1. Исследование работы модульных и интеграционных тестов

Далее на рисунке 1 приведен скриншот страницы запуска автоматических тестов на GitHub, демонстрирующий успешную работу модульных и интеграционных тестов.

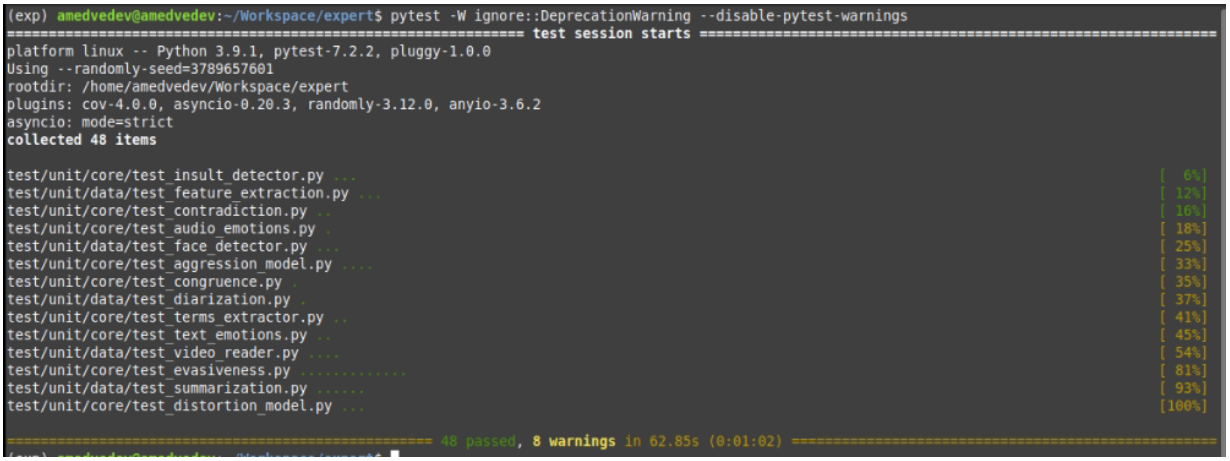
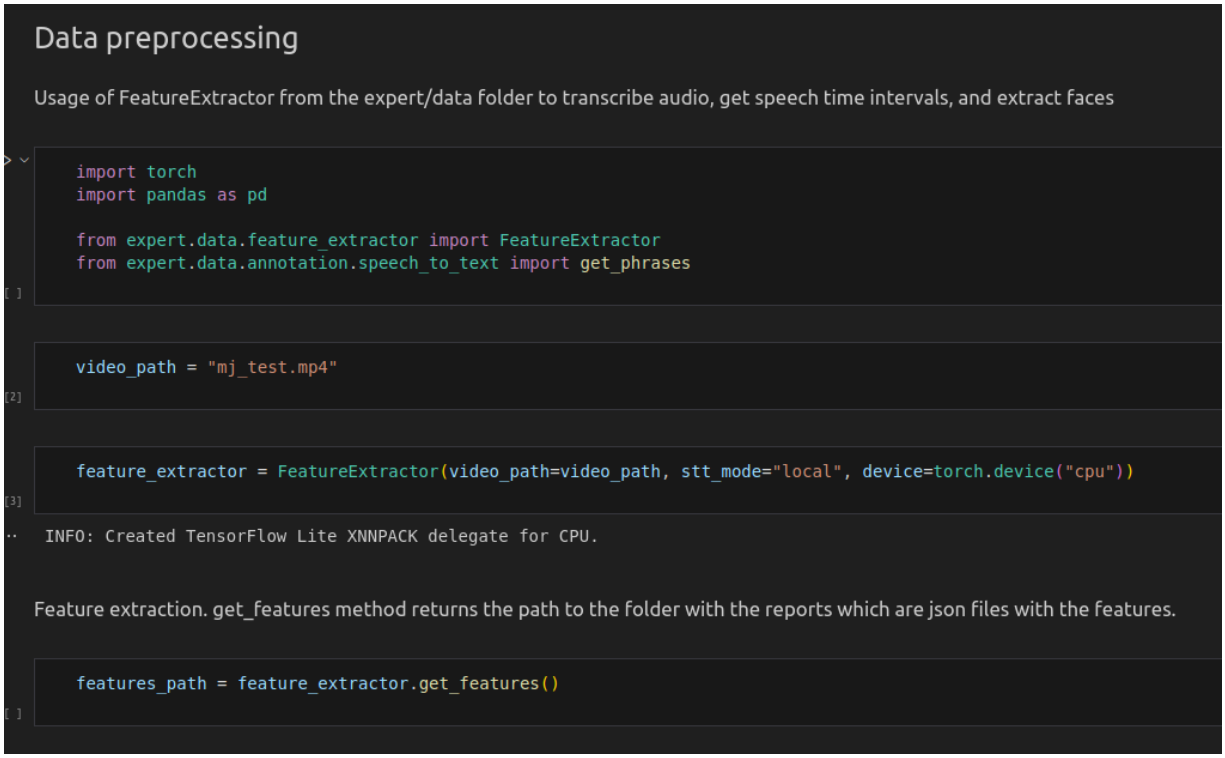


Рисунок 1 – Скриншот страницы запуска автоматических тестов на GitHub

2. Исследование возможностей решения прикладных задач ИИ в промышленности

Рассмотренные примеры доступны по ссылке: https://github.com/expertspec/expert/tree/main/examples.

2.1 Описание эмоционального состояния и оценка рассогласованности эмоций (модуль конгруэнтности)



Transcribation

```
import json

with open(features_path + "/transcription.json") as f:
    transcribation = json.load(f)
```

+ Code + Markdown

```
phrases = get_phrases(transcribation)
```

```
pd.set_option('display.max_rows', 162)
pd.set_option('display.max_colwidth', 10000)

pd.DataFrame(phrases)
```

	time	text
0	[0.98, 11.66]	Not from unnamed sources and not from unsanctioned, but for me. So that you can get a good idea of
1	[11.66, 22.78]	how I'm feeling, ask me questions, and hear what I truly believe at this moment in time about usada, and it's ongoing examination of me. I
2	[22.78, 32.92]	have been patient. I have been cooperative, and I have done all I can do to provide you, so with information that I have knowledge of, because I believe in
3	[32.92, 43.1]	a drug free sport, and having a drug free Olympic team. Throughout all of this, I have maintained my sincere belief that if the process is fair,
4	[43.7, 53.82]	that in the end, the truth would prevail, and my name would be cleared. However, the events of the last several weeks have led me more
5	[53.82, 64.72]	in sadness than in anger, to the conclusion that you sawdda is not engaged in a fair process. Let's all review the facts. I
6	[64.72, 74.9]	have never, ever failed a drug test. I have taken over 160 drug tests. I have taken tests
7	[74.9, 85.22]	before, during, and after the 2000 Olympics, and have never failed a test. Musada has no information
8	[85.22, 95.46]	that shows that I have ever failed a test, because simply I have never failed a test, and no information exists anywhere
9	[95.46, 105.54]	to even suggest that I have ever failed a test. I have truthfully answered every question asked of me under oath by
10	[105.54, 115.6]	the government and the federal grand jury during the Balco investigation. I took the extraordinary step of asking for this grand jury testimony to
11	[115.6, 126.34]	be made public to assist Musada in its examination and its efforts to get to the truth. I truthfully answered every question asked
12	[126.34, 136.58]	of me by Usada during the three hour May 24th meeting, a meeting that took place only because I asked for the meeting. I made publicly
13	[136.58, 147.48]	available the Balco information that Usada presented to me and questioned me about, and I have demonstrated that the information had no connection to me. There
14	[147.48, 157.64]	exists no one who can truthfully testify that I have ever used performance enhancing drugs simply for the reason that I never have. Despite

Congruence

```
import torch

from expert.core.congruence import CongruenceDetector
```

By default, for preprocessing reports, "temp" directory and a folder named as the analysed video file are created in it. All other files are taken by the name of the modules

```
cong_detector = CongruenceDetector(video_path=video_path,
                                   features_path=features_path + "/features.json",
                                   face_image=features_path + "/faces/0.jpg",
                                   diarization_path=features_path + "/diarization.json",
                                   transcription_path=features_path + "/transcription.json",
                                   device=torch.device("cpu"))
```

get_congruence method returns the dictionary with features {"emotions": emotions_data, "congruence": cong_data} and creates json files in correspondig folder.

```
results = cong_detector.get_congruence()
```

Visualization of results

Pyplot is used to create interactive data. There may be a conflict with pytorch-lightning, it is not used in congruence

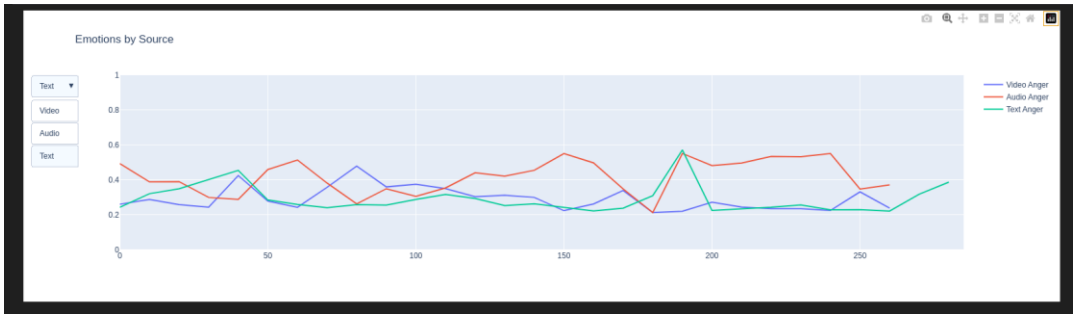
!pip install plotly

!pip install nbformat>=4.2.0

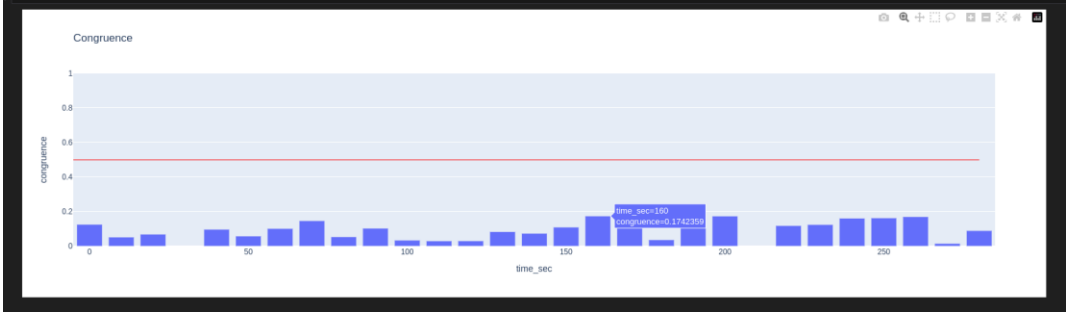
```
import plotly.express as px
import plotly.graph_objects as go

video_emo = pd.DataFrame(results["emotions"]["video"])
audio_emo = pd.DataFrame(results["emotions"]["audio"])
text_emo = pd.DataFrame(results["emotions"]["text"])
```

Visualization of the emotions by the modality

```
trace_video = go.Scatter(x=video_emo['time_sec'], y=video_emo['video_anger'], name='Video Anger')
trace_audio = go.Scatter(x=audio_emo['time_sec'], y=audio_emo['audio_anger'], name='Audio Anger')
trace_text = go.Scatter(x=text_emo['time_sec'], y=text_emo['text_anger'], name='Text Anger')

# Create buttons to switch between datasets
updatemenus = [{'buttons': [
    {'method': 'update', 'label': 'Video', 'args': [{'y': [video_emo['video_anger'], video_emo['video_neutral'], video_emo['video_happiness']]}]},
    {'method': 'update', 'label': 'Audio', 'args': [{'y': [audio_emo['audio_anger'], audio_emo['audio_neutral'], audio_emo['audio_happiness']]}]},
    {'method': 'update', 'label': 'Text', 'args': [{'y': [text_emo['text_anger'], text_emo['text_neutral'], text_emo['text_happiness']]}]}
], 'direction': 'down', 'showactive': True}]

# Create layout with buttons
layout = go.Layout(title='Emotions by Source', updatemenus=updatemenus)

fig = go.Figure(data=[trace_video, trace_audio, trace_text], layout=layout)
fig.update_xaxes(range=[0, text_emo['time_sec'].max()+5])
fig.update_yaxes(range=[0, 1])
fig.show()
```




```
Visualization of the congruence level

congr_data = pd.DataFrame(results["congruence"])

fig = px.bar(congr_data, x="time_sec", y="congruence", title='Congruence')
fig.add_shape(type='line',
              x0=-5, y0=0.5,
              x1=video_emo["time_sec"][video_emo.shape[0]-1], y1=0.5,
              line=dict(color='red', width=1))
fig.update_yaxes(range=[0, 1])
fig.show()
```



2.2 Анализ обращения в колл-центр. Аннотация, выделение фрагментов, статистика и выделение тематик.

Audio preprocessing

```
import torch

from examples.cases import example_utils
from examples.cases.call_stats import DialogStats, call_statistic
from expert.data.annotation import speech_to_text
from expert.data.diarization.speaker_diarization import SpeakerDiarization
```

```
file_path = "examples/cases/test_aud.wav"

Diarization = SpeakerDiarization(audio=file_path, device=torch.device("cpu"))
speakers = Diarization.apply()
```

```
transcribation = speech_to_text.transcribe_video(file_path, lang="ru")
```

```
all_words = speech_to_text.get_all_words(transcribation)[0]
```

```
timestamps = []
for speaker in speakers:
    for elem in speakers[speaker]:
        timestamps.append((elem, speaker))
```

```
sentences = example_utils.sentences_with_time(timestamps, all_words)
```

Connecting sentences with speech intevals form diarization

```
ci = example_utils.place_words(timestamps, sentences[1])

for num, i in enumerate(ci):
    timestamps[i[1]][0][0] = min(timestamps[i[1]][0][0], sentences[1][num][1])

timestamps.sort()

sentences = example_utils.sentences_with_time(timestamps, all_words)

sentences[0]
```

```
{0: (' Здравствуйте', ([0, 3], 'SPEAKER_01')),
 1: (' Мне нужна помощь. Я забыла, но вдруг внезапно обнаружила, что у меня есть карточка вашего банка. И на ней написано И тут я вспоминаю, что мне в паре карточку впалили.',
  ([2, 36], 'SPEAKER_00')),
 2: (' Проверьте', ([36, 37], 'SPEAKER_01')),
 3: (' и скажите, что мне надо делать с этой карточкой, какие плюшки она мне дает.',
  ([37, 42], 'SPEAKER_00')),
 4: (' ДС?', ([42, 44], 'SPEAKER_01')),
 5: (' Нет, через О В секунду', ([44, 50], 'SPEAKER_00')),
 6: (' Я даже не знаю, какой кот у этой карточки. Сейчас посмотрим, скажу вам.',
  ([50, 57], 'SPEAKER_01')),
 7: ('', ([51, 55], 'SPEAKER_00')),
 8: (' Обычно на карточках расписываются. У меня тут подлинные стахи.',
  ([63, 68], 'SPEAKER_00')),
 9: ('', ([68, 69], 'SPEAKER_01')),
 10: (' Так, дата рождения ваша?', ([74, 76], 'SPEAKER_01')),
 11: (' Четвёртая, двенадцать, с отвощенного.', ([76, 80], 'SPEAKER_00')),
 12: (' Так, и Да, верно назвали. Так, у вас пакет',
  ([84.0, 92], 'SPEAKER_01')),
 13: (' услуг нового.', ([92, 95], 'SPEAKER_00')),
```

Extracting of specific statistics

```
from examples.cases.call_stats import DialogStats, call_statistic
```

```
report = call_statistic(Diarization.audio, Diarization.sr, sentences[0], speakers)
```

```
Перебивал SPEAKER_00
00:02 - 00:03
Перебивал SPEAKER_01
00:36 - 00:36
Перебивал SPEAKER_00
00:37 - 00:37
Перебивал SPEAKER_01
00:42 - 00:42
Перебивал SPEAKER_00
00:44 - 00:44
Перебивал SPEAKER_01
00:50 - 00:50
```

```
pd.DataFrame(report)
```

Python

Продолжительность разговора сек	Тишина в начале сек	Тишина в конце сек	Длительность тишины сек	Доля тишины	Продолжительность речи SPEAKER_00	Скорость речи SPEAKER_00	Количество интервалов SPEAKER_00	Доля речи SPEAKER_00	Продолжительность речи SPEAKER_01	Скорость речи SPEAKER_01	Количество интервалов SPEAKER_01	Доля речи SPEAKER_01	Продолжительность одновременной речи	одно
414.84	0	-0.16	69.0	6.012174	218.52	128.983018	21	0.631853	227.32	127.887393	24	0.657298	50	

Analysis of emotions by speech

```
from expert.core.congruence import audio_emotions
```

17]

```
speakers = example_utils.get_rounded_intervals(speakers)
```

48]

```
CoAudio = audio_emotions.audio_analysis.AudioAnalysis(video_path=file_path,
                                                      stamps=speakers,
                                                      speaker="SPEAKER_00",
                                                      sr=44100)
```

52]

Emotions of the first speaker

```
audio_0_emotions = CoAudio.predict()
```

53]

```
[W NNPACK.cpp:53] Could not initialize NNPACK! Reason: Unsupported hardware.
```

Emotions of the second speaker

```
CoAudio = audio_emotions.audio_analysis.AudioAnalysis(video_path=file_path,
                                                      stamps=speakers,
                                                      speaker="SPEAKER_01",
                                                      sr=44100)
audio_1_emotions = CoAudio.predict()
```

55]

```
emo_audio_0 = pd.DataFrame(audio_0_emotions)
emo_audio_1 = pd.DataFrame(audio_1_emotions)
```

56]

```
anger_speaker_1 = emo_audio_1.loc[(emo_audio_1['audio_anger'] > emo_audio_1['audio_neutral']) & (emo_audio_1['audio_anger'] >  emo_audio_1['audio_happiness'])]
happy_speaker_1 = emo_audio_1.loc[(emo_audio_1['audio_happiness'] > emo_audio_1['audio_neutral']) & (emo_audio_1['audio_happiness'] >  emo_audio_1['audio_anger'])]

anger_speaker_0 = emo_audio_0.loc[(emo_audio_0['audio_anger'] > emo_audio_0['audio_neutral']) & (emo_audio_0['audio_anger'] >  emo_audio_0['audio_happiness'])]
happy_speaker_0 = emo_audio_0.loc[(emo_audio_0['audio_happiness'] > emo_audio_0['audio_neutral']) & (emo_audio_0['audio_happiness'] >  emo_audio_0['audio_anger'])]
```

```
from termcolor import colored

from datetime import datetime
```

Show sentences of the first speaker said with aggression

```
for time in anger_speaker_0['time_sec']:
    for elem in sentences[0]:
        if sentences[0][elem][1][1] == 'SPEAKER_00' and sentences[0][elem][1][0][0] <= time and  sentences[0][elem][1][0][1] >= time:
            print(datetime.fromtimestamp(time).strftime("%M:%S"), '  ', colored(sentences[0][elem][0], 'red'))
```

```
03:03     Сейчас по комиссии.
```

Show sentences of the first speaker said with happiness

```
for time in happy_speaker_0['time_sec']:
    for elem in sentences[0]:
        if sentences[0][elem][1][1] == 'SPEAKER_00' and sentences[0][elem][1][0][0] <= time and  sentences[0][elem][1][0][1] >= time:
            print(datetime.fromtimestamp(time).strftime("%M:%S"), '  ', colored(sentences[0][elem][0], 'green'))
```

```
01:32     услуг нового.
02:32     скачать это приложение. И, конечно, это не так. Я живу. Какие у меня еще варианты есть? В отделении банка. В отделении банка спрашивают, вам помогут активировать вам вс
02:43
03:08     Я не очень поняла, когда
03:13     не очень поняла, когда просто пустая карта смысл в ее. Но вроде как, какая-то сумма должна быть или что-то там должно покупаться с этой карты.
```

An example of the result obtained with AutoTM analysis for every speaker

```
result = pd.read_csv('./AutoTM/src/result.csv')
```

```
result[['speakers', 'topic_1', 'topic_2']]
```

	speakers	topic_1	topic_2
0	speaker_00	Тарифный план по обслуживанию карты	Бонусы при использовании карты
1	speaker_01	Тарифный план по обслуживанию карты	Бонусы при использовании карты