Sistema experto

Alumno: Mayra vianney rodriguez lopez

Semestre: 7mo

Turno: matutino

Materia: Inteligencia artificial

Introducción

El proyecto tiene como finalidad de recomendar videojuegos mediante las reglas para definir que videojuegos le pueden gustar con esas reglas, esta hecho en el lenguaje Python, que se estructura de la siguiente forma: librerías(numpy, fuzzywuzzy), base de conocimiento(definimos las reglas), base de hechos(datos de los videojuegos), motor de inferencia(evaluar reglas y hechos para hacer recomendaciones).

Guia:

Sistema experto de recomendaciones de videojuegos.

Librerias:

Numpy:

Es el paquete más básico pero poderoso para la computación científica y la manipulación de datos en Python. Nos permite trabajar con matrices y matrices multidimensionales.

Fuzzywuzzy:

Es una biblioteca de Python que se utiliza para realizar comparación de cadenas de texto (string matching). Fue desarrollada inicialmente por SeatGeek y facilita la comparación de similitudes entre dos cadenas de texto.

Base de conocimientos:

La base de conocimientos define las reglas y criterios de clasificación de los géneros de videojuegos.

```python
def define_rules():
    rules = {
        "accion": ["ritmo rapido", "combate", "aventura"],
        "estrategia": ["planificacion", "tacticas", "gestion de recursos"],
        "rompecabezas": ["resolucion de problemas", "logica", "acertijos"],
        "simulacion": ["realista", "simulacion", "vida real"],
        "deporte": ["ritmo rapido", "competencia", "habilidad"],
        "cooperativo": ["cooperativo", "trabajo en equipo", "multijugador", "multijugador local"]
    }
    return rules
```

Función define rules: Crea un diccionario donde las claves son géneros de videojuegos y los valores son listas de características asociadas a esos géneros.

Reglas: Cada clave representa un género de videojuegos (genero, "accion", "estrategia") y cada valor es una lista de características que definen ese género (e.g., ["ritmo rapido", "combate", "aventura"]).

Base de Hechos:

a base de hechos contiene datos sobre videojuegos y sus características.

```python
def define_facts():
    facts = {
        "fortnite": ["ritmo rapido", "combate", "accion", "multijugador"],
        "jurassic world": ["planificacion", "gestion de recursos"],
        "sims": ["planificacion", "simulacion"],
        "sims4": ["planificacion", "simulacion", "realista"],
        "battlefield": ["resolucion de problemas", "logica", "combate", "multijugador"],
        "call of duty": ["ritmo rapido", "combate", "accion", "multijugador"],
        "civilization vi": ["planificacion", "tacticas", "gestion de recursos"],
        "portal": ["resolucion de problemas", "logica", "acertijos"],
        "flight simulator": ["realista", "simulacion", "vida real"],
        "overwatch": ["ritmo rapido", "combate", "accion", "multijugador"],
```

Función define_facts:

Crea un diccionario donde las claves son nombres de videojuegos y los valores son listas de características asociadas a esos videojuegos.

Hechos:

Cada clave representa un videojuego (genero, "fortnite", "sims4") y cada valor es una lista de características que describen el juego (e.g., ["ritmo rapido", "combate", "accion", "multijugador"]).

Motor de Inferencia:

El motor de inferencia evalúa las reglas y hechos para hacer recomendaciones basadas en las preferencias del usuario.

```python
def infer(preferences, rules, facts):
    recomendaciones = []
    justificacion = []

    for game, features in facts.items():
        for preference in preferences:
            match_ratio = fuzz.token_set_ratio(preference, " ".join(features))
            if match_ratio > 80:  # Umbral de coincidencia
                recomendaciones.append(game)
                justificacion.append(f"El juego '{game}' coincide con la preferencia '{preference}' con una relación de coincidencia de {match_ratio}%.")
                break

    return recomendaciones, justificacion
```

Función infer:

Toma las preferencias del usuario, las reglas y los hechos como entrada, y devuelve una lista de recomendaciones y justificaciones.

Bucles:

Recorre cada juego y sus características en facts. Para cada juego, compara cada preferencia del usuario con las características del juego usando fuzz.token_set_ratio.

fuzz.token_set_ratio:

Compara la preferencia del usuario con las características del juego y devuelve una puntuación de coincidencia entre 0 y 100.

Umbral de coincidencia:

Si la coincidencia es mayor que 80, el juego se añade a las recomendaciones y se almacena una justificación que explica por qué se recomendó ese juego.

Justificaciones:

Se almacenan explicaciones que describen la coincidencia entre la preferencia del usuario y las características del juego.

Subsistema de Justificación:

El subsistema de justificación explica las recomendaciones al usuario

```
def print_justificacion(justificacion):
    print("Justificaciones de las recomendaciones:")
    for justification in justificacion:
        print(f"- {justification}")
```

Función print_justifications:

Toma una lista de justificaciones y las imprime.

Impresión de justificaciones:

Imprime cada justificación en una nueva línea, proporcionando al usuario una explicación de por qué se recomendaron ciertos juegos.

Recomendación de Videojuegos

Esta función coordina el proceso de recomendaciones.

```
def recommend_games(preferences, rules, facts):
    recomendaciones, justificacion = infer(preferences, rules, facts)
    return recomendaciones, justificacion
```

Función recommend_games:

Llama a la función infer con las preferencias del usuario, las reglas y los hechos, y devuelve las recomendaciones y justificaciones resultantes.

Función Principal

La función principal gestiona la interacción con el usuario.

```python
def main():
    # Definir reglas y hechos
    rules = define_rules()
    facts = define_facts()

    # Imprimir las reglas
    print("Ingrese sus preferencias de videojuegos basadas en las siguientes reglas:")
    for genre, criteria in rules.items():
        print(f"{genre.capitalize()}: {', '.join(criteria)}")

    while True:
        # Solicitar preferencias del usuario
        user_preferences = input("Ingrese sus preferencias de videojuegos (separadas por comas) o 'termine' para salir: ").lower()

        if user_preferences == "termine":
            print("Gracias por utilizar el sistema de recomendación de videojuegos.")
            break

        user_preferences = user_preferences.split(", ")

        # Recomendar videojuegos basados en las preferencias del usuario
        recomendaciones, justificacion = recommend_games(user_preferences, rules, facts)

        if recomendaciones:
            print("Basado en sus preferencias, podría gustarle jugar a los siguientes juegos:")
            for game in recomendaciones:
                print(f"- {game}")
            print_justificacion(justificacion)
        else:
            print("Lo siento, no se encontraron juegos que coincidan con sus preferencias.")

if __name__ == "__main__":
    main()
```

Función main:

Gestiona la interacción con el usuario.

Imprime las reglas:

Muestra las reglas definidas al usuario.

Entrada del usuario:

Solicita las preferencias del usuario hasta que el usuario ingrese "termine".

División de preferencias:

Divide las preferencias del usuario en una lista.

Llama a recommend_games:

Genera recomendaciones basadas en las preferencias del usuario.

Imprime las recomendaciones:

Muestra las recomendaciones y justificaciones al usuario.

Decisiones tomadas:

Función main: Gestiona la interacción con el usuario.

Imprime las reglas: Muestra las reglas definidas al usuario.

Entrada del usuario:

Solicita las preferencias del usuario hasta que el usuario ingrese "termine".

División de preferencias:

Divide las preferencias del usuario en una lista.

Llama a recommend_games:

Genera recomendaciones basadas en las preferencias del usuario.

Imprime las recomendaciones:

Muestra las recomendaciones y justificaciones al usuario.

Objetivos Alcanzados

Implementación de Reglas y Hechos:

Se creó un sistema que clasifica videojuegos en función de sus características y se almacenaron datos relevantes sobre los juegos.

Sistema de Recomendación:

Se desarrolló un motor de inferencia que compara las preferencias del usuario con las características de los videojuegos para generar recomendaciones.

Justificación de Recomendaciones:

Se proporcionaron explicaciones detalladas sobre por qué se recomendaron ciertos juegos, mejorando la transparencia del sistema.

Interacción con el Usuario:

Se creó una interfaz sencilla para que los usuarios ingresen sus preferencias y reciban recomendaciones basadas en esas preferencias.

Desafíos Enfrentados

Asegurar que el sistema pueda manejar sinónimos, diferencias léxicas y variaciones en el orden de las palabras en las preferencias del usuario.

Gestionar la variabilidad en la información proporcionada sobre los juegos y las preferencias del usuario.

Aprendizajes Obtenidos

Importancia de la Normalización:

La normalización y estandarización de las características y preferencias ayuda a mejorar la precisión en la comparación de datos.

Eficiencia en la Comparación de Texto:

Herramientas como fuzz.token_set_ratio son útiles para manejar variaciones en el texto, pero es necesario ajustar los parámetros para obtener resultados óptimos.

Feedback del Usuario:

Proporcionar justificaciones claras para las recomendaciones ayuda a aumentar la confianza del usuario en el sistema.

Aplicaciones Potenciales

Recomendación de Productos:

El enfoque puede adaptarse a sistemas de recomendación de productos en comercio electrónico, ayudando a los usuarios a encontrar productos que se ajusten a sus preferencias.

Sistemas de Asesoramiento:

Puede utilizarse en sistemas de asesoramiento en áreas como educación, salud y finanzas para recomendar opciones basadas en las necesidades del usuario.

Personalización de Experiencias:

Se puede aplicar en plataformas de streaming, juegos y redes sociales para personalizar la experiencia del usuario en función de sus intereses y comportamientos.