Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Кафедра вычислительной математики и механики

Лабораторная работа № 3

по дисциплине: «Интеллектуальные ИСИТ»

Выполнила:

студентка группы ИСТ-19-1б

Дорогина И.В.

Проверил:

ассистент кафедры ВММБ

Нетбай Г. В.

Пермь, 2022

**Описание метода**

Структура семантической сети отображается графически с помощью узлов и соединяющих их дуг. Узла иногда именуются объектами, а дуги – связами. Связи в семантической сети применяются для представления отношений, а узлы, как правило, служат для представления физических объектов, концепций или ситуаций.

В семантический сетях часто встречаются следующие отношения:

1. Иерархические;

2. Функциональные;

3. Количественные;

4. Временные;

5. Атрибутивные;

6. Пространственные;

7. Казуальные.

**Постановка задания**

В рамках лабораторной работы необходимо спроектировать БЗ с представлением знаний в виде семантических сетей, а также реализовать оболочку экспертной системы, работающую с данной БЗ.

**Ход работы**

В рамках лабораторной работы по созданию семантической сети для реализации выбрана тема «Литература».

На рисунке 1 представлена семантическая сеть с иерархическими связями, соответствующая поставленной теме.

Рассмотрим типы имеющихся связей:

1. ЖАНР – связывает произведение и тот литературный жанр, к которому оно относится;

2. КИНОАДАПТАЦИЯ – связывает произведение и имеющиеся фильмы им вдохновленные;

3. ГЕРОЙ ПРОИЗВЕДЕНИЯ – связывает какого-либо героя произведения с произведением, в котором он встречается.

|  |
| --- |
|  |
| Рисунок 1 – Семантическая сеть «Литература» |

Первым этапом разработки программной оболочки для построенной семантической сети выделено описание базы знаний. Она составлена из 2 частей:

1. *Entity* – сопоставляет все объекты семантической сети (*name*) и присвоенные им идентификаторы (*id*);

2. *Connection* – описывает пары объектов (*parent / child*) и связь между ними (*connect*).

Данная база знаний составлена в *YAML*-файле (Приложение 1), что позволяет легко воспринимать информацию в нем, а также быстро и легко обрабатывать её с помощью языка программирования *Python*.

На втором этапе разработки программной оболочки были добавлены функции для обработки информации:

1. *toIdConvert* – конвертация имени объекта (*name*) в его идентификатор (*id*);

2. *toNameConvert* – конвертация идентификатора (*id*) объекта в его имя (*name*);

3. *searchConnections* – поиск по *id* среди *parent* в *connection*;

4. *processing* – процесс работы с каждой поступившей на вход парой значений *name* и *id*;

5. *handler* – главная функция оболочки.

На листингах 1 и 2, соответственно, приведены примеры реализации взаимообратных функций *toIdConvert* и *toNameConvert*.

На вход данные функции принимают либо имя объекта семантической сети, либо его идентификатор. В результате поиска необходимого известного параметра в *entity* на выход функции выдают либо идентификатор, либо имя объекта, соответственно.

|  |
| --- |
| def toIdConvert(name):  for element in data['entity']:  if element['name'] == name:  id = element['id']  return id |
| Листинг 1. Реализация функции *toIdConvert* |
|  |
| def toNameConvert(id):  for element in data['entity']:  if element['id'] == id:  name = element['name']  return name |
| Листинг 2. Реализация функции *toNameConvert* |

На листинге 3 приведена реализация функции *searchConnections*. Поиск осуществляется перебором всех имеющихся наборов параметров в словаре *connection*. Если не было найдено ни одного совпадения имени объекта и связи, то результатом будет записано сообщение «Указанная сущность не поддерживает выбранную связь».

|  |
| --- |
| def searchConnections(id, connectType):  search\_result = []  for element in data['connection']:  if element['parent'] == id  and element['connect'] == connectType:  search\_result.append(element['child'])  if len(search\_result) == 0:  search\_result.append("Указанная сущность не поддерживает выбранную связь")  return search\_result |
| Листинг 3 – Реализация функции *searchConnections* |

Рассмотрим алгоритм работы программы:

1. Загрузка БЗ из *YAML*-файла;

2. Запрос у пользователя имени объекта, для которого будет осуществляться поиск в семантической сети. Предусмотрен неограниченный ввод имени, чтобы в программу не попало значения, отсутствующего в БЗ;

3. Аналогично, запрос у пользователя связи для поиска по семантической сети;

4. Запуск в работу функции *handler* (листинг 4), которая в свою очередь вызывает *processing* с параметрами, введенными пользователем;

5. Полученные *child* для указанных связи и имени объекта также проверяются на наличие уже у них интересующей связи.

6. В результате поиска все найденные *child* выводятся в консоль.

|  |
| --- |
| def handler(entity, connect):  result = {}  result[entity] = processing(entity, connect)  tmp = result.copy()  intermediate\_result = {}  end = False  while not(end):  for key, value in tmp.items():  for element in value:  if element != "Указанная сущность не поддерживает выбранную связь":  intermediate\_result[element] = processing(element, connect)  result.update(intermediate\_result)  tmp.clear()  tmp.update(intermediate\_result)  count = 0  for key, value in intermediate\_result.items():  count = count + 1 if value == "Указанная сущность не поддерживает выбранную связь" else count  end = True if count == len(intermediate\_result) else False  intermediate\_result.clear()  print("\nОтвет:")  for key, value in result.items():  print(key, ":", end=' ')  for element in value:  if element != "Указанная сущность не поддерживает выбранную связь":  print(element, end=', ')  print() |
| Листинг 4 – Реализация функции *handler* |

Листинг разработанной экспертной системы на основе семантической сети представлен в Приложении 2.

**Примеры работы программы**

Ниже представлено несколько примеров работы программы и выводы, которые она сделала, опираясь на имеющуюся семантическую сеть:

|  |  |
| --- | --- |
| 1. |  |
| 2. |  |
| 3. |  |

**Заключение**

В рамках лабораторной работы были сформированы навыки построения прототипа экспертной системы, использующего семантическую модель представления знаний.

Результатом работы являются:

* Описание базы знаний на *YAML* ;
* Оформленная схема семантической сети;
* Функционирующая экспертная система, основанная на составленной базе знаний.

Приложение 1

**Скрипт семантической сети в *YAML*-файле**

entity:

- id: id01

name: Убийство на поле для гольфа

- id: id02

name: Убийство в Восточном экспрессе

- id: id03

name: Гарри Поттер и Принц-полукровка

- id: id04

name: Дубровский

- id: id05

name: Мастер и Маргарита

- id: id06

name: Детектив

- id: id07

name: Роман

- id: id08

name: Убийство в восточном экспрессе (2010)

- id: id09

name: Harry Potter and the Half-Blood Prince (2009)

- id: id10

name: Дубровский (2014)

- id: id11

name: Мститель (1959)

- id: id12

name: Pilatus and Andere (1972)

- id: id13

name: Мастер и Маргарита (1994)

- id: id14

name: Эркюль Пуаро

- id: id15

name: Severus Snape

- id: id16

name: Маша Троекурова

- id: id17

name: Воланд

connection:

- parent: id01

child: id06

connect: Жанр

- parent: id02

child: id07

connect: Жанр

- parent: id02

child: id08

connect: Киноадаптация

- parent: id03

child: id06

connect: Жанр

- parent: id03

child: id09

connect: Киноадаптация

- parent: id04

child: id07

connect: Жанр

- parent: id04

child: id10

connect: Киноадаптация

- parent: id04

child: id11

connect: Киноадаптация

- parent: id05

child: id07

connect: Жанр

- parent: id05

child: id12

connect: Киноадаптация

- parent: id05

child: id13

connect: Киноадаптация

- parent: id14

child: id01

connect: Герой произведения

- parent: id14

child: id02

connect: Герой произведения

- parent: id15

child: id03

connect: Герой произведения

- parent: id16

child: id04

connect: Герой произведения

- parent: id17

child: id05

connect: Герой произведения

Приложение 3

**Листинг экспертной системы**

import yaml

def toIdConvert(name):

for element in data['entity']:

if element['name'] == name:

id = element['id']

return id

def toNameConvert(id):

for element in data['entity']:

if element['id'] == id:

name = element['name']

return name

def searchConnections(id, connectType):

search\_result = []

for element in data['connection']:

if element['parent'] == id and element['connect'] == connectType:

search\_result.append(element['child'])

if len(search\_result) == 0:

search\_result.append("Указанная сущность не поддерживает выбранную связь")

return search\_result

def processing(entity, connect):

id = entity if "id" in entity else toIdConvert(entity)

result = searchConnections(id, connect)

final\_answer = {}

tmp = []

for element in result:

if element != "Указанная сущность не поддерживает выбранную связь":

tmp.append(toNameConvert(element))

else:

tmp.append(element)

final\_answer[entity] = tmp

return final\_answer[entity]

def handler(entity, connect):

result = {}

result[entity] = processing(entity, connect)

tmp = result.copy()

intermediate\_result = {}

end = False

while not(end):

for key, value in tmp.items():

for element in value:

if element != "Указанная сущность не поддерживает выбранную связь":

intermediate\_result[element] = processing(element, connect)

result.update(intermediate\_result)

tmp.clear()

tmp.update(intermediate\_result)

count = 0

for key, value in intermediate\_result.items():

count = count + 1 if value == "Указанная сущность не поддерживает выбранную связь" else count

end = True if count == len(intermediate\_result) else False

intermediate\_result.clear()

print("\nОтвет:")

for key, value in result.items():

print(key, ":", end=' ')

for element in value:

print(element, end=', ')

print()

with open("Base.yaml", "r", encoding="utf-8") as strokes:

data = yaml.safe\_load(strokes)

all\_entities = []

for element in data['entity']:

all\_entities.append(element['id'])

entity = ""

correct = 0

while correct == 0:

entity = input("Введите сущность: ")

for element in data['entity']:

correct = correct + 1 if element['name'] == entity else correct

correct = 0

while correct == 0:

connect = input("Введите тип связи: ")

for element in data['connection']:

correct = correct + 1 if element['connect'] == connect else correct

handler(entity, connect)