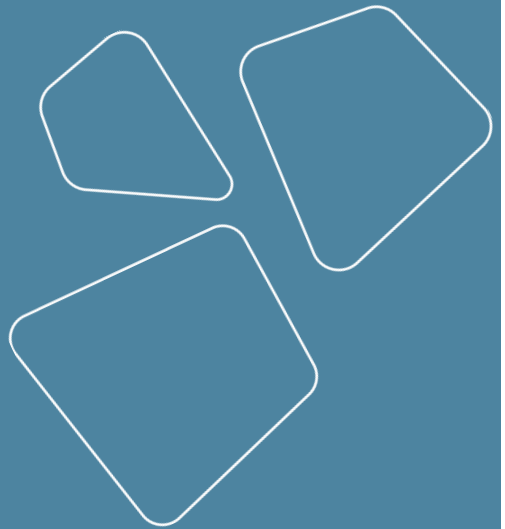
Данные: хранение и управление

Vladislav Goncharenko

MIPT, MSU, fall 2023

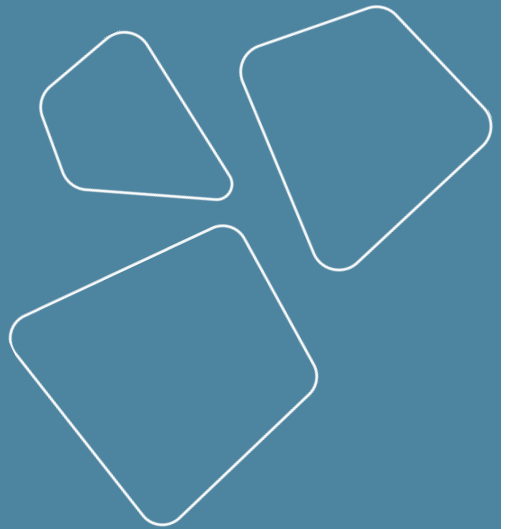
Recap



- Хранение кода
- Git servers
- Разработка в больших командах
  - Дистрибуция кода
- CI/CD
- Форматирование кода
- Inline документация

- pre-commit
- инструметы контроля качества

Outline



- Источники данных
- Хранение данных
  - локально
  - удалённо
  - распределённо
- Data engineering
  - ETL and ELT
  - datahub
- Experiments parametrization
  - Hydra

Источники данных

girafe
ai

01

Источники данных

- Первичные (Primary)
  - Ввод с клавиатуры
  - Web проекты
    - Типичная архитектура: kafka - hadoop - airflow - ml
    - курсы по web и big data
  - Фото и видео камеры
    - Беспилотники
- Вторичные (Secondary)
  - sql запросы к базе данных
  - Web scrapping
  - LLM
  - etc...

Характеристики источников

Factor	Primary Data	Secondary Data
Definition	Primary Data refers to the first-hand data collected by the team. It is collected based on the researcher's needs.	Secondary Data has been collected by other teams in the past. It does not necessarily need to be aligned with the researcher's requirements.
Data	Real-time Data	Historical Data
Process	Time Consuming	Quick and Easy
Cost	Expensive	Economical
Collection Time	Long	Short
Available In	Raw and Crude form	Refined form
Accuracy and Reliability	Very high	Relatively less
Examples	Personal Interviews, Surveys, Observations, etc.	Websites, Articles, Research Papers, Historical Data, etc.

Классы данных

Structured Data	Unstructured Data
Structured Data is organized and has a fixed and defined schema.	Unstructured Data does not have any pre-defined structure to it.
Structured data is **quantitative** data that consists of **numbers and values**.	Unstructured data is **qualitative** data that consists of **text, images, audio, video,** etc.
Structured Data is stored in **Relational Databases** and **Data warehouses**.	Unstructured Data generally is stored in **Data Lakes.**
Structured Data is stored in tabular formats such as **SQL databases.**	It is typically stored in **NoSQL databases**.
It is collected from sensors, network logs, web server logs, OLTP systems, etc.	It is sourced from email messages, word-processing documents, pdf files, etc.
It requires **less storage** and is **highly scalable.**	It needs **more storage** space and is **difficult to scale**.

Типы данных

- Табличные
- Структурированные
  - Текст
  - Изображение
  - Граф
  - ....

Хранение данных

girafe
ai

02

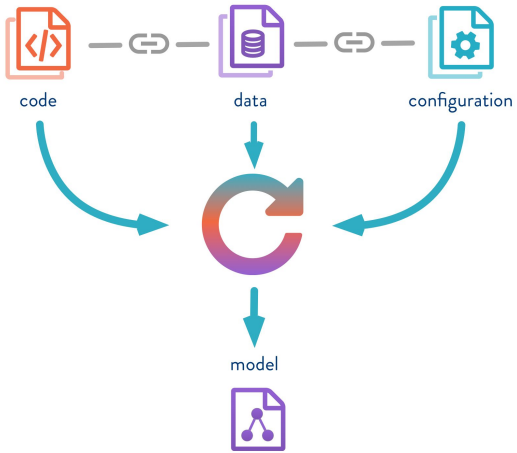
Хранение данных

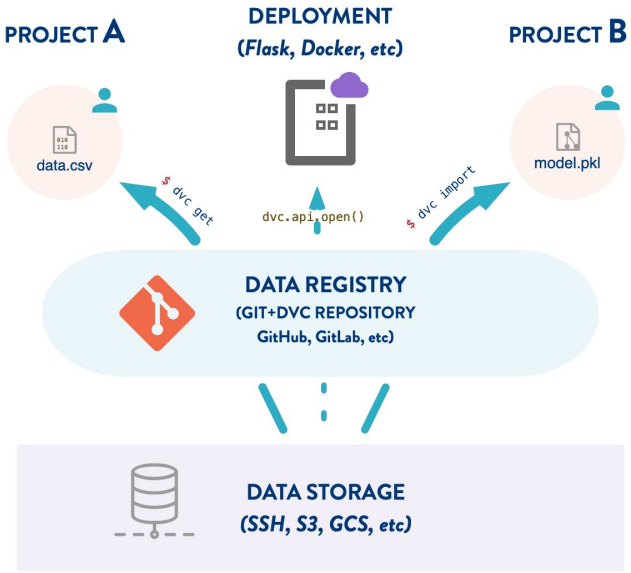
Актуально как для датасетов, так и для готовых моделей

- Локально
- Удалённо
- Распределённо
  - небольшие данные
    - DVC
    - git LFS
  - большие данные
    - парадигма MapReduce
    - стек hadoop или система YTsaurus

Data Version Control (DVC)

- git for data is DVC (tutorials: one, two)
- Versioning and Access submodules





Data Engineering

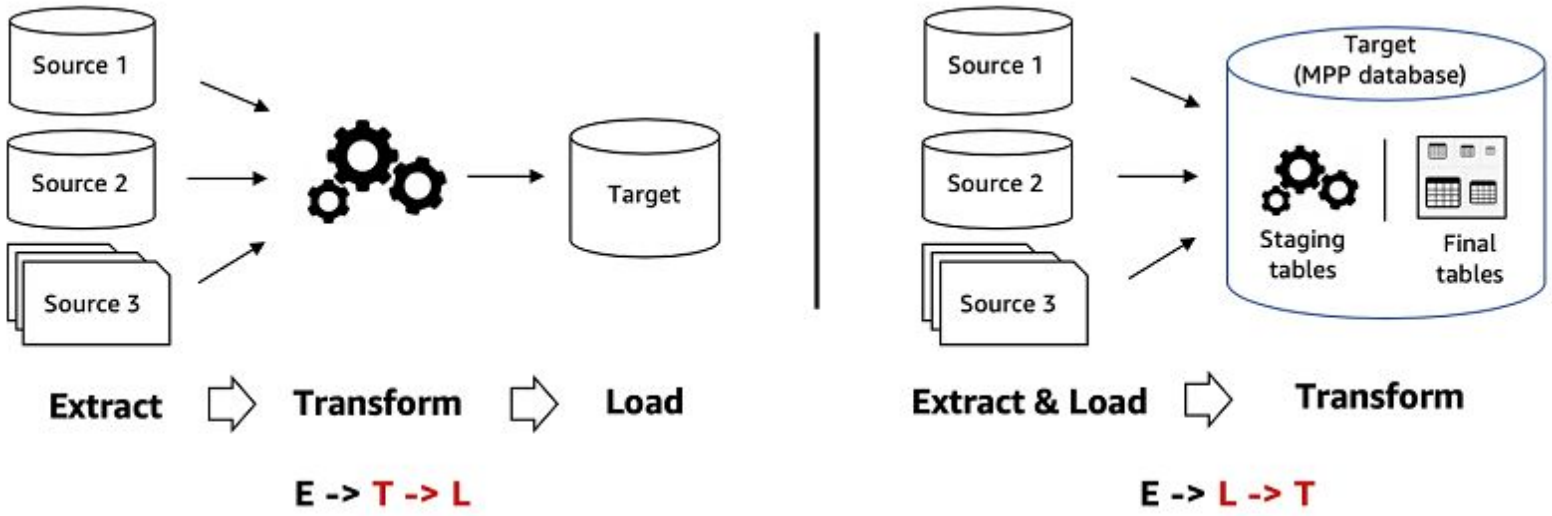
girafe ai

03

ETL and ELT

ETL: extract, transform, load

ELT: extract, load, transform



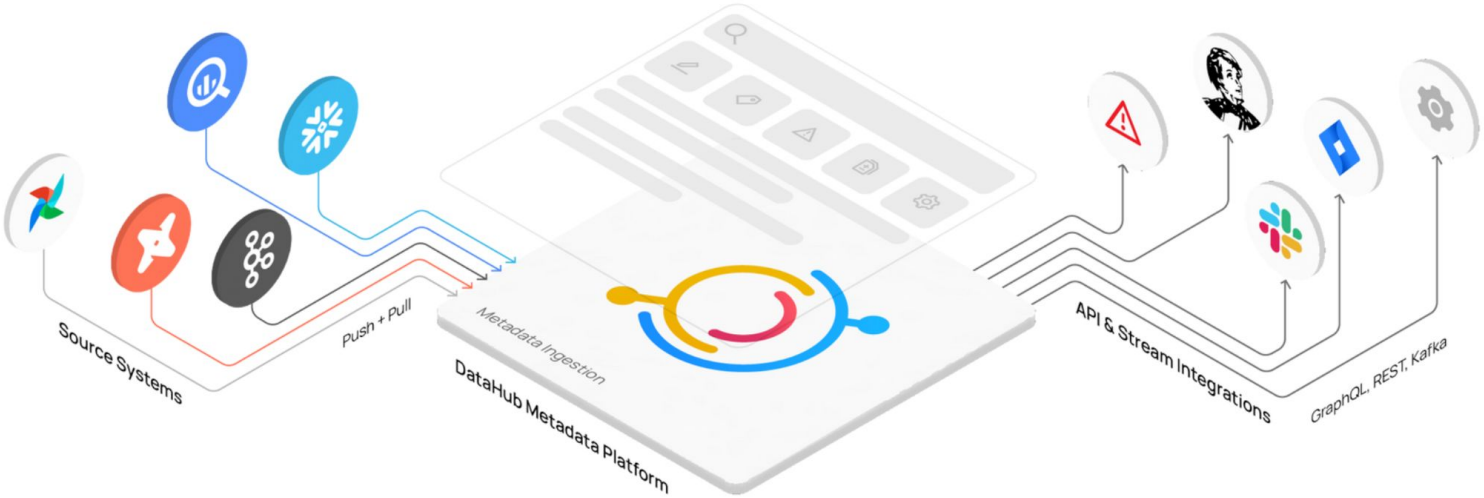
ETL vs ELT

ELT	ETL
ELT tools do not require additional hardware	ETL tools require specific hardware with their own engines to perform transformations
Mostly Hadoop or NoSQL database to store data.Rarely RDBMS is used	RDBMS is used exclusively to store data
As all components are in one system, loading is done only once	As ETL uses staging area, extra time is required to load the data
Time to transform data is independent of the size of data	The system has to wait for large sizes of data. As the size of data increases, transformation time also increases
It is cost effective and available to all business using SaaS solution	Not cost effective for small and medium business
The data transformed is used by data scientists and advanced analysts	The data transformed is used by users reading report and SQL coders
Creates ad hoc views.Low cost for building and maintaining	Views are created based on multiple scripts.Deleting view means deleting data
Best for unstructured and non-relational data. Ideal for data lakes. Suited for very large amounts of data	Best for relational and structured data. Better for small to medium amounts of data

Data Hub

Единое место для хранения метаинформации обо всех данных

https://datahubproject.io/



Experiments parametrization

girafe ai

04

Написал и запустил модель - а что дальше?

- Прийти к лучшей модели за один запуск - нереально
- Требуется большое количество экспериментов
- Как не запутаться?
- А если вас несколько человек?
- Ещё же и в продакшен катить...

Варианты решений

- Копировать папку с проектом - уже проходили
- Заводить ветки в гите - неудобно
- Использовать конфиги!

Библиотека для работы с конфигурациями

https://github.com/facebookresearch/hydra

Пример приложения

- Конфиг задаётся через yaml или dataclass
- Функции оборачиваются в hydra.main (и другие)
- При запуске можно указать нужный конфиг
- Подробнее на семинаре

my_app_type_error.py

```python
from dataclasses import dataclass

import hydra
from hydra.core.config_store import ConfigStore

@dataclass
class MySQLConfig:
    host: str = "localhost"
    port: int = 3306

cs = ConfigStore.instance()
# Registering the Config class with the name 'config'.
cs.store(name="config", node=MySQLConfig)

@hydra.main(version_base=None, config_name="config")
def my_app(cfg: MySQLConfig) -> None:
    # pork should be port!
    if cfg.pork == 80:
        print("Is this a webserver?!")

if __name__ == "__main__":
    my_app()
```

YAML для хранения значений

Поддерживаются:

- дефолтные значения
- переменные
- иерархические конфиги

```yaml
defaults:
  - files: mnist
  - _self_
paths:
  log: ./runs
  data: ${hydra:runtime.cwd}/../data/raw
params:
  epoch_count: 20
  lr: 5e-5
  batch_size: 128
```

Иерархические конфиги

```python
@dataclass
class MySQLConfig:
    host: str = "localhost"
    port: int = 3306

@dataclass
class UserInterface:
    title: str = "My app"
    width: int = 1024
    height: int = 768

@dataclass
class MyConfig:
    db: MySQLConfig = field(default_factory=MySQLConfig)
    ui: UserInterface = field(default_factory=UserInterface)

cs = ConfigStore.instance()
cs.store(name="config", node=MyConfig)

@hydra.main(version_base=None, config_name="config")
def my_app(cfg: MyConfig) -> None:
    print(f"Title={cfg.ui.title}, size={cfg.ui.width}x{cfg.ui.height} pixels")
```

Интеграция с популярными библиотеками

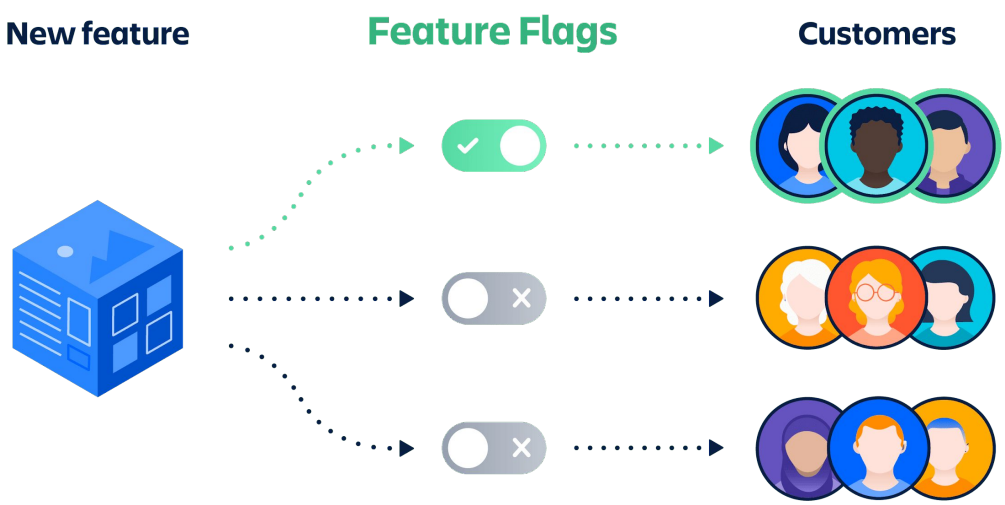
- hydra-zen: ванильная обёртка над Гидрой
- lightning-hydra-template: шаблон для популярной train loop библиотеки
- hydra-torch: самый популярный фреймворк для deep learning

Дальше - больше

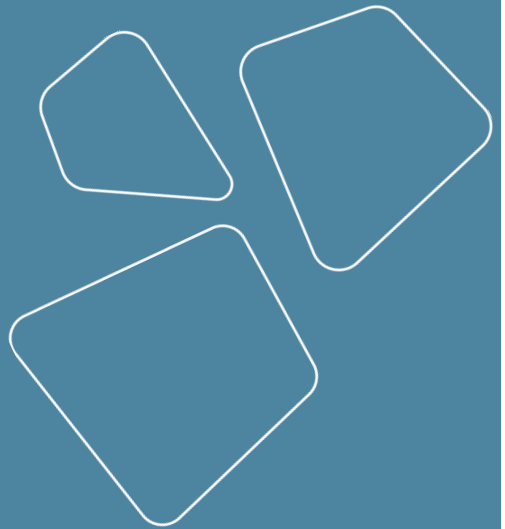
Имея конфиги мы можем упаковать приложение в Docker контейнер

Развернуть докер с нужным конфигом и раздавать эту функциональность части пользователей

Может быть частью системы АБ тестов



Revise



- Источники данных
- Хранение данных
  - локально
  - удалённо
  - распределённо
- Data engineering
  - ETL and ELT
  - datahub
- Experiments parametrization
  - Hydra

Спасибо за внимание!

Вопросы?