Создания ИИ-агента для построения карт депонирования углерода лесом с использованием **LangChain** и LLM

Разберем пример создания ИИ-агента для построения карт депонирования углеро да лесом с использованием **LangChain** и LLM.

Архитектура решения

Компоненты системы:

- **LLM** (например, GPT-4, Llama 3) — ядро рассуждений;
- **LangChain** — оркестратор взаимодействия;
- **Геопространственные библиотеки** (GDAL, GeoPandas, Shapely) — работа с картами;
- **Базы данных** (PostGIS, SQLite) — хранение таксационных данных;
- **Визуализация** (Folium, Plotly) — отображение результатов.

Пошаговая реализация

Шаг 1. Установка зависимостей

bash

```
pip install langchain langchain-openai geopandas folium matplotlib scikit-learn
```

Шаг 2. Настройка LangChain-агента

python

```
from langchain.agents import initialize_agent, AgentType
from langchain.llms import OpenAI
from langchain.tools import tool
import geopandas as gpd
import pandas as pd

# Инициализация LLM
llm = OpenAI(temperature=0, model_name="gpt-4")

# Загрузка таксационных данных (пример структуры)
forest_data = gpd.read_file("forest_inventory.geojson")
# Поля: area_ha, tree_species, age, dbh_cm, height_m, biomass_t_ha
```

Шаг 3. Создание специализированных инструментов

python

```
@tool
def calculate_carbon_stock(area_ha: float, biomass_t_ha: float) -> float:
```

```
    """
    Расчёт запаса углерода на участке.
    Коэффициент конверсии биомассы в углерод: 0.5
    """
    carbon_stock = area_ha * biomass_t_ha * 0.5  # т C/га
    return carbon_stock

@tool
def filter_forest_areas(min_age: int, species: str) -> gpd.GeoDataFrame:
    """
    Фильтрация участков по возрасту и породе.
    Возвращает GeoDataFrame с геометрией и атрибутами.
    """
    filtered = forest_data[
        (forest_data['age'] >= min_age) &
        (forest_data['tree_species'] == species)
    ]
    return filtered

@tool
def create_carbon_map(carbon_data: gpd.GeoDataFrame, output_path: str):
    """
    Создание интерактивной карты депонирования углерода.
    Использует Folium для визуализации.
    """
    m = folium.Map(location=[55.75, 37.62], zoom_start=6)

    for idx, row in carbon_data.iterrows():
        folium.CircleMarker(
            location=[row.geometry.centroid.y, row.geometry.centroid.x],
            radius=row['carbon_stock'] / 100,  # масштабирование радиуса
            popup=f"Участок {idx}: {row['carbon_stock']:.1f} т C",
            color="green" if row['carbon_stock'] > 50 else "orange",
            fill=True
        ).add_to(m)

    m.save(output_path)
    return f"Карта сохранена в {output_path}"
```

Шаг 4. Инициализация агента

python

```
tools = [calculate_carbon_stock, filter_forest_areas, create_carbon_map]
```

```
agent = initialize_agent(
    tools,
    llm,
    agent_type=AgentType.STRUCTURED_CHAT_ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True
)
```

Шаг 5. Запуск агента с запросом

python

```
query = """
Построй карту депонирования углерода для еловых лесов старше 80 лет.
Используй таксационные данные из forest_inventory.geojson.
Для каждого участка рассчитай запас углерода (биомасса × 0.5) и визуализируй результат.
Выведи таблицу с суммарным запасом углерода по участкам.
"""

response = agent.run(query)
print(response)
```

Пример работы агента

Входные данные (таксация):

Участок	Площадь, га	Порода	Возраст, лет	Биомасса, т/га
1	5.2	Ель	95	220
2	3.8	Сосна	70	180
3	6.1	Ель	120	280

Действия агента:

1. Фильтрует участки: порода = «Ель», возраст ≥ 80 лет (участки 1 и 3).
2. Рассчитывает запас углерода:
   - Участок 1: $5{,}2\ \text{га} \times 220\ \text{т/га} \times 0{,}5 = 572\ \text{т C}$;
   - Участок 3: $6{,}1\ \text{га} \times 280\ \text{т/га} \times 0{,}5 = 854\ \text{т C}$.

3. Создаёт интерактивную карту с маркерами, размер которых отражает запас углер ода.
4. Формирует отчёт:

   Итоговый запас углерода: $572+854=1426$ т C.

Оптимизация и расширения

Улучшения:

- Добавление инструмента для прогноза динамики углерода (модель роста леса).
- Интеграция спутниковых данных (Sentinel-2) для актуализации биомассы.
- Поддержка многопользовательского доступа через веб-интерфейс (Streamlit/Dash) .
- Кэширование результатов для ускорения повторных запросов.

Типичные проблемы и решения:

- **Ошибка геокодирования:** проверка CRS (система координат) данных.
- **Перегрузка LLM:** ограничение длины контекста, использование Retrieval-Augment ed Generation (RAG).
- **Неточность расчётов:** валидация формул агентом через вызов калькулятора.

Вывод

LangChain позволяет создать гибкого ИИ-агента, который:

- понимает сложные запросы на естественном языке;
- автоматизирует геопространственный анализ;
- интегрирует специализированные инструменты;
- выдаёт результаты в виде карт и таблиц.

Такой агент может использоваться экологами, лесоводами и климатическими анал итиками для оперативного мониторинга углеродного баланса лесов.