МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
«Санкт-Петербургский национальный исследовательский
университет информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И
КОМПЬЮТЕРНОЙ ТЕХНИКИ

Отчёт по модулю №1

по дисциплине

«Системы искусственного интеллекта»

Выполнил ***:*** Студент группы
P3334 Баянов Равиль
Динарович
Преподаватель:
Авдюшина А. Е.

Санкт-Петербург, 2024

Оглавление

Введение

В первых двух лабораторных работах мы столкнулись с изучением истоков систем искусственного интеллекта. Начало ИИ строится на основе логических рассуждений. И первая практическая идея построения интеллектуальных систем была реализована на декларативном языке программирования Prolog, построенном на основе логики, фактов и правил. Первая часть первой лабораторной работы продвигает цель научить нас применять идеи последовательных логических действий на языке Prolog. Познакомившись с предикатами и синтаксисом Prolog, мы хотим научиться определить для себя границы применимости систем искусственного интеллекта и научиться строить простейшие базы знаний. Вторая же часть знакомит нас с понятием онтологий и их значимостью в сфере ИИ. Благодаря программе Protégé, мы научимся создавать примитивные онтологии, на основе которых в будущем могут строиться системы искусственного интеллекта. Ну и визуализировать полученные знания нам поможет вторая лабораторная работа, которая в свою очередь предлагает нам уже построенные нами базу знаний и онтологию реализовать на любом языке программирования в виде рекомендательной системы. Первый модуль плавно знакомит нас с элементами систем ИИ, которые строятся на основе логики и на основе данных.

Анализ требований

Основные требования к системе поддержки принятия решений это:

- Точность и полнота
- Гибкость
- Ясность и удобство взаимодействия с системой

Рекомендательная система обязана быть быстрой и полной. Пользователь не должен получать ответы на свои запросы, которые будут не точными или лживыми. Именно поэтому такую систему нужно постоянно поддерживать и обновлять, чтобы такая система всегда выдавала актуальный ответ.

Основные требования к БЗ и онтологии:

- Плотная связь онтологии и БЗ
- Структурированность и логичность
- Чёткость и ясность субъектов, объектов и отношений (предикатов)

Базы знаний и онтологии — это неразрывные вещи. Абсолютно все онтологии (семантические сети и паутины) строятся на основе какой-то определённой БЗ. Именно поэтому не должно быть никаких расхождений. Также онтология и БЗ должны быть чётко структурированы и удобочитаемы. Иначе люди и другие системы, пользующиеся этими онтологиями, могут не так интерпретировать данные. Онтологии для этого и созданы, чтобы укомплектовать какую-то предметную область в приятную глазу концептуальную схему.

Изучение основных концепций и инструментов

Базы знаний и онтологии — это два важных подхода к организации знаний в системах ИИ. Базы знаний базируются на основе хранения данных и связей между этими данными на основе предикатов, правил и фактов. Фактами являются элементарными единицами в создании баз знаний. Правила же собираются из фактов и создают более сложные знания в базе. Всё это собирается в логические выводы о предметной области. И это помогает нам в принятии решений в определённых ситуациях.

Онтологии помогают нам формализовать данные, хранящиеся в базе знаний с помощью схем, семантических сетей и графов. Онтологии базируются на классах, свойствах, отношениях и ограничениях (подобно объектно-ориентированному программированию). Чем сложнее БЗ, тем сильнее она зависима от её формализации. Но при увеличении базы знаний онтология может перестать быть читаемой.

Prolog – очень мощный язык декларативной парадигмы. Он оперирует предикатами, фактами и правилами. Он позволяет находить всевозможные решения логических задач, по-своему перебирая факты. Он выводит логические выводы по запросу из базы знаний. Prolog обладает рекурсией и массивами, что позволяет строить сложные правила, на основе которых можно искать интересные логические пути и применять их в создании систем ИИ. Prolog умеет сопоставлять факты между собой, сравнивать их и с помощью заглушек регулирует объём поиска нужных данных.

Для начала изучения Prolog поможет простой и понятный интерпретатор SWI-Prolog. Этот инструмент обладает OWL и RDF форматами, которые позволяют сохранять базы знаний в текстовом формате. С помощью RDF (расширения для XML) можно создать на основе данной БЗ онтологию. А с помощью SPARQL запросов можно обращаться в базе знаний, написанной на Prolog.

Реализация системы искусственного интеллекта (системы поддержки принятия решения)

Я реализовал СППР на языке программирования Java. Вся логика хранится в классе Data, которые является как бы хранилищем для фактов и правил. И приложение обращаясь к этой базе выдаёт пользователю ответ.

Код:

```
1.  public class Main {
2.      public static void main(String[] args) {
3.          Supervisor supervisor = new Supervisor();
4.          supervisor.run();
5.      }
6.  }
```

```
1.  public class Main {
2.      public static void main(String[] args) {
3.          Supervisor supervisor = new Supervisor();
4.          supervisor.run();
5.      }
6.  }import model.Brawler;
7.  import model.Data;
8.
9.  import java.util.Locale;
10. import java.util.Objects;
11. import java.util.Scanner;
12.
13. public class Supervisor {
14.     Scanner console = new Scanner(System.in);
15.     Data data;
16.
17.     Supervisor() {
18.         this.data = new Data();
19.         data.getArray().add(new Brawler("shelly", "common", 3000, 7400));
20.         data.getArray().add(new Brawler("nita", "rare", 1920, 8000));
21.         data.getArray().add(new Brawler("bull", "rare", 880, 10000));
22.         data.getArray().add(new Brawler("el_primo", "rare", 760, 12000));
23.         data.getArray().add(new Brawler("colt", "rare", 4320, 5600));
24.         data.getArray().add(new Brawler("tick", "super_rare", 1280, 4400));
25.         data.getArray().add(new Brawler("penny", "super_rare", 1960, 6400));
26.         data.getArray().add(new Brawler("carl", "super_rare", 1480, 8000));
27.         data.getArray().add(new Brawler("rico", "super_rare", 640, 5600));
28.         data.getArray().add(new Brawler("bo", "epic", 1280, 7200));
29.         data.getArray().add(new Brawler("piper", "epic", 3400, 4600));
30.         data.getArray().add(new Brawler("bea", "epic", 1600, 5000));
31.         data.getArray().add(new Brawler("emz", "epic", 1040, 7200));
32.         data.getArray().add(new Brawler("tara", "mythic", 960, 6200));
33.         data.getArray().add(new Brawler("mortis", "mythic", 2000, 7600));
34.         data.getArray().add(new Brawler("buzz", "mythic", 840, 9600));
35.         data.getArray().add(new Brawler("max", "mythic", 640, 6600));
36.         data.getArray().add(new Brawler("chuck", "mythic", 1080, 9000));
37.         data.getArray().add(new Brawler("sandy", "legendary", 1800, 7600));
```

```
38.         data.getArray().add(new Brawler("leon", "legendary", 960, 6800));
39.         data.getArray().add(new Brawler("crow", "legendary", 640, 4800));
40.         data.getArray().add(new Brawler("surge", "legendary", 2360, 6800));
41.         data.getArray().add(new Brawler("kordelius", "legendary", 1400, 6400));
42.         data.getArray().add(new Brawler("amber", "legendary", 4200, 6400));
43.         data.getArray().add(new Brawler("spike", "legendary", 1080, 5200));
44.         data.getMap_good_team().put("max", "surge");
45.         data.getMap_good_team().put("tara", "sandy");
46.         data.getMap_good_team().put("sandy", "tara");
47.         data.getMap_good_team().put("surge", "max");
48.     }
49.
50.     // Запуск приложения
51.     void run() {
52.         System.out.println("Добро пожаловать в базу знаний по мобильной игре Brawl Stars");
53.         while (true) {
54.             System.out.print(">>>");
55.             String str = console.nextLine();
56.             str = str.toLowerCase().trim();
57.             String[] facts = str.split(",");
58.             handle_facts(facts);
59.             if (Objects.equals(str, "exit")) {
60.                 System.out.println("Пока!");
61.                 System.exit(0);
62.             }
63.         }
64.     }
65.
66.     void handle_facts(String[] facts) {
67.         for (String fact : facts) {
68.             if (fact.equals("help")) {
69.                 System.out.println("Введите help, чтобы увидеть формат запросов");
70.                 System.out.println("Формат строки: <факт>, <факт> и так далее");
71.                 System.out.println("Примеры фактов, которые можно запросить у базы знаний:");
72.                 System.out.println("все бравлеры\nбравлеры <редкость> (например: бравлеры rare)\nкто
    сильнее <бравлер> <бравлер> (например: Кто сильнее max shelly)" +
73.                         "\n бой <бравлер> <бравлер>\n редкость <бравлер>\n синергия <бравлер>");
74.             } else {
75.                 if (fact.equals("все бравлеры")) {
76.                     data.getAllBrawlers();
77.                 } else {
78.                     String[] words = fact.split(" ");
79.                     if (words[0].equals("бравлеры")) {
80.                         data.getAllBrawlersByRarity(words[1]);
81.                     } else if (words[0].equals("редкость")) {
82.                         data.rarityByBrawlerName(words[1]);
83.                     } else if (words[0].equals("синергия")) {
84.                         data.getBrawlerWithGoodTeam(words[1]);
85.                     } else if (words[0].equals("бой")) {
86.                         data.fight(words[1], words[2]);
87.                     } else if (words[0].equals("кто") && words[1].equals("сильнее")) {
88.                         data.morePower(words[2], words[3]);
89.                     } else {
90.                         System.out.println("Факт: '" + fact + "' введён некорректно");
91.                     }
92.                 }
93.             }
94.
95.         }
96.     }
97. }
```

```
1.  package model;
2.
3.  import java.lang.reflect.Array;
4.  import java.util.ArrayList;
5.  import java.util.HashMap;
6.
7.  public class Data {
8.      private ArrayList<Brawler> array = new ArrayList<>();
9.      private HashMap<String, String> map_good_team = new HashMap<>();
10.
11.     public ArrayList<Brawler> getArray() {
12.         return array;
13.     }
14.
15.     public void setArray(ArrayList<Brawler> array) {
16.         this.array = array;
17.     }
18.
19.     public HashMap<String, String> getMap_good_team() {
20.         return map_good_team;
21.     }
22.
23.     public void setMap_good_team(HashMap<String, String> map_good_team) {
24.         this.map_good_team = map_good_team;
25.     }
26.
27.     public void getAllBrawlers() {
28.         this.array.forEach(x -> System.out.printf("name: " + x.name + "\n" +
29.                 "rarity: " + x.rarity + "\n" +
30.                 "damage: " + x.damage + "\n" +
31.                 "health: " + x.health + "\n\n"));
32.     }
33.
34.     public void getAllBrawlersByRarity(String rarity) {
35.         this.array.stream().filter(x -> x.rarity.equals(rarity)).forEach(x -> System.out.println("name: " +
    x.name));
36.     }
37.
38.     public void getBrawlerWithGoodTeam(String name) {
39.         if (this.map_good_team.get(name) == null) {
40.             System.out.print("Для данного бравлера нет синергии либо такого бравлера вообще не
    существует\n");
41.         } else {
42.             System.out.println(this.map_good_team.get(name));
43.         }
44.     }
45.
46.     public void morePower(String name1, String name2) {
47.         int damage1 = 0;
48.         int damage2 = 0;
49.         for (Brawler x : array) {
50.             if (x.name.equals(name1)) {
51.                 damage1 = x.damage;
52.             }
53.         }
54.         for (Brawler x : array) {
55.             if (x.name.equals(name2)) {
56.                 damage2 = x.damage;
57.             }
58.         }
59.         if (damage2 == 0) {
```

```
60.             System.out.println("Бравлер " + name2 + "не найден");
61.             return ;
62.         }
63.         if (damage1 == 0) {
64.             System.out.println("Бравлер " + name1 + "не найден");
65.             return ;
66.         }
67.         if (damage2 > damage1) {
68.             System.out.println("more power:" + name2);
69.         } else if (damage1 > damage2) {
70.             System.out.println("more power:" + name1);
71.         } else {
72.             System.out.println("brawlers have the same damage");
73.         }
74.     }
75.
76.     public void rarityByBrawlerName(String name) {
77.         this.array.stream().filter(x -> x.name.equals(name)).forEach(x -> System.out.println("rarity: " +
    x.rarity));
78.     }
79.
80.     public void fight(String name1, String name2) {
81.         int damage1 = 0;
82.         int health1 = 0;
83.         int damage2 = 0;
84.         int health2 = 0;
85.         for (Brawler x : array) {
86.             if (x.name.equals(name1)) {
87.                 damage1 = x.damage;
88.                 health1 = x.health;
89.             }
90.         }
91.         for (Brawler x : array) {
92.             if (x.name.equals(name2)) {
93.                 damage2 = x.damage;
94.                 health2 = x.health;
95.             }
96.         }
97.         if (damage2 == 0) {
98.             System.out.println("Бравлер " + name2 + "не найден");
99.             return ;
100.        }
101.        if (damage1 == 0) {
102.            System.out.println("Бравлер " + name1 + "не найден");
103.            return ;
104.        }
105.        int flag = 2;
106.        while (true) {
107.            if (health1 <= 0) {
108.                flag = 1;
109.                break;
110.            }
111.            if (health2 <= 0) {
112.                flag = 0;
113.                break;
114.            }
115.            health1 -= damage2;
116.            health2 -= damage1;
117.        }
118.        if (flag == 1) {
119.            System.out.println("winner:" + name2);
120.        } else if (flag == 0) {
```

```
            System.out.println("winner:" + name1);
        } else {
            System.out.println("Haven't winner");
        }
    }
}
```

```
package model;

public class Brawler {
    String name;
    String rarity;
    int damage;
    int health;

    public Brawler(String name, String rarity, int damage, int health) {
        this.damage = damage;
        this.health = health;
        this.rarity = rarity;
        this.name = name;
    }
}
```

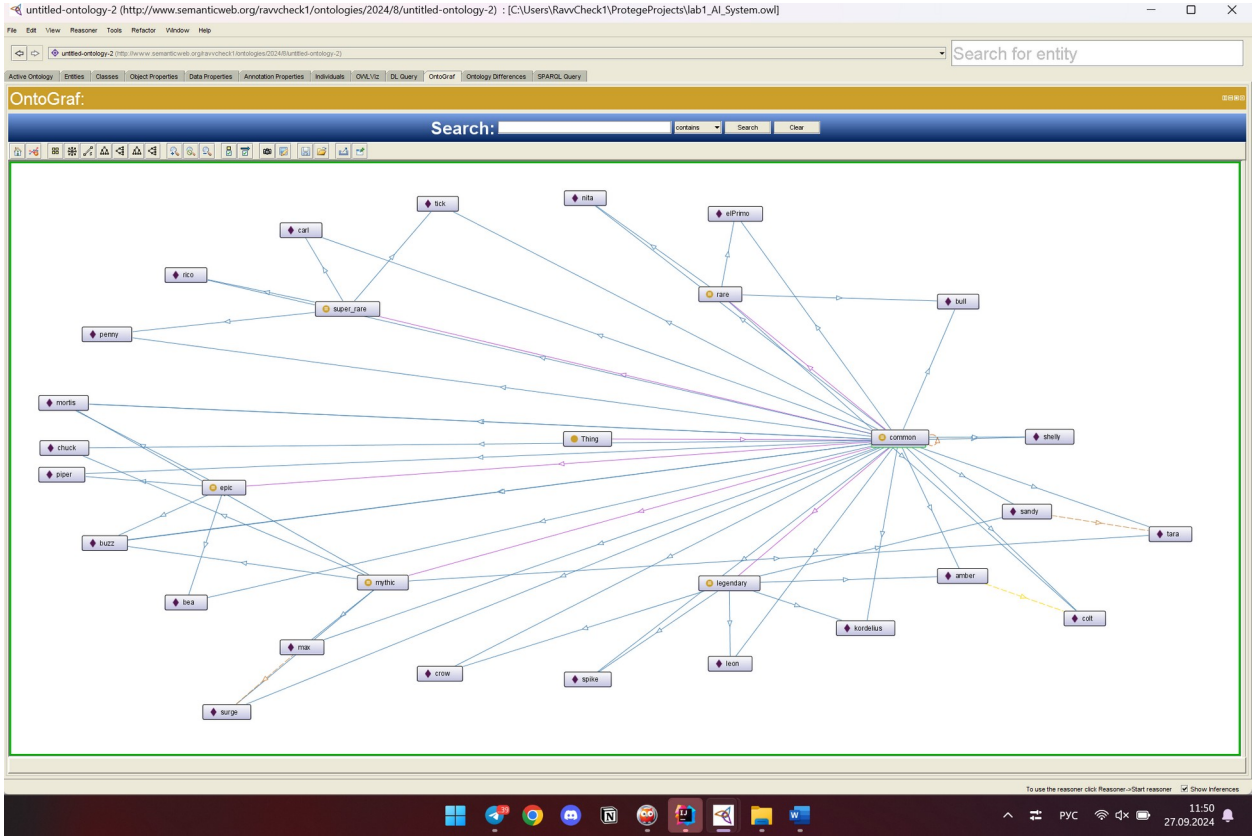
Оценка и интерпретация результатов

Построив БЗ на основе предметной области по мобильной игре Brawl Stars, я считаю, что требования к реализации онтологий и БЗ соблюдены. БЗ и онтология связаны. БЗ и онтологии полны, удобочитаемы и точны. В целом по написанной рекомендательной системе можно вполне получить все необходимые знания по игре.

Запросы на Prolog:

```
?- more_power(shelly, max, Result).
Result = shelly.

?- damage_by_brawler(piper, X).
X = 3400.

?- find_brawlers_by_rarity(legendary, Brawlers).
Brawlers = [crow, spike, leon, amber, kordelius, surge, sandy].

?- fight(piper, bull).
piper наносит удар по bull на 3400 урона. Осталось здоровья у bull: 6600
bull наносит удар по piper на 880 урона. Осталось здоровья у piper: 3720
piper наносит удар по bull на 3400 урона. Осталось здоровья у bull: 3200
bull наносит удар по piper на 880 урона. Осталось здоровья у piper: 2840
piper наносит удар по bull на 3400 урона. Осталось здоровья у bull: -200
piper побеждает!
true.

?- good_team(tara, X).
X = sandy
```

Онтограф (Семантическая сеть):



Запросы на рекомендательной системе:

```
>>>все бравлеры
name: shelly
rarity: common
damage: 3000
health: 7400

name: nita
rarity: rare
damage: 1920
health: 8000

name: bull
rarity: rare
damage: 880
health: 10000

name: el_primo
rarity: rare
damage: 760
health: 12000

name: colt
rarity: rare
damage: 4320
health: 5600
```

```
>>>редкостmax
Факт: 'редкостmax' введён некорректно
>>>редкость max
rarity: mythic
>>>синергия sandy
tara
>>>синергия kordelius
Для данного бравлера нет синергии либо такого бравлера вообще не существует
```

Запросы на SPARQL Query:

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subject ?predicate ?object

WHERE { ?subject ?predicate ?object }

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?object
    WHERE { ?subject ?rdfs:subClassOf ?object }
```

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?object
    WHERE { ?subject ?predicate owl:DatatypeProperty }
```

Исходя из полученных результатов систему ещё можно расширять до более глубоких знаний, но уже этого достаточно, чтобы получать всю информацию о персонажах, о том, кто сильнее, и о том какую сыгранность персонажи имеют между собой.

Вывод

Преимущества системы ИИ на базе Prolog, баз знаний и онтологий:

1. **Логический вывод и решение задач**:
    - **Prolog** предоставляет мощный механизм для логического вывода и решения задач. Он использует правила и факты для нахождения решений на основе логических связей. Это позволяет эффективно решать задачи, которые требуют поиска оптимальных решений или анализа сложных взаимосвязей.
2. **Гибкость и масштабируемость**:
    - Системы, основанные на базах знаний и онтологиях, легко расширяемы. Новые факты, правила и концепции могут быть добавлены без необходимости полного изменения системы. Это обеспечивает гибкость в обновлении базы знаний по мере появления новой информации.
3. **Автоматизация принятия решений**:
    - Разработанная система может автоматизировать процесс принятия решений, основанный на анализе данных и логическом выводе. Это особенно полезно в областях, где необходимо принимать сложные решения на основе большого количества параметров и ограничений.

Потенциальные применения разработанной системы:

1. **Экспертные системы**:
    - Система может быть использована для разработки экспертных систем в различных областях (например, диагностика заболеваний, юридическая поддержка, финансовые консультации). База знаний содержит правила и факты, которые позволяют системе делать точные выводы и предлагать решения на основе предоставленной информации.
2. **Автоматизация бизнес-процессов**:
    - В бизнесе такие системы могут применяться для автоматизации принятия решений, оптимизации логистических операций, оценки рисков или планирования производства. Логический вывод и структурированные знания позволяют находить

оптимальные решения в условиях ограничений и неопределенности.

3. **Системы поддержки принятия решений (СППР)**:
    - Разработанная система может быть использована для разработки СППР в различных областях, таких как управление персоналом, планирование ресурсов, оптимизация процессов и т.д. СППР могут анализировать множество факторов и помогать пользователям принимать обоснованные решения на основе логических выводов.
4. **Обработка естественного языка**:
    - **Prolog** может применяться для задач обработки естественного языка (NLP), таких как синтаксический анализ, генерация ответов на вопросы и работа с грамматическими структурами. Система может использоваться в чат-ботах, системах поиска информации и инструментах автоматического перевода.