ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

Лекция 14

Форматы JSON, XML и Excel

Оспан Ә.Ғ., PhD, доцент-исследователь

КазНУ, кафедра ИИ и Big Data

2026-2027 учебный год

Цель и содержание лекции

Цель и результаты обучения

- Выбрать структуру хранения по смыслу данных
- Преобразовать вложенные записи в проверяемые таблицы
- Подтвердить сохранность после записи и повторного чтения

Содержание лекции

1. **Формат и схема**
2. **JSON**
3. **Нормализация списков**
4. **XML-дерево**
5. **Excel и проверка обмена**

Формат выбирают по структуре и назначению

Формат	Структура	Преимущество	Проверка
CSV	Одна плоская таблица	Простой обмен	Разделитель, кодировка, типы
JSON	Объекты и массивы	Вложенные записи	Ключи, типы, списки
XML	Дерево элементов	Явная разметка	Теги, атрибуты, пространства имен
Excel	Книга, листы, ячейки	Просмотр и оформление	Листы, заголовки, формулы

Выбор формата следует из смысла записи и способа использования.

Схема и проверка важнее успешного чтения

1

Разбор файла

Проверить синтаксис выбранным парсером.

2

Схема и типы

Проверить ключи, ограничения и явные преобразования.

3

Нормализация и расчет

Выбрать единицу строки и выполнить анализ.

4

Запись и обратное чтение

Сравнить ключи, число записей и значения.

- Корректный синтаксис не гарантирует корректные предметные данные.
- Сравнение смысла полезнее буквального совпадения файлов.

JSON и Python: соответствие основных типов

JSON	Python после чтения	Что учитывать
object	dict	Ключи JSON всегда строки
array	list	Порядок элементов сохраняется
string / number	str / int или float	Проверить ожидаемый тип
true / false / null	True / False / None	Пустое и неизвестное имеют разный смысл

load/dump работают с файлом; loads/dumps - со строкой.

Полный локальный JSON вместо отсутствующего файла

Пример Python

```
from pathlib import Path
import json
import pandas as pd

out = Path('work_results')
out.mkdir(exist_ok=True)
recipes = [
    dict(id=101, cuisine='local', ingredients=['rice', 'onion']),
    dict(id=102, cuisine='med', ingredients=['cheese', 'onion']),
    dict(id=103, cuisine='local', ingredients=['rice']),
]
with (out/'recipes_demo.json').open('w', encoding='utf-8') as f:
    json.dump(recipes, f, ensure_ascii=False, allow_nan=False)
```

Контракт записи

- id - уникальное целое; cuisine - строка.
- ingredients - список текстовых значений.

Воспроизводимость

- Данные полностью заданы в коде.
- Файл пишется в созданный work_results.

Вложенный список требует выбора единицы анализа

Длинная форма

- Одна строка - пара «рецепт - ингредиент».
- Удобна для частот и группировок связей.
- Повторные пары могут искажать число рецептов.

Широкая форма

- Одна строка - рецепт; столбцы ing_* содержат 0/1.
- Сумма индикаторов дает число различных ингредиентов.
- Префикс отделяет ингредиенты от id и cuisine.

Определите, что считаете: упоминания, связи или рецепты.

Нормализуем список и считаем число рецептов

Пример Python

```
with (out/'recipes_demo.json').open(encoding='utf-8') as f:
    loaded = json.load(f)
table = pd.DataFrame(loaded)
long = table.explode('ingredients', ignore_index=True)
pairs = long[['id', 'ingredients']].drop_duplicates()
counts = pairs['ingredients'].value_counts().sort_index()
wide = pd.crosstab(pairs['id'], pairs['ingredients'])
wide = wide.gt(0).astype('int64').add_prefix('ing_')
print(counts.to_dict())
```

Результат: {'cheese': 1, 'onion': 2, 'rice': 2}

Длинная форма

- explode создает отдельную строку на элемент.
- drop_duplicates задает подсчет разных рецептов.

Широкая форма

- crosstab строит связь id и ингредиента.
- gt(0) преобразует частоты в индикаторы.

Три представления одной учебной записи

Представление	Одна строка означает	Рецепт 101
JSON	Объект с вложенным списком	ingredients=['rice','onion']
Длинная таблица	Связь id и ингредиента	101-rice; 101-onion
Широкая таблица	Рецепт с индикаторами	ing_rice=1; ing_onion=1; ing_cheese=0
Сохранение порядка	Требует отдельной позиции	В индикаторах порядок не хранится

Преобразование без потерь возможно только при явно заданном смысле.

Обратное преобразование проверяет сохранность

1

Выбрать ing_*

Метаданные id и cuisine не являются ингредиентами.

2

Восстановить списки

Включить имена столбцов с индикатором 1.

3

Записать и прочитать

Использовать исходную схему JSON.

4

Сравнить смысл

Проверить id, cuisine и наборы ингредиентов.

- Отступы и порядок ключей не требуют текстового равенства.
- При значимом порядке ингредиентов храните позицию элемента.

XML: корень, дочерние элементы и атрибуты

Узел ElementTree

- tag - имя; text - текстовое значение.
- attrib хранит словарь атрибутов узла.
- Дочерние элементы образуют дерево с одним корнем.

Извлечение по схеме

- findtext ищет дочерний элемент по имени.
- int/float явно преобразуют текст в число.
- Вложенность и пространства имен нельзя игнорировать.

Не каждый XML имеет естественное представление одной плоской таблицей.

Создаем доверенное локальное XML-меню

Пример Python

```
import xml.etree.ElementTree as ET
root = ET.Element('menu')
for name, price, weight in [('Салат', 450, 180),
                            ('Рис', 600, 250)]:
    dish = ET.SubElement(root, 'dish', name=name)
    ET.SubElement(dish, 'price').text = str(price)
    ET.SubElement(dish, 'weight').text = str(weight)
ET.ElementTree(root).write(out/'menu_demo.xml', encoding='utf-8')
parsed = ET.parse(out/'menu_demo.xml').getroot()
menu = pd.DataFrame([
    dict(name=e.get('name'), price=int(e.findtext('price')),
         weight=int(e.findtext('weight'))) for e in parsed])
print(menu)
```

Создание

- Element задает корень; SubElement - дочерние узлы.
- Атрибут name отличается от текстовых price и weight.

Чтение

- ET.parse возвращает дерево локального файла.
- Числовые поля приводятся к int по схеме.

Excel: два листа и обязательное обратное чтение

Пример Python

```
book = out / 'format_demo.xlsx'
with pd.ExcelWriter(book, engine='openpyxl') as writer:
    table[['id', 'cuisine']].to_excel(
        writer, sheet_name='recipes', index=False)
    menu.to_excel(writer, sheet_name='menu', index=False)
    sheet = writer.sheets['menu']
    sheet.freeze_panes = 'A2'
    sheet.auto_filter.ref = sheet.dimensions
read_menu = pd.read_excel(
    book, sheet_name='menu', engine='openpyxl')
pd.testing.assert_frame_equal(menu, read_menu)
print('Данные меню сохранены')
```

Результат: Данные меню сохранены

Запись

- Контекст завершает сохранение книги.
- Каждый лист получает явное имя; index=False.

Проверка

- read_excel читает выбранный лист.
- Сравниваем значения и типы после обмена.

Инварианты показывают, что именно сохранилось

Объект	Инвариант	Способ проверки
Рецепты	Уникальные id и те же связи	id -> cuisine и набор ингредиентов
Длинная/широкая формы	Число различных ингредиентов	Сумма индикаторов по рецепту
XML-меню	Число блюд, типы, цены и масса	Повторное чтение и явные преобразования
Excel	Листы и табличные значения	read_excel и сравнение DataFrame

Успешная запись файла - начало проверки, а не ее завершение.

Ошибки при обмене данными

Разбор

- Не использовать eval для JSON.
- Не считать успешный parse проверкой схемы.

Преобразование

- Не терять ключи, атрибуты и значимый порядок.
- Не смешивать число рецептов и упоминаний.

Сохранность

- Не принимать оформление за единицы измерения.
- Не пропускать обратное чтение и инварианты.

Главные результаты лекции

1. Формат выбирают по структуре записи и назначению обмена.
2. JSON требует проверки ожидаемой схемы после десериализации.
3. Длинная и широкая формы имеют разные единицы строки.
4. XML-атрибуты, текст, узлы и порядок требуют явного соглашения.
5. Excel и другие выходы проверяют повторным чтением и инвариантами.

Проверка понимания и самостоятельная работа

1. Когда json.load вернет скаляр вместо списка?
2. Чем отличаются строки длинной и широкой таблиц ингредиентов?
3. Почему не каждый XML без потерь становится одной таблицей?
4. Что возвращает read_excel при sheet_name=None?
5. Какие свойства данных важнее совпадения текстов JSON?

Самостоятельная работа

Добавьте повторный onion и пустой список в отдельный эксперимент. Задайте политику обработки и сравните число упоминаний с числом рецептов.

Документация и исходные материалы

Документация

1. Python: JSON

https://docs.python.org/3/library/json.html

2. Python: ElementTree

https://docs.python.org/3/library/xml.etree.elementtree.html

3. pandas: ExcelWriter

https://pandas.pydata.org/docs/reference/api/pandas.ExcelWriter.html

Основа лекции

- Лекция 14 JSON XML Excel.pptx
- Лекция 14.pptx
- Лабораторная работа 14 .docx
- Лабораторная работа 14 с ответами.docx