Trabajo Final - Diseño de Sistema Experto
Algoritmia y Lógica Computacional

Aboulafia Gerardo,
Barquet Amélie

Diciembre 2024

Indice

1. Introducción

En la era de la transformación digital, los sistemas expertos se han convertido en herramientas esenciales para resolver problemas complejos al emular el razonamiento humano en dominios específicos. Los sistemas expertos son aplicaciones informáticas que incorporan cierta experiencia no algorítmica para resolver ciertos tipos de problemas. Los sistemas expertos tienen una serie de componentes principales y se relacionan con individuos en varios roles[1]. Estos componentes principales son:

- **Base de conocimientos**: una representación declarativa de la experiencia, a menudo en reglas SI ENTONCES;
- **Almacenamiento de trabajo**: los datos que son específicos del problema que se está resolviendo;
- **Motor de inferencia**: el código en el núcleo del sistema, que deriva recomendaciones de la base de conocimientos y los datos específicos del problema en el almacenamiento de trabajo;
- **Interfaz de usuario**: el código que controla el diálogo entre el usuario y el sistema.

Spotify es una de las plataformas de streaming de música más populares a nivel global, ofreciendo acceso a millones de canciones, podcasts y videos de artistas de todo el mundo. Proporciona experiencias personalizadas a través de sus algoritmos de recomendación, que sugieren nueva música basada en los hábitos de escucha de los usuarios.

En este proyecto, nuestro objetivo es diseñar un sistema experto que recomiende canciones a los usuarios basándose en sus preferencias musicales, utilizando datos de Spotify. El sistema genera una radio de canciones, que en el contexto de los servicios de streaming de música, se refiere a una lista de reproducción curada de pistas similares en género, estilo o características a una canción. Para ello, el sistema experto le solicitará el género del que quiere que sea la radio; luego, una canción en la que el usuario quiere basar la radio, y por último el artista que la interpreta (de esta manera se soluciona el problema de canciones con mismo nombre de artistas diferentes). Esta funcionalidad permite a los usuarios descubrir nueva música que se alinea con sus gustos.

Nuestro sistema experto aprovecha nuestro propio historial de escucha para crear una base de conocimientos, asegurando que las recomendaciones no solo se basen en la popularidad general, sino también en nuestras preferencias personales. Al integrar datos de la API de Spotify, extraemos características musicales detalladas de cada pista, que luego se utilizan en algoritmos como K-Nearest Neighbors (KNN) para encontrar y recomendar canciones similares a las que el usuario ya disfruta.

[1] Merritt, D. (2000). *Building Expert Systems in Prolog*. Amzi! Inc.

El proyecto abarca la creación de un autómata finito determinista (AFD) para modelar el flujo lógico del sistema, garantizando una interacción estructurada y predecible con el usuario. Además, analizamos la complejidad del sistema, enfocándonos en el rendimiento de las consultas a la base de conocimientos y la eficiencia del algoritmo KNN al manejar grandes conjuntos de datos.

2. Generación de la Base de Conocimientos

En el contexto de un sistema experto, la base de conocimientos es un componente fundamental que almacena información y hechos sobre un dominio específico, permitiendo al sistema realizar inferencias y tomar decisiones informadas. En nuestro sistema, la base de conocimientos consiste en una colección de datos sobre canciones, artistas y características musicales, basada en las canciones que más escuchamos nosotros. De esta manera, el sistema actúa como si nosotros mismos estuviéramos recomendando canciones al usuario, proporcionando recomendaciones personalizadas y alineadas con nuestras preferencias musicales.

2.1. Utilización de la API de Spotify

Para construir una base de conocimientos robusta y representativa de nuestras preferencias, optamos por extraer información de la API de Spotify, un servicio que permite acceder programáticamente a los datos de Spotify, como canciones, artistas, álbumes y características de audio. La API proporciona endpoints que facilitan la recuperación de datos detallados sobre pistas musicales, incluyendo métricas como bailabilidad, energía, tempo y más.

Es importante mencionar que la API de Spotify tiene una limitación que permite solicitar características de audio para un máximo de 100 canciones por petición y 200 canciones al obtener pistas de una playlist. Para superar esta restricción y evitar saturar cada consulta, implementamos estrategias de ciclos que nos permitieron iterar sobre nuestras playlists y realizar múltiples solicitudes sin exceder los límites establecidos por la API.

2.2. Proceso de Extracción y Transformación de Datos

El proceso inició con la selección de 12 playlists de música que habitualmente escuchamos, asegurando así una base de conocimientos alineada con nuestras preferencias musicales reales y variadas. Finalmente, contamos con 3256 canciones de 10 géneros diferentes (pop, rap, rock, reggaetón, salsa, jazz, r&b, soul, indie, electro). A continuación, detallamos los pasos seguidos:

1. **Extracción de Canciones de las Playlists:** Utilizamos el endpoint de la API de Spotify para obtener todas las canciones contenidas en cada playlist seleccionada. Debido a la limitación de 200 canciones por petición, implementamos un ciclo que segmentaba las solicitudes en bloques

manejables, asegurando la recuperación de todas las pistas sin exceder los límites de la API.

Listing 1: Función para obtener pistas de la playlist con manejo de paginación

```
    tracks = []
    offset = 0
    limit = 100  # Maximo permitido por solicitud
    while True:
        response = spotify_api.playlist_tracks(
            playlist_id, limit=limit, offset=offset)
        tracks.extend(response['items'])
        if response['next']:
            offset += limit
        else:
            break
    return tracks
```

2. **Procesamiento de Canciones y Obtención de Géneros:** Una vez extraídas las canciones, procesamos cada pista para obtener información relevante, como el nombre, el artista, el álbum y los géneros asociados al artista. Este paso es crucial para categorizar las canciones según su género musical, lo cual es fundamental para el estado q_1 del sistema.

3. **Obtención de Características de Audio:** Utilizando los track IDs de las canciones, solicitamos a la API de Spotify las características de audio de cada pista. Dado que la API solo permite obtener características de hasta 100 canciones por solicitud, implementamos un ciclo que dividía la lista de IDs en lotes de 100.

   Listing 2: División de track_ids en lotes de 100 para obtener características de audio

```
def get_audio_features(track_ids):
    audio_features = []
    for i in range(0, len(track_ids), 100):
        batch = track_ids[i:i+100]
        features = spotify_api.audio_features(batch)
        audio_features.extend(features)
    return audio_features
```

   Las características de audio extraídas incluyen:

   - **Danceability (Bailabilidad):** Indica qué tan adecuada es una pista para bailar, basada en una combinación de elementos musicales como tempo, estabilidad del ritmo, fuerza del pulso y regularidad general. Un valor más alto significa que es más bailable.
   - **Energy (Energía):** Representa una medida perceptual de intensidad y actividad. Las pistas energéticas suelen sentirse rápidas, ruidosas y rítmicamente intensas.

- **Key (Clave):** La tonalidad estimada de la pista. Se representa como un entero entre 0 y 11, donde cada número corresponde a una nota musical específica.
- **Loudness (Volumen):** El nivel promedio de decibelios (dB) de la pista.
- **Mode (Modo):** Indica la modalidad (mayor o menor) de una pista, representada por 0 (menor) y 1 (mayor).
- **Speechiness (Discurso):** Detecta la presencia de palabras habladas en una pista.
- **Acousticness (Acústica):** Mide qué tan acústica es una pista.
- **Instrumentalness (Instrumentalidad):** Predice si una pista no contiene voces.
- **Liveness (Vitalidad):** Detecta la presencia de una audiencia en la grabación.
- **Valence (Valencia):** Describe la positividad musical transmitida por una pista.
- **Tempo:** El tempo general estimado de una pista en pulsaciones por minuto (BPM).

Estas características son esenciales para el algoritmo de K-Nearest Neighbors (KNN) utilizado en el sistema.

4. **Integración de Datos:** Combinamos las características de audio con los datos iniciales de las canciones, enriqueciendo así cada entrada con información detallada que será utilizada en las recomendaciones.

5. **Creación de DataFrames y Consolidación:** Para facilitar el manejo y manipulación de los datos, convertimos la información en DataFrames utilizando la librería `pandas`. Concatenamos los DataFrames de las diferentes playlists para formar una base de conocimientos unificada.

Listing 3: Creación y consolidación de DataFrames

```
# Creaci n de un DataFrame para una vista tabular de
    los datos
df_playlist = pd.DataFrame(songs_data)

# Concatenaci n de DataFrames
df_consolidated = pd.concat([df1, df2, ..., df12],
    ignore_index=True)
```

2.3. Conversión a Hechos Prolog

Con los datos consolidados, procedimos a convertir la base de conocimientos en un formato compatible con Prolog. Para ello:

1. **Formateo de Datos:** Aseguramos que todos los valores de texto estuvieran correctamente escapados para evitar errores en Prolog.

Listing 4: Escapado de comillas simples en los valores

```
# Escapado de comillas simples en los valores
row = row.apply(lambda x: str(x).replace("'", "\\'") if
    isinstance(x, str) else x)
```

2. **Generación de Hechos:** Creamos una representación en Prolog de cada canción, definiendo un predicado `song` que incluye todos los atributos relevantes.

```
song(
  genre(Género),
  name('Nombre de la Canción'),
  artist('Nombre del Artista'),
  album('Nombre del Álbum'),
  popularity(Popularidad),
  artist_popularity(PopularidadArtista),
  danceability(Bailabilidad),
  energy(Energía),
  key(Clave),
  loudness(Volumen),
  mode(Modo),
  speechiness(Discurso),
  acousticness(Acústica),
  instrumentalness(Instrumentalidad),
  liveness(Vitalidad),
  valence(Valencia),
  tempo(Tempo)
).
```

3. **Escritura en Archivo .pl:** Iteramos sobre cada fila del DataFrame y escribimos los hechos en un archivo `.pl`, que posteriormente es cargado en Prolog.

Listing 5: Función para generar hechos Prolog desde el DataFrame

```
def dataframe_to_prolog_facts(df, file_path):
    with open(file_path, 'w') as f:
        for _, row in df.iterrows():
            fact = f"song(genre({row['genre']}),␣name('{
                row['name']}'),␣...).\n"
            f.write(fact)
```

2.4. Beneficios y Justificación

La utilización de la API de Spotify y la conversión de los datos a Prolog nos proporcionó una base de conocimientos:

- **Personalizada:** Basada en las canciones que más escuchamos nosotros, lo que mejora la relevancia y autenticidad de las recomendaciones.
- **Rica en Información:** Con detalles cuantitativos y cualitativos de cada canción.
- **Escalable:** La metodología permite agregar más playlists y actualizar la base de conocimientos según sea necesario.

2.5. Consultas a la Base de Conocimientos en Prolog

A continuación, se detallan algunas consultas comunes realizadas en el sistema:

- **Búsqueda de canciones por género musical:**

Listing 6: Consulta para buscar canciones por género musical

```
song(genre('pop'), name(SongName), artist(ArtistName), _
    , _, _, _, _, _, _, _, _, _, _, _, _).
```

- **Identificación de canciones con alta bailabilidad:**

Listing 7: Consulta para identificar canciones con alta bailabilidad

```
song(_, name(SongName), _, _, _, danceability(
    Danceability), _, _, _, _, _, _, _, _, _, _),
    Danceability > 0.8.
```

- **Listado de canciones populares de un artista específico:**

Listing 8: Consulta para listar canciones populares de un artista específico

```
song(_, name(SongName), artist('Nombre del Artista'), _,
     popularity(Popularity), _, _, _, _, _, _, _, _, _,
    _, _, _), Popularity > 70.
```

Estas consultas demuestran la flexibilidad de Prolog y su utilidad para construir recomendaciones basadas en características específicas.

3. Implementación de KNN para Encontrar Canciones Similares

3.1. Descripción Matemática del Algoritmo KNN

El algoritmo *K-Nearest Neighbors* (KNN) es un método de aprendizaje supervisado que se utiliza tanto para clasificación como para regresión. En nuestro caso, lo aplicamos para encontrar canciones similares basándonos en ciertas características cuantitativas extraídas de cada pista. El objetivo es, dada una canción de referencia, identificar las K canciones más cercanas en términos de distancia en el espacio de características.

Matemáticamente, el algoritmo KNN opera de la siguiente manera:

1. **Representación de las Canciones en un Espacio de Características:** Cada canción se representa como un vector en un espacio R^n, donde n es el número de características consideradas.

2. **Definición de la Métrica de Distancia:** Utilizamos la distancia euclidiana para medir la similitud entre canciones. La distancia euclidiana entre dos vectores $x = (x_1, x_2, \ldots, x_n)$ y $y = (y_1, y_2, \ldots, y_n)$ se define como:
$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

3. **Normalización de las Características:** Para asegurar que todas las características contribuyan de manera equitativa al cálculo de la distancia, aplicamos una normalización *min-max* a cada una de ellas. La normalización se realiza de la siguiente manera:
$$x_i^{\text{norm}} = \frac{x_i - x_i^{\min}}{x_i^{\max} - x_i^{\min}}$$
donde $x_i^{\min}$ y $x_i^{\max}$ son el valor mínimo y máximo de la característica i en el conjunto de datos.

4. **Cálculo de Distancias:** Para la canción de referencia x, calculamos la distancia a cada una de las canciones y_j en el conjunto de datos filtrado por género:
$$d(x, y_j) = \sqrt{\sum_{i=1}^{n} \left(x_i^{\text{norm}} - y_{j,i}^{\text{norm}}\right)^2}$$

5. **Selección de los K Vecinos Más Cercanos:** Ordenamos las canciones y_j en función de la distancia $d(x, y_j)$ y seleccionamos las K canciones con las distancias más pequeñas.

6. **Generación de la Recomendación:** Las K canciones seleccionadas se presentan al usuario como recomendaciones similares a la canción de referencia.

3.2. Normalización de Datos con Min-Max

La normalización es un paso crucial en nuestro modelo, ya que las características extraídas de las canciones pueden tener escalas muy diferentes. Por ejemplo, la característica *Tempo* puede variar entre 0 y 250, mientras que la *Valence* está en un rango de 0 a 1. Sin normalización, las características con valores numéricos más grandes tendrían una influencia desproporcionada en el cálculo de la distancia euclidiana.

Aplicamos la normalización *min-max* a cada característica individualmente, escalando sus valores al rango [0, 1]. Esto asegura que cada característica contribuya de manera equitativa al cálculo de la distancia, evitando sesgos en las recomendaciones.

3.3. Selección de Características

Para el modelo KNN, seleccionamos las siguientes características cuantitativas proporcionadas por la API de Spotify, las cuales ya fueron descritas anteriormente: **Popularity**, **Danceability**, **Energy**, **Key**, **Loudness**, **Mode**, **Speechiness**, **Acousticness**, **Instrumentalness**, **Liveness**, **Valence** y **Tempo**.

3.4. Ajustes y Refinamiento del Modelo

En un primer intento, incluimos la característica *Artist Popularity* en el conjunto de características utilizadas por el modelo. Sin embargo, al realizar pruebas con esta configuración, observamos que las recomendaciones generadas tendían a incluir varias canciones de los mismos artistas repetidamente. Esto reducía la variedad en las recomendaciones y limitaba el descubrimiento de nueva música por parte del usuario. La Figura 1 ilustra este comportamiento, donde se puede apreciar que las canciones recomendadas repiten el artista varias veces. Se pueden observar solamente 5 artistas a lo largo de la radio de 10 canciones.

A diferencia de esto, al eliminar la característica *Artist Popularity* del conjunto de características, obtuvimos recomendaciones más variadas en términos de artistas. La Figura 2 muestra que las canciones recomendadas provienen de diferentes artistas, aunque mantienen similitudes en sus características musicales con la canción de referencia. En esta oportunidad, ningún artista se repite en la radio (salvo por el artista ingresado por el usuario).

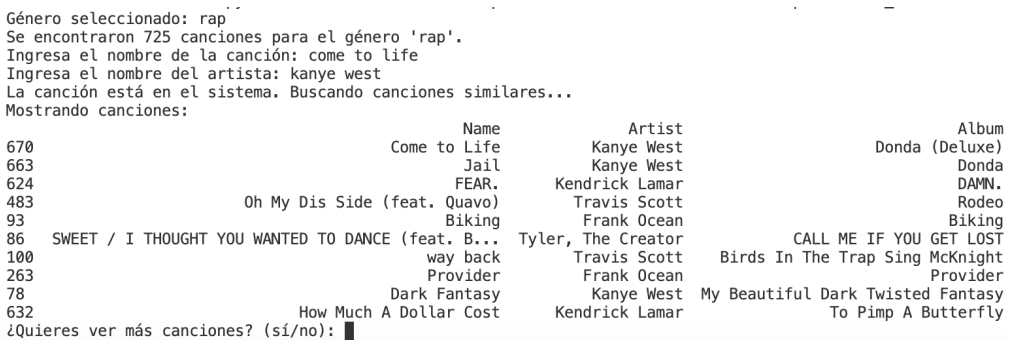

Figura 1: Radio generada incluyendo la característica *Artist Popularity*.

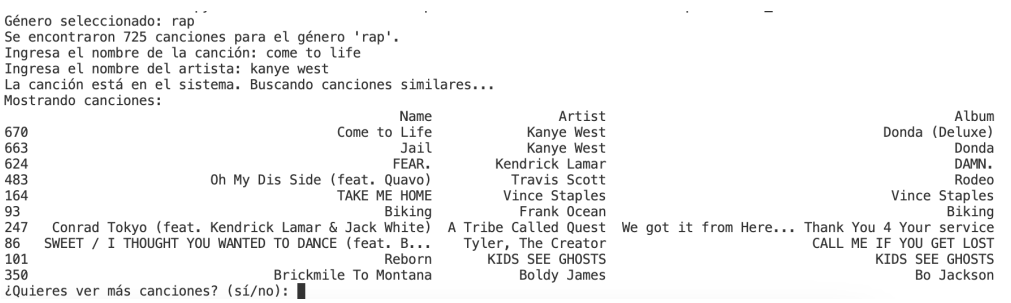

Figura 2: Radio generada sin incluir la característica *Artist Popularity*.

Nuestro objetivo es que la radio permita al usuario descubrir música más variada y no limitarse a los mismos artistas. Por ello, decidimos eliminar la característica *Artist Popularity* de nuestro conjunto de características.

Tras este ajuste, el modelo KNN comenzó a generar recomendaciones más diversas, incluyendo canciones de diferentes artistas que comparten características musicales similares con la canción de referencia. Esto mejora la experiencia del usuario al permitirle explorar y descubrir nueva música dentro del género seleccionado.

3.5. Implementación

La implementación del algoritmo KNN se realizó utilizando la librería scikit-learn de Python, que proporciona herramientas eficientes para el aprendizaje automático. El procedimiento seguido fue:

1. **Preparación de los Datos:** Los datos de las canciones filtradas por género se cargan en un *DataFrame* de *pandas*.

2. **Normalización:** Aplicamos la normalización *min-max* a las características seleccionadas.

3. **Entrenamiento del Modelo:** Utilizamos el modelo `NearestNeighbors` de `scikit-learn`, ajustándolo con los datos normalizados.

4. **Consulta al Modelo:** Dada la canción de referencia, extraemos sus características normalizadas y consultamos el modelo para obtener las K canciones más cercanas.

5. **Presentación de Resultados:** Las canciones recomendadas se presentan al usuario, mostrando información relevante como el nombre de la canción, el artista y el álbum.

3.6. Resultados y Observaciones

Tras la implementación y ajustes realizados, el sistema es capaz de generar recomendaciones de canciones que son similares a la canción de referencia en términos de sus características musicales, pero que provienen de una variedad más amplia de artistas. Esto enriquece la experiencia del usuario y cumple con el objetivo de ayudar a descubrir nueva música dentro de sus preferencias.

La eliminación de la característica *Artist Popularity* resultó ser una decisión clave para mejorar la diversidad en las recomendaciones. Este hallazgo resalta la importancia de seleccionar adecuadamente las características en un modelo de aprendizaje automático, considerando no solo su impacto en la precisión, sino también en la calidad y utilidad de los resultados para el usuario final.

4. Autómata Finito Determinista

Un autómata finito determinista (AFD) es un modelo matemático que sólo puede estar en un único estado después de leer cualquier secuencia de entradas. La característica clave de un AFD es su determinismo, para cada estado y cada símbolo de entrada, existe exactamente una transición posible hacia otro estado. En otras palabras, para cada símbolo procesado, hay solo un estado al que el autómata puede hacer transición a partir de su estado actual.

Un AFD se define formalmente mediante sus cinco componentes $A = (Q, \Sigma, \delta, q_0, F)$ un autómata finito determinista donde:

- Q: Conjunto finito de estados.
- Σ: Conjunto finito de símbolos de entrada.
- δ: Función de transición, δ se representa mediante arcos entre los estados y las etiquetas sobre los arcos. Si q es un estado y a es un símbolo de entrada, entonces $\delta(q, a)$ es el estado p tal que existe un arco etiquetado a que va desde q hasta p.
- q_0: Estado inicial, $q_0 \in Q$.

- F: Conjunto de estados finales, $F \subseteq Q$.

En este trabajo, los componentes del AFD toman los siguientes valores:

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$.
- $\Sigma = \{\text{Género}, \text{Canción}\}$.
- Estado inicial: q_0.
- Estados finales: $F = \{q_7\}$.

4.1. Estados del AFD

La Figura 1 muestra las transiciones entre los estados del AFD.

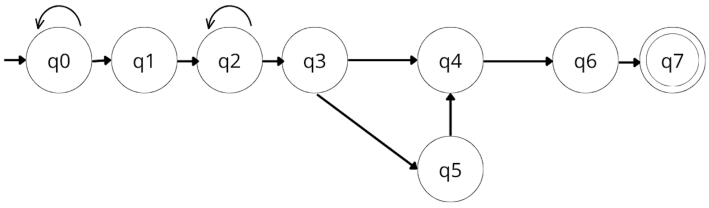

Figura 3: Estados del AFD generado.

A continuación, se describen las funciones de cada estado:

- Estado q_0: El sistema espera que el usuario ingrese un género musical. Si no se ingresa ningún género, se realiza una transición de bucle al mismo estado. Al ingresar un género, se avanza al estado q_1.
- Estado q_1: El sistema filtra las canciones en la base de conocimiento según el género ingresado. El conjunto resultante servirá como base para comparar la canción proporcionada por el usuario.
- Estado q_2: Se solicita al usuario que proporcione una canción. Si no se ingresa ninguna canción, el sistema permanece en este estado. Si se ingresa una canción, se avanza al estado q_3.
- Estado q_3: El sistema verifica si la canción proporcionada ya está en la base de conocimiento filtrada por género: si la canción está en la playlist, se avanza al estado q_4. Si la canción no está, el sistema pasa al estado q_5 realiza una búsqueda en la API de Spotify, la agrega a la base de conocimiento, y luego avanza al estado q_4.
- Estado q_4: Una vez que la canción está en la playlist o ha sido añadida desde Spotify, se devuelve una lista de 10 canciones recomendadas (radio) a través de un KNN. Este compara canciones considerando características extraídas de la API de Spotify.

- Estado q_5: El sistema busca la canción a través de la API de Spotify, la agrega a la base de conocimiento y pasa a q_4. En este estado, el sistema hace un request a la API sobre las búsquedas que coinciden con el nombre y el artista ingresado por el usuario. Ordena los resultados en forma descendente por *Popularity* y elige la primera fila para pasar al siguiente estado.

- Estado q_6: El sistema consulta al usuario si desea obtener más canciones: si la respuesta es afirmativa, se pasa al estado q_4. Si la respuesta es negativa, se avanza al estado q_7.

- Estado q_7: Estado final donde se concluye la consulta.

El diseño de este autómata garantiza que el sistema funcione de manera secuencial y determinista, adaptándose a las entradas proporcionadas por el usuario de manera eficiente.

5. Complejidad del Sistema

5.1. Complejidad de las Consultas al Sistema Experto

La base de conocimientos de nuestro sistema experto está construida en Prolog y contiene hechos que describen cada una de las 3657 canciones. Cada canción se representa como un hecho con sus características asociadas. Esto implica que el número total de hechos y reglas en la base de conocimiento (N) es igual al total de canciones almacenadas. Dado que cada consulta puede potencialmente revisar estos hechos, la complejidad en el peor caso está relacionada linealmente con N.

En el proceso de consulta, por ejemplo, cuando un usuario selecciona un género, el sistema carga todas las canciones asociadas a ese género. Supongamos que un género tiene aproximadamente G canciones ($G < N$). Si la consulta es más específica, como buscar una canción en particular, se pueden dar dos casos:

- Si la canción está en el sistema, la complejidad es $O(1)$, ya que Prolog verifica directamente si el hecho existe.

- Si la canción no está en el sistema, el sistema procede a buscarla en la API de Spotify, lo que implica operaciones adicionales fuera del ámbito de Prolog.

5.2. Complejidad de KNN

El algoritmo K-Nearest Neighbors (KNN) se utiliza para encontrar canciones similares basándose en características cuantitativas. Después de filtrar por género, trabajamos con un subconjunto de G canciones (donde $G < 3657$). Cada canción tiene D características (en nuestro caso, $D = 20$). Para encontrar

las K canciones más similares (con un valor $K = 10$), realizamos los siguientes cálculos:

El cálculo de distancias tiene una complejidad temporal de $O(G \cdot D)$. Esto se debe a que para cada una de las G canciones calculamos la distancia respecto a la canción de referencia utilizando las D características. Por ejemplo, si G es 500 y D es 20, se realizan $500 \cdot 20 = 10,000$ operaciones básicas.

Para optimizar el sistema, reducimos el espacio de búsqueda trabajando solo con canciones del mismo género, lo que disminuye N a G. También implementamos el algoritmo utilizando librerías optimizadas que aprovechan operaciones vectorizadas y cálculos en paralelo.

5.3. Ejemplo de complejidad para una ejecución

Supongamos que el género seleccionado tiene 500 canciones ($G = 500$). En este caso, se realizan 500 cálculos de distancia, cada uno con 20 operaciones (una por característica). El total de operaciones básicas es $500 \cdot 20 = 10,000$. Luego, con K=10, mantenemos un *heap* de 10 elementos para seleccionar las canciones más cercanas, lo que es eficiente tanto en tiempo como en recursos.

Tanto las consultas al sistema experto como el algoritmo KNN muestran un comportamiento eficiente con los datos actuales. En el caso del sistema experto, la complejidad es lineal y manejable gracias al tamaño moderado de la base de conocimiento y a las optimizaciones implementadas.

El algoritmo KNN, aunque adecuado para nuestro sistema, tiene limitaciones inherentes en términos de escalabilidad. Con datasets grandes o de alta dimensionalidad, KNN puede volverse computacionalmente costoso debido a su dependencia de cálculos de distancia en tiempo de consulta. Sin embargo, en nuestro caso, al trabajar con un tamaño de datos moderado ($G \approx 500$) y una dimensionalidad razonable ($D = 20$), el algoritmo proporciona tiempos de respuesta adecuados.

6. Evidencia y comprobación de funcionamiento

Para comprobar el funcionamiento del sistema experto diseñado se presentan algunos resultados clave. Concretamente, se muestran las pruebas de dos canciones seleccionadas que varían en artista y género. Estas pruebas permitieron observar empíricamente el desempeño del sistema bajo diferentes condiciones.

Uno de los aspectos fundamentales evaluados fue la coherencia de las recomendaciones generadas por el sistema, especialmente en relación con el género de la canción ingresada y el género seleccionado por el usuario. Los resultados muestran que, cuando la canción ingresada pertenece al mismo género que el seleccionado (por ejemplo, una canción del género Pop con el género Pop seleccionado), el sistema funciona de manera óptima, generando una radio con canciones que eran altamente relevantes y empíricamente similares a la canción de entrada. Esto puede observarse en la figura 4. Además en esta observamos

uno de los caminos posibles del sistema, que es cuando la canción ingresada ya está en la base de conocimiento.

```
Género seleccionado: pop
Se encontraron 1043 canciones para el género 'pop'.
Ingresa el nombre de la canción: skinny
Ingresa el nombre del artista: billie eilish
La canción está en el sistema. Buscando canciones similares...
Mostrando canciones:
                          Name            Artist                                                 Album
101                      SKINNY      Billie Eilish                                  HIT ME HARD AND SOFT
426           How to disappear       Lana Del Rey                               Norman Fucking Rockwell!
65                     The 30th      Billie Eilish                                          Guitar Songs
114  Margaret (feat. Bleachers)       Lana Del Rey  Did you know that there's a tunnel under Ocean...
519            Pluto Projector  Rex Orange County                                                   Pony
385                    Coaster             Khalid                                       American Teen
43                     logical     Olivia Rodrigo                                               GUTS
396             Sweet Creature       Harry Styles                                       Harry Styles
71                   Love song       Lana Del Rey                               Norman Fucking Rockwell!
645    Norman fucking Rockwell       Lana Del Rey                               Norman Fucking Rockwell!
¿Quieres ver más canciones? (sí/no):
```

Figura 4: Ejemplo de radio generada.

Sin embargo, cuando la canción ingresada no coincidía en género con el género seleccionado, se identificaron inconsistencias en la radio generada. En estos casos, la radio incluía la canción ingresada como punto de referencia, pero las otras canciones recomendadas pertenecían exclusivamente al género seleccionado, sin necesariamente guardar una evidente relación empírica o contextual con la canción de entrada, como se puede observar en la figura número 5. En este caso, se seleccionó el género Rock y se observó que las canciones recomendadas están más alineadas con este género en particular, en lugar de ajustarse a las características de la canción específica. Esto podría deberse a que el sistema realiza un filtrado inicial de la base de conocimientos completa, priorizando el género antes que otros atributos de la canción.

```
Género seleccionado: rock
Se encontraron 443 canciones para el género 'rock'.
Ingresa el nombre de la canción: es el amor
Ingresa el nombre del artista: estelares
La canción no está en el sistema. Buscando en Spotify...
Ejecutando algoritmo para generar canciones similares...
Mostrando canciones:
                                 Name             Artist                                  Album
443                      Es el Amor          Estelares                              Las Antenas
224                  Shot In The Dark              AC/DC                                 POWER UP
253  (I Can't Get No) Satisfaction - Mono  The Rolling Stones  The Rolling Stones In Mono (Remastered 2016)
189         Junk Of The Heart (Happy)          The Kooks                          Junk Of The Heart
444                      Es el Amor          Estelares                       Es el Amor - Single
440                    Threat of Joy         The Strokes                       Future Present Past
422          Dude (Looks Like A Lady)          Aerosmith                                Big Ones
61         El lado soleado de la calle  El Cuarteto De Nos                                Porfiado
175          I Was Made For Lovin' You               KISS                                 Dynasty
20                    You're So Vain         Carly Simon                              No Secrets
¿Quieres ver más canciones? (sí/no):
```

Figura 5: Ejemplo de radio generada.

Estas observaciones destacan la importancia de una adecuada correspon-

dencia entre la canción ingresada y el género seleccionado para garantizar la calidad de las recomendaciones, y permiten identificar áreas de mejora en el funcionamiento del sistema experto.

7. Conclusiones

Se cumplió con el objetivo de crear un sistema experto que permita ofrecer recomendaciones musicales personalizadas basadas en las preferencias iniciales del usuario. En este trabajo se ha logrado diseñar y desarrollar un sistema experto capaz de recomendar canciones integrando información proveniente de la API de Spotify con una base de conocimientos en Prolog y técnicas de aprendizaje automático como KNN. A través de la construcción de una base de datos personalizada y alineada con las preferencias musicales de los autores, fue posible generar recomendaciones más cercanas a los gustos y hábitos de escucha reales.

La implementación del algoritmo KNN demostró ser efectiva al identificar canciones similares en función de características cuantitativas estandarizadas, como la energía, la bailabilidad o la valencia. La decisión de excluir la popularidad del artista como atributo permitió obtener resultados más diversos, evitando la concentración excesiva de artistas en la lista de recomendaciones. De este modo, se incrementó el potencial de descubrimiento de nuevas canciones y se propició una experiencia más rica para el usuario.

Por otra parte, el modelo de autómata finito determinista contribuyó a estructurar el flujo lógico del sistema, garantizando una interacción coherente y determinista. La inclusión de mecanismos de validación, tanto para el ingreso de géneros como para la identificación y adición de canciones faltantes, logró asegurar la consistencia y robustez del proceso de recomendación.

En términos de rendimiento, la complejidad asociada a las consultas en Prolog y al cálculo de distancias en KNN resultó manejable para el tamaño de datos considerado. Las optimizaciones aplicadas y el filtrado por género disminuyeron el espacio de búsqueda, permitiendo ofrecer tiempos de respuesta adecuados.

Finalmente, las evaluaciones realizadas evidenciaron que la calidad de las recomendaciones depende en gran medida de la coherencia entre la canción seleccionada y el género elegido, destacando así la importancia de un adecuado alineamiento entre las preferencias declaradas por el usuario y la información almacenada en la base de conocimientos. Estas observaciones abren la puerta a futuras mejoras, tales como el ajuste dinámico de atributos y la incorporación de técnicas de aprendizaje profundo, con el fin de perfeccionar las sugerencias y ampliar el alcance del sistema experto desarrollado.

Referencias

[1] Hopcroft, John E., Rajeev Motwani, y Jeffrey D. Ullman. *Teoría de autómatas, lenguajes y computación.* 3ª edición. Pearson, 2007. Disponible en:

https://docencia.eafranco.com/materiales/teoriacomputacional/books/TeoriaDeAutomatas,lenguajesYComputacion-Hopcroft.pdf

[2] Hughes, Chris. *Understanding Spotify Web API*. Engineering at Spotify, 9 de marzo de 2015. Disponible en: https://engineering.atspotify.com/2015/03/understanding-spotify-web-api/

[3] Merritt, Dennis. *Building Expert Systems in Prolog*. Amzi! Inc., 2000.

[4] Lopez Yse, Diego. *K-Nearest Neighbor (KNN) Explained*. Pinecone, 30 de junio de 2023. Disponible en: https://www.pinecone.io/learn/k-nearest-neighbor/