

Основы работы с данными для ИИ

Лекция 7 Автоматизация получения данных: парсинг API и скрапинг

Доц
Екатерина Александровна

Преподаватель кафедры информатики и вычислительного эксперимента

Об автоматизации данных

Цели автоматизированного получения данных

- Исключение ручного копирования информации.
- Обеспечение своевременного доступа к актуальным данным.
- Повышение точности и уменьшение числа ошибок.
- Возможность массового сбора и обработки информации.

Примеры применения:

- Финансовые приложения, автоматически загружающие курсы валют.
- Системы сравнения цен, собирающие данные из интернет-магазинов.
- Аналитические сервисы, мониторящие новости и социальные сети.

Об автоматизации данных

Существует 2 подхода к извлечению данных со страниц сайта

Извлекать данные из HTML-кода страницы сайта

Плюсы — этот способ прост и работает всегда, так как код страницы всегда доступен пользователю

Минусы — этот способ может работать долго (несколько секунд), если часть данных генерирует java script (например, данные появляются только после прокручивания страницы или нажатия кнопки)

Использовать API сайта

Плюсы — быстрее первого способа и не зависит от изменений структуры html-страницы

Минус — не у всех сайтов есть открытое API

Метод 1. API (Application Programming Interface)

Определение

API — это формализованный интерфейс, предоставляемый сервисом или приложением, через который программист может запрашивать данные или выполнять действия

Принцип работы

Пользовательское приложение формирует HTTP-запрос (например, GET или POST)

Сервер обрабатывает запрос и возвращает ответ в структурированном виде (обычно JSON или XML)

Метод 1. API (Application Programming Interface)

Пример ответа API (JSON)

```
{  
  "city": "Москва",  
  "temperature": 23,  
  "condition": "ясно"  
}
```

Метод 1. API (Application Programming Interface)

Преимущества

- Структурированность: данные приходят в удобной для обработки форме
- Легальность: использование API официально разрешено владельцем сервиса
- Скорость: минимальные накладные расходы при передаче данных

Недостатки

- Возможны ограничения по количеству запросов (rate limits)
- Доступ к API может быть платным или требовать авторизации
- Иногда API предоставляет лишь часть информации, доступной на сайте

Метод 1. API (Application Programming Interface)

Преимущества

- Структурированность: данные приходят в удобной для обработки форме
- Легальность: использование API официально разрешено владельцем сервиса
- Скорость: минимальные накладные расходы при передаче данных

Недостатки

- Возможны ограничения по количеству запросов (rate limits)
- Доступ к API может быть платным или требовать авторизации
- Иногда API предоставляет лишь часть информации, доступной на сайте

Форматы данных. Основные текстовые форматы

JSON (JavaScript Object Notation) —
вложенные структуры

```
{  
  "name": "Alice",  
  "age": 25,  
  "city": "Berlin"  
}
```

XML (eXtensible Markup Language) —
тегированное представление:

```
<person>  
  <name>Alice</name>  
  <age>25</age>  
</person>
```

CSV (Comma-Separated Values) —
табличный формат:

```
Name, Age, City  
Alice, 25, Berlin  
Bob, 30, Paris
```

Parquet — это файл с таблицей, сохранённой по столбцам (columnar). Думай как про «умный Excel для больших данных»: хранит типы, сжимается и читается очень быстро, особенно когда нужны **пару столбцов** из огромной таблицы. Расширение: .parquet.

Главная идея

- Данные лежат **столбиками**, а не построчно →
 - быстрее читать **нужные колонки**,
 - лучше сжатие (одинаковые значения рядом),
 - можно быстро **фильтровать** по диапазонам (движки смотрят мини/макси в метаданных).

Данные в таблице

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

Текстовый файл CSV

A1	B1	C1	A2	B2	C2	A3	B3	C3
----	----	----	----	----	----	----	----	----

Файл в формате Parquet

A1	A2	A3	B1	B2	B3	C1	C2	C3
----	----	----	----	----	----	----	----	----

HDF5 — для сложных научных данных

HDF5 — это один файл-контейнер, внутри которого можно хранить много разных массивов (картинки, звуковые сигналы, признаки и т.п.) с папками и подписями.

```
data.h5
├ raw/                ← «папка»
│   └ camera1         ← подпапка
│       └ frames       ← массив (dataset)
├ proc/
│   └ features         ← массив (dataset)
└ labels               ← массив (dataset)
```



ПЕРЕВЕДИ В **HDF5**,
ТАК БОЛЬШЕ ВЛЕЗЕТ!

JPEG (Joint Photographic Experts Group)

Тип: Сжатие с потерями (lossy)

Применение: Фотографии, веб-изображения

Плюсы:

- Маленький размер файла
- Поддержка миллионов цветов (24 бит/пиксель)

Минусы:

- Артефакты сжатия при низком качестве
- Нет прозрачности (альфа-канала)



PNG (Portable Network Graphics)

Тип: Сжатие без потерь (lossless)

Применение: Графика с прозрачностью, скриншоты, логотипы

Плюсы:

- Поддержка прозрачности (альфа-канал)
- Чёткие контуры текста и графики

Минусы:

- Большой размер, чем JPEG
- Не подходит для фотореалистичных изображений.



TIFF (Tagged Image File Format)

Тип: Без потерь или с потерями (зависит от настроек)

Применение: Полиграфия, медицинские снимки, архивное хранение

Плюсы:

- Высокое качество
- Поддержка слоёв и метаданных

Минусы:

- Очень большой размер файла
- Ограниченная поддержка в вебе



Инструменты для анализа и обработки данных. Pandas

Назначение:

Библиотека для работы со структурированными данными (аналог Excel в Python).

Ключевые возможности:

- Работа с таблицами (DataFrame) и временными рядами
- Загрузка данных из CSV, Excel, SQL, JSON
- Очистка данных (обработка пропусков, дубликатов)
- Группировка, агрегация, сводные таблицы
- Слияние и соединение таблиц



Основные структуры данных:

- Series - одномерный массив с метками
- DataFrame - двумерная таблица с именованными колонками

Применение в ИИ:

- Предобработка данных перед обучением моделей
- Анализ признаков (feature engineering)

Назначение:

Библиотека для эффективных численных вычислений.

Ключевые особенности:

- Многомерные массивы (ndarray)
- Векторизованные операции (без циклов)
- Линейная алгебра, Фурье-анализ
- Интеграция с C/C++ и Fortran

Преимущества перед списками Python:

- В 10-100 раз быстрее
- Меньший расход памяти
- Удобный синтаксис

Применение в ИИ:

- Основа для других библиотек (TensorFlow, PyTorch)
- Операции с тензорами



Назначение:

Библиотека классического машинного обучения.

Основные модули:

- Предобработка данных (StandardScaler, OneHotEncoder)
- Классификация (SVM, Random Forest), Регрессия (Linear Regression)
- Кластеризация (K-means)
- Метрики качества (accuracy, F1-score)

Преимущества:

- Единый API для всех алгоритмов
- Хорошая документация
- Интеграция с NumPy/pandas

Применение:

- Прототипирование моделей
- Базовые задачи ML



Инструменты для анализа и обработки данных. SQL

Назначение:

Язык для работы с реляционными базами данных.

Основные команды:

- SELECT - выбор данных
- JOIN - соединение таблиц
- GROUP BY - группировка
- WHERE - фильтрация

Интеграция с Python:

- Библиотеки: sqlite3, psycopg2 (PostgreSQL)
- ORM: SQLAlchemy
- В pandas: `pd.read_sql()`

Применение в ИИ:

- Извлечение данных для обучения
- Агрегация признаков



Matplotlib – основа визуализации в Python

Назначение:

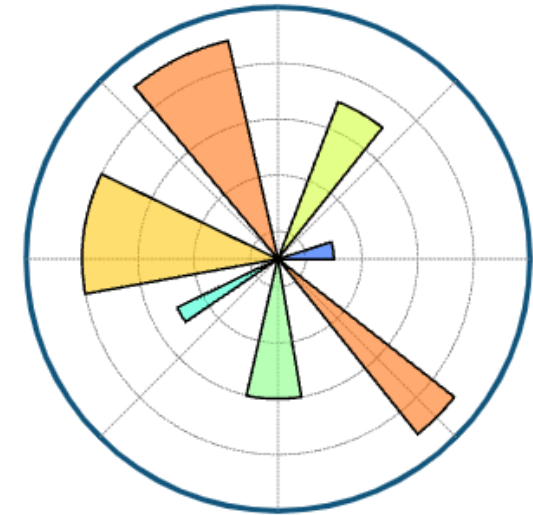
Библиотека для создания статических, анимированных и интерактивных визуализаций.

Ключевые особенности:

- Низкоуровневый контроль над каждым элементом графика
- Кастомизация: цвета, подписи, легенды, сетка
- Возможность сохранения в PDF/SVG (для публикаций)

Поддержка множества типов графиков:

- Линейные (plot)
- Столбчатые (bar)
- Гистограммы (hist)
- Круговые (pie)
- Точечные (scatter)



Назначение:

Высокоуровневая библиотека на основе Matplotlib для статистической визуализации.

Ключевые возможности:

- Встроенные стили и цветовые палитры
- Упрощенный синтаксис для сложных визуализаций
- Интеграция с pandas DataFrames

Специализированные графики:

- heatmap (тепловые карты)
- pairplot (матрица корреляций)
- violinplot (диаграммы скрипки)
- Implot (регрессионные графики)



Инструменты для визуализации данных. Яндекс Даталенс

Назначение:

Облачный сервис для создания интерактивных дашбордов и аналитических отчетов.

Ключевые функции:

- Подключение различных источников данных: CSV, Excel, SQL-базы, API
- Визуальный конструктор графиков
- Интерактивные фильтры и срезы
- Совместная работа в реальном времени

Типы визуализаций:

- Стандартные (столбчатые, линейные)
- Гео-карты
- Кастомные виджеты
- Динамические фильтры



Машинное обучение (каталоги/площадки)

- **Hugging Face Datasets** — большое хранилище открытых датасетов (лицензия на карточке набора)
- **UCI ML Repository** — классические открытые наборы (разные открытые лицензии)
- **OpenML** — открытые наборы и задачи (лицензия указана у набора)
- **Kaggle Datasets** — много public-датасетов (лицензии разные, читать карточку).

Компьютерное зрение (изображения/разметка)

- **COCO** — объекты/сегментация (аннотации под CC BY 4.0).
- **Open Images** — масштабный набор (аннотации CC BY 4 сами изображения — по лицензиям источников, проверяйте).
- **OpenLogo, OpenDataCam** примеры — тематические open-наборы (смотреть лицензию в репозитории)

Источники данных. Лицензии

Лицензия	Коммерция	Нужна подпись автора	Нужно делиться изменениями	Простое правило
CC0 / Public Domain	✓	✗ (желательно указать источник)	✗	Можно брать и использовать где угодно
CC BY	✓	✓	✗	Используй, но обязательно подпиши автора
CC BY-SA	✓	✓	✓ (если публикуешь изменённые данные)	Подпиши автора и публикуй изменения под той же лицензией
ODbL (OSM и др.)	✓	✓	✓ для переделанных баз	Карты/картинки — ок с подписью; новые базы данных — под ODbL
CC BY-NC	✗	✓	✗	Только некоммерческое использование

Откуда брать данные для ИИ?

Есть 3 базовых пути:

1. Скрапинг — берём открытые данные с сайтов
2. Синтетика — генерируем похожие на реальные
3. Разметка — люди ставят метки (классы, границы, теги)

Данные: существуют

Аналитики:



Что это: программно «считываем» открытую страницу и вытаскиваем нужные кусочки (заголовки, цены, тексты).

Зачем: когда API нет, а на сайте данные есть.

Как сделать (3 шага):

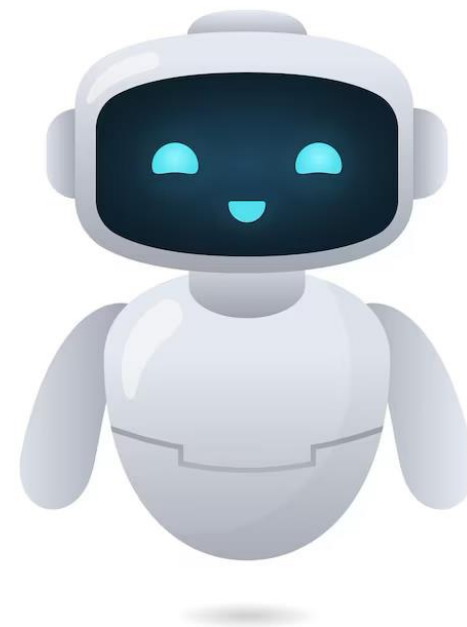
1. Открой страницу и найди нужные элементы (класс/тег).
2. Напиши короткий скрипт → заведи текст/числа.
3. Сохрани в CSV/Excel.

Правила безопасности (запомнить):

1. Обязательно изучить robots.txt и правила сайта (ToS).
2. Не делать больше 1–2 запроса/сек.
3. Не трогать персональные данные

Robots.txt — вежливые правила для ботов

- Это файл по адресу `https://сайт/robots.txt`, где владелец сайта пишет **рекомендации** для роботов (по стандарту Robots Exclusion Protocol).
- В нём указывают **какие пути можно/нельзя** сканировать, иногда желаемую паузу между запросами (Crawl-delay) и ссылку на `sitemap.xml`.
- Это **не закон**, но это принятая норма поведения. Игнорировать — значит вести себя «невежливо» и рисковать, что вас заблокируют.



Пример файла robots.txt

```
User-agent: *  
Disallow: /private/  
Allow: /public/  
Crawl-delay: 2  
Sitemap:  
https://example.com/sitemap.xml
```

Расшифровка:

- всем ботам (*) нельзя ходить в /private/, можно — в /public/
- необходимо делать паузу ~2 сек между запросами
- карта сайта — где быстрее найти все страницы

Важно: кроме robots.txt, есть ещё метки robots в <meta> и заголовок X-Robots-Tag — они тоже говорят ботам «не индексируй». Для учебного скрейпинга это хороший сигнал **не трогать** такие страницы

ToS (Terms of Service) — юридические правила сайта

- Это «условия использования» — часто внизу страницы (Terms, ToS, Пользовательское соглашение).
- В ToS прямо пишут, что **разрешено/запрещено**: автоматический доступ, скрапинг, лимиты запросов, логин/платный контент, коммерческое использование данных и т.д.
- Нарушение ToS = **риск блокировки, претензий**, а в некоторых юрисдикциях — правовые последствия.

Типичные пункты ToS:

- «Запрещён автоматический сбор данных без письменного разрешения.»
- «Макс. 100 запросов за 5 минут.»
- «Контент нельзя перепубликовать/продавать.»
- «Доступ только через официальный API.»

Как действовать правильно (пошагово)

- Проверьте, есть ли API. Если есть — используйте его вместо скрейпинга.
- Посмотрите robots.txt. Разрешены ли нужные разделы? Есть ли Crawl-delay?
- Прочитайте ToS. Нет ли запрета на автоматический доступ/перепубликацию? Есть ли лимиты?
- Идентифицируйте бота. Укажите честный User-Agent и контакт (email/сайт).
- Ставьте паузы. 1–2 запроса/сек — безопасный ориентир, или следуйте Crawl-delay.
- Не трогайте PII. Не собирайте личные данные (ФИО, телефоны и т.п.) без законных оснований.
- Кэшируйте и не дублируйте. Сохраняйте ответы, не запрашивайте одно и то же постоянно.
- Соблюдайте лицензии. Даже если «собирать можно», использовать контент (особенно коммерчески) может быть нельзя.

Генерация данных. Синтетические данные (Faker, GPT)

Что это: искусственно созданные примеры, похожие на реальные (имена, адреса, тексты, изображения).

Зачем: когда реальных данных мало/нельзя поделиться; выравниваем классы, тренируемся без приватных данных.

Как сделать:

1. Решить, какие поля нужны (схема).
2. Сгенерировать N строк/примеров.
3. Отметить `synthetic=true`, проверить, что данные похожи на реальность

```
from faker import Faker
fake = Faker("ru_RU")
row = {"name": fake.name(),
      "city": fake.city(), "age":
      fake.random_int(18,70)}
print(row)
```

Генерация данных. Разметка

Что это: люди добавляют метки к данным:

- для картинок: классы/рамки;
- для текста: тональность/темы;
- для аудио: расшифровка речи.

Зачем: без меток модель «не понимает», что предсказывать.

Как сделать:

- Составить чёткие правила (что считать «кошкой», что «не кошкой»)
- Сделать «пилот» на 50 примерах — поправить правила
- Разметить основной объём

Инструменты: Label Studio (универсально), CVAT (картинки/видео).

Форматы (часто): COCO/YOLO (CV), CSV/JSON (тексты).

Этапы работы с данными в ИИ

1) Сбор(*API, датчики, ручной ввод*)

- выбираем источник и формат (JSON/CSV/изображения/аудио)
- фиксируем схему и частоту/периодичность
- сохраняем «как есть» в raw/ + логируем дату/версию

2) Очистка(*шум, пропуски*)

- приводим типы, убираем дубликаты и аномалии
- заполняем пропуски/фильтруем мусор
- кладём результат в clean/

3) Преобразование(*нормализация, кодирование*)

- скейлим числовые признаки, кодируем категориальные (one-hot/label)
- делим на train/val/test без утечки
- для изображений: resize, аугментации

4) Анализ(*статистика, визуализация*)

- проверяем распределения и баланс классов
- ищем корреляции/связи
- фиксируем инсайты и решения в отчёте

5) Моделирование(*обучение ML*)

- базовая модель → метрика → улучшения (регуляризация, фичи, тюнинг)
- контроль переобучения: валид. метрика, early stopping
- сохраняем веса/версии/окружение