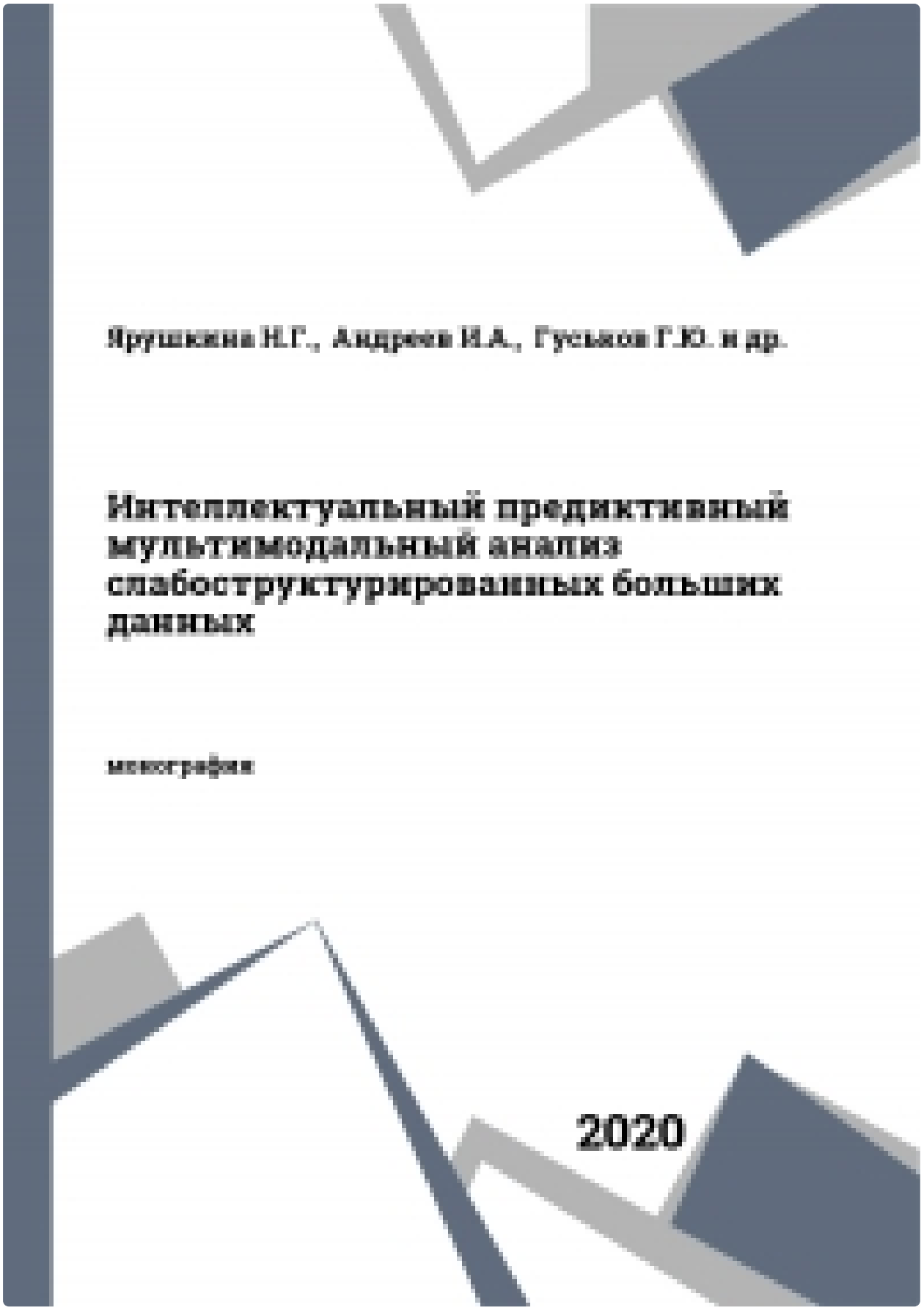
2025-01-27

Методы и алгоритмы обработки слабоструктурированных данных



Интеллектуальный предиктивный мультимодальный анализ слабоструктурированных больших данных / Н. Г. Ярушкина, И. А. Андреев, Г. Ю. Гуськов [и др.]. — Ульяновск : Ульяновский

государственный технический университет, 2020. – 221 с. – ISBN 978-5-9795-2088-9. – Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/106136.html (дата обращения: 07.10.2023).

Воронина, В. В. Разработка веб-сервисов для анализа слабоструктурированных информационных ресурсов : учебное пособие / В. В. Воронина. – Ульяновск : УлГТУ, 2016. – 165 с. – ISBN 978-5-9795-1564-9. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/165020 (дата обращения: 30.01.2024).

Понятие слабоструктурированных данных

Какие данные вообще бывают?

1. Структурированные данные (табличные данные, чаще всего используются в реляционных СУБД и т.п.);
2. Слабоструктурированные данные (или полуструктурированные — semi-structured) — языки разметки + любые данные, к которым можно применить некоторую модель;
3. Неструктурированные данные (unstructured) — произвольные тексты, изображения, видео, аудио, данные без четко заранее заданной структуры и без требований к какому-либо определенному расположению сущностных элементов.

Пример неструктурированных данных — эссе в свободной форме с набором дат, чисел, какой-либо фактологической информации, перечисленные элементы расположены в документе (тексте) в произвольных местах, их вид относительно друг друга может отличаться (24 марта, 28.04, 31-е мая 25 г.) и никак заранее не оговаривается.

Если такое эссе обязательно должно быть разбито на смысловые блоки (тезис, антитезис, синтез), определен обязательный формат упоминания дат в тексте (например, только 28.04.2025, 31.05.2025), чисел, фактов, изображений, оговорить правила использования указанных элементов в каждом из смысловых блоков, то такое эссе можно рассматривать уже как **слабоструктурированные данные**.

Если рассмотреть электронную коммуникацию и в целом интернет, слабоструктурированные данные играют в них ведущую роль.

В консервативном подходе слабоструктурированные данные предполагают наличие эксплицитной (явной) структуры, явной модели — маркеров, тэгов, служебных символов и иных средств явного определения семантически значимых компонентов и задания иерархии/структуры для документа в целом и для отдельных записей.

> **Note**
>
> Это отличает слабоструктурированные данные от классического реляционного подхода, где у данных есть внешняя схема.
> Такой подход называют как «бессхемным» (Schemaless), так и самоописываемым — имена тэгов, маркеры, иные служебные элементы напрямую всегда хранятся вместе с самими записями.

Пример такого подхода — JSON, JavaScript Object Notation

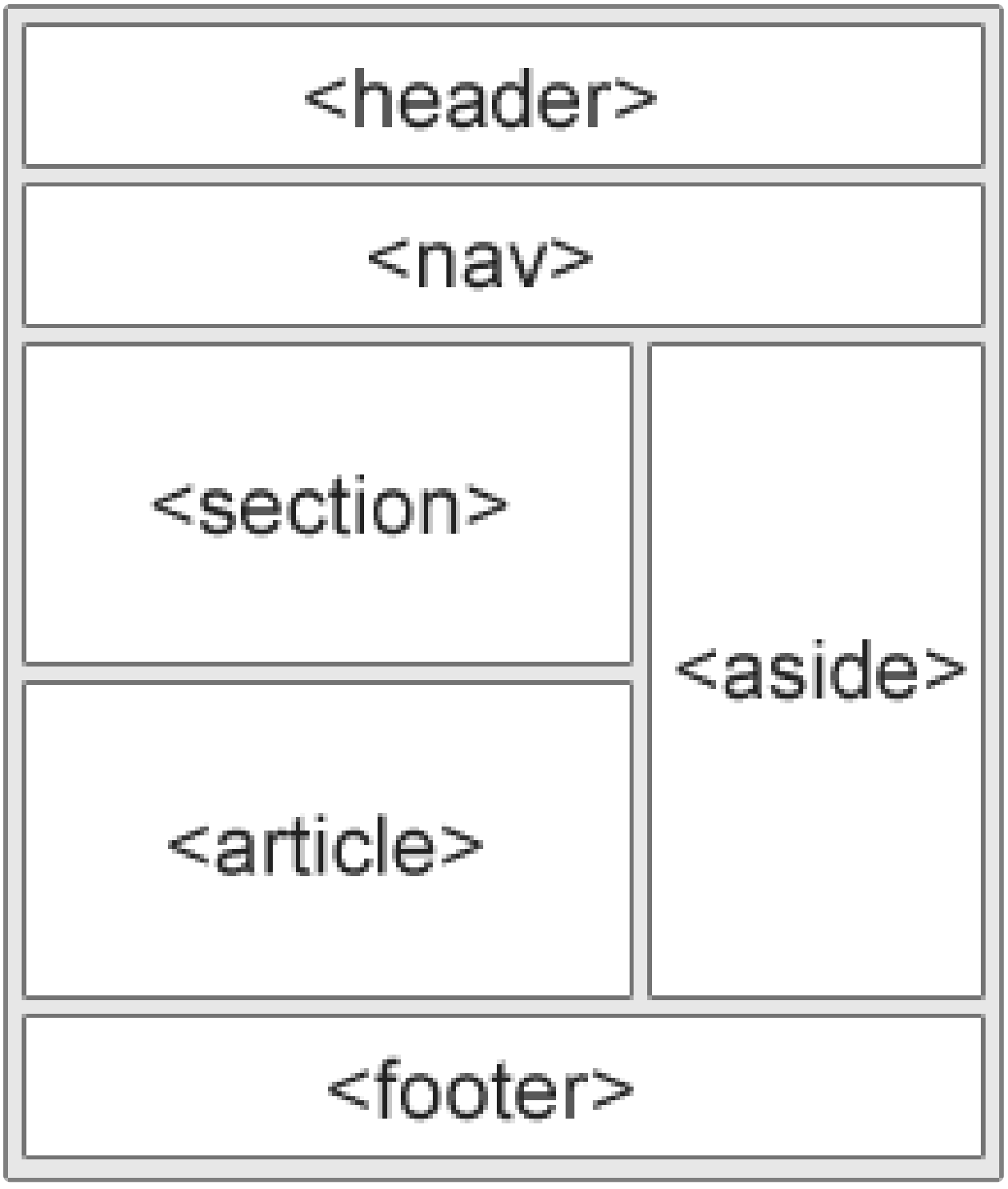
```
{
    "brand": "BMW",
    "age": 5.0,
    "mileage": 53000.0,
```

```
    "owners": ["H. Dubua", "T. Moor", "L. Salander"]
}
```

Другие примеры — SOAP, XML, HTML, YAML.

HTML elements reference - HTML: HyperText Markup Language | MDN

HTML Semantic Elements



Сюда же можно добавить методологии структуризации в рамках программных интерфейсов — к примеру, REST.

В частности, RESTful Web API также предполагает наличие определенных семантических блоков:

```
@app.post('/project/new')
async def create_project() -> Response:
    pass

@app.get('/project/{project_id}')
async def get_project_by_id(project_id:int) -> Project:
    pass

@app.put('/project/{project_id}')
async def update_project_by_id(project_id:int) -> Response:
    pass

@app.delete('/project/{project_id}')
async def delete_project_by_id(project_id:int) -> Response:
    pass
```

Плюсы и минусы слабоструктурированных данных

Плюсы

- гибкость схемы/модели данных
- переносимость — легкая интеграция систем и баз данных с разными моделями данных за счет использования простого самоописываемого подхода (пример — JSON как де-факто стандарт обмена данными в вебе)
- удобство навигации по данным для человека
- независимость от изменений схем данных иных сервисов, подсистем, связанных систем (файл с выгрузкой данных останется

читаемым даже при смене модели данных в будущем)

- масштабируемость — легко распределить обработку таких данных по кластеру (в том числе посредством готовых решений — Hadoop + Spark, например)
- возможность более глубокого анализа на основе контекстуальной информации — за счет того, что метаданные и семантические тэги расположены непосредственно вместе со самим данными, можно выявить большее число закономерностей через сопоставление данных и их контекста (например, семантика или тональность текста сообщения + дата и время отправки)

Минусы

- избыточность
- усложнение машинной обработки, особенно в случае с «человекоориентированными» подходами к модели данных (пример — YAML)
- комплексность — в отличие от структурированных данных можно наткнуться на целый зоопарк форматов, тэгов, метаданных и т.д.
- отсутствие стандартизации — каждый может придумать свою модель данных, к которой нужно будет придумывать свой подход к обработке
- сниженная производительность — в том числе неэффективные запросы
- сниженная безопасность — если не принимать дополнительных мер по обеспечению безопасности слабоструктурированных данных, их можно прочитать и/или перенести/скопировать без специальных средств

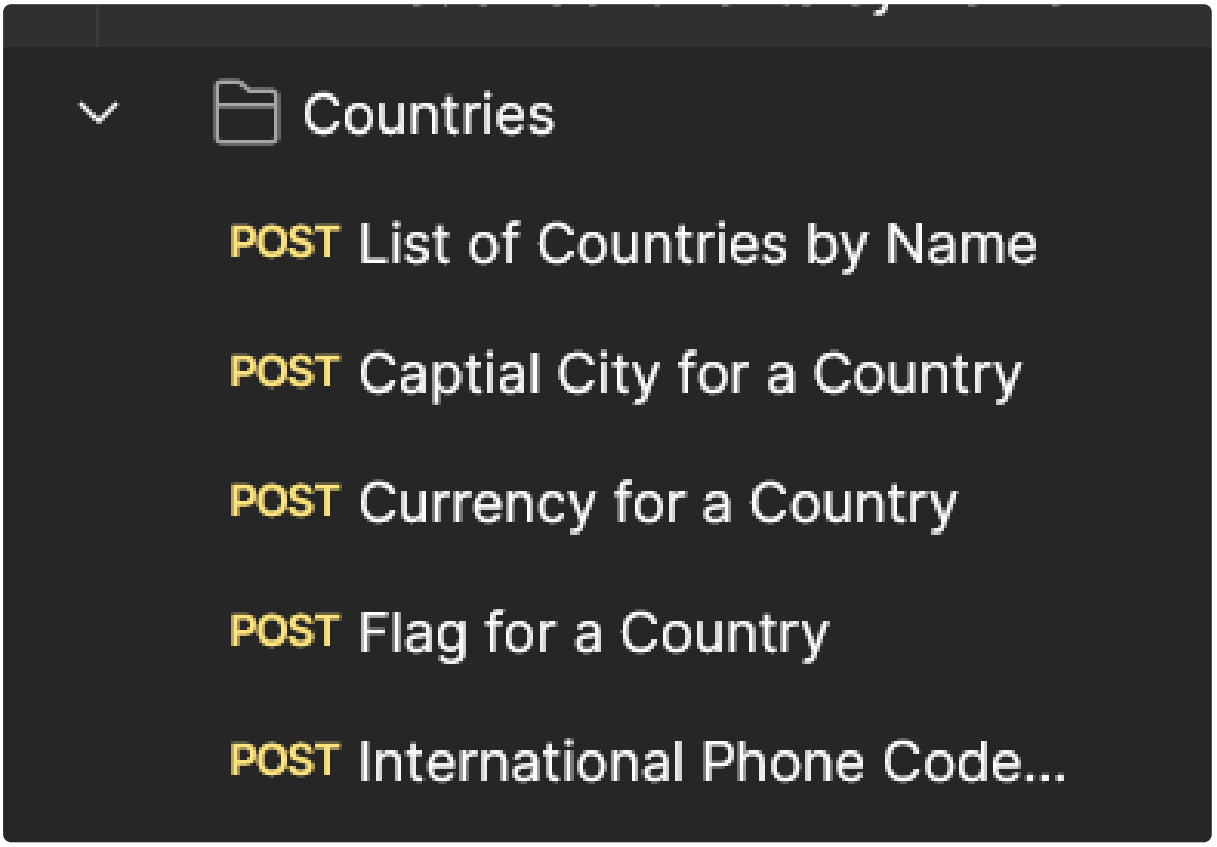
Лабораторная работа №1: SOAP-сервисы

Список: https://www.postman.com/cs-demo/postman-customer-org-s-public-workspace/documentation/eebj1yq/public-soap-apis

Требуется реализовать программу, которая будет принимать запрос пользователя в относительно свободной форме (например, «столица России»; допустимые категории запросов нужно вывести в начале работы с программой), формировать корректный запрос и затем получать данные при помощи методов любого SOAP-сервиса в данном списке. На выходе необходимо представить понятный пользователю ответ на его запрос.

Обратите внимание, что для каждого сервиса есть несколько POST-методов.

Программа должна полностью утилизировать весь функционал выбранного SOAP-сервиса.

> Например, в блоке Continents (https://www.postman.com/cs-demo/postman-customer-org-s-public-workspace/request/34m7jic/list-of-continents-by-name) есть всего один метод – CountryInfoService.wso, но во многих других блоках методов больше.
> Необходимо, чтобы все методы были охвачены программой:
>
> 
>