Основы работы с данными в ML

План лекции

Зачем хранить и версионировать данные

Хранить данные - возможность повторно воспользоваться данными

Пример: Интернет-магазин, задача рекомендаций.
Нужно рекомендовать только те товары которые есть в наличие. Система управления магазином не хранит историю когда доступности товара.
Для обучения на ретроспективных данных и честной оценки метрик нам нужно позаботиться о сохранении этой истории

Версионирование данных - одна из составляющих для возможность воспроизвести эксперимент.Версионировать - означает возможность хранить несколько версий одного и того же объекта данных и переключаться между версиями.

- Удаление - это специальный маркер, а не фактическое удаление
- Изменение объекта - по сути новый объект

Зачем хранить и версионировать данные

Другие примеры изменения данных:

- CV система собирающая картинки и дообучающаяся со временем
- Конкурс “шляпа” - каждую неделю обновление корпуса слов
- Продакшен система обработки CV - дообучение раз в две недели. Необходимо мониторить качество при дообучении

Версионирование в файловой системе

- Почему git не помошник в версионировании данных
- При решении задачи версионирования в файловой системе хорошо принять решение о разделении данных на чанки
- Некоторые файловые системы имеют встроенные системы версионирования
    - Linux - NIFLS
    - MacOS - Time Machine
- Для универсальности - проект DVC
- Или Quilt

Версионирование данных в облаках

Некоторые облачные системы поддерживают версионирование данных:

- AWS S3 buckets - 5GB на 12 месяцев (есть ограничения по запросам)
- AZURE workspace - 200$ на 30 дней.
- GCS - есть грант на использование 300$ для новых аккаунтов

Версионирование данных в базах данных

- SQL
  - Данные меняются - в каждой таблице есть поле версии и признак того что объект удален. Изменение - новый объект. Удаление объекта - метка
  - Использование триггеров - перегружает и замедляет базу (ок, если это не продакшен)
  - Postgres Point in time recovery
  - Структура таблиц - SQLAlchemy - только инкрементальные изменения
  - MariaDB - версионирование в коробке (не знаю кто этим пользовался)

Версионирование данных в базах данных (продолжение)

- NoSQL
    - Нет проблем со структурой
    - Подход тот же что и в файловой системе но все версии одного объекта данных можно хранить внутри самого объекта. Пример MongoDB
- Datalakes - вещь в себе. Обычно есть версионирование. Дружите с архитектором вашего Datalake :)

Подробнее про dvc

DVC - надстройка над git

1. Установка и инициализация

```
pip install dvc
cd [your_project]
dvc init
>>> new file:   .dvc/.gitignore
>>> new file:   .dvc/config
git commit -m "Initialize DVC"
```

2. Добавляем данные

```
dvc add data/data.json
git add data/data.xml.dvc data/.gitignore
git commit -m "Add raw data"
```

3. Подключаем облачное хранилище

```
dvc remote add -d storage s3://your-bucket/your-storage
git commit .dvc/config -m "Configure remote storage"
```

4. Отправляем данные в харнилище

```
dvc push
```

5. Получаем данные из хранилища

```
dvc pull
```

6. Добавление данных

```
dvc add data/data.json
git commit data/data.xml.dvc -m "Dataset updates"
dvc push
```

7. Переключаемся между версиями

```
git checkout <...>
dvc checkout
```

8. Python API

```python
import dvc.api

with dvc.api.open(
        'get-started/data.xml',
        repo='https://github.com/iterative/dataset-registry'
        ) as fd:
    # ... fd - можно работать как с обычным файлом.
```

Выводы

- Воспроизводимость эксперимента - это не только гиперпараметры и код, но и данные
- Возможность лучше контролировать результат дообучения моделей
- DVC - наш друг

Семинар

- Ставим DVC
- Инициализируем DVC в проекте `dvc init | git commit -m "Initialize DVC"`
- Добавляем файл с данными `dvc add data/corpus.txt | git add data/data.xml.dvc data/.gitignore | git commit -m "Add raw data"`
- Создаем бакет для хранения https://console.cloud.google.com/storage/browser
- Подключаем и настраиваем GCP CLI
- Подключаем бакет для хранения данных в облаке `dvc remote add -d myremote gs://dmia_dvc | git commit .dvc/config -m "Configure remote storage"`
- Сохраняем данные в хранилище `dvc push`
- Добавляем новые файлы, меняем старые `dvc add | git commt | dvc push`
- Возвращаемся к старым экспериментам `git checkout | dvc checkout`
- Немного про pipleine и эксперименты