.

2. Establecer una base de conocimientos y reglas sobre géneros musicales y canciones que pertenecen a estos, que serán las salidas del programa representando las sugerencias musicales.

3. Aplicar la Fusificación, completar un sistema de inferencia, y defuzzificar la información entrante con la base de conocimientos y reglas propuestas.

4. Aplicar la inferencia de Mamdani.

2. Desarrollo

2.1. Géneros, Canciones y Escalas

El programa es desarrollado en el lenguaje de programación **Python** versión 3.10, utilizando las librerías **Scikit-Fuzzy** versión 0.4.2, **Numpy** versión 1.23.0 y **Matplotlib** versión 3.5.2. Lo anterior nos permite trabajar con elementos y funciones de la Lógica Difusa, como por ejemplo, las funciones de pertenencia.

Se consideran 4 géneros de musica, los cuales son:

- Pop
- Rock
- Rap
- Blues

Además, se consideran los siguientes atributos, con escalas de rangos bajos, medios y altos:

- Número de integrantes del artista
- Intensidad de sonido de la canción
- Ritmo de la canción

Utilizando *arange* de Numpy y las funciones *trimf* y *trapmf* de Scikit-Fuzzy, se definen escalas y gráficos con valores asociados para los distintos rangos de géneros y atributos.

Número de integrantes del artista:

- Cantidad baja: el número de integrantes oscila entre 1 y 3 participantes
- Cantidad media: el número de integrantes oscila entre 4 y 6 participantes
- Cantidad alta: el número de integrantes oscila entre 6 y 10 participantes

El gráfico que representa lo anterior es el siguiente:

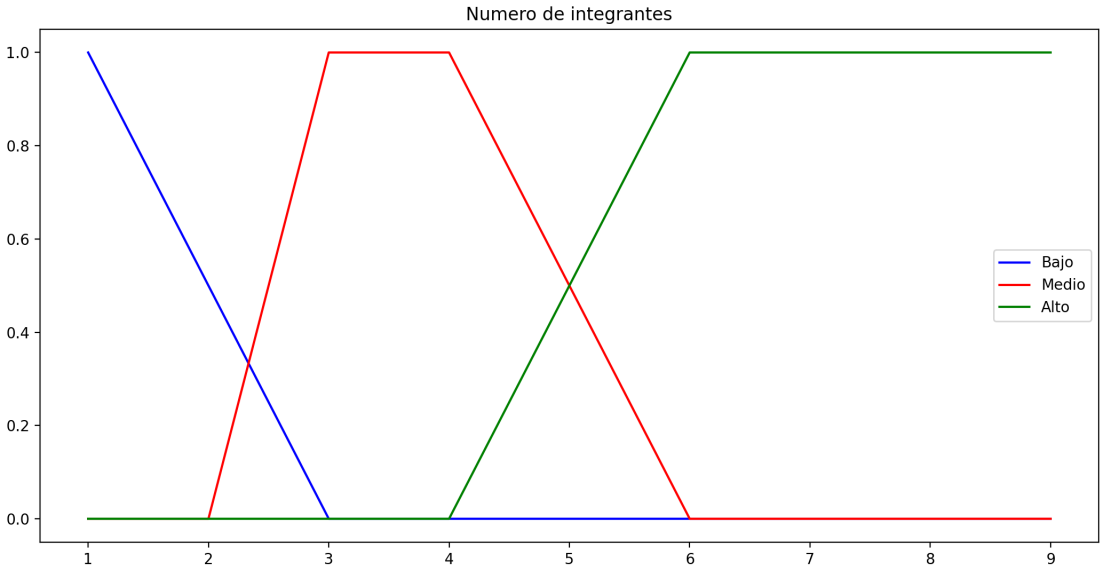

Gráfico 1: Escala del número de integrantes del artista

Intensidad de sonido de la canción:

- Intensidad baja: oscila en el rango entre 1 y 3 de intensidad
- Intensidad media: oscila en el rango entre 3 y 6 de intensidad
- Intensidad alta: oscila en el rango entre 6 y 9 de intensidad

El gráfico que representa lo anterior es el siguiente:

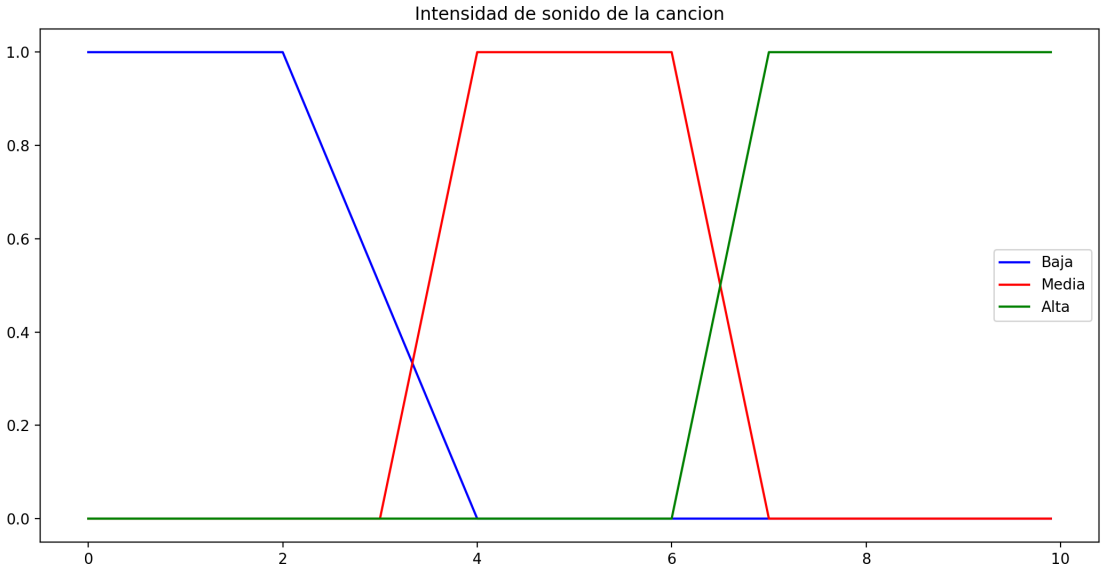

Gráfico 2: Escala de la intensidad de sonido de una canción

Ritmo de una canción:

- Ritmo bajo: oscila en el rango entre 1 y 3 de ritmo
- Ritmo medio: oscila en el rango entre 3 y 6 de ritmo
- Ritmo alto: oscila en el rango entre 6 y 9 de ritmo

El gráfico que representa lo anterior es el siguiente:

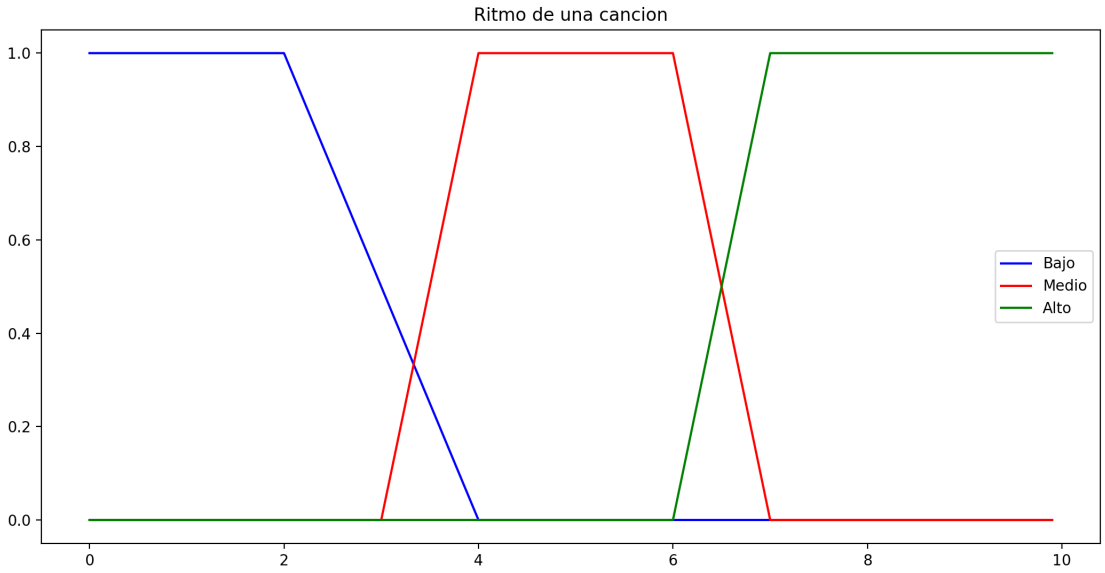

Gráfico 3: Escala del ritmo de una canción

Para los géneros, se consideró tener un gráfico donde aproximadamente cada género pudiera tener un espacio de área similar.

- Rock: Se considera Rock cuando se esta en el rango desde 0 hasta 3
- Rap: Se considera Rap cuando se esta en el rango desde 3 hasta 5
- Blues: Se considera Blues cuando se esta en el rango desde 5 hasta 7
- Pop: Se considera Pop cuando se esta en el rango desde 7 hasta 9

El gráfico que representa lo anterior es el siguiente:

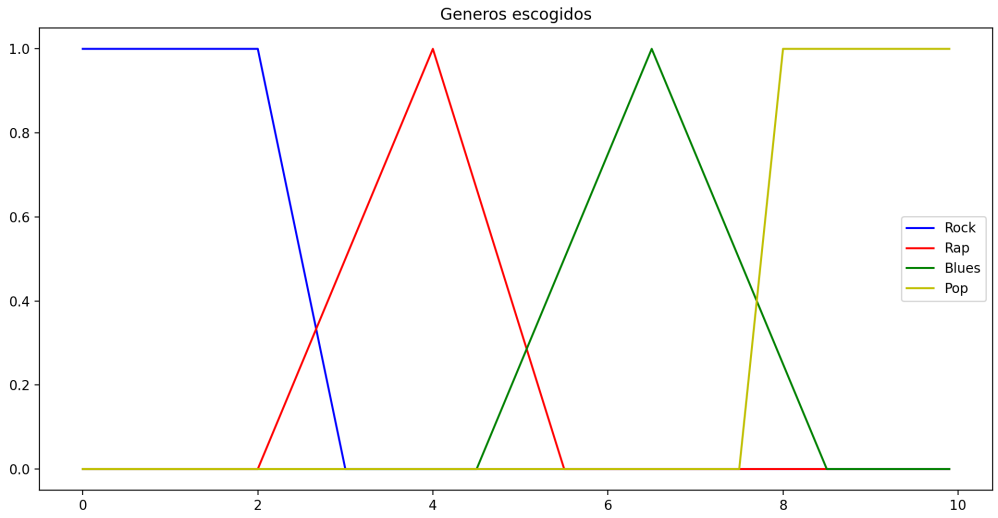

Gráfico 4: Escala de los géneros escogidos

Para poder entregar resultados correctamente, se tienen listas de canciones de los 4 géneros escogidos, donde cada lista contiene sublistas correspondientes a una canción. Cada canción tiene un nombre, una cantidad de integrantes del artista, un rango de intensidad y un rango de ritmo. Por ejemplo:

```
# Lista de canciones de POP
canciones_pop = [["Maroon 5- She will be loved", "integrantes_medio", "intensidad_media", "ritmo_medio"],
                 ["Michael Jackson - Beat it", "integrantes_bajo", "intensidad_media", "ritmo_alto"],
                 ["The Weeknd - Blinding Lights", "integrantes_bajo", "intensidad_baja", "ritmo_alto"],
                 ["Hunting high and low - Take on me", "integrantes_bajo", "intensidad_media", "ritmo_alto"],
                 ["The Weeknd - Hype", "integrantes_bajo","intensidad_baja", "ritmo_medio"],
                 ["The Weeknd - Starboy", "integrantes_bajo", "intensidad_baja", "ritmo_bajo"],
                 ["The Chainsmokers - Closer", "integrantes_bajo", "intensidad_baja", "ritmo_alto"],
                 ["Christian Kuria - Losing You", "integrantes_alta","intensidad_baja","ritmo_alto"],
                 ["Michael Jackson - Billie Jean", "integrantes_bajo", "intensidad_media", "ritmo_alto"],
                 ["Frank Blunt - The Dirt Road", "integrantes_alto", "intensidad_baja", "ritmo_medio"],
                 ["BTS - Butter", "integrantes_alta", "intensidad_alta","ritmo_alto"],
                 ["Kairi Yagi (Vivy) - Sing My Pleasure", "integrantes_bajo", "intensidad_media", "ritmo_alto"],
                 ["Yellowcard - Way Way", "integrantes_medio", "intensidad_media", "ritmo_alto"]]

# Lista de canciones de ROCK
canciones_rock = [
                 ["Billy Talent - Red Flag", "integrantes_medio", "intensidad_alta", "ritmo_bajo"],
                 ["Phil Collins - In The Air Tonight", "integrantes_bajo","intensidad_media", "ritmo_medio"],
                 ["Autopilot Off - Clockworks", "integrantes_medio", "intensidad_alta", "ritmo_bajo"],
                 ["Caesars - Jerk It Out", "integrantes_medio", "intensidad_alta", "ritmo_medio"],
                 ["Mike Oldfield - Nuclear", "integrantes_bajo", "intensidad_alta", "ritmo_bajo"],
                 ["Brandon Yates - Olympus Mons", "integrantes_bajo", "intensidad_alta", "ritmo_bajo"],
                 ["Nirvana - Come as you are", "integrantes_bajo", "", "ritmo_bajo"],
                 ["Wolfmother - Woman", "integrantes_medio", "intensidad_alta", "ritmo_bajo"],
                 ["Set It Off - Wolf In Sheeps Clothing","integrantes_bajo","intensidad_alta", "ritmo_medio"],
                 ["The Who - Pinball Wizard","integrantes_alta","intensidad_media","ritmo_medio"],
                 ["The Peggies - FootPrints", "integrantes_bajo","intensidad_media","ritmo_alto"],
                 ["Sambomaster - Seishun kyousoukyoku", "integrantes_bajo", "intensidad_alta", "ritmo_bajo"],
                 ]
```

Figura 1: Listas de canciones de los géneros Pop y Rock

A diferencia del Taller 1, no se debe usar una base de conocimientos creada mediante Prolog, por lo que todo el Taller se desarrolla en un solo archivo de nombre **"taller2.py".**

2.2. Funciones e Interfaz Gráfica

Para el desarrollo del taller, se utilizan diferentes funciones, las cuales son:

- **obtenerCaracteristica1:** Obtiene un string correspondiente al número de integrantes de un artista para así poder utilizar el string a la hora de filtrar las canciones. Recibe tres arreglos, correspondientes a las cantidades baja, media y alta de la característica. El string retornado representa la característica escogida.

- **obtenerCaracteristica2:** Obtiene un string correspondiente a la intensidad de sonido de una canción para así poder utilizar el string a la hora de filtrar las canciones. Recibe tres arreglos, correspondientes a las cantidades baja, media y alta de la característica. El string retornado representa la característica escogida.

- **obtenerCaracteristica3:** Obtiene un string correspondiente al ritmo de una canción para así poder utilizar el string a la hora de filtrar las canciones. Recibe tres arreglos, correspondientes a las cantidades baja, media y alta de la característica. El string retornado representa la característica escogida.

- **obtenerCanciones:** Obtiene las canciones que cumplen con las características dadas, retornando el resultado en una lista. Para ello, recibe todas las canciones de un género junto con las características a buscar. En el caso de que no pueda encontrar canciones, busca las canciones que por lo menos coincidan con el número de integrantes del artista. Si aún así no se logran encontrar canciones, se retornan las canciones del género, de esta forma asegurando que el usuario obtenga resultados asegurados.

- **llenar_datos_grafico:** Llena los datos de un gráfico con la información correspondiente, recibe cuatro arreglos y, como tal, no tiene salida.

- **mostrarIntergantes:** Crea el gráfico de escala de número de integrantes y lo muestra en pantalla. No tiene entrada, es llamada mediante la interfaz gráfica.

- **mostrarIntensidades:** Crea el gráfico de escala de intensidades de sonido y lo muestra en pantalla. No tiene entrada, es llamada mediante la interfaz gráfica.

- **mostrarRitmo:** Crea el gráfico de escala de ritmos y lo muestra por pantalla. No tiene entrada, es llamada mediante la interfaz gráfica.
- **mostrarGeneros:** Crea el gráfico de escala de géneros y lo muestra por pantalla. No tiene entrada, es llamada mediante la interfaz gráfica.
- **main_interfaz:** Función maestra que recibe la información proporcionada a través de la interfaz gráfica y la procesa para realizar el proceso de lógica difusa, para así obtener el género y canciones recomendadas. También muestra los gráficos correspondientes si así se desea.

Además, se utilizan las siguientes funciones de la librería Scikit-Fuzzy:

- **Trimf:** Genera un gráfico en forma de triángulo para la escala dada
- **Trapmf:** Genera un gráfico en forma de trapecio para la escala dada
- **Interp_membership:** Genera las funciones de pertenencia de los conjuntos difusos
- **Defuzz:** Obtiene el valor de la defuzzificación, utilizando un método. Para este taller, se pide que el método utilizado sea el método de Bisector de Área (BOA)

Utilizando las librerías **Tkinter** y **Custom Tkinter**, se desarrolla una interfaz gráfica (GUI), la cual permite que un usuario pueda ingresar sus preferencias respondiendo preguntas respecto a las características anteriormente mencionadas de una canción. El programa se encarga de buscar cuáles son las canciones que coinciden con sus preferencias (O alternativamente, que se parecen a sus preferencias) y mostrarlas a través de la interfaz gráfica. El usuario también puede decidir si observar los gráficos de escala para los atributos y géneros, y también si desea ver todos los gráficos de resultados generados a través del proceso de la Lógica Difusa.

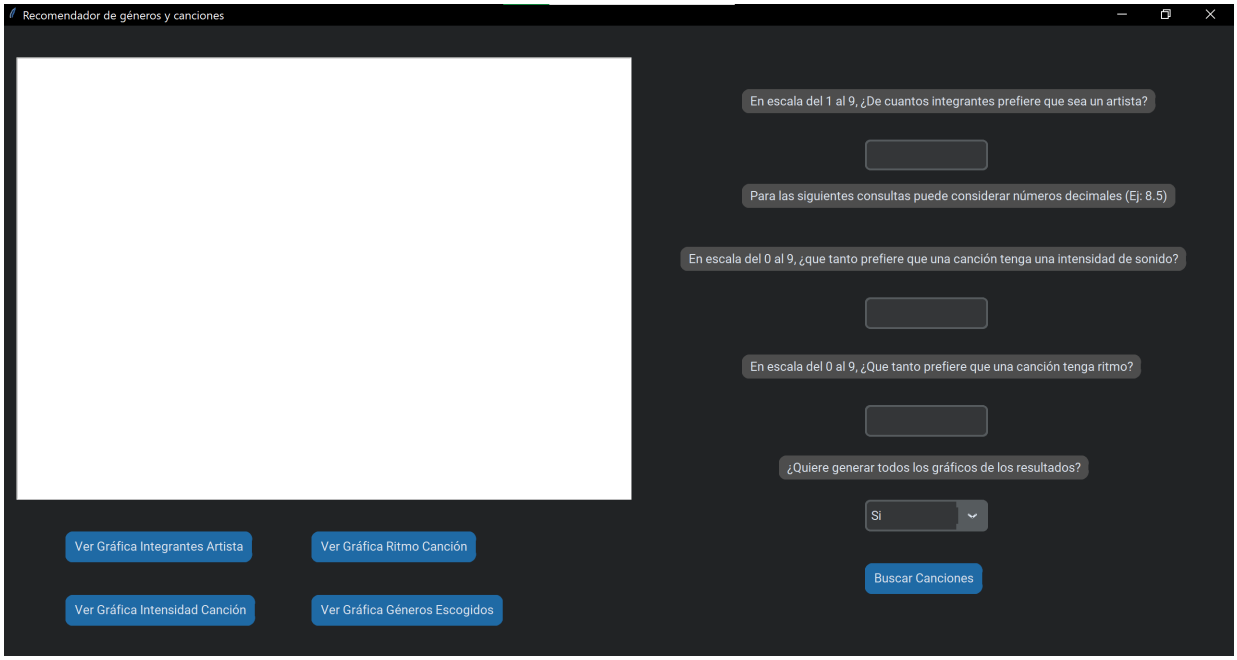

Figura 2: Interfaz gráfica desarrollada.

2.3. Explicación y funcionamiento del código

Respecto al código de taller2.py, en el bloque principal se tienen las listas de canciones, las cuales ya fueron mencionadas.

Luego de eso, viene la creación de escalas, de la cual también ya se habló anteriormente. Con las escalas ya generadas, se utilizan las funciones *trimf* y *trapmf* para generar los gráficos asociados a cada cantidad de los atributos y los géneros.

Se inicializa y configura la interfaz gráfica, se configuran las entradas a proporcionar, la listbox y los botones a utilizar. Al apretar el botón de "Buscar Canciones", se llama a la función *main_interfaz()*. A través de la interfaz gráfica, esta función recibe la siguiente información proporcionada a través del usuario como respuesta a las siguientes preguntas:

- En escala del 1 al 9, ¿De cuantos integrantes prefiere que sea un artista?
- En escala del 0 al 9 (puede considerar decimales) ¿que tanto prefiere que una canción tenga una intensidad de sonido?
- En escala del 0 al 9 (puede considerar decimales) ¿Que tanto prefiere que una canción tenga ritmo?

En la función *main_interfaz()* se realizan las verificaciones correspondientes para las entradas. Es decir, que cada entrada corresponda a un número y que este en el rango de valores permitidos. En el caso de que una verificación de un resultado negativo, se muestra una ventana para decirle al usuario que la entrada es errónea

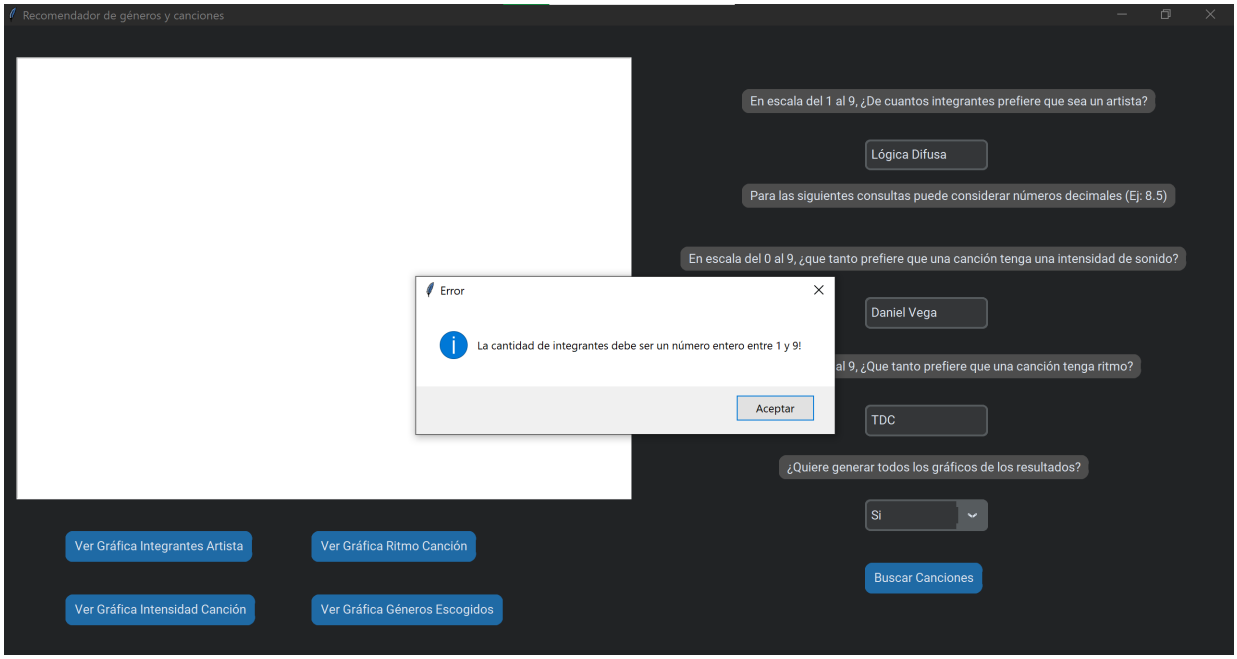

Figura 3: Uno de los mensajes de error al introducir mal la entrada.

Utilizando las entradas proporcionadas por el usuario, se utiliza la función *Interp_membership* para crear las funciones de pertenencias. Cada función de pertenencia recibe la escala del atributo, la cantidad a la cual se le esta creando la función de pertenencia y la entrada del usuario. También se utiliza la funciones *obtenerCaracteristica1*, *obtenerCaracteristica2* y *obtenerCaracteristica3*, para así guardar los rangos preferidos por el usuario, lo cual será útil a la hora de encontrar las canciones que se acomodan a las preferencias del usuario.

Posterior a eso, se utiliza la inferencia de Mamdani para crear las siguientes reglas a utilizar.

1. Si la cantidad de integrantes es media, con intensidad de sonido fuerte y ritmo medio, **ENTONCES** el género es **RAP**

2. Si la cantidad de integrantes es medio, con intensidad de sonido fuerte y ritmo bajo, **ENTONCES** el género es **ROCK**

3. Si la cantidad de integrantes es media o la intensidad de sonido es baja y el ritmo es medio, **ENTONCES** el género es **BLUES**

4. Si la cantidad de integrantes es alto o con intensidad de sonido baja y ritmo alto, **ENTONCES** el género es **POP**

5. Si la cantidad de integrantes es baja, con intensidad de sonido media y ritmo alto, **ENTONCES** el género es **POP**

6. Si la cantidad de integrantes es alto, con intensidad de sonido alta y ritmo bajo, **ENTONCES** el género es **ROCK**

7. Si la cantidad de integrantes baja, con intensidad de sonido media y ritmo bajo, **ENTONCES** el género es **RAP**

8. Si la cantidad de integrantes es baja o, la intensidad de sonido media y ritmo medio, **ENTONCES** el género es **BLUES**

9. Si la cantidad de integrantes es baja, con intensidad de sonido media y ritmo medio, **ENTONCES** el género es **BLUES**

10. Si la cantidad de integrantes es medio, con intensidad de sonido alto y ritmo alto, **ENTONCES** el género es **RAP**

11. Si la cantidad de integrantes es alta, con intensidad de sonido alta y ritmo medio, **ENTONCES** el género es **ROCK**

12. Si la cantidad de integrantes es alta, con intensidad de sonido media y ritmo bajo, **ENTONCES** el género es **ROCK**

13. Si la cantidad de integrantes es baja, con intensidad de sonido alta y ritmo alto, **ENTONCES** el género es **POP**

14. Si la cantidad de integrantes es media, con intensidad de sonido media y ritmo bajo, **ENTONCES** el género es **POP**

15. Si la cantidad de integrantes es media o, la intensidad de sonido es baja y el ritmo es bajo, **ENTONCES** el género es **BLUES**

Con las reglas ya definidas, se asocia cada regla a su género correspondiente, utilizando la función de Numpy *fmin* para así obtener la asociación. Luego se aplica la Agregación, donde se utiliza Fmax de Numpy para obtener 1 solo gráfico con los valores de corte asociados. Una vez que finaliza la Agregación, se utiliza la función *defuzz* la cual recibe la escala de géneros, la agregación y el método a aplicar, el cual para el caso de este taller es el método de Bisector de Área.

Con el valor ya defuzzificado, se puede obtener el género preferido, utilizando nuevas funciones de pertenencias para cada género. Esto hará que se obtengan distintos arreglos, donde se utiliza la función *max* para así obtener cual es el género preferido.

Con el género preferido ya obtenido, se asigna cual es la lista de canciones a utilizar para la función *obtenerCaciones*. Una vez que ya se obtienen las canciones, se muestra por la interfaz gráfica el género preferido, junto con las canciones asociadas. Si el usuario eligió la opción de ver los gráficos de los resultados, entonces se va a mostrar un mensaje de éxito y se verán todos los gráficos generados a través del proceso. Si el usuario lo desea, puede presionar uno de los botones para ver algunas de las escalas de su interés. Se debe tener en cuenta que solo se puede tener abierta 1 ventana de Matplotlib a la vez, por lo que si el usuario quiere ver una escala y luego ver otra, debe primero cerrar la ventana de la primera escala.

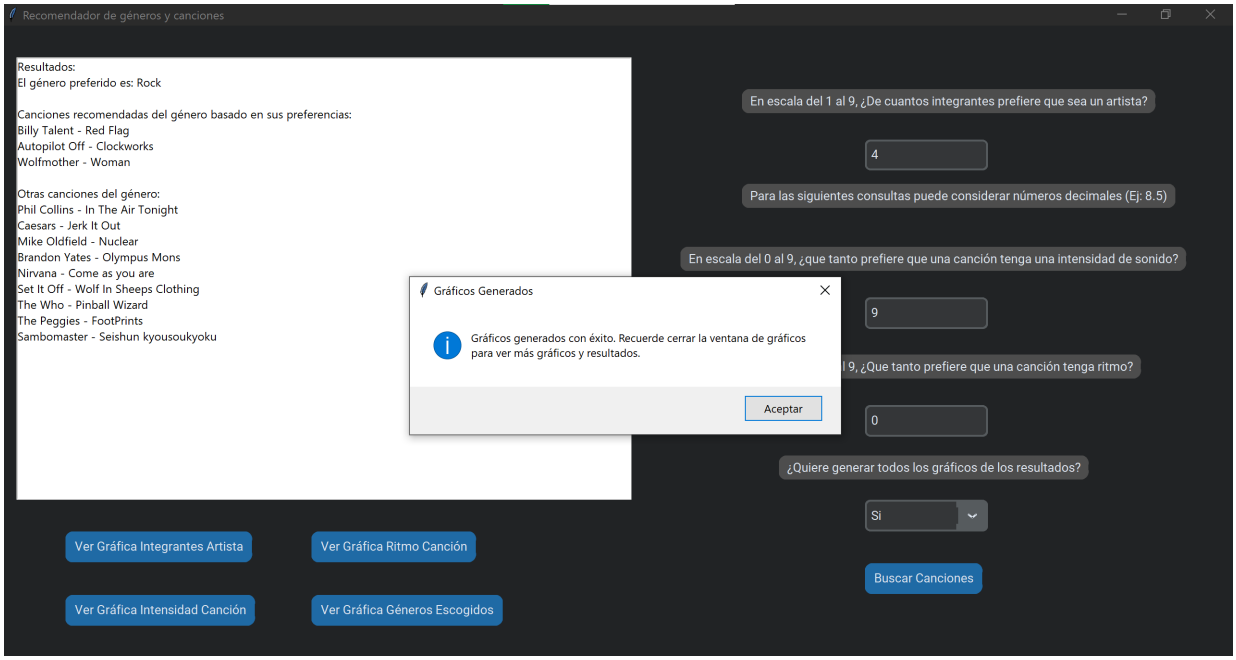

Figura 4: Mensaje que se muestra al usuario al generar algún gráfico.

El usuario es libre de probar combinaciones y encontrar distintas recomendaciones de géneros y canciones, finalizando la ejecución del código una vez que se cierra la interfaz gráfica.

3. Análisis de resultados

3.1. Análisis de la obtención de resultados

Después de finalizar con la entrega de la información pedida para analizar en el software, este comenzará con el análisis de los datos recibidos a través de las variables internas del programa (como por ejemplo "la cantidad de integrantes"), estas variables anteriormente mencionadas se muestran al usuario después de entregar la respuesta. Tras comparar y defuzzificar, el programa usará esta respuesta para determinar cuál será el género que se le entregará al usuario. Cuando el programa consiga la respuesta tras la defuzzificación, comparará esta información con las variables designadas para los géneros, cuando termine y logre identificar cuál será el género, empezará con la segunda parte de la búsqueda la cual será identificar con los datos entregados del usuario cuál es la canción que más concuerde con este. Ya que se tiene el género, se determinará la canción entre las guardadas en el software comparándolas con los datos entregados anteriormente, está siendo la cantidad de integrantes, la intensidad y por último su ritmo. Cuando obtenga la canción o por lo menos una con características similares a los datos entregados por el usuario el programa la entregará mediante la interfaz el género con la canción obtenida para luego mostrar las gráficas de las variables entregadas y con los pasos de la defuzzificación.

3.2. Gráficos resultantes

Al momento de utilizar la interfaz gráfica, el usuario puede decidir si desea que se generen y se muestren por pantalla todos los gráficos resultantes. Si el usuario escoge la opción "Sí", se mostrará una ventana de Matplotlib que contiene todos los gráficos generados a través del proceso de la Lógica Difusa.

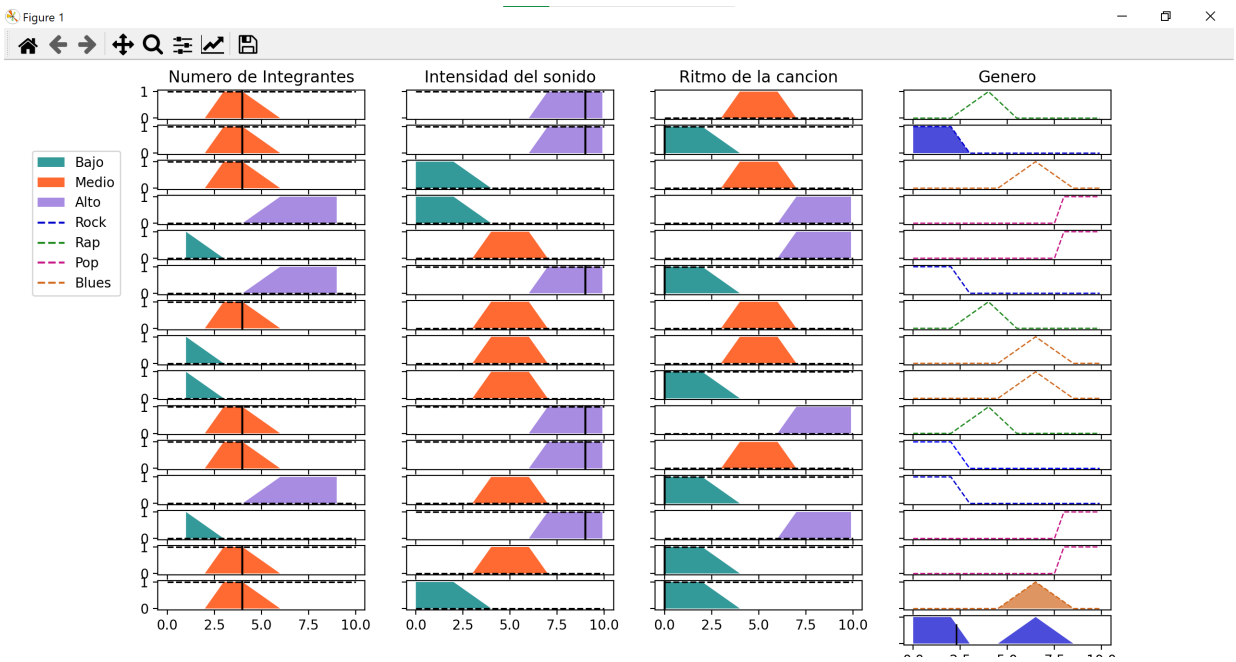

Figura 5: Ventana de los gráficos de resultados generados para un caso donde el género preferido es Rock.

Para comprender mejor los gráficos, estos se realizaron utilizando colores: bajo se representa mediante verde agua, medio mediante el naranjo y alto mediante el morado. En la primera columna se tiene el número de integrantes para cada regla, lo mismo para la segunda columna, se tiene la intensidad del sonido y en la tercera columna esta el ritmo. Con una linea negra se representa el valor obtenido a través de la función de pertenencia, que también se relaciona con la entrada. En la última columna, se tienen los géneros, obtenidos a través de la agregación. En el último gráfico de la columna de géneros, está el gráfico resultante final, el cual se marca del color correspondiente al género preferido: azul representa Rock, verde a Rap, rosado a Pop y café chocolate a Blues.

Probando distintas entradas se pueden obtener distintos gráficos. El valor que se marca en el gráfico final representa el valor de defuzzificación encontrado. Es posible que las entradas den lugar a un resultado donde mas de un género parece ser preferido y es ahí donde el valor de defuzzificación juega un rol importante, ya que es el encargado de finalmente decidir cual es el género preferido.

3.3. Análisis de modelos

Con lo anterior explicado, se utilizará el ejemplo con las variables 4 integrantes, 9 de intensidad y 1 de ritmo, que da un resultado de Rock con sus correspondientes canciones con algunas otras recomendaciones, esta respuesta se puede relacionar con la realidad con los datos comunes entre las bandas que producen música rock[**1**], con esto dicho se puede relacionar esta cantidad pedida con las bandas de rock debido a que estas mismas generalmente son de 4 integrantes, pasando con la intensidad de 9, esto se compara con las canciones producidas en este género ya que tienden a ser más alta en decibelios que las demás canciones que se producen. Y por último el ritmo 1 se puede comparar con las canciones de rock debido a que estás a diferencia de las demás canciones que tienen un ritmo fluido, las de rock generalmente son poco fluidas.

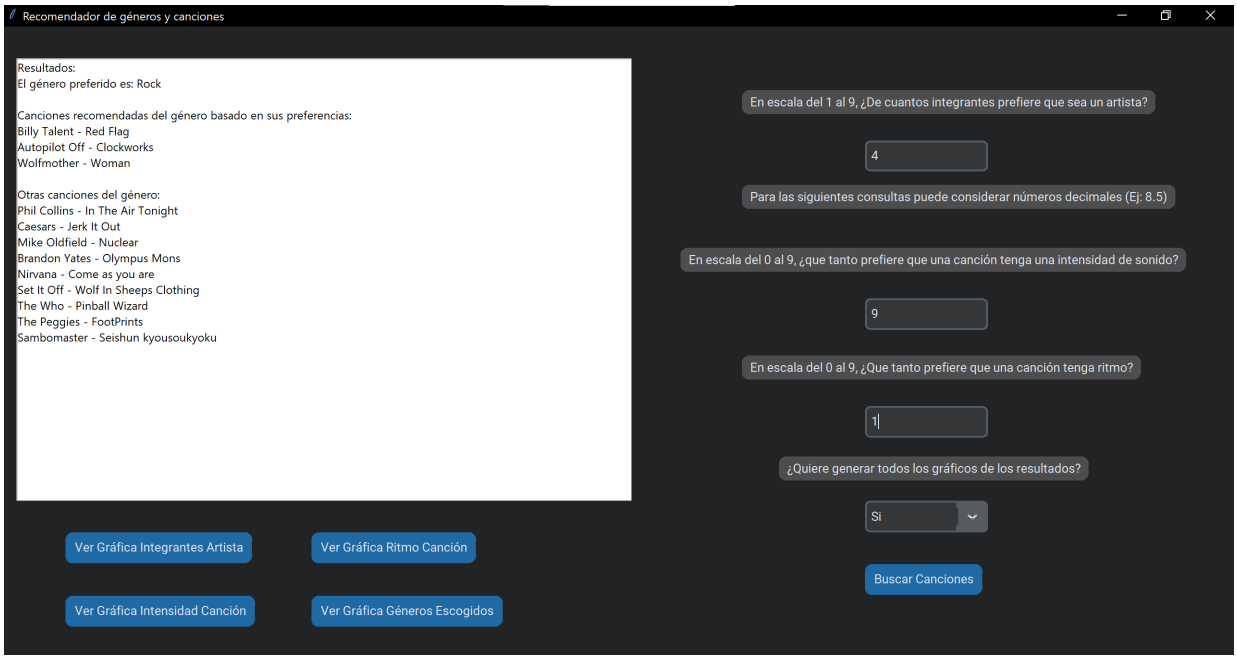

Figura 6: Ejemplo 1 - Rock.

Los gráficos resultantes para el ejemplo anterior corresponden a la figura 5. Con todo lo mencionado anteriormente, se puede concluir que este modelo es adecuado, ya que representa la realidad de la mayoría de las bandas de Rock.

Continuando con otro ejemplo, en este se utilizan las siguientes variables: 1 integrante, 1 de intensidad, 9 de ritmo. Con la información ya proporcionada en el programa

da como resultado 'Pop' al igual que el resultado anterior entregaría una canción que se ajuste con el dato entregados, con eso dicho se puede relacionar estas entradas con lo que generalmente se puede considerar como género Pop, pasando a los integrantes en el pop especialmente en la actualidad[**2**] se puede tener pocos integrantes debido a la posibilidad de utilizar programas que pueden hacer música, respecto a la intensidad, como estas canciones no son demasiado fuerte en sonido si no que son para más bailar, no son tan intensas en la mayoría de canciones de Pop, y por último sobre el ritmo, como se dijo anteriormente esta son usadas principalmente para bailar, ya que las canciones tienen mucho ritmo para que la gente baile.

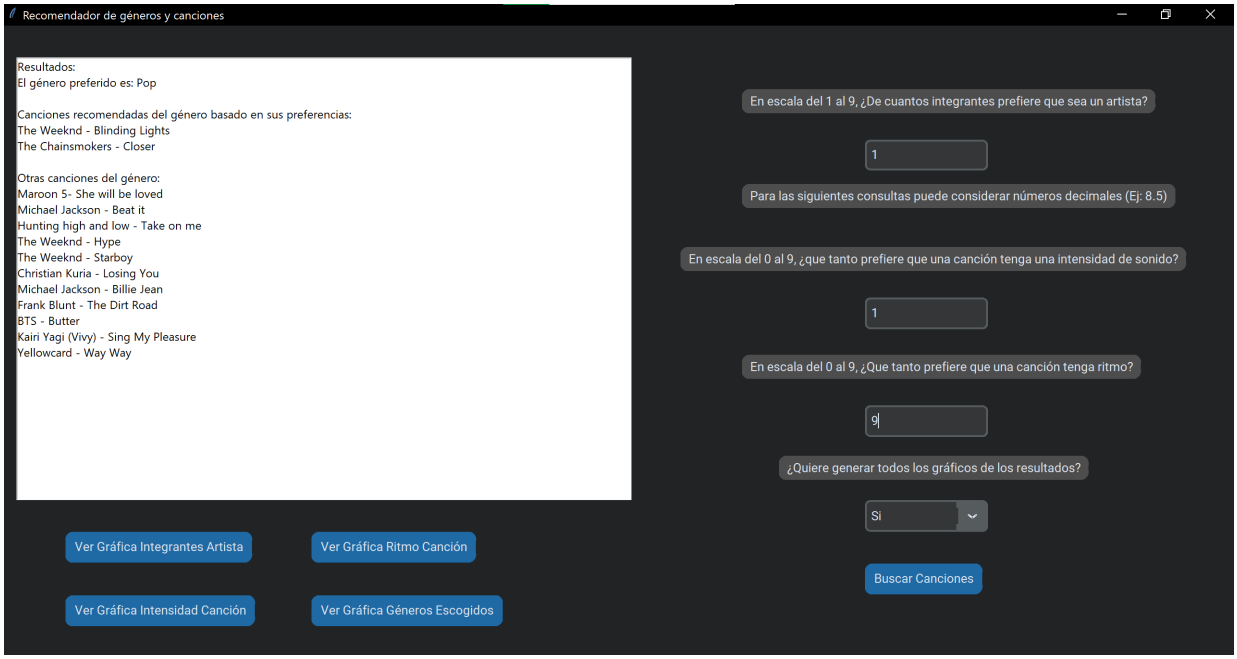

Figura 7: Ejemplo 2 - Pop.

De lo anterior se obtienen las siguientes gráficas:

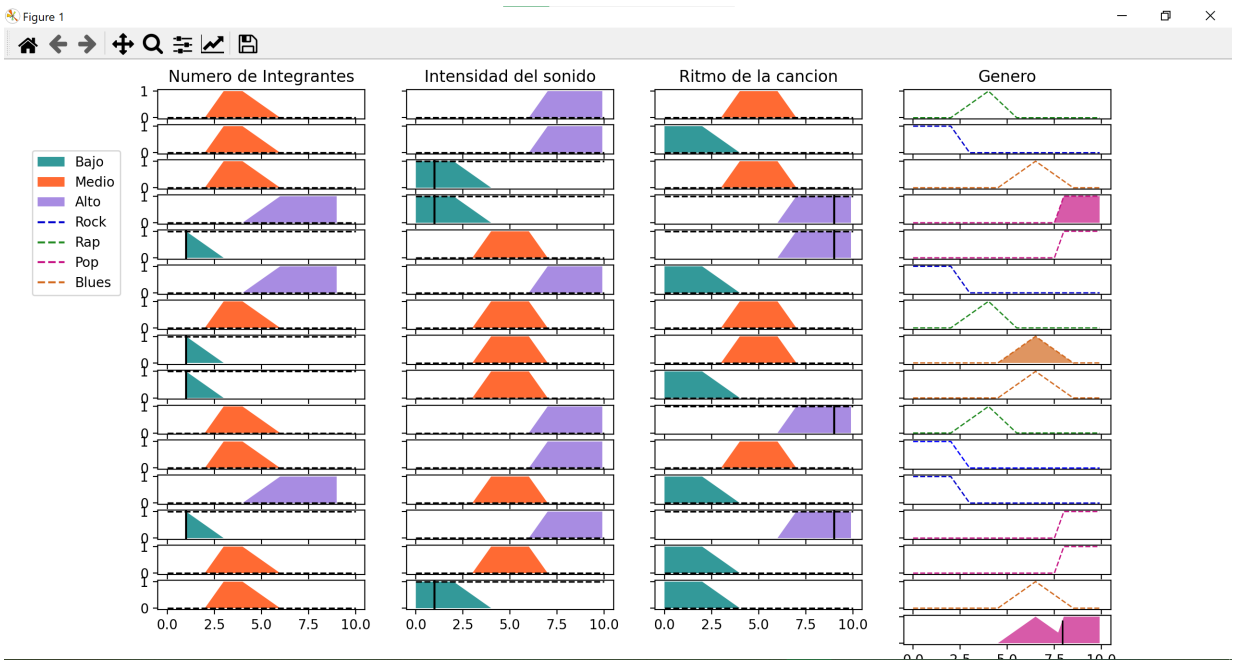

Figura 8: Gráficos obtenidos de un caso donde el género preferido es Pop.

Se puede notar que para la tercera agregación, se obtiene Pop cuando se tienen muchos integrantes, baja intensidad y alto ritmo. Esto también refleja la realidad, ya que hay grupos de Pop donde se tienen muchos integrantes (Los grupos Coreanos son un gran ejemplo), pero el ritmo no deja de ser menor, lo cual es lo que mas destaca en la música Pop. Por lo tanto, nuevamente se puede decir que se tiene un buen modelo, ya que refleja la realidad de la música Pop.

3.4. Relación con la teoría

En el anterior informe se vio la Lógica de Primer Orden (LPO) que se caracteriza por conocer completamente los valores y no lleva nada de ambigüedad; la Lógica Difusa, al contrario, permite la posibilidad de trabajar y/o manipular la noción de ambigüedad.

Pasando a la relación de esta lógica con la teoría, se recibe información que no se sabe cuál es la exacta forma de clasificarla, como la cantidad de integrantes de una banda (por ejemplo, cuando se consideraría que son muchos en una banda) para luego, utilizando variables de pertenencia se le da a estos valores un grado de conocimiento que nos permite saber que es lo que quiere el usuario. Tras tener estos "valores de pertenencia", se utilizan las "funciones de pertenencia"que en conjunto a los "valores de pertenencia", permiten ver

la "pertenencia", es decir, que tanto grado es designado por un valor (ejemplo: mucho, poco) a los datos recibidos, finalizando con esto se utiliza la defuzzificación que permite determinar los valores en todos los puntos de pertenencia para así obtener un valor final y designar cuál es el género al que le pertenece. Por lo tanto, la Lógica Difusa es una poderosa herramienta a la hora de querer comprobar si alguna información refleja la realidad.

Para ver más gráficas sobre resultados, ver figuras 10 y 11 en Anexos.

4. Conclusión

Se cumplieron los objetivos planteados: se obtuvieron los datos del usuario; se creó una base de información y reglas sobre géneros musicales junto con canciones que pertenecen a estos; se procesaron estos datos a través de la fusificación y sistema de inferencia para, posteriormente, defuzzificar y realizar la recomendación musical al aplicar la inferencia de Mamdani durante el trayecto de este cálculo.

Para este taller la principal limitante fue la parametrización de las canciones y géneros musicales, en primer lugar, ya que fue necesario establecer de manera arbitraria los rangos de valores considerados para los atributos que se aplican a cada canción; y, en segundo lugar, cuando se realizó la estimación numérica para calificar la ocasión en que se recomendaría un género u otro.

El código aplica Lógica Difusa gracias a las librerías externas de Python como Scikit-fuzzy. Este programa se encarga, primeramente, de identificar y procesar los datos entregados por el usuario y, posteriormente, mostrar los resultados en la interfaz gráfica donde el usuario verá las canciones recomendadas según sea el caso. Además, entrega los gráficos asociados al cálculo, si así lo desea el usuario.

Una mejora para este proyecto sería que el usuario pueda acceder a una muestra de lo que el programa considera como los extremos de los parámetros de las características de una canción, como integrantes, ritmo e intensidad; con el fin de que el operador provea información precisa y así obtenga mejores resultados.

En el taller anterior se utilizó la Lógica de Primer Orden, donde se trabajo con el lenguaje de programación Prolog, junto con Python. Si se realiza una rápida comparación, se puede observar que es posible realizar una tarea utilizando LPO o Lógica Difusa, como es el caso de el trabajo realizado en los Talleres 1 y 2. Pero la gran diferencia es que para la Lógica Difusa, los atributos a considerar para sugerir canciones y géneros son mas precisos, ya que se relacionan más con la realidad que en el caso de LPO.

Se espera que la experiencia de este taller sirva para futuros proyectos donde se deba utilizar la Lógica Difusa.

5. Bibliografía y Referencias

[**1**] PromocionMusical.es. (2019). “Las 100 Mejores Bandas de Rock”. Articulo Online. Recuperado de: https://promocionmusical.es/mejores-bandas-de-rock

[**2**] Fernández, M. (2021). ”10 cantantes de pop actuales que tienes que conocer”. Reportaje Online. Recuperado de: https://www.espectaculosbcn.com/cantantes-de-pop-actuales/

6. Anexos

README.md

```
Instrucciones de Uso:
1. Este programa funciona con cualquier versión de Python 3. Sin embargo, para poder ejecutarlo, se
deben tener algunas  extensiones, las cuales se instalan utilizando PIP en la CMD de Windows (O terminal
de linux). Todas las extensiones debe ser instaladas, utilizando los comandos que se describen a continuación.
* Primero, se debe instalar Numpy. Se debe utilizar "pip install numpy"
* La segunda extensión necesaria es Matplotlib. Se instala con "pip install matplotlib"
* La tercera extensión necesaria es Scikit-Fuzzy. Se instala con "pip install scikit-fuzzy"
* La cuarta extensión necesaria es Tkinter. Se instala con "pip install tk"
* La última extensión necesario es Custom Tkinter. Se instala con "pip install customtkinter"
2. Con las extensiones ya instaladas, se debe tener la carpeta src del código, donde dentro de ella se
encuentra el archivo "taller2.py"
3. Se ejecuta el archivo "taller2.py" (Presionar F5 si se esta en IDLE o Ejecutar Codigo si se esta utilizando
Visual Studio Code). Si todas las extensiones fueron instaladas correctamente entonces se abrirá la
interfaz gáfica
4. El siguiente paso es colocar los valores preferidos para la cantidad de integrantes, intensidad y ritmo.
* Si lo desea, el usuario puede visualizar las distintas gráficas de escalas presionando los botones
correspondientes. Note que si abre una ventana de gráfico, deberá cerrarla para abrir otra ventana de gráfico.
* Además de los valores, el usuario debe escoger si desea poder visualizar todos los gráficos de resultados
generados.
5. Con los valores ya colocados, al presionar el boton "Buscar Canciones", la interfaz gráfica pasará los datos
ingresados al programa. El programa mostrará, mediante la interfaz, el género preferido junto con las
canciones recomendadas del género.
* Notar que la cantidad de integrantes deben ser valores enteros de 1 al 9. La intensida y ritmo pueden
ser números enteros o decimales del 0 al 9. En caso de que lo anterior no se respete, el programa mostrará
un mensaje para que el usuario pueda volver a ingresar las entradas.
6. Repetir desde el paso 4 hasta que se desee cerrar la interfaz gráfica, finalizando la ejecución
del programa.
```

Figura 9: Archivo README.md que contiene las instrucciones de ejecución del programa.

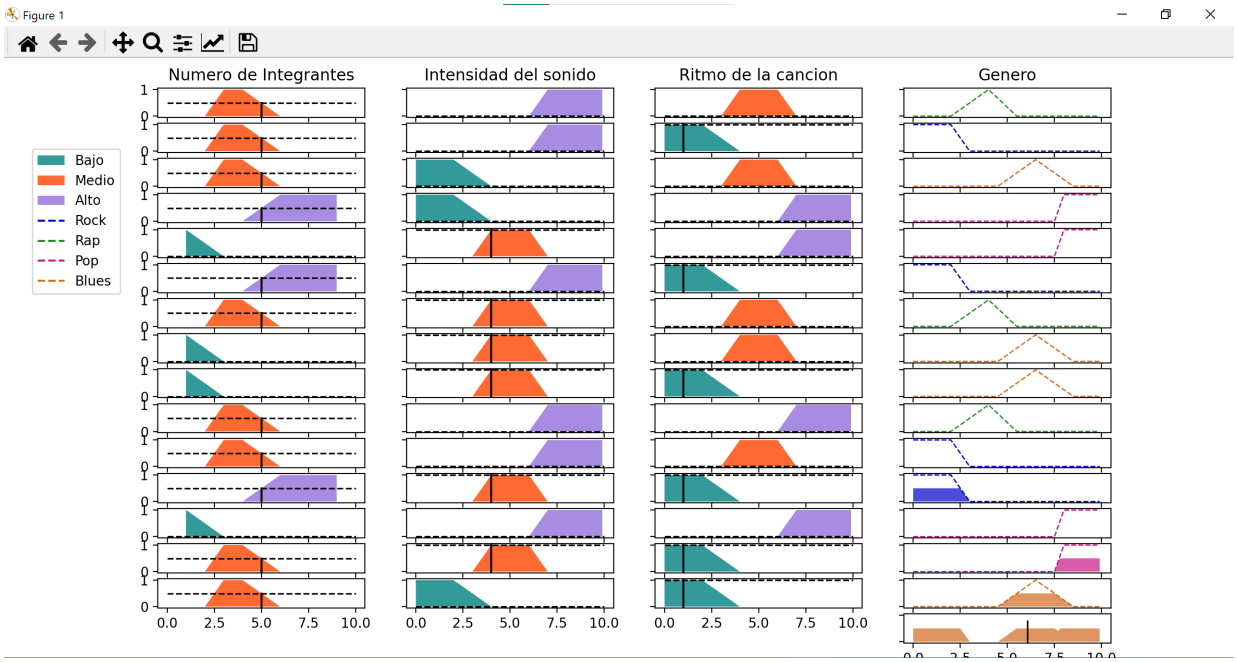

Figura 10: Gráficos resultantes en un caso donde el género preferido es Blues.

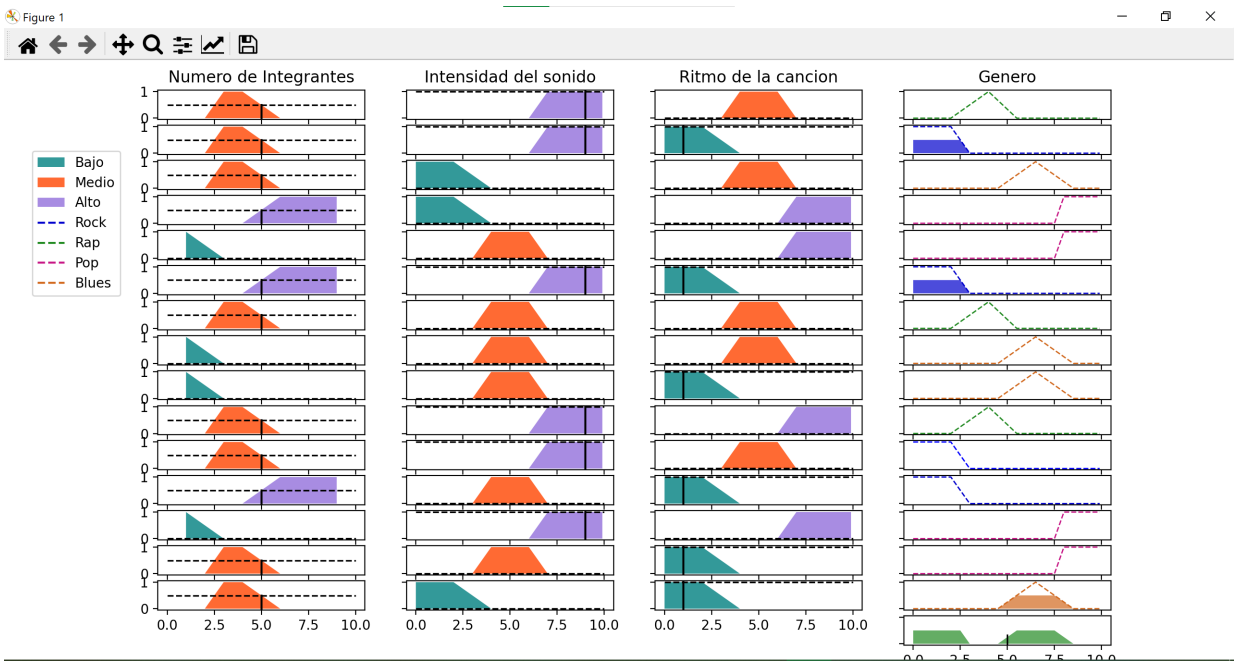

Figura 11: Gráficos resultantes en un caso donde el género preferido es Rap.

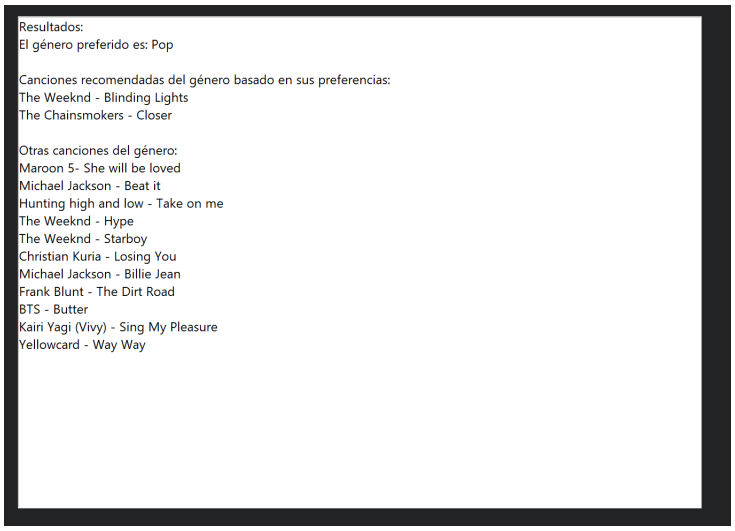

Figura 12: Una vista mas clara a los resultados en un caso donde el género preferido es Pop.