Графы знаний: Фракции персонажей игры Honkai:Star Rail

Предметная область: Взаимосвязи персонажей и их принадлежность к фракциям в игре Honkai Star Rail

Задача: Создание структурированного знания о персонажах и их фракциях для анализа, упрощения навигации по лору игры и предоставления базы для интеграции с другими приложениями

Для кого: Для фанатов игры, аналитиков игрового лора, создателей контента, разработчиков фан проектов и интерактивных приложений

Выполнено магистрантами 2 курса AITH: Гузенко Мария и Дерюга Полина

[1]:
```
!pip install rdflib
```

```
Requirement already satisfied: rdflib in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (7.1.1)
Requirement already satisfied: isodate<1.0.0,>=0.7.2 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from rdflib)
(0.7.2)
Requirement already satisfied: pyparsing<4,>=2.1.0 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from rdflib)
(3.2.0)

WARNING: You are using pip version 21.2.3; however, version 24.3.1 is available.
You should consider upgrading via the 'C:\Users\mashik\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip' comman
d.
```

[13]:
```
import warnings
import os
warnings.simplefilter("ignore")
```

[2]:
```
from rdflib import URIRef, BNode, Literal, Namespace, Graph
from rdflib.namespace import Namespace, NamespaceManager
from rdflib.plugins import sparql
from rdflib.namespace import RDF, RDFS, XSD
from rdflib.serializer import Serializer
import rdflib
```

Загружаем скелет онтологии без экземпляров

[9]:
```
g = Graph()
```

[4]:
```
g.parse('C:/Users/mashik/Protege-5.6.4-win/Protege-5.6.4/projects/kg_hsr_turtle', format="turtle")
```

[4]:
```
<Graph identifier=N2e4e58834e8e43a9ac4e6a529a495571 (<class 'rdflib.graph.Graph'>)>
```

Парсинг с сайтов

проблема: больше всего информации на англоязычных сайтах, но большинство имен и названий переводится некорректно с английского на русский (т.к. игра переводится напрямую с китайского) -> использовать переводчик не вариант

решение: парсить с русских сайтов, сверяясь с английской вики и добавляя изменения вручную (разногласий мало, так что отдельный скрипт писать смысла нет

сайт №1: https://honkai-star-rail.fandom.com/ru/wiki/Фракции

[5]:
```
import requests
from bs4 import BeautifulSoup
```

[6]:
```
url = "https://honkai-star-rail.fandom.com/ru/wiki/Фракции"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")
```

[7]:
```
data = []
```

[8]:
```
# Находим таблицу с нужным классом
table = soup.find("table", class_="wikitable sortable hover-row")

# Проверяем, нашли ли таблицу
if table:
    # Получаем все строки таблицы (включая заголовок)
    rows = table.find_all("tr")[1:]  # Пропускаем первую строку, это заголовок таблицы

    # Перебираем строки и извлекаем данные
    for row in rows:
        cols = row.find_all("td")  # Находим все ячейки в строке

        # Если строка не пуста
        if len(cols) > 0:
            faction_name = cols[0].get_text(strip=True) if len(cols) > 0 else None
            eon_name = cols[1].get_text(strip=True) if len(cols) > 1 else None
            leader_name = cols[2].get_text(strip=True) if len(cols) > 2 else None
            subordinate_name = cols[3].get_text(strip=True) if len(cols) > 3 else None
            species_name = cols[4].get_text(strip=True) if len(cols) > 4 else None
            location_name = cols[5].get_text(strip=True) if len(cols) > 5 else None

            # Сохраняем данные в словарь и добавляем его в список
            data.append({
                "faction": faction_name,
                "eon": eon_name,
                "leader": leader_name,
                "subordinates": subordinate_name,
                "species": species_name,
                "location": location_name
            })
else:
    print("Таблица с классом 'wikitable sortable hover-row' не найдена.")
```

обработка полученных значений

[10]:
```
import re

def process_field(field_value):
    """
    Функция для обработки строковых значений:
    - Добавление пробела между словами, если они были слиты.
    - Разделение значений, если они разделены запятой.
    """
    field_value = re.sub(r'([a-za-яё])([A-ZA-ЯЁ])', r'\1,\2', field_value)  # запятая перед заглавной буквой

    # Разделение значений, если есть запятая
    return [x.strip() for x in field_value.split(",")]
```

[11]:
```
for entry in data:
    # Применяем функцию к каждому полю
    entry["species"] = process_field(entry["species"])
    entry["subordinates"] = process_field(entry["subordinates"])
    entry["location"] = process_field(entry["location"])
    entry["leader"] = process_field(entry["leader"])

    print(entry)
```

```
{'faction': 'Безымянные', 'eon': 'Акивили', 'leader': ['Неизвестно'], 'subordinates': ['Звёздный экспресс'], 'species': ['Люди', 'Видьядхар
а'], 'location': ['Звёздный экспресс']}
{'faction': 'Звёздный экспресс', 'eon': 'Акивили', 'leader': ['Химеко', 'Пом-Пом'], 'subordinates': ['Нет'], 'species': ['Люди', 'Видьядхар
а'], 'location': ['Звёздный экспресс']}
{'faction': 'Охотники за Стелларонами', 'eon': 'Неизвестно', 'leader': ['Элио'], 'subordinates': ['Нет'], 'species': ['Люди'], 'location':
['Неизвестно']}
{'faction': 'Альянс Сяньчжоу', 'eon': 'Лань', 'leader': ['Хуа(адмирал', 'Облачных Рыцарей)Генералы-арбитры'], 'subordinates': ['Облачные рыц
ари', 'Комиссия по предсказаниям', 'Комиссия по полётам', 'Комиссия по алхимии', 'Комиссия по ремёслам', 'Комиссия по балансу', 'Комиссия де
сяти владык'], 'species': ['Жители Сяньчжоу', 'Видьядхара', 'Лисий народ'], 'location': ['Яоцин Сяньчжоу', 'Лофу Сяньчжоу', 'Юйцюэ Сяньчжо
у', 'Чжумин Сяньчжоу', 'Фанху Сяньчжоу', 'Сюйлин Сяньчжоу']}
{'faction': 'Галактические рейнджеры', 'eon': 'Лань', 'leader': ['Ла Манча'], 'subordinates': ['Неизвестно'], 'species': ['Неизвестно'], 'lo
cation': ['Неизвестно']}
{'faction': 'Доктора Хаоса', 'eon': 'IX', 'leader': ['Неизвестно'], 'subordinates': ['Неизвестно'], 'species': ['Неизвестно'], 'location':
['Неизвестно']}
{'faction': 'Устройство IX', 'eon': 'IX', 'leader': ['Неизвестно'], 'subordinates': ['Неизвестно'], 'species': ['Неизвестно'], 'location':
['Неизвестно']}
{'faction': 'Обитатели Изобилия', 'eon': 'Яоши', 'leader': ['Неизвестно'], 'subordinates': ['Ученики Санктус Медикус'], 'species': ['Вьюнокр
ылые', 'Борисинцы', 'Гуигнгнмы', 'Мясоеды'], 'location': ['Неизвестно']}
```

сайт №2: https://wiki.hoyolab.com/pc/hsr/aggregate/104

страница использует JavaScript для динамической загрузки данных, поэтому вспоминаем как использовать Selenium

[12]:
```
!pip install selenium
```

```
Requirement already satisfied: selenium in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (4.27.1)
Requirement already satisfied: urllib3[socks]<3,>=1.26 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from se
lenium) (2.2.3)
Requirement already satisfied: trio-websocket~=0.9 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from seleni
um) (0.11.1)
Requirement already satisfied: typing_extensions~=4.9 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from sel
enium) (4.12.2)
Requirement already satisfied: trio~=0.17 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from selenium) (0.2
7.0)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from seleniu
m) (2024.8.30)
Requirement already satisfied: websocket-client~=1.8 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from sele
nium) (1.8.0)
Requirement already satisfied: cffi>=1.14 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from trio~=0.17->sel
enium) (1.17.1)
Requirement already satisfied: idna in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from trio~=0.17->selenium)
(3.10)
Requirement already satisfied: exceptiongroup in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from trio~=0.17-
```

[13]:
```
!pip install webdriver-manager
```

[13]:
```
!pip install webdriver-manager
```

```
Requirement already satisfied: webdriver-manager in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (4.0.2)
Requirement already satisfied: python-dotenv in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from webdriver-ma
nager) (1.0.1)
Requirement already satisfied: requests in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from webdriver-manage
r) (2.32.3)
Requirement already satisfied: packaging in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from webdriver-manage
r) (24.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from request
s->webdriver-manager) (2.2.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from r
equests->webdriver-manager) (3.4.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from request
s->webdriver-manager) (2024.8.30)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from requests->web
driver-manager) (3.10)

WARNING: You are using pip version 21.2.3; however, version 24.3.1 is available.
You should consider upgrading via the 'C:\Users\mashik\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip' comm
and.
```

[14]:
```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
```

[15]:
```
# Настройка Selenium
chrome_options = Options()
chrome_options.add_argument("--headless")  # Без графического интерфейса
chrome_options.add_argument("--disable-gpu")  # Для стабильности

# Настройка ChromeDriver
service = Service(ChromeDriverManager().install())
driver = webdriver.Chrome(service=service, options=chrome_options)

try:
    # Открываем страницу
    url = "https://wiki.hoyolab.com/pc/hsr/aggregate/104?lang=ru-ru"
    driver.get(url)

    # Ждем загрузки
    wait = WebDriverWait(driver, 20)
    wait.until(lambda d: d.execute_script("return document.readyState") == "complete")

    # Прокрутка страницы для подгрузки всех данных
    last_height = driver.execute_script("return document.body.scrollHeight")
    while True:
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(2)
        new_height = driver.execute_script("return document.body.scrollHeight")
        if new_height == last_height:
            break
        last_height = new_height
```

```
    # Поиск элементов
    character_elements = driver.find_elements(By.CLASS_NAME, "rpg-character-card-name")

    # Сохранение имен
    characters = [element.text for element in character_elements if element.text]
    print("Имена персонажей:", characters)

finally:
    # Закрываем браузер
    driver.quit()
```

```
Имена персонажей: ['Первопроходец: Память (Скоро)', 'Аглая (Скоро)', 'Великая Герта (Скоро)', 'Фуга (Скоро)', 'Воскресенье', 'Раппа', 'Моцзэ', 'Линша', 'Фэйсяо', 'Март 7 - Охота', 'Цзяоцю', 'Юньли', 'Первопроходец (Гармония)', 'Яшма', 'Светлячок', 'Бутхилл', 'Зарянка', 'Галлахер', 'Авантюрин', 'Ахерон', 'Миша', 'Искорка', 'Чёрный Лебедь', 'Сюэи', 'Доктор Рацио', 'Жуань Мэй', 'Ханья', 'Аргенти', 'Хохо', 'Гуйнайфэнь', 'Топаз и Счетовод', 'Цзинлю', 'Рысь', 'Дань Хэн: Пожиратель Луны', 'Фу Сюань', 'Лука', 'Кафка', 'Блэйд', 'Юйкун', 'Лоча', 'Серебряный Волк', 'Байлу', 'Яньцин', 'Сушан', 'Цзин Юань', 'Тинъюнь', 'Цинцюэ', 'Первопроходец (Сохранение)', 'Хук', 'Сампо', 'Клара', 'Пела', 'Наташа', 'Гепард', 'Сервал', 'Зеле', 'Броня', 'Герта', 'Аста', 'Арлан', 'Вельт', 'Химеко', 'Дань Хэн', 'Март 7', 'Первопроходец (Разрушение)']
```

сайт №3: https://honkai-star-rail.fandom.com/ru/wiki/Пути

[16]:
```
url = "https://honkai-star-rail.fandom.com/ru/wiki/Пути"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")
```

[17]:
```
paths = {}
```

[18]:
```
# Находим таблицы с нужным классом
tables = soup.find_all("table", class_="wikitable sortable")

for table in tables:
    # Проходим по строкам таблицы, исключая заголовок
    for row in table.find_all("tr")[1:]:
        cells = row.find_all("td")

        # Вытаскиваем названия путей и имена эонов
        if len(cells) >= 2:
            path = cells[0].get_text(strip=True)
            aeon = cells[1].get_text(strip=True)
            paths[path] = aeon

print(paths)
```

```
{'Разрушение': 'Нанук', 'Охота': 'Лань', 'Эрудиция': 'Нус', 'Гармония': 'Шипе', 'Небытие': 'IX', 'Сохранение': 'Клипот', 'Изобилие': 'Яоши', 'Освоение': 'Акивили', 'Ненасытность': 'Ороборос', 'Радость': 'Аха', 'Красота': 'Идрила', 'Распространение': 'Тайззиронт', 'Энигмата': 'Мифус', 'Порядок': 'Эна', 'Равновесие': 'ХуХ', 'Завершённость': 'Терминус', 'Память': 'Фули', 'Постоянство': 'Лун'}
```

Добавляем экземпляры классов в граф

[10]:
```
# Пространство имен для онтологии
base_uri = "http://www.semanticweb.org/mashik/ontologies/2024/11/untitled-ontology-3/"

# Исключаем значения "Нет" и "Неизвестно"
excluded_values = {"Неизвестно", "Нет"}
```

[20]:
```python
def add_faction_to_graph(graph, faction_name):
    """Добавляет фракцию в граф RDF"""
    faction_class_uri = URIRef(base_uri + "Фракция")
    if faction_name and faction_name not in excluded_values:
        faction_uri = URIRef(base_uri + faction_name.replace(" ", "_"))
        if not (faction_uri, RDF.type, faction_class_uri) in graph:
            graph.add((faction_uri, RDF.type, faction_class_uri))
            graph.add((faction_uri, RDFS.label, Literal(faction_name, datatype=XSD.string)))
```

[21]:
```python
def add_eon_to_graph(graph, eon_name):
    """Добавляет эона в граф RDF"""
    eon_class_uri = URIRef(base_uri + "Эон")
    if eon_name and eon_name not in excluded_values:
        eon_uri = URIRef(base_uri + f"Эон_{eon_name.replace(' ', '_')}")
        if not (eon_uri, RDF.type, eon_class_uri) in graph:
            graph.add((eon_uri, RDF.type, eon_class_uri))
            graph.add((eon_uri, RDFS.label, Literal(eon_name, datatype=XSD.string)))
```

[22]:
```python
def add_subordinates_to_graph(graph, subordinates, base_uri, excluded_values):
    """Добавляет подчиненные организации в граф RDF

    Args:
        graph: RDF-граф
        subordinates: Список подчиненных организаций
        base_uri: Базовый URI для создания новых экземпляров
        excluded_values: Набор значений, которые необходимо пропустить
    """
    # URI подклассов
    subclass_uris = {
        "Комиссия": URIRef(base_uri + "Комиссия"),
        "Отдел": URIRef(base_uri + "Отдел"),
        "Союз": URIRef(base_uri + "Союз"),
        "Клан": URIRef(base_uri + "Клан"),
        "Силы_обороны": URIRef(base_uri + "Силы_обороны"),
    }

    for subordinate in subordinates:
        # Пропускаем исключённые значения
        if subordinate in excluded_values:
            continue

        # Определяем подкласс по ключевым словам
        if "Комиссия" in subordinate:
            subclass_uri = subclass_uris["Комиссия"]
        elif "Отдел" in subordinate:
            subclass_uri = subclass_uris["Отдел"]
        elif "Союз" in subordinate:
            subclass_uri = subclass_uris["Союз"]
        elif "Клан" in subordinate:
            subclass_uri = subclass_uris["Клан"]
        else:
            subclass_uri = subclass_uris["Силы_обороны"]
```

```
        # Уникальный URI экземпляра
        subordinate_instance_uri = URIRef(base_uri + subordinate.replace(" ", "_"))

        # Проверяем, существует ли уже экземпляр в графе
        if not (subordinate_instance_uri, RDF.type, subclass_uri) in graph:
            # Добавляем экземпляр в граф
            graph.add((subordinate_instance_uri, RDF.type, subclass_uri))
            graph.add((subordinate_instance_uri, RDFS.label, Literal(subordinate, datatype=XSD.string)))
```

23]:
```
def add_character_to_graph(graph, character_name):
    """
    Добавляет персонажа в граф RDF как экземпляр класса 'Последователь_пути'
    """
    character_class_uri = URIRef(base_uri + "Последователь_пути")
    if character_name and character_name not in excluded_values:
        # Создаем URI для экземпляра персонажа
        character_uri = URIRef(base_uri + f"Персонаж_{character_name.replace(' ', '_')}")

        # Проверяем, существует ли экземпляр уже в графе
        if not (character_uri, RDF.type, character_class_uri) in graph:
            # Добавляем экземпляр и его свойства
            graph.add((character_uri, RDF.type, character_class_uri))
            graph.add((character_uri, RDFS.label, Literal(character_name, datatype=XSD.string)))
```

24]:
```
# Добавляем фракции, организации и эонов в граф
for entry in data:
    add_faction_to_graph(g, entry.get('faction'))
    add_eon_to_graph(g, entry.get('eon'))

    subordinates = entry.get("subordinates", [])
    add_subordinates_to_graph(g, subordinates, base_uri, excluded_values)

# Добавляем персонажей в граф
for character in characters:
    add_character_to_graph(g, character)
```

25]:
```
g.serialize(destination="kg_hsr_turtle", format="turtle")
```

25]:
```
<Graph identifier=N2e4e58834e8e43a9ac4e6a529a495571 (<class 'rdflib.graph.Graph'>)>
```

Добавляем ObjectProperty

37]:
```
def link_subordinates_to_factions(graph, data, property_name="состоит_в"):
    """
    Связывает подчиненные организации с фракциями с помощью ObjectProperty 'состоит_в'

    :param graph: Граф RDF
    :param data: Список словарей с информацией о фракциях и их подчиненных организациях
    :param property_name: Название ObjectProperty
    """
    # Создаем URI для ObjectProperty
    property_uri = URIRef(base_uri + property_name)
```

```
    for entry in data:
        faction_name = entry.get("faction")
        subordinates = entry.get("subordinates", [])

        # Пропускаем, если фракция или подчиненные отсутствуют
        if not faction_name or faction_name in excluded_values:
            continue

        # Создаем URI для фракции (range)
        faction_uri = URIRef(base_uri + faction_name.replace(" ", "_"))

        for subordinate in subordinates:
            # Пропускаем исключённые значения
            if not subordinate or subordinate in excluded_values:
                continue

            # Создаем URI для подчиненной организации (domain)
            subordinate_uri = URIRef(base_uri + subordinate.replace(" ", "_"))

            # Добавляем связь "состоит_в" между организацией и фракцией
            graph.add((subordinate_uri, URIRef(property_uri), faction_uri))
```

[38]:
```
def add_paths_and_link_to_aeons(graph, paths, property_name="кем_основан"):
    """
    Добавляет элементы класса "Путь" и связывает их с эонами через ObjectProperty 'кем_основан'

    :param graph: Граф RDF
    :param paths: Словарь, где ключ - название пути, а значение - название эона
    :param property_name: Название ObjectProperty
    """
    # Создаем URI для ObjectProperty
    property_uri = URIRef(base_uri + property_name)

    for path_name, aeon_name in paths.items():
        # Создаем URI для пути (класс "Путь")
        path_uri = URIRef(base_uri + path_name.replace(" ", "_"))

        # Получаем URI для эона из графа
        aeon_uri = URIRef(base_uri + aeon_name.replace(" ", "_"))

        # Добавляем путь в граф как экземпляр класса "Путь"
        graph.add((path_uri, RDF.type, URIRef(base_uri + "Путь")))

        # Добавляем связь "кем_основан" между путем и эоном
        graph.add((path_uri, property_uri, aeon_uri))
```

[35]:
```
# связываем экземпляры классов с помощью ObjectProperties 'состоит_в' и 'кем_основан'
link_subordinates_to_factions(g, data)
add_paths_and_link_to_aeons(g, paths)
```

[36]:
```
g.serialize(destination="kg_hsr_turtle", format="turtle")
```

[36]:
```
<Graph identifier=N2e4e58834e8e43a9ac4e6a529a495571 (<class 'rdflib.graph.Graph'>)>
```

Добавляем DataProperty

[30]:
```
def add_location_to_faction(data, excluded_values, graph, base_uri):
    """
    Добавляет локации и связывает их с фракциями через DataProperty 'локация'

    :param data: Список словарей с информацией о фракциях и их локациях
    :param excluded_values: Список исключенных значений
    :param graph: Граф RDF
    :param base_uri: Пространство имен для онтологии
    """
    for entry in data:
        faction_name = entry.get("faction")
        location_list = entry.get("location", [])

        # Пропускаем исключённые значения
        if faction_name in excluded_values:
            continue

        # Создаем уникальный URI для фракции
        faction_instance_uri = URIRef(base_uri + faction_name.replace(" ", "_"))

        # Привязываем локацию к фракции через DataProperty
        for location in location_list:
            if location in excluded_values:
                continue

            # Уникальный URI для каждой локации
            location_instance_uri = URIRef(base_uri + location.replace(" ", "_"))

            # Добавляем DataProperty для локации
            graph.add((faction_instance_uri, URIRef(base_uri + "локация"), Literal(location, datatype=XSD.string)))
```

[31]:
```
# Добавляем локации в граф
add_location_to_faction(data, excluded_values, g, base_uri)
```

[32]:
```
g.serialize(destination="kg_hsr_turtle", format="turtle")
```

[32]:
```
<Graph identifier=N2e4e58834e8e43a9ac4e6a529a495571 (<class 'rdflib.graph.Graph'>)>
```

Далее в граф были внесены правки вручную в Protege:

- классы сделаны непересекающимися (с помощью свойства DisjointWith)
- созданы подклассы организаций
- добавлены несколько ObjectProperties
- настроены характеристики свойств (транзитивность, симметричность, функциональность)
- добавлены несколько DataProperties

Запросы к графу

[12]:
```
g.parse('C:/Users/mashik/Protege-5.6.4-win/Protege-5.6.4/projects/kg_hsr', format="turtle")
```

[12]:
```
<Graph identifier=N24396b8b4747424d8729a750b616930a (<class 'rdflib.graph.Graph'>)>
```

1. Какие фракции следуют пути сохранения?

[58]:
```
query = """
PREFIX untitled-ontology-3: <http://www.semanticweb.org/mashik/ontologies/2024/11/untitled-ontology-3/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?название_фракции
WHERE {
  ?путь rdfs:label "Сохранение" ;
        rdf:type untitled-ontology-3:Путь.
  ?фракция rdf:type untitled-ontology-3:фракция ;
           untitled-ontology-3:следует_пути ?путь ;
           rdfs:label ?название_фракции.
}
"""
results = g.query(query)
```

[61]:
```
factions = [row[0].split('/')[-1].replace('_', ' ') for row in results]
factions_str = ', '.join(factions)
print(f"Фракции, следующие пути Сохранения: {factions_str}")
```

```
Фракции, следующие пути Сохранения: Архитекторы, Десять каменных сердец
```

2. Кто из персонажей состоит в фракции "Безымянные"?

[41]:
```
query = """
PREFIX untitled-ontology-3: <http://www.semanticweb.org/mashik/ontologies/2024/11/untitled-ontology-3/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?label
WHERE {
    ?фракция rdfs:label "Безымянные" ;
             rdf:type untitled-ontology-3:фракция.

    ?персонаж rdf:type untitled-ontology-3:Персонаж;
              untitled-ontology-3:состоит_в ?фракция;
              rdfs:label ?label.
}
"""
results = g.query(query)
```

[43]:
```
# Сохраняем и выводим результаты
ch = [row[0].split('/')[-1].replace('_', ' ') for row in results]

# Формируем строку для вывода
ch_str = ', '.join(ch)
print(f"Персонажи, состоящие во фракции 'Безымянные': {ch_str}")
```

```
Персонажи, состоящие во фракции 'Безымянные': Вельт, Дань Хэн, Март 7, Миша, Химеко, Первопроходец
```

3. Как связан Альянс Сяньчжоу с другими фракциями?

[45]:
```
query = """
PREFIX untitled-ontology-3: <http://www.semanticweb.org/mashik/ontologies/2024/11/untitled-ontology-3/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?relation ?relatedFactionLabel
WHERE {
    ?фракция rdfs:label "Альянс Сяньчжоу" ;
             rdf:type untitled-ontology-3:Фракция.

    ?фракция ?relation ?другаяФракция .

    ?другаяФракция rdf:type untitled-ontology-3:Фракция ;
                   rdfs:label ?relatedFactionLabel .

    FILTER (?relation IN (untitled-ontology-3:враждует_с, untitled-ontology-3:в_союзе_с))
}
"""
results = g.query(query)
```

[47]:
```
# Обработка и вывод результатов
print("Результаты:")
for row in results:
    relation = row.relation.split('/')[-1]  # Убираем URI и оставляем только название отношения
    faction_label = row.relatedFactionLabel
    print(f"Альянс Сяньчжоу {relation} {faction_label}")
```

```
Результаты:
Альянс Сяньчжоу в_союзе_с Архитекторы
Альянс Сяньчжоу в_союзе_с Безымянные
Альянс Сяньчжоу в_союзе_с Корпорация межзвёздного мира
Альянс Сяньчжоу враждует_с Легион Антиматерии
Альянс Сяньчжоу враждует_с Обитатели Изобилия
```

4. Какие пути основаны эонами?

[48]:
```
query = """
PREFIX untitled-ontology-3: <http://www.semanticweb.org/mashik/ontologies/2024/11/untitled-ontology-3/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?эонLabel ?путьLabel
WHERE {
    ?путь rdf:type untitled-ontology-3:Путь ;
          rdfs:label ?путьLabel ;
          untitled-ontology-3:кем_основан ?эон .
    ?эон rdf:type untitled-ontology-3:Эон ;
         rdfs:label ?эонLabel .
}
"""
results = g.query(query)
```

[49]:
```
print("Эоны и их пути:")
for row in results:
    print(f"{row.эонLabel} основал {row.путьLabel}")
```

```
Эоны и их пути:
Шипе основал Гармония
Эна основал Гармония
Терминус основал Завершённость
Яоши основал Изобилие
Идрила основал Красота
IX основал Небытие
Акивили основал Освоение
Лань основал Охота
Фули основал Память
Шипе основал Порядок
Эна основал Порядок
ХуХ основал Равновесие
Аха основал Радость
Нанук основал Разрушение
Тайэзиронт основал Распространение
Клипот основал Сохранение
Мифус основал Энигмата
Нус основал Эрудиция
```

Создаем документацию автоматически с помощью pyLode

[4]:
```
!pip install pylode
```

```
Collecting pylode
  Downloading pylode-3.2.0-py3-none-any.whl (693 kB)
Requirement already satisfied: httpx<1.0.0,>=0.25.0 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from pylod
e) (0.27.2)
Collecting Markdown<4.0.0,>=3.3.7
  Downloading Markdown-3.7-py3-none-any.whl (106 kB)
Requirement already satisfied: rdflib<8.0.0,>=7.0.0 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from pylod
e) (7.1.1)
Collecting html5lib<2.0,>=1.1
  Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
Collecting dominate<3.0.0,>=2.6.0
  Downloading dominate-2.9.1-py2.py3-none-any.whl (29 kB)
Requirement already satisfied: webencodings in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from html5lib<2.0,
>=1.1->pylode) (0.5.1)
Requirement already satisfied: six>=1.9 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from html5lib<2.0,>=1.
1->pylode) (1.16.0)
Requirement already satisfied: idna in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from httpx<1.0.0,>=0.25.0-
>pylode) (3.10)
```

[19]:
```
from rdflib import Graph
from pylode.profiles.ontpub import OntPub
```

[27]:
```
!pip install rdflib-jsonld
```

[31]:
```python
# 1. Задаем путь к owl файлу
file_path_owl = Path('C:/Users/mashik/Protege-5.6.4-win/Protege-5.6.4/projects/kg_hsr_ttl')


# 2. Проверяем, существует ли файл
if not file_path_owl.exists():
    print(f"Ошибка: Файл {file_path_owl} не найден.")
else:
    # 3. Использование pyLODE
    try:
        # Инициализируем OntPub, передаем путь к .owl файлу
        od = OntPub(ontology=str(file_path_owl))

        # Генерируем HTML
        od.make_html(destination="kg_hsr_doc.html")

        print("Документация успешно сгенерирована в kg_hsr_doc.html")

    except Exception as e:
       print(f"Ошибка при генерации документации: {e}")
```

```
INFO:root:Loading background ontologies from a pickle file
Документация успешно сгенерирована в kg_hsr_doc.html
```

Генерируем VoID

[32]:
```python
from rdflib.void import generateVoID
```

[33]:
```python
# 2. Вызов generateVoID
void_graph, dataset_uri = generateVoID(g)

# 3. Вывод или сохранение VoID

print("VoID description:")
print(void_graph.serialize(format="turtle"))
```

```
VoID description:
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://example.org/Dataset> a void:Dataset ;
    void:classPartition <http://example.org/Dataset_class0>,
        <http://example.org/Dataset_class1>,
        <http://example.org/Dataset_class10>,
        <http://example.org/Dataset_class11>,
        <http://example.org/Dataset_class12>,
        <http://example.org/Dataset_class13>,
        <http://example.org/Dataset_class14>,
        <http://example.org/Dataset_class15>,
        <http://example.org/Dataset_class16>,
```

Графовые эмбеддинги

[12]:
```
!pip install tensorflow==2.9.0
!pip install ampligraph
```

```
Collecting tensorflow==2.9.0
  Downloading tensorflow-2.9.0-cp310-cp310-win_amd64.whl (444.1 MB)
Collecting google-pasta>=0.1.1
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
Collecting absl-py>=1.0.0
  Downloading absl_py-2.1.0-py3-none-any.whl (133 kB)
Collecting libclang>=13.0.0
  Downloading libclang-18.1.1-py2.py3-none-win_amd64.whl (26.4 MB)
Requirement already satisfied: setuptools in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from tensorflow==2.
9.0) (57.4.0)
Collecting termcolor>=1.1.0
  Downloading termcolor-2.5.0-py3-none-any.whl (7.8 kB)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from t
ensorflow==2.9.0) (4.12.2)
Collecting protobuf>=3.9.2
  Downloading protobuf-5.29.3-cp310-abi3-win_amd64.whl (434 kB)
Requirement already satisfied: numpy>=1.20 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from tensorflow==2.
9.0) (1.26.4)
```

[25]:
```
import pandas as pd
import numpy as np
import re
import ampligraph
import tensorflow as tf
from ampligraph.evaluation import train_test_split_no_unseen
from ampligraph.latent_features import ScoringBasedEmbeddingModel
from ampligraph.latent_features.loss_functions import get as get_loss
from ampligraph.latent_features.regularizers import get as get_regularizer
from ampligraph.evaluation import mr_score, mrr_score, hits_at_n_score
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
```

[15]:
```
g = Graph()
g.parse('C:/Users/mashik/Protege-5.6.4-win/Protege-5.6.4/projects/kg_hsr_2', format="xml")

# Извлечение триплетов
query = """
PREFIX untitled-ontology-3: <http://www.semanticweb.org/mashik/ontologies/2025/0/12/untitled-ontology-3/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subjectLabel ?property ?objectLabel
WHERE {
  ?subject ?property ?object .
  ?property rdf:type owl:ObjectProperty .
  ?subject rdf:type ?Персонаж;
           rdfs:label ?subjectLabel.
  ?object rdfs:label ?objectLabel.
}
"""

# Выполнение запроса
results = g.query(query)
```

[16]:
```
# Преобразование результатов в DataFrame
data = []
for row in results:
    data.append([str(row.subjectLabel), str(row.property), str(row.objectLabel)])

df = pd.DataFrame(data, columns=["Subject", "Predicate", "Object"])
```

[17]:
```
# Функция для извлечения имени свойства из URL
def extract_property_name(url):
    # Если есть символ #, то берем все после него
    match = re.search(r'#(.*)', url)
    if match:
        return match.group(1)  # Возвращаем часть после '#'
    else:
        # Если # нет, берем последний сегмент пути после последнего '/'
        return url.split('/')[-1]  # Часть после последнего слэша

# Применяем функцию к столбцу Predicate
df['Predicate'] = df['Predicate'].apply(extract_property_name)
```

[18]:
```
df
```

[18]:

	Subject	Predicate	Object
0	Акивили	в_плохих_отношениях_с	Аха
1	Аха	в_плохих_отношениях_с	Акивили
2	Лань	в_плохих_отношениях_с	Яоши
3	Авантюрин	в_плохих_отношениях_с	Воскресенье
4	Ахерон	в_плохих_отношениях_с	Бутхилл
...	...	...	...
2013	Цзинлю	является_учителем	Блэйд
2014	Цзинлю	является_учителем	Цзин Юань
2015	Юйкун	является_учителем	Тинъюнь
2016	Юньли	является_учителем	Март 7
2017	Яньцин	является_учителем	Март 7

2018 rows × 3 columns

[19]:
```
# Сохраняем датафрейм как CSV
df.to_csv('triplets.csv', index=False, header=True, encoding='utf-8-sig')
```

[22]:
```
triples = df.to_numpy()
# Разделение на обучающую и тестовую выборки с сохранением всех узлов
X_train, X_valid = train_test_split_no_unseen(np.array(triples), test_size=300)
```

[23]:
```
print('Train set size: ', X_train.shape)
print('Test set size: ', X_valid.shape)
```

```
Train set size:  (1718, 3)
Test set size:  (300, 3)
```

[26]:
```
# Обучение модели векторных представлений (графовые эмбеддинги)
model = ScoringBasedEmbeddingModel(k=100,
                                   eta=20,
                                   scoring_type='ComplEx',
                                   seed=0)

# Optimizer, loss and regularizer definition
optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4)
loss = get_loss('multiclass_nll')
regularizer = get_regularizer('LP', {'p': 3, 'lambda': 1e-5})

# Compilation of the model
model.compile(optimizer=optimizer, loss=loss, entity_relation_regularizer=regularizer)
```

[27]:
```
model.fit(X_train,
          batch_size=int(X_train.shape[0] / 50),
          epochs=300, # Number of training epochs
          verbose=True # Displays a progress bar.
          )
```

```
Epoch 1/300
52/52 [==============================] - 3s 66ms/step - loss: 102.5428
Epoch 2/300
52/52 [==============================] - 0s 9ms/step - loss: 102.5191
Epoch 3/300
52/52 [==============================] - 0s 9ms/step - loss: 102.4989
Epoch 4/300
52/52 [==============================] - 0s 9ms/step - loss: 102.4781
Epoch 5/300
52/52 [==============================] - 0s 9ms/step - loss: 102.4567
Epoch 6/300
52/52 [==============================] - 0s 9ms/step - loss: 102.4353
Epoch 7/300
52/52 [==============================] - 0s 9ms/step - loss: 102.4134
Epoch 8/300
52/52 [==============================] - 1s 10ms/step - loss: 102.3907
Epoch 9/300
52/52 [==============================] - 0s 9ms/step - loss: 102.3672
```

[28]:
```
# Оценка модели векторных представлений
ranks = model.evaluate(X_valid,
                       use_filter={'train': X_train,
                                   'test': X_valid},
                       corrupt_side='s,o',
                       verbose=True)
```

```
31/31 [==============================] - 4s 131ms/step
```

[29]:
```
mr = mr_score(ranks)
mrr = mrr_score(ranks)

print("MRR: %.2f" % (mrr))
print("MR: %.2f" % (mr))

hits_10 = hits_at_n_score(ranks, n=10)
print("Hits@10: %.2f" % (hits_10))
hits_3 = hits_at_n_score(ranks, n=3)
print("Hits@3: %.2f" % (hits_3))
hits_1 = hits_at_n_score(ranks, n=1)
print("Hits@1: %.2f" % (hits_1))
```

```
MRR: inf
MR: -2.06
Hits@10: 0.99
Hits@3: 0.99
Hits@1: 0.99
```

Задача многоклассовой классификации: Определить, в каких отношениях с другими будет персонаж, исходя из его фракции

[34]:
```
!pip install adjustText
```

```
Collecting adjustText
  Downloading adjustText-1.3.0-py3-none-any.whl (13 kB)
Requirement already satisfied: matplotlib in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from adjustText) (3.
9.2)
Requirement already satisfied: numpy in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from adjustText) (1.26.4)
Requirement already satisfied: scipy in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from adjustText) (1.10.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from matplotl
ib->adjustText) (1.4.7)
Requirement already satisfied: cycler>=0.10 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from matplotlib->a
djustText) (0.12.1)
Requirement already satisfied: packaging>=20.0 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from matplotlib
->adjustText) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from matplotli
b->adjustText) (3.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from matplotli
b->adjustText) (1.3.1)
Requirement already satisfied: pillow>=8 in c:\users\mashik\appdata\local\programs\python\python310\lib\site-packages (from matplotlib->adju
stText) (11.0.0)
```

[35]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
from adjustText import adjust_text
```

[36]:
```
# Классификация

organizations = [
    "Вечногорящий Особняк",
    "Звёздный экспресс",
    "Клан Гончих",
    "Клан Дубов",
    "Клан Ирисов",
    "Клан Люцерн",
    "Клан Ночных Дроздов",
    "Комиссия десяти владык",
    "Комиссия по алхимии",
    "Комиссия по балансу",
    "Комиссия по полётам",
    "Комиссия по предсказаниям",
    "Комиссия по ремёслам",
    "Облачные рыцари",
    "Отдел консолидации бизнеса",
    "Отдел логистики строительных материалов",
    "Отдел мотивации",
    "Отдел развития маркетинга",
    "Отдел стратегических инвестиций",
    "Отдел технологий",
    "Отдел традиционных проектов",
    "Среброгривые Стражи",
    "Ученики Санктус Медикус"
]
```

```python
def filter_out_organizations(df, organizations):
    """
    Убирает строки, где Subject или Object совпадают с названиями организаций.
    """
    # Фильтруем DataFrame, оставляя только строки, где Subject и Object не принадлежат списку организаций
    filtered_df = df[~df["Subject"].isin(organizations) & ~df["Object"].isin(organizations)]
    return filtered_df

# Фильтруем данные

df_filtered = filter_out_organizations(df.copy(), organizations)
print("\nПосле фильтрации:")
print(df_filtered.head())
```

```
После фильтрации:
     Subject             Predicate       Object
0    Акивили  в_плохих_отношениях_с          Аха
1        Аха  в_плохих_отношениях_с      Акивили
2       Лань  в_плохих_отношениях_с         Яоши
3  Авантюрин  в_плохих_отношениях_с  Воскресенье
4     Ахерон  в_плохих_отношениях_с      Бутхилл
```

[37]:
```python
fractions = df_filtered[df_filtered["Predicate"] == "состоит_в"]
# Убираем лишние столбцы, оставляем только персонажей и их фракции
fraction_map = fractions[["Subject", "Object"]]
fraction_map.columns = ["Character", "Fraction"]
print(fraction_map)
```

```
          Character                     Fraction
1685      Авантюрин       Десять каменных сердец
1686      Авантюрин  Корпорация межзвёздного мира
1688        Аргенти               Рыцари Красоты
1689          Байлу              Альянс Сяньчжоу
1690          Блэйд      Охотники за Стелларонами
...             ...                          ...
1951           Яшма       Десять каменных сердец
1952           Яшма  Корпорация межзвёздного мира
1954  Первопроходец                   Безымянные
1956          Герта              Общество гениев
1957      Жуань Мэй              Общество гениев

[162 rows x 2 columns]
```

[38]:
```python
df_merged = df_filtered.merge(fraction_map, how="left", left_on="Subject", right_on="Character")
df_merged.drop(columns=["Character"], inplace=True)
df_merged.rename(columns={"Fraction": "Fraction_Subject"}, inplace=True)

df_merged = df_merged.merge(fraction_map, how="left", left_on="Object", right_on="Character", suffixes=('_subj', '_obj'))
df_merged.drop(columns=["Character"], inplace=True)
df_merged.rename(columns={"Fraction": "Fraction_Object"}, inplace=True)

print("\nОбъединенные данные:")
print(df_merged.head())

le = LabelEncoder()
df_merged["Fraction_Subject"] = le.fit_transform(df_merged["Fraction_Subject"].astype(str))
df_merged["Fraction_Object"] = le.fit_transform(df_merged["Fraction_Object"].astype(str))
df_merged['Predicate'] = le.fit_transform(df_merged['Predicate'])
df_merged['Subject'] = le.fit_transform(df_merged['Subject'].astype(str))
df_merged['Object'] = le.fit_transform(df_merged['Object'].astype(str))
```

```
print("\nДанные после Label Encoding:")
print(df_merged.head())
```

```
Объединенные данные:
     Subject             Predicate       Object        Fraction_Subject  \
0    Акивили  в_плохих_отношениях_с          Аха                     NaN
1        Аха  в_плохих_отношениях_с      Акивили                     NaN
2      Лань  в_плохих_отношениях_с         Яоши                     NaN
3  Авантюрин  в_плохих_отношениях_с  Воскресенье  Десять каменных сердец
4  Авантюрин  в_плохих_отношениях_с  Воскресенье  Десять каменных сердец

  Fraction_Object
0             NaN
1             NaN
2             NaN
3           Семья
4           Семья

Данные после Label Encoding:
   Subject  Predicate  Object  Fraction_Subject  Fraction_Object
0        2          0       8                 0                0
1        8          0       2                 0                0
2       42          0     104                 0                0
3        1          0      17                 6               13
4        1          0      17                 6               13
```

[39]:
```
# Получение эмбеддингов для субъектов и объектов
entity_embeddings = model.get_embeddings(np.unique(df_merged['Subject'].to_list() + df_merged['Object'].to_list()))
entity_to_embedding = {entity: emb for entity, emb in zip(np.unique(df_merged['Subject'].to_list() + df_merged['Object'].to_list()), entity_embe
```

```
106 triples containing invalid keys skipped! You can use `ScoringBasedEmbeddingModel.get_invalid_keys` or`DataIndexer.get_invalid_keys` to find
out which keys are invalid.
```

[40]:
```
# Сопоставление эмбеддингов с исходными данными
def get_entity_embedding(entity, entity_to_embedding):
    try:
        return entity_to_embedding[entity]
    except KeyError:
        return np.zeros(100)

df_merged["Subject_Emb"] = df_merged["Subject"].apply(lambda x: get_entity_embedding(x, entity_to_embedding))
df_merged["Object_Emb"] = df_merged["Object"].apply(lambda x: get_entity_embedding(x, entity_to_embedding))
```

[41]:
```
# Создаем признаки для классификатора (конкатенируем эмбеддинги и фракции)
def create_features(row):
    return np.concatenate((row["Subject_Emb"], row["Object_Emb"], [row["Fraction_Subject"], row["Fraction_Object"]]))

df_merged["Features"] = df_merged.apply(create_features, axis=1)
```

[42]:
```
# Разделение на обучающую и тестовую выборки
X = np.array(df_merged["Features"].tolist())
y = df_merged["Predicate"]
X_train_emb, X_test_emb, y_train_emb, y_test_emb = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

[43]:
```python
# Обучение XGBoost с эмбеддингами
xgb_emb = XGBClassifier(n_estimators=500, max_depth=5, objective="multi:softmax", random_state=42)
xgb_emb.fit(X_train_emb, y_train_emb)
```

[43]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=5, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=500, n_jobs=None,
              num_parallel_tree=None, objective='multi:softmax', ...)
```

[44]:
```python
# Оценка модели с эмбеддингами
y_pred_emb = xgb_emb.predict(X_test_emb)
print("\nОтчет классификации с использованием эмбеддингов:")
print(classification_report(y_test_emb, y_pred_emb))
print(f"Accuracy с использованием эмбеддингов: {accuracy_score(y_test_emb, y_pred_emb)}")
```

```
Отчет классификации с использованием эмбеддингов:
              precision    recall  f1-score   support

           0       0.89      0.56      0.69       154
           1       0.00      0.00      0.00         9
           2       0.94      0.98      0.96      2397
           3       0.00      0.00      0.00         5
           4       0.00      0.00      0.00         6
           5       0.31      1.00      0.48        11
           7       0.67      0.92      0.78       108
           8       0.00      0.00      0.00        40
           9       0.00      0.00      0.00        20
          10       0.00      0.00      0.00        20

    accuracy                           0.92      2770
   macro avg       0.28      0.35      0.29      2770
weighted avg       0.89      0.92      0.90      2770

Accuracy с использованием эмбеддингов: 0.9205776173285198
```

[45]:
```python
# Модель, выдающая самый частый класс
from sklearn.dummy import DummyClassifier

dummy_clf = DummyClassifier(strategy="most_frequent")
dummy_clf.fit(X_train_emb, y_train_emb)
y_pred_dummy = dummy_clf.predict(X_test_emb)
print("\nОтчет классификации для модели, выдающей самый частый класс:")
print(classification_report(y_test_emb, y_pred_dummy))
print(f"Accuracy для модели, выдающей самый частый класс: {accuracy_score(y_test_emb, y_pred_dummy)}")
```

```
Отчет классификации для модели, выдающей самый частый класс:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       154
           1       0.00      0.00      0.00         9
           2       0.87      1.00      0.93      2397
           3       0.00      0.00      0.00         5
           4       0.00      0.00      0.00         6
           5       0.00      0.00      0.00        11
           7       0.00      0.00      0.00       108
           8       0.00      0.00      0.00        40
           9       0.00      0.00      0.00        20
          10       0.00      0.00      0.00        20

    accuracy                           0.87      2770
   macro avg       0.09      0.10      0.09      2770
weighted avg       0.75      0.87      0.80      2770

Accuracy для модели, выдающей самый частый класс: 0.8653429602888086
```

[46]:
```python
# Модель с One-Hot Encoding
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

categorical_features = ["Fraction_Subject", "Fraction_Object"]
preprocessor = ColumnTransformer(
    transformers=[
        ('onehot', OneHotEncoder(handle_unknown='ignore'), categorical_features)])

pipeline_ohe = Pipeline(steps=[('preprocessor', preprocessor),
                               ('classifier', XGBClassifier(n_estimators=500, max_depth=5, objective="multi:softmax", random_state=42))])

X_ohe = df_merged[categorical_features]
y_ohe = df_merged["Predicate"]
X_train_ohe, X_test_ohe, y_train_ohe, y_test_ohe = train_test_split(X_ohe, y_ohe, test_size=0.2, random_state=42, stratify=y_ohe)

pipeline_ohe.fit(X_train_ohe, y_train_ohe)
y_pred_ohe = pipeline_ohe.predict(X_test_ohe)
print("\nОтчет классификации с использованием One-Hot Encoding:")
print(classification_report(y_test_ohe, y_pred_ohe))
print(f"Accuracy с использованием One-Hot Encoding: {accuracy_score(y_test_ohe, y_pred_ohe)}")
```

```
Отчет классификации с использованием One-Hot Encoding:
              precision    recall  f1-score   support

           0       0.89      0.56      0.69       154
           1       0.00      0.00      0.00         9
           2       0.94      0.98      0.96      2397
           3       0.00      0.00      0.00         5
           4       0.00      0.00      0.00         6
           5       0.31      1.00      0.48        11
           7       0.67      0.92      0.78       108
           8       0.00      0.00      0.00        40
           9       0.00      0.00      0.00        20
          10       0.00      0.00      0.00        20

    accuracy                           0.92      2770

   macro avg       0.28      0.35      0.29      2770
weighted avg       0.89      0.92      0.90      2770

Accuracy с использованием One-Hot Encoding: 0.9205776173285198
```

[ ]: