

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Факультет информатики
Кафедра технической кибернетики

Отчет по лабораторной работе №2

Дисциплина: «Инженерия данных»

Тема: «Инференс и обучение НС»

Выполнил: Иванов И.А.

Группа: 6231-010402D

Самара 2024

Часть 1. Построение пайплайн для инференса данных.

Шаг 1. Разработка и реализация DAG-а

В рамках первого задания необходимо реализовать пайплайн, который реализует систему "Автоматического распознавания речи" для видеофайлов.

Построенный пайплайн будет выполнять следующие действия поочередно:

- Производить мониторинг целевой папки на предмет появления новых видеофайлов.
- Извлекать аудиодорожку из исходного видеофайла.
- Преобразовывать аудиодорожку в текст с помощью нейросетевой модели.
- Формировать конспект на основе полученного текста.
- Формировать выходной .pdf файл с конспектом.

Для реализации описанных действий мы будем использовать DockerOperator, а также FileSensor для получения необходимого видеофайла.

Для работы task-а по ожиданию получения нового видео необходимо создать новое подключение к airflow. Для создания подключения переходим в Airflow по адресу <http://localhost:8080/connection/list/>

Далее для того, чтобы можно было преобразовать наш аудиофайл в текст, а после получить из него summary, необходимо зарегистрироваться на <https://huggingface.co/> и получить токен API с правами записи для возможности отправки и получения запросов к сайту.

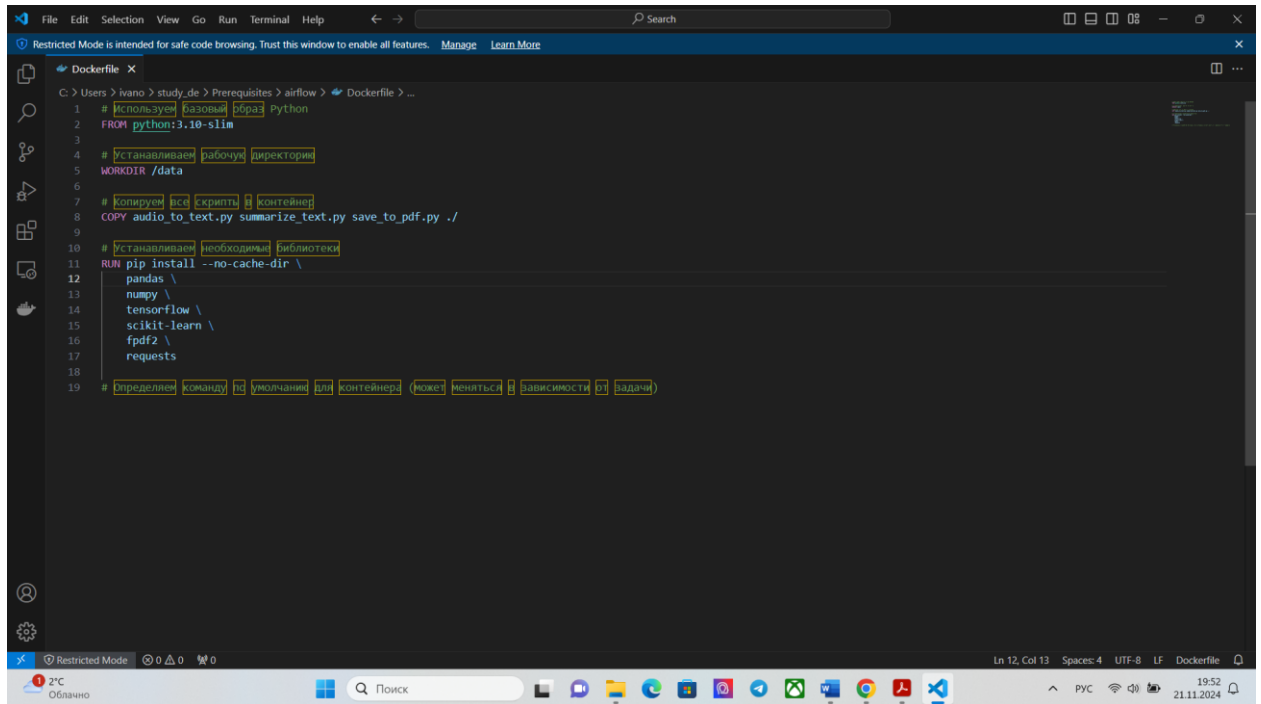
Для сохранения конспекта в PDF, необходимо было использовать библиотеку fpdf. Создадим необходимый для этого образ в Docker, который будет содержать в себе необходимые библиотеки для выполнения всей лабораторной работы. Процесс представлен ниже.

Вначале создадим Dockerfile который будет содержать в себе инструкции по сборке и разворачиванию контейнера. Контейнер мы создаем на основе tensorflow, который нам пригодится при выполнении второй части работы. Так же добавим следующие библиотеки, которые нам понадобятся в будущем:

- Scikit-learn
- Numpy

- Pandas
- FPDF

Пример Dockerfile, который я использовала в лабораторной работе приведен на рисунке ниже, а также в репозитории с решением лабораторной работы. После того как Dockerfile создан переходим в консоль и выполним процесс сборки.



```
C:\Users\ivano\study_de>Prerequisites>airflow> Dockerfile > ...
1 # Используем базовый образ Python
2 FROM python:3.10-slim
3
4 # Устанавливаем рабочую директорию
5 WORKDIR /data
6
7 # Копируем все скрипты в контейнер
8 COPY audio_to_text.py summarize_text.py save_to_pdf.py ./
9
10 # Устанавливаем необходимые библиотеки
11 RUN pip install --no-cache-dir \
12     pandas \
13     numpy \
14     tensorflow \
15     scikit-learn \
16     fpdf2 \
17     requests
18
19 # Определяем команду по умолчанию для контейнера (может меняться в зависимости от задачи)
```

Создание Dockerfile

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

Dockerfile dag_h2_task_1.py X
C:\Users\ivano> study.de > Prerequisites > airflow > dags > dag_h2_task_1.py
1 from datetime import datetime
2 from airflow import DAG
3 from airflow.providers.docker.operators.docker import DockerOperator
4 from airflow.sensors.filesystem import FileSensor
5 from docker.types import Mount # импортируем Mount
6
7 default_args = {
8     'owner': 'IlyaSwallow',
9     'start_date': datetime(2024, 9, 28),
10    'retries': 1,
11 }
12
13 dag = DAG(
14     'audio_to_pdf',
15     default_args=default_args,
16     description='DAG for extracting audio, transforming to text, summarizing, and saving as PDF',
17     schedule_interval=None,
18 )
19
20 wait_for_new_file = FileSensor(
21     task_id='wait_for_new_file',
22     poke_interval=10, # Interval to check for new files (in seconds)
23     filepath='/opt/airflow/data',
24     fs_conn_id='bombit_connection', # Target folder to monitor
25     dag=dag,
26 )
27
28 extract_audio = DockerOperator(
29     task_id='extract_audio',
30     image='jrottenberg/ffmpeg',
31     command='-i /data/input_video.mp4 -vn -acodec copy /data/audio.aac',
32     mounts=[Mount(source='/data', target='/data', type='bind')],
33     docker_url='tcp://docker-proxy:2375',
34     dag=dag,
35 )
```

Написанный DAG

Теперь после всех необходимых настроек и приготовлений, мы можем запустить наш DAG. Для этого переходим в airflow: <http://localhost:8080/home> и находим наш только что созданный DAG:

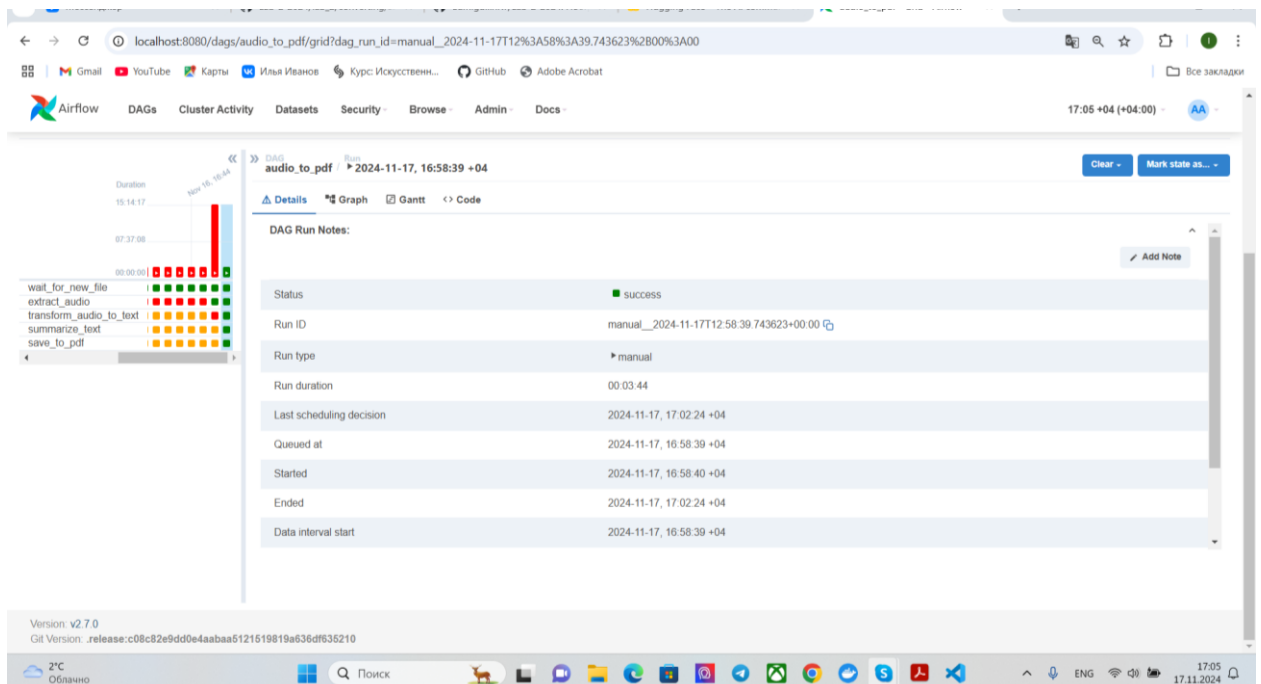


Рисунок 6 – Запуск DAG-а.

В качестве исходного видео использовался фрагмент из кинофильма «Крестный отец». После чего мы получали аудиодорожку, которая использовалась в качестве основы для получения текстового файла.

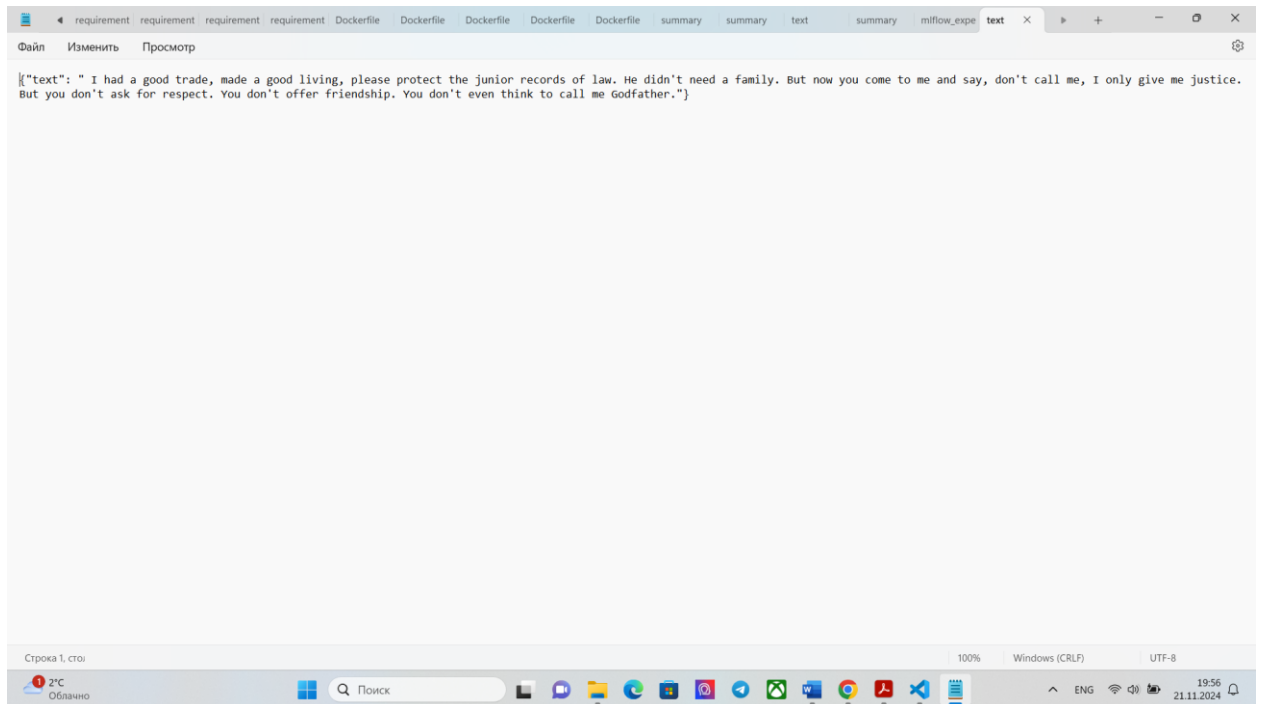


Рисунок 11 – Результат работы huggingface по преобразованию аудио в текст

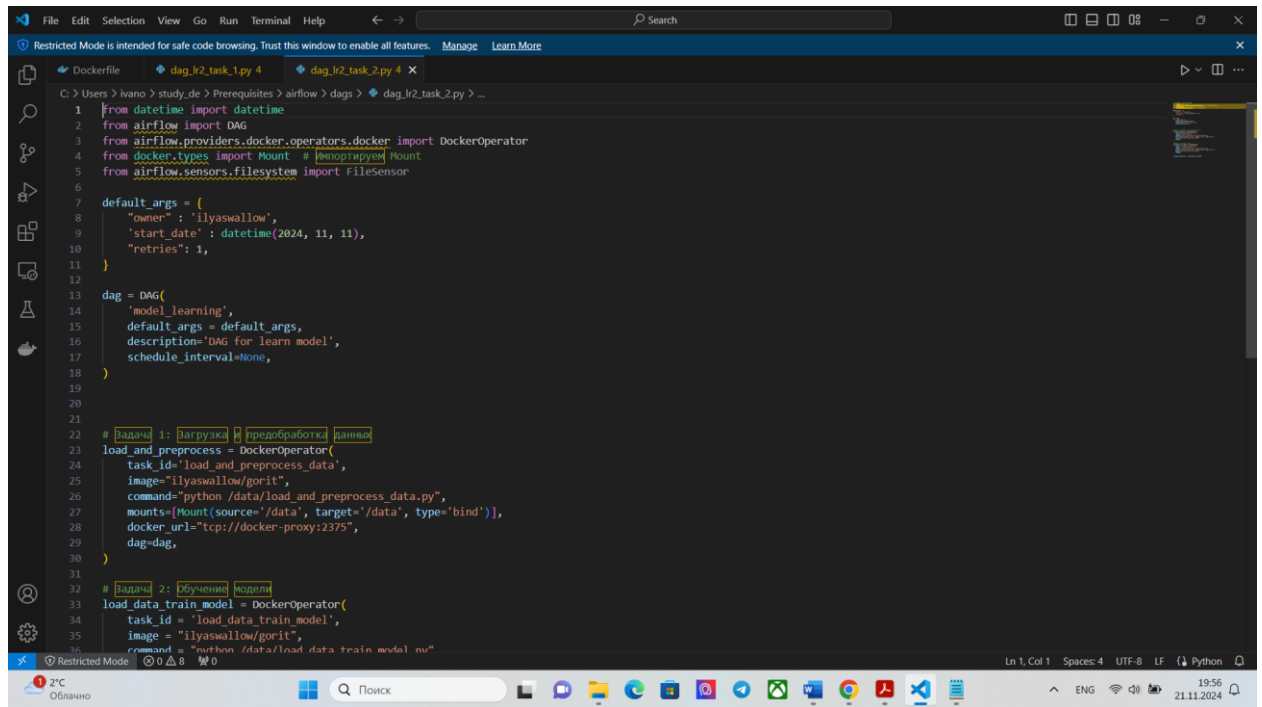
Далее полученный результат мы еще раз передавали huggingface для получения уже конспекта по отправленному нами файлу. Полученный результат записывали pdf-файл.

Получилось неплохо. Перейдем ко второй части.

Часть 2. Пайплайн, который реализует систему автоматического обучения/дообучения нейросетевой модели

В рамках второй части лабораторной работы нам необходимо было разработать пайплайн, который реализует систему автоматического обучения/дообучения нейросетевой модели.

Шаг 1. Разработка DAG



```
1 from datetime import datetime
2 from airflow import DAG
3 from airflow.providers.docker.operators.docker import DockerOperator
4 from docker.types import Mount # [неоптиризм] Mount
5 from airflow.sensors.filesystem import FileSensor
6
7 default_args = {
8     "owner": "ilyaswallow",
9     "start_date": datetime(2024, 11, 11),
10    "retries": 1,
11 }
12
13 dag = DAG(
14     'model_learning',
15     default_args = default_args,
16     description='DAG for learn model',
17     schedule_interval=None,
18 )
19
20
21 # Задача 1: Загрузка и предобработка данных
22 load_and_preprocess = DockerOperator(
23     task_id='load_and_preprocess_data',
24     image='ilyaswallow/gorit',
25     command='python /data/load_and_preprocess_data.py',
26     mounts=[Mount(source='/data', target='/data', type='bind')],
27     docker_url='tcp://docker-proxy:2375',
28     dag=dag,
29 )
30
31
32 # Задача 2: Обучение модели
33 load_data_train_model = DockerOperator(
34     task_id='load_data_train_model',
35     image = "ilyaswallow/gorit",
36     command = "nuthon /data/load_data_train_model.py"
```

Пайплайн

DAG запускал код, который получал датасет вин `load_wine` из `sklearn.datasets`, после чего мы проводили разбиение данных. Которые передаются в нейросеть, после чего модель проходит обучение. Процесс обучения логируется.

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

C:\Users\ivano> study.de > Prerequisites > airflow > data > load_data_train_model.py > ...

1 import os
2 import numpy as np
3 import tensorflow as tf
4 from sklearn.model_selection import train_test_split
5 from sklearn.datasets import load_breast_cancer
6 from tensorflow.keras.models import Sequential
7 from tensorflow.keras.layers import Dense
8 from tensorflow.keras.callbacks import CSVLogger
9 import pandas as pd
10
11 # Путь к папке с предварительно обработанными данными
12 data_dir = '/data/lab2/goy'
13
14 # Загрузка данных из файла
15 X_train = pd.read_csv(os.path.join(data_dir, 'X_train.csv')).values
16 X_test = pd.read_csv(os.path.join(data_dir, 'X_test.csv')).values
17 y_train = pd.read_csv(os.path.join(data_dir, 'y_train.csv')).values.ravel()
18 y_test = pd.read_csv(os.path.join(data_dir, 'y_test.csv')).values.ravel()
19
20 model = Sequential([
21     Dense(30, input_shape=(X_train.shape[1],), activation='relu'), # Входной слой из 30 нейронов с функцией активации ReLU
22     Dense(15, activation='relu'), # Скрытый слой из 15 нейронов с функцией активации ReLU
23     Dense(1, activation='sigmoid') # Выходной слой из 1 нейрона с сигмоидальной активацией для бинарной классификации
24 ])
25
26 # Компилируем модель, указывая оптимизатор, функцию потерь и метрики для оценки
27 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
28
29 # Надаем директорию для сохранения логов обучения
30 log_dir = '/data/logs'
31 os.makedirs(log_dir, exist_ok=True)
32
33 # Определяем файл для записи логов обучения и создаем CSV логгер
34 log_file = os.path.join(log_dir, 'training_logs.csv')
35 csv_logger = CSVLogger(log_file, append=True)
```

Код обучения модели.

Шаг 2. Запуска DAG-а.

В процессе запуска DAG-а модель была обучена и показала какие-то результаты, которые мы записали в файл. В итоге получили вот такой лог обучения:

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

C:\Users\ivano> study.de > Prerequisites > airflow > data > logs > training_logs.csv

1 epoch,accuracy,loss,val_accuracy,val_loss
2 0,0.6285714507102966,0.6924117803573608,0.6228070259094238,0.691617272796631
3 1,0.6285714507102966,0.6908723711967468,0.6228070259094238,0.6901804804801941
4 2,0.6285714507102966,0.6894702911376953,0.6228070259094238,0.6889445781707764
5 3,0.6285714507102966,0.6881193518638611,0.6228070259094238,0.6875426769256592
6 4,0.6285714507102966,0.6866475939750671,0.6228070259094238,0.6863250732421875
7 5,0.6285714507102966,0.68526291847229,0.6228070259094238,0.6851486563682556
8 6,0.6285714507102966,0.6840927600860596,0.6228070259094238,0.6837889552116394
9 7,0.6285714507102966,0.6827404499053955,0.6228070259094238,0.6826493740081787
10 epoch,accuracy,loss,val_accuracy,val_loss
11 0,0.6285714507102966,0.6924459934234619,0.6228070259094238,0.6915672421455383
12 8,0.6285714507102966,0.6816219687461853,0.6228070259094238,0.6815829277038574
13 1,0.6285714507102966,0.6908851265907288,0.6228070259094238,0.6900966167449951
14 9,0.6285714507102966,0.6805641651153564,0.6228070259094238,0.6806546449661255
15 2,0.6285714507102966,0.6893822550773621,0.6228070259094238,0.6887136101722717
16 10,0.6285714507102966,0.679581880569458,0.6228070259094238,0.6797240376472473
17 3,0.6285714507102966,0.6878746151924133,0.6228070259094238,0.6873668432235718
18 11,0.6285714507102966,0.678657054901123,0.6228070259094238,0.6787829995155334
19 4,0.6285714507102966,0.6776757836341858,0.6228070259094238,0.6781039237976074
20 12,0.6285714507102966,0.6805426301956177,0.6228070259094238,0.6801566104469299
21 13,0.6285714507102966,0.6768389040261841,0.6228070259094238,0.67727047204097131
22 5,0.6285714507102966,0.6852272740947144,0.6228070259094238,0.6840672100295593
23 14,0.6285714507102966,0.676019012920809,0.6228070259094238,0.676504075271012
24 6,0.6285714507102966,0.6840904951095581,0.6228070259094238,0.6837022304534912
25 7,0.6285714507102966,0.6828017234802246,0.6228070259094238,0.6826764041215515
26 15,0.6285714507102966,0.6752216815948486,0.6228070259094238,0.6757284998893738
27 8,0.6285714507102966,0.6816319823265076,0.6228070259094238,0.6816965341567993
28 16,0.6285714507102966,0.6743949055671692,0.6228070259094238,0.6750533580780029
29 9,0.6285714507102966,0.6805561184883118,0.6228070259094238,0.6805976629257202
30 17,0.6285714507102966,0.6736469864845276,0.6228070259094238,0.6744007468223572
31 10,0.6285714507102966,0.6794567108154297,0.6228070259094238,0.6796071529388428
32 18,0.6285714507102966,0.6730405688285828,0.6228070259094238,0.67317155318260193
33 11,0.6285714507102966,0.6784725189208984,0.6228070259094238,0.678658628082275
34 19,0.6285714507102966,0.6723023653030396,0.6228070259094238,0.6731364130973816
35 12,0.6285714507102966,0.6775026321411133,0.6228070259094238,0.6778765320777893
36 20,0.6285714507102966,0.6717407703300658,0.6228070259094238,0.6734633011580095
```

Лог процесса обучения нейросети.

Заключение

В заключении хотелось бы отметить полезные навыки, полученные в результате выполнения лабораторной работы:

1. Работа с DAG в Airflow
2. Работс сетями на huggingface