**✅ 1. FUNCIONAMIENTO GENERAL DEL SISTEMA**

### **Entrada del sistema (input del usuario)**

* Género favorito del usuario (acción, comedia, drama, etc.)
* Nivel de preferencia por ese género (muy alto, medio, bajo)
* Edad del usuario
* Idioma preferido (español, inglés, etc.)
* Formato preferido (serie o película)
* Historial o intereses previos
* Nivel de apertura a nuevos géneros o formatos (valor difuso)
* Características de contenido disponibles (extraídas desde la ontología)

### **Procesamiento**

1. **Ontología (RDFLib + OWL-RL):**
   * Se modela el dominio (géneros, tipos de formatos, directores, idiomas, actores, etc.).
   * Se razona sobre el grafo y se infieren relaciones.
   * Se traducen algunas de estas inferencias a **hechos** para el sistema experto.
2. **Lógica difusa (Scikit-Fuzzy):**
   * Modela preferencias del usuario de forma imprecisa (por ejemplo, "muy interesado en acción", "ligeramente interesado en comedia").
   * El resultado se transforma en valores que se usan como **condiciones** en reglas de Experta.
3. **Sistema experto (Experta):**
   * Recibe hechos de la ontología y valores de la lógica difusa.
   * Aplica reglas para recomendar series o películas con base en estos datos.

### **Salida del sistema (recomendación)**

* Lista de series o películas recomendadas al usuario.
* Justificación basada en reglas activadas e inferencias ontológicas y difusas.

## **✅ 2. ELEMENTOS A CREAR EN CADA COMPONENTE Y SU INTEGRACIÓN**

### **A. ONTOLOGÍA (RDFLib + OWL-RL)**

#### **Clases a definir (rdfs:Class)**

* Serie, Pelicula, Contenido, Genero, Usuario, Director, Idioma, Actor, Formato, Recomendacion.

#### **Relaciones jerárquicas (rdfs:subClassOf)**

* Serie y Pelicula son subclases de Contenido.
* Comedia, Drama, Acción, etc., son subclases de Genero.

#### **Propiedades (rdf:Property)**

* tieneGenero, tieneFormato, tieneIdioma, tieneDirector, esRecomendadoPara.

#### **Jerarquía de propiedades (rdfs:subPropertyOf)**

* tieneGeneroPrincipal ⊆ tieneGenero

#### **Dominios y rangos**

* tieneGenero → dominio: Contenido, rango: Genero
* esRecomendadoPara → dominio: Contenido, rango: Usuario

#### **Inferencias generadas (que se traducen a hechos de Experta)**

* Si Película X tiene género Acción y Acción es subclase de Contenido Popular, se infiere que Película X pertenece a Contenido Popular.
* Si Película Y tiene tieneIdioma Español → el usuario que prefiere Español puede recibirla como recomendación.
* Estas inferencias se pasan como hechos (Fact) a Experta:

engine.declare(Genero\_favorito(genero="Acción"))



### **B. LÓGICA DIFUSA (Scikit-Fuzzy)**

#### **Variables difusas**

* **Interés\_usuario** (bajo, medio, alto)
* **Apertura\_nuevos\_generos** (cerrado, neutral, abierto)
* **Adecuación\_edad** (niño, adolescente, adulto)

**Definición del Universo y Variables**

El universo de discurso de cada variable está definido de la siguiente manera:

interés del usuario: valores entre 0 y 10, con incrementos de 0.1.

Apertura a nuevos géneros: valores entre 0 y 10, con incrementos de 0.1.

adecuación de la edad: valores entre 0 y 10, con incrementos de 0.1.

recomendabilidad(salida): valores entre 0 y 100, con incrementos de 1.

**Definición de Funciones de Pertenencia**

#### **interés del usuario**

* **bajo**: triangular, (0, 0, 4)
* **medio**: trapezoidal, (media=5, sigma=1.5)
* **alto**: gausiana
* **muy interesado**: concentración de **alto** (potencia 2)

#### **apertura a nuevos géneros**

* **cerrado**: trapezoidal,
* **neutral**: gaussiana, (media=5, sigma=1.5)
* **abierto**: triangular
* **ligeramente abierto**: dilatación de **abierto** (potencia 1.3)

#### **adecuación de la edad**

* **niño**: triangular, (0, 0, 3)
* **adolescente**: trapezoidal, (2, 4, 6, 8)
* **adulto**: gaussiana, (media=9, sigma=1)

#### **Reglas difusas**

****Si (Interés\_usuario ES alto) Y (Apertura\_nuevos\_generos ES abierto) Entonces (Recomendabilidad ES alta)

#### **Proceso de defuzzificación**

* Usar **centroide** para obtener un valor numérico final de recomendabilidad.
* Este valor puede alimentar a Experta así:

engine.declare(Fact(recomendabilidad=0.82))



### **C. SISTEMA EXPERTO (Experta)**

#### **Clases de hechos**

****class Usuario(Fact): pass

class Genero\_favorito(Fact): pass

class Idioma\_preferido(Fact): pass

class Formato\_preferido(Fact): pass

class Recomendabilidad(Fact): pass # valor difuso defuzzificado

class Inferencia\_ontologica(Fact): pass # resultado del razonador

class Recomendacion(Fact): pass

#### **Reglas (Ejemplos)**

****@Rule(Genero\_favorito(genero='Acción'),

Idioma\_preferido(idioma='Español'),

Recomendabilidad(valor=P(lambda v: v > 0.7)))

def recomendar\_accion():

print("Recomendamos películas de acción en español.")

self.declare(Recomendacion(titulo="John Wick"))



## **✅ 3. INTEGRACIÓN DE LOS COMPONENTES**

**** ┌─────────────┐

│ Usuario │

└────┬────────┘

│

┌──────────▼─────────────┐

│ Datos de entrada del │

│ usuario (gustos, edad) │

└────┬──────┬────────────┘

│ │

┌─────────▼─┐ ┌─▼─────────────┐

│ Ontología │ │ Lógica Difusa│

└────┬──────┘ └────┬──────────┘

│ Inferencias │ Recomendabilidad (0.0 a 1.0)

▼ ▼

┌────────────────────┐

│ SISTEMA EXPERTO │ (Experta)

└───────┬────────────┘

▼

┌────────────────────────┐

│ Recomendaciones finales│

└────────────────────────┘



## **🧩 Ejemplo de Flujo Concreto**

1. El usuario ingresa: le gusta la acción, prefiere películas, idioma español.
2. La ontología infiere que “Película X” tiene esas características.
3. La lógica difusa determina una recomendabilidad del 0.82.
4. Se declara en Experta:

engine.declare(Genero\_favorito(genero="Acción"))

engine.declare(Idioma\_preferido(idioma="Español"))

engine.declare(Recomendabilidad(valor=0.82))

1.   
   Se activan reglas que recomiendan “Película X” y se imprime la recomendación.

¿Te gustaría que te prepare una plantilla base en código para cada componente y cómo conectarlos?