федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий
Кафедра управления и интеллектуальных технологий

Отчёт по лабораторной работе №2
По дисциплине «Методы и алгоритмы обработки данных и изображений»

Выполнили студенты: Михайловский М. Ю., Озеров С. Д.
Группа: А-02м-25
Бригада: 2
Проверил: Бородкин А. А.

Москва 2025

Содержание

1 ВЫПОЛНЕНИЕ РАБОТЫ

1.1 Выявление голосовой активности

Имеется звуковой файл в формате .wav, где диктор на протяжении минуты зачитывает текст. В данной записи имеются значительные шумы, которые громче самого диктора. График первых 1,5 секунд этого сигнала представлен на рис. 1.1.

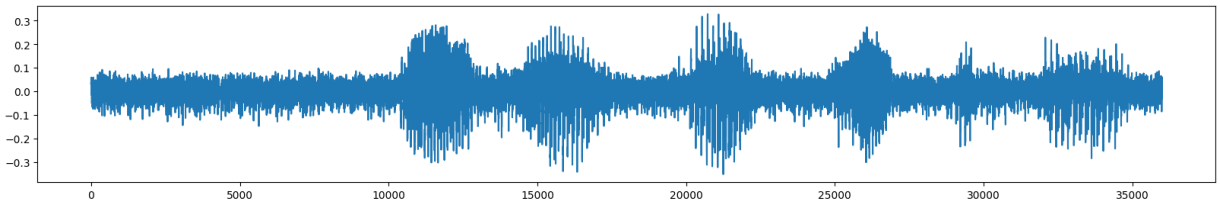

Рис. 1.1. Исходный речевой сигнал

Для выявления голосовой активности была написана программа, представленная на листинге 3. Первые 0.4 секунды были вырезаны в качестве фрагмента с примером шума. Этот фрагмент был разбит на кадры с перекрытием, и для каждого кадра рассчитана кратковременная энергия речевого сигнала:

$$E = \frac{1}{N}\sum_{k=1}^{N} s_k^2, \tag{1}$$

где s_k – отсчёты речевого сигнала.

Для шума было получено пороговое значение $E_{\text{пор}} = \max \mathbf{E} = \max\left(E_1 \ \ldots \ E_M\right)$, где значение M – количество кадров, на которое был разбит сигнал

Это значение будет использоваться для выявления речевой активности. Кадры, для которых $E < E_{\text{пор}}$ будут считаться участками молчания. Для рассматриваемого сигнала было получено значение $E_{\text{пор}} = 1{,}63 \cdot 10^{-3}$. Результаты выявления участков речи представлены на рис. 1.2.

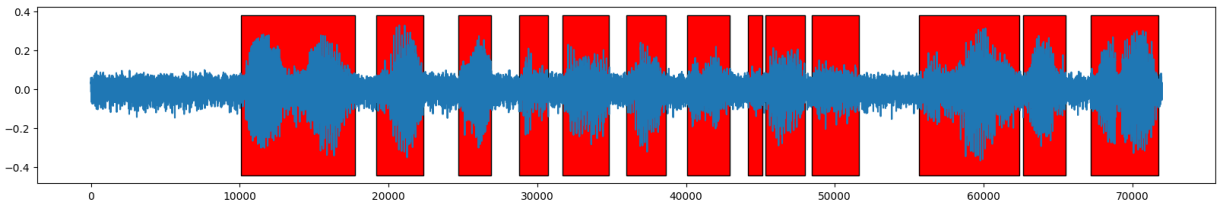

Рис. 1.2. Результат работы алгоритма выявления участков речи

Также этот алгоритм будет использован далее в работе.

1.2 Фильтрация шумов

Для фильтрации шумов было рассмотрено два подхода. Был реализован алгоритм, представленный в листинге 4.

Первый подход основан на функции *reduce_noise* из библиотеки **noisereduce**. Это модификация метода спектральных вычитаний. Для разных параметров, этот алгоритм позволяет по-разному устранить шум, в том числе полностью его заглушить. Однако, для параметров, полностью устраняющих шум, сам речевой сигнал становится сильно заглушённым, в следствие чего малоразборчивым.

Если подбирать параметры так, чтобы шум снижался не полностью, получается добиться хорошей слышимости при низком значении шума.

В этой модификации на фильтрацию влияют следующие параметры:

- **prop_decrease=1** – представляет собой коэффициент пропорциональности, определяющий то, какая часть спектра шума вычитается из исходного спектра. Это аналог параметра A из классического метода спектрального вычитания;
- **thresh_n_mult_nonstationary=1.5** – пороговое значение, в количестве среднеквадратических отклонений (СКО) сигнала от СКО шума, для того, чтобы в сигнале было принято наличие речи. Косвенный аналог B из классического метода, представляющего собой минимальный уровень спектра.

В результате фильтрации этим методом один из полученных сигналов имел вид, представленный на рис. 1.3.

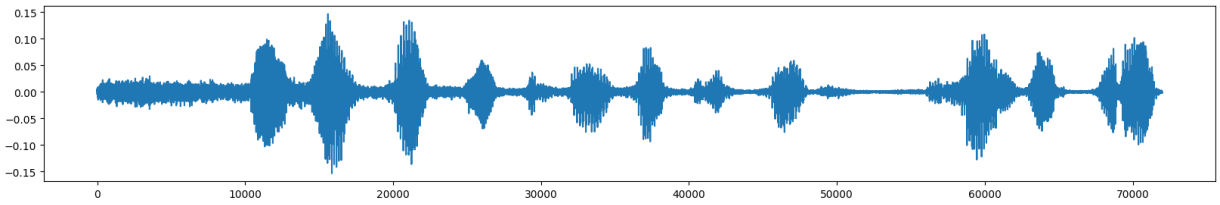

Рис. 1.3. Фильтрация шумов через reduce_noise

Второй подход основан на использовании цифровых фильтров низких частот. Если шум является высокочастотной аддитивной составляющей, то он таким фильтром может быть снижен.

Для фильтрации было использовано следующее разностное уравнение. $y[0] = \frac{s[0]}{\sqrt{1-\alpha^2}}$:

$$y[n] = \alpha \cdot y[n-1] + \sqrt{1-\alpha^2} \cdot s[n] \tag{2}$$

Здесь полученный вектор **y** – представляет собой отфильтрованный сигнал **s**. Полученный результат для $\alpha = 0{,}99$ представлен на рис. 1.4. В отфильтрованном сигнале всё ещё слышен шум, однако он стал значительно тише, по сравнению с голосом диктора. Сам голос диктора сохранился разборчивым.

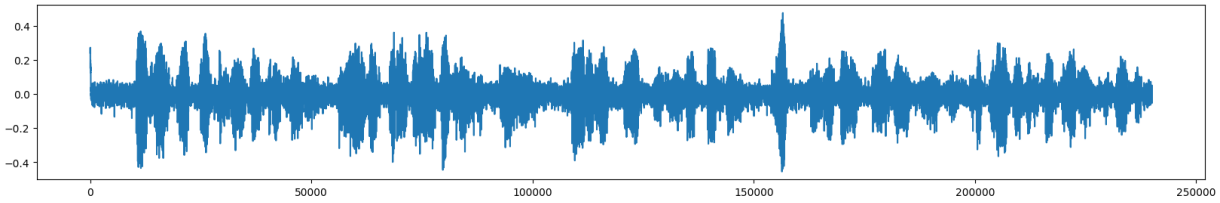

Рис. 1.4. Фильтрация шумов через цифровой низкочастотный фильтр

1.3 Запись и сегментация голосовых команд

Была написана программа для записи реализаций голосовых команд, представленная на листинге 2. С помощью неё были записаны выборки команд, по выбору направлений и действий роботом, в ходе исследования территории.

В данном алгоритме первая секунда используется для определения $E_{\text{пор}}$ шума. Все последующие фрагменты, на которых выявлена голосовая активность, и которые не являются слишком короткими, вырезаются в отдельные .wav файлы.

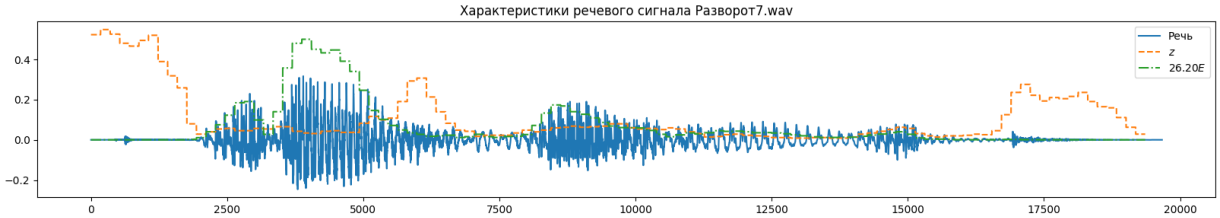

Рис. 1.5. Фильтрация шумов через цифровой низкочастотный фильтр

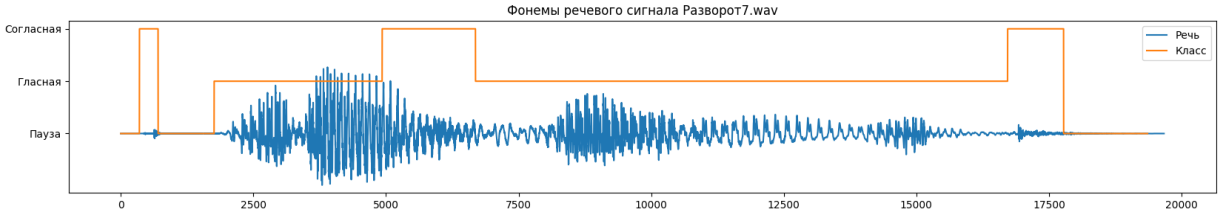

Рис. 1.6. Фильтрация шумов через цифровой низкочастотный фильтр

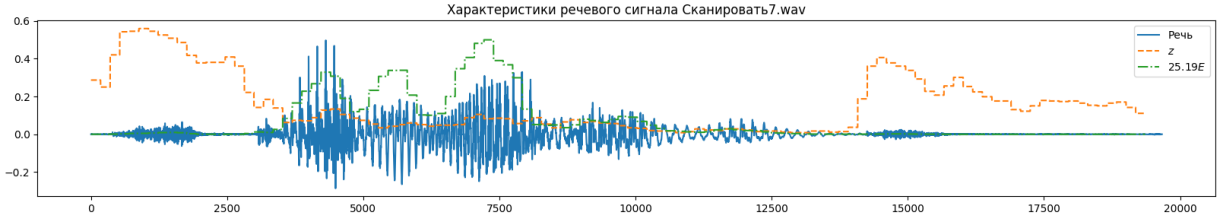

Рис. 1.7. Фильтрация шумов через цифровой низкочастотный фильтр

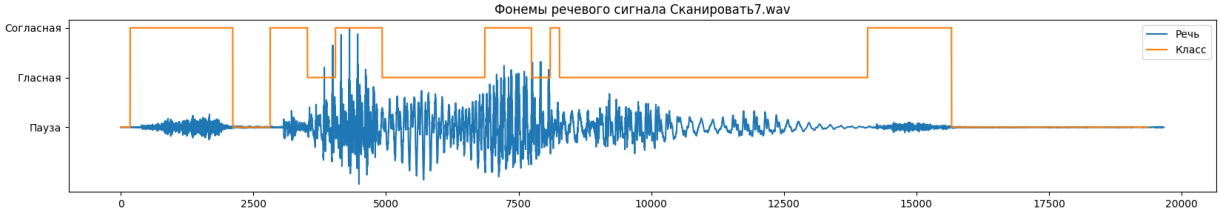

Рис. 1.8. Фильтрация шумов через цифровой низкочастотный фильтр

Для классификации фонем был написан алгоритм, описанный в листинге 5. Здесь сформирована функция, которая по данным пороговым значениям кратковременной энергии речевого сигнала $E_{\text{пор}}$ и числу нулей интенсивности $z_{\text{пор}}$ классифицирует кадры речевого сигнала на классы: пауза, гласный, согласный. Пороговые значения были подобраны вручную.

Рассчитанные характеристики для кадров, представлены на рис. 1.5, 1.7. А классифицированные фонемы, приведены на рис. 1.6, 1.8. Были взяты два слова, из ранее записанного словаря.

Можно видеть, что чётко определяются гласные. Некоторые согласные приняты за гласные, что может быть связано с их малым временем произношения в середине слова, а также с особенностями произношения конкретного диктора.

2 ВЫВОДЫ

В данной работе были рассмотрены 3 темы по обработке речевых сигналов.

Был рассмотрен алгоритм выявления голосовой активности. Он успешно выделил участки речи, в анализируемом сигнале.

Были рассмотрены два алгоритма фильтрации шумов. Метод, основанный на анализе спектров, позволяет снизить шум до требуемого уровня, за счёт чёткости речи. Метод, основанный на цифровой фильтрации, частично снижает наличие шума в сигнале.

Был реализован алгоритм записи голосовых команд, и автоматической вырезки их из одной записи. Для команд был реализован алгоритм сегментации, который проводит классификацию кадров на паузы, гласные и согласные. Этот алгоритм хорошо определяет наличие пауз, но склонен пропускать некоторые согласные в угоду гласным.

Приложение А. Листинги

Листинг 1. Файл dataprocessing.py

```
import numpy as np
import soundfile as sf
import librosa.feature as ft

def dataToEnergy(data, sr, frame_time, frame_shift, do_print = True):
    frameWidth = int(frame_time * sr)
    frameShift = int(frame_shift * frameWidth)

    frameCount = int(data.size / (frameWidth - frameShift)) - 1
    if do_print:
        print(f'{frameWidth=}')
        print(f'{frame_shift=}')
        print(f'{frameCount=}')

    E = [0.0] * frameCount
    sh = 0
    df = frameWidth - frameShift
    for i in range(frameCount):
        En = 0
        for j in range(frameWidth):
            sn = float(data[j + sh] ** 2)
            En = En + sn

        E[i] = En / frameWidth
        sh = sh + df

    return E

def getEMax(file_name, seconds_from_start, frame_time, frame_shift):
    data, samplerate = sf.read(file_name)

    noise_indices = int(seconds_from_start * samplerate)
    noise_data = data[:noise_indices]

    E = dataToEnergy(noise_data, samplerate, frame_time, frame_shift, do_print=False)
    return np.max(E)

def VAD(data, sr, frame_time, frame_shift, noise_frame_end, eTh):
    frameWidth = int(frame_time * sr)
    frameShift = int(frame_shift * frameWidth)

    frameCount = int(data.size / (frameWidth - frameShift)) - 1

    maxY = np.max(data)
    df = frameWidth - frameShift
    E = [0] * frameCount
    sh = 0
    for i in range(frameCount):
        En = 0

```

```
        for j in range(frameWidth):
            sn = float(data[j + sh] ** 2)
            En = En + sn

        E[i] = En / frameWidth
        sh = sh + df

    if noise_frame_end > 0:
        ePorog = 0
        for j in range(0, noise_frame_end):
            if ePorog < E[j]:
```

Листинг 2. Файл recording.py

```
import wave
import pyaudio as py
import keyboard
from array import array
import os
import dataprocessing as dp
import numpy as np
import soundfile as sf

def record_voice(file_to_save, descriptive = True):
    CHUNK = 1024
    FORMAT = py.paInt16
    CHANNELS = 1
    RATE = 22050

    p = py.PyAudio()
    stream = p.open(format=FORMAT, channels=CHANNELS, rate=RATE, input=True,
    ↪  frames_per_buffer=CHUNK)

    data_all = array('h')

    recording = True

    def key_press(e):
        nonlocal recording
        if e.name == 'enter':
            recording = False
            keyboard.unhook_all()

    keyboard.hook(key_press)

    start_prompted = False
    while recording:
        if stream.is_active():
            if not start_prompted:
                start_prompted = True
                if descriptive:
                    print("Началась запись")

```

```
            data = stream.read(CHUNK, exception_on_overflow=False)
            data_all.extend(array('h', data))

    if descriptive:
        print('Вышли из цикла')

    stream.stop_stream()
    stream.close()
    p.terminate()
    if descriptive:
        print('Запись остановлена')

    wav = wave.open(file_to_save, "wb")
    wav.setnchannels(CHANNELS)
    wav.setframerate(RATE)
    wav.setsampwidth(p.get_sample_size(FORMAT))
    wav.writeframes(data_all)
    wav.close()
    if descriptive:
        print(f"Запись сохранена в {file_to_save}")

def main():
    FRAME_TIME = 20E-3
```

Листинг 3. Выявление участков голосовой активности

```
import numpy as np
import soundfile as sf
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import noisereduce as nr
import dataprocessing as dp
import librosa.feature as ft
from enum import Enum

# Считывание файла с речью
data, samplerate = sf.read('Y.wav')
seconds_to_plot = 1.5
indices_to_plot = int(seconds_to_plot * samplerate)
plt.rcParams["figure.figsize"] = (20,3)
plt.plot(data[:indices_to_plot])
plt.show()

# Сохранение шума в начале
noise_seconds = 0.4
noise_indices = int(noise_seconds * samplerate)
noise_data = data[:noise_indices]
dp.resave('Y.wav', 0, noise_indices, 'Y_noise.wav')

# Получение выборки кратковременных энергий кадров речевого сигнала
frame_time = 20e-3
frame_shift = 0.5

```

```
28
29 E = dp.dataToEnergy(noise_data, samplerate, frame_time, frame_shift, do_print=False)
30 E_noise_max = np.max(E)
31 print(f'{E_noise_max=:.2e}')
32 plt.rcParams["figure.figsize"] = (6,4)
33 plt.plot(E)
34 plt.show()
35
36 # Выявление голосовой активности
37 vadData, tickData = dp.VAD(data, samplerate, frame_time, frame_shift, noise_frame_end=0, eTh=
   ↪  E_noise_max)
38
39 seconds_to_plot = 3
40 indices_to_plot = int(seconds_to_plot * samplerate)
41 ticks = []
42 for tick in tickData:
43         if tick > indices_to_plot:
44                 break
45
46         ticks.append(tick)
47
48 plt.rcParams["figure.figsize"] = (20,3)
49 fig, ax = plt.subplots()
50 for i in range(int(len(ticks) / 2)):
51         left_tick = ticks[2*i]
52         right_tick = ticks[2*i + 1]
53         rect = Rectangle((left_tick, np.min(data)), width=right_tick - left_tick, height =
          ↪  np.max(data) - np.min(data), facecolor='r', linewidth=1, edgecolor='k')
54         ax.add_patch(rect)
55
56 ax.plot(data[:indices_to_plot])
57 plt.show()
```

Листинг 4. Фильтрация шума

```
1 # Метод recude_noise (Модификация метода спектральных вычитаний)
2 filtered_data = nr.reduce_noise(y=data, sr=samplerate, y_noise=noise_data, prop_decrease=1,
  ↪  thresh_n_mult_nonstationary=1.5, stationary=False)
3 plt.rcParams["figure.figsize"] = (20,3)
4 plt.plot(filtered_data[:indices_to_plot])
5 plt.show()
6
7 sf.write('Y_filtered12.wav', filtered_data, samplerate)
8
9 # Цифровая фильтрация
10 def digital_filter(data, alpha):
11         N = len(data)
12         y = np.zeros(N)
13         y[0] = data[0] / np.sqrt(1 - alpha**2)
14         for n in range(1, N):
15         y[n] = alpha * y[n-1] + np.sqrt(1 - alpha**2) * data[n]
16
17         return y
18
19 seconds_to_plot = 10
```

```
indices_to_plot = int(seconds_to_plot * samplerate)

filtered_data2 = digital_filter(data, 0.99)
plt.rcParams["figure.figsize"] = (20,3)
plt.plot(filtered_data2[:indices_to_plot])
plt.show()

sf.write('Y_filtered2.wav', filtered_data2, samplerate)
```

Листинг 5. Определение типов фонем

```
word1_path = './notebook/Разворот/Разворот7.wav'
word2_path = './notebook/Сканировать/Сканировать7.wav'

data1, samplerate1 = sf.read(word1_path)
data2, samplerate2 = sf.read(word2_path)

frame_time = 16e-3
frame_length = int(frame_time * samplerate1)
frame_shift_ratio = 0.5
frame_shift = int(frame_shift_ratio * frame_length)

class Phoneme(Enum):
        Pause = 0
        Vowel = 1
        Consonant = 2

        def __int__(self):
                return self.value

def scale_to_plot(data):
        to_plot_data = []
        for zero_crossing_rate in data:
        to_plot_data += [zero_crossing_rate] * frame_shift

        return to_plot_data

def calc_plot_features(data, samplerate, frame_length, frame_shift, E_max, z_max, data_name:str =
↪  ''):
        # Расчёт характеристик
        E = dp.dataToEnergy(data, samplerate, frame_time, frame_shift_ratio, do_print=False)
        E = np.array(E)
        E_max_calc = np.max(E)
        scaler = 0.5/E_max_calc
        E *= scaler
        zero_crossing_rates = ft.zero_crossing_rate(data, frame_length=frame_length,
        ↪  hop_length=frame_shift, center=False)

        # Классификация фонем для кадров
        phoneme_classes = np.empty(len(E), dtype=object)
        pause_mask = E < E_max
        vowel_mask = (~pause_mask) & (zero_crossing_rates[0] < z_max)
        consonant_mask = ~(pause_mask | vowel_mask)

        phoneme_classes[pause_mask] = Phoneme.Pause
```

```
43          phoneme_classes[vowel_mask] = Phoneme.Vowel
44          phoneme_classes[consonant_mask] = Phoneme.Consonant
45
46          # Преобразование для построения графиков
47          to_plot_crossing_rate = scale_to_plot(zero_crossing_rates.flatten())
48          to_plot_E = scale_to_plot(E)
49          to_plot_classes = scale_to_plot(phoneme_classes)
50
51          plt.title(f'Характеристики речевого сигнала {data_name}')
52          plt.plot(data, label='Речь')
53          plt.plot(to_plot_crossing_rate, '--', label='$z$')
54          plt.plot(to_plot_E, '-.', label=f'${scaler:.2f}E$')
55          plt.legend()
56          plt.show()
57
58          plt.title(f'Фонемы речевого сигнала {data_name}')
59          plt.plot(data, label='Речь')
60          plt.plot(0.25*np.int16(to_plot_classes), label='Класс')
61          plt.yticks(
62          [0, 0.25, 0.5],
63          ['Пауза', 'Гласная', 'Согласная']
64          )
65          plt.legend()
66          plt.show()
67
68          return E, zero_crossing_rates
69
70  E_max = 1e-4
71  z_max = 0.08
72
73  calc_plot_features(data1, samplerate1, frame_length, frame_shift, E_max, z_max,
    ↪  word1_path.split('/')[-1])
74  calc_plot_features(data2, samplerate2, frame_length, frame_shift, E_max, z_max,
    ↪  word2_path.split('/')[-1])
```