Факультет программной инженерии и компьютерной техники

Системы искусственного интеллекта

Лабораторная работа №3

Разработка системы поддержки принятия решения на основе базы знаний или онтологии

Преподаватель: Кугаевских Александр Владимирович
Выполнил: Ле Чонг Дат
Группа: P33302

2023 г.

1 Введение

Целью этой лабораторной работы является разработка программы (рекомендательной системы), которая будет использовать базу знаний или онтологию для предоставления рекомендаций на основе введенных пользователем данных. (Knowledge-based support system)

2 Задание

- Создать программу, которая позволяет пользователю ввести запрос через командную строку. Например, информацию о себе, своих интересах и предпочтениях в контексте выбора видеоигры - на основе фактов из БЗ (из первой лабы)/Онтологии(из второй).
- Использовать введенные пользователем данные, чтобы выполнить логические запросы к БЗ/Онтологии.
- На основе полученных результатов выполнения запросов, система должна предоставить рекомендации или советы, связанные с выбором из БЗ или онтологии.
- Система должна выдавать рекомендации после небольшого диалога с пользователем.

2.1 Пример

Входная строка: Мне 13 лет, мне нравятся: RPG, инди-игры

2.2 Нужно

Спарсить строку, разбить на факты, построить запрос, используя эти предикаты. (Формат входной строки фиксированный, искать частичное соответсвие подстроки не нужно)

2.3 Критерии оценки

- Корректность и эффективность реализации системы поддержки принятия решения.
- Способность программы адекватно использовать базу знаний или онтологию для выдачи рекомендаций.
- Качество тестирования и обработка ввода пользователя.
- Качество документации и описание работы системы.

3 Работа

В этом разделе мы подробно рассмотрим процесс создания и тестирования базы знаний на Prolog, посвященной видеоиграм.
Код Protolog

Листинг 1: Prolog code для базы знаний о видеоиграх

```
% Knowledge Base for Video Games

% Facts with one argument
game(minecraft).
game(the_witcher_3).
game(legend_of_zelda).
game(super_mario).
game(doom).
game(halo).
game(final_fantasy).
game(overwatch).
game(elder_scrolls_skyrim).
game(dark_souls).
genre(action).
genre(adventure).
genre(rpg).
genre(shooter).
genre(platformer).
platform(pc).
platform(playstation).
platform(xbox).
platform(nintendo).
platform(mobile).

% Facts with two arguments
released_on(minecraft, pc).
released_on(minecraft, playstation).
released_on(minecraft, xbox).
released_on(the_witcher_3, pc).
released_on(the_witcher_3, playstation).
released_on(the_witcher_3, xbox).
released_on(legend_of_zelda, nintendo).
released_on(super_mario, nintendo).
released_on(doom, pc).
released_on(doom, xbox).
belongs_to_genre(minecraft, adventure).
belongs_to_genre(the_witcher_3, action).
belongs_to_genre(legend_of_zelda, adventure).
belongs_to_genre(super_mario, platformer).
belongs_to_genre(doom, shooter).
belongs_to_genre(halo, shooter).
belongs_to_genre(final_fantasy, rpg).
belongs_to_genre(overwatch, shooter).
belongs_to_genre(elder_scrolls_skyrim, rpg).
belongs_to_genre(dark_souls, action).

% Rules
popular_game(Game) :- game(Game), released_on(Game, pc),
    belongs_to_genre(Game, adventure).
available_on_multiple_platforms(Game) :- released_on(
    Game, Platform1), released_on(Game, Platform2),
    Platform1 \= Platform2.
is_rpg(Game) :- game(Game), belongs_to_genre(Game, rpg).
```

```
is_action_adventure(Game) :- game(Game), belongs_to_genre(Game, action), belongs_to_genre(Game, adventure).
classic_game(Game) :- game(Game), (belongs_to_genre(Game, platformer) ; belongs_to_genre(Game, adventure)).

genre_on_platform(Game, Genre, Platform) :- game(Game), belongs_to_genre(Game, Genre), released_on(Game, Platform).

released_on_both_platforms(Game, Platform1, Platform2) :- game(Game), released_on(Game, Platform1), released_on(Game, Platform2), Platform1 \= Platform2.



% Queries
% Simple query to find if a game is an RPG
% ?- is_rpg(the_witcher_3).

% Query using logical operators
% ?- game(Game), released_on(Game, pc), belongs_to_genre(Game, action).

% Query using variables
% ?- released_on(Game, playstation), belongs_to_genre(Game, Genre).

% Query that requires the application of rules
% ?- popular_game(Game).

% Commented documentation throughout the code explains each fact, predicate, and rule.
```

Код Python

Листинг 2: Ppython код для предоставления рекомендаций

```
import re
from pyswip import Prolog

def match_pattern(user_input):
    patterns = {
        "List all popular games available on PC.": "popular_game(Game)",
        "Show games released on multiple platforms.": "available_on_multiple_platforms(Game)",
        "Identify games that are categorized as RPG.": "is_rpg(Game)",
        "Which games belong to both the action and adventure genres?": "is_action_adventure(Game)",
        "I'm looking for (.+) games released on (.+).": "genre_on_platform(Game, {}, {})",
        "Find games available on both (.+) and (.+).": "
```

```
            released_on_both_platforms(Game, {}, {})"
    }

    for pattern, query in patterns.items():
        match = re.match(pattern, user_input)
        if match:
            return query, match.groups()

    return None, None

def query_prolog(query, params):
    prolog = Prolog()
    prolog.consult('base-knowledge.pl')
    query_string = query.format(*params)
    return list(prolog.query(query_string))

def main():
    while True:
        user_input = input("Enter your query (or 'exit'
            to quit): ")
        if user_input.lower() == 'exit':
            break

        query, params = match_pattern(user_input)
        if query:
            results = query_prolog(query, params)
            print("Results:", results)
        else:
            print("No matching pattern found.")

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

Проверка

В этом разделе представлены результаты тестирования кода.

```
datlt@CA-00618:~/Documents/itmo/ai-system/lab3$ python __init__.py
Enter your query (or 'exit' to quit): List all popular games available on PC.
Results: [{'Game': 'minecraft'}]
Enter your query (or 'exit' to quit): I'm looking for shooter games released on pc.
Results: [{'Game': 'doom'}]
Enter your query (or 'exit' to quit): I'm looking for adventure games released on xbox.
Results: [{'Game': 'minecraft'}]
Enter your query (or 'exit' to quit): Find games available on both pc and xbox.
Results: [{'Game': 'minecraft'}, {'Game': 'the_witcher_3'}, {'Game': 'doom'}]
Enter your query (or 'exit' to quit): Identify games that are categorized as RPG.
Results: [{'Game': 'final_fantasy'}, {'Game': 'elder_scrolls_skyrim'}]
Enter your query (or 'exit' to quit): Show games released on multiple platforms.
Results: [{'Game': 'minecraft'}, {'Game': 'minecraft'}, {'Game': 'minecraft'}, {'Game': 'minecraft'}, {'Game': 'minec
raft'}, {'Game': 'minecraft'}, {'Game': 'the_witcher_3'}, {'Game': 'the_witcher_3'}, {'Game': 'the_witcher_3'}, {'Gam
e': 'the_witcher_3'}, {'Game': 'the_witcher_3'}, {'Game': 'the_witcher_3'}, {'Game': 'doom'}, {'Game': 'doom'}]
Enter your query (or 'exit' to quit): exit
datlt@CA-00618:~/Documents/itmo/ai-system/lab3$
```

Рис. 1: Результаты тестирования программы рекомендаций

4 Вывод

В ходе данной лабораторной работы была успешно разработана рекомендательная система, использующая базу знаний для предоставления персонализированных рекомендаций. Система демонстрирует эффективность в обработке пользовательских запросов и точность в выдаче соответствующих рекомендаций. Этот проект не только подчеркивает важность интеграции баз знаний в разработке программного обеспечения, но и открывает перспективы для дальнейших усовершенствований в области интеллектуального анализа данных.