**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |  |
| --- | --- | --- | --- |
| СОГЛАСОВАНО  Научный руководитель,  доцент департамента  программной инженерии факультета компьютерных наук,  канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.З.Ахметсафина  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | УТВЕРЖДАЮ  Академический руководитель образовательной программы  «Программная инженерия»  профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_ г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **Программа для адаптации музыкального произведения под определенный стиль на основе машинного обучения**  **Текст программы**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.507600-01 12 01-1-ЛУ** | | |
|  |  | |
| Исполнитель  студент группы БПИ175  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / И. О. Балбин /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | |
|  | | |
|  | |  |

**Москва 2017**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729.507600-01 12 01-1-ЛУ |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **Программа для адаптации музыкального произведения под определенный стиль на основе машинного обучения**  **Текст программы**  **RU.17701729.507600-01 12 01-1**  **Листов 18** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
| **Москва 2017** | | | |  |

**СОДЕРЖАНИЕ**

[1. ТЕКСТ ПРОГРАММЫ 2](#_Toc483243963)

[Приложение 1 17](#_Toc483243964)

# ТЕКСТ ПРОГРАММЫ

**run.py**

from app import app

**config.py**

import os

basedir = os.path.dirname(\_\_file\_\_)

class Config(object):

UPLOAD\_FOLDER = basedir + '\\uploads'

**\_\_init\_\_.py**

from flask import Flask

from config import Config

app = Flask(\_\_name\_\_)

app.config.from\_object(Config)

from app import routes

**routes.py**

from flask import render\_template, request, flash, redirect, url\_for, send\_from\_directory

from app import app

from app.wavNN import calc\_new\_music

from app.midiNN.remix import lets\_do\_it

from multiprocessing import Process

import os

ALLOWED\_EXTENSIONS\_MID = set(['mid'])

ALLOWED\_EXTENSIONS\_WAV = set(['wav'])

def allowed\_file(filename, mode):

if mode == 'MID':

return '.' in filename and \

filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS\_MID

else:

return '.' in filename and \

filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS\_WAV

@app.route('/uploads/<filename>')

def uploaded\_file(filename):

return send\_from\_directory(app.config['UPLOAD\_FOLDER'],

filename)

@app.route('/index', methods=['GET', 'POST'])

@app.route('/', methods=['GET', 'POST'])

def index():

return render\_template("index.html")

@app.route('/process\_data/wav/', methods=['Get','POST'])

def wavadapt():

if request.method == 'POST':

# check if the post request has the file part

if ('style' not in request.files) or ('music' not in request.files):

return redirect(request.url)

style = request.files['style']

music = request.files['music']

if style.filename == '' or music.filename == '':

return redirect(request.url)

if style and allowed\_file(style.filename, 'wav') and music and allowed\_file(music.filename, 'wav'):

music.save(os.path.join(app.config['UPLOAD\_FOLDER'], "music.wav"))

style.save(os.path.join(app.config['UPLOAD\_FOLDER'], "style.wav"))

if os.path.isfile('{}/out.wav'.format(app.config['UPLOAD\_FOLDER'])):

os.remove('{}/out.wav'.format(app.config['UPLOAD\_FOLDER']))

p = Process(target=calc\_new\_music)

p.start()

return redirect(url\_for('wait'))

return render\_template("wav.html")

@app.route('/process\_data/mid/', methods=['Get','POST'])

def midadapt():

if request.method == 'POST':

if 'music' not in request.files:

return redirect(request.url)

music = request.files['music']

if music.filename == '':

return redirect(request.url)

if music and allowed\_file(music.filename, 'MID'):

mode = request.form['style']

music.save(os.path.join(app.config['UPLOAD\_FOLDER'], "music.mid"))

if os.path.isfile('{}/out.mid'.format(app.config['UPLOAD\_FOLDER'])):

os.remove('{}/out.mid'.format(app.config['UPLOAD\_FOLDER']))

p = Process(target=lets\_do\_it, args=(mode,))

p.start()

return redirect(url\_for('wait'))

return render\_template("mid.html")

@app.route('/wait', methods=['GET', 'POST'])

def wait():

if request.method == 'POST':

if 'out.mid' in os.listdir(app.config['UPLOAD\_FOLDER']):

return redirect(url\_for('uploaded\_file',

filename="out.mid"))

if 'out.wav' in os.listdir(app.config['UPLOAD\_FOLDER']):

return redirect(url\_for('uploaded\_file',

filename="out.wav"))

return render\_template("waitpage.html")

return render\_template("waitpage.html")

**train.py**

import random

import ngram

import glob

from keras import Sequential

from keras.layers import LSTM, Dense

from music21 import converter, instrument, note, chord, stream

from music21.ext import joblib

import music21

from sklearn.preprocessing import LabelBinarizer

from app import wild\_card

#from heapq import nlargest

#import operator

import app.wild\_card

from app.music\_controller import get\_msg, create\_midi

def train(file):

# Тут храним ноты

notes = []

# Счетчик для ограничения кол-ва мелодий

nn = 0

print('Engage!')

# Тусуемся в папке

##for file in glob.glob("midi/mario/test/\*.mid"):

# Добавляем в хранилище еще ноты

notes.extend(get\_msg(file))

#nn+=1

#if nn > 20:

#break

# Сохраняем на всякий склеиную мелодию

#create\_midi(notes)

print(notes)

# exit()

print("Создаем словарь всех возможных нот...")

encoder = LabelBinarizer()

encoder.fit(notes)

joblib.dump(encoder, "app/encoders/LabelBinarizer.sav")

data = encoder.transform(notes)

print(len(data[0]))

print("Создаем ngram для поиска наиболее похожих нот...")

notes\_set = set(notes)

ngram\_notes = list(notes\_set)

G = ngram.NGram(ngram\_notes)

joblib.dump(G, "app/encoders/ngram.sav")

print("Создаем последовательности нот для обучения...")

look\_back = 3

dataX, dataY = wild\_card.create\_dataset(data, look\_back)

print("Перемешиваем мелодии...")

combined = list(zip(dataX, dataY))

random.shuffle(combined)

dataX[:], dataY[:] = zip(\*combined)

print("Создаем тренировочные и тестовые данные...")

size\_train = int(len(dataX)\*0.8)

trainX = dataX[:size\_train]

trainY = dataY[:size\_train]

testX = dataX[size\_train:]

testY = dataY[size\_train:]

print('Создаем нейросеть...')

model = Sequential()

# model.add(LSTM(256, input\_shape=(look\_back, len(data[0])), return\_sequences=True))

# model.add(Dense(256, activation='relu'))

# model.add(LSTM(256))

# model.add(Dense(256, activation='relu'))

# model.add(Dense(len(data[0])))

model.add(LSTM(64, input\_shape=(look\_back, len(data[0]))))

model.add(Dense(32, activation='relu'))

model.add(Dense(len(data[0]), activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['acc'])

acc = []

print("Обучаем...")

for i in range(50):

print("EPOCH " + str(i))

out = model.fit(trainX, trainY, epochs=4, batch\_size=32, verbose=1, shuffle=True)

eval = model.evaluate(testX, testY, verbose=1)

print(eval)

acc.append(eval[1])

#model.save("models/mario" + str(eval[1]) + ".h5")

model.save("app/models/mario.h5")

#exit()

**wavNN.py**

import librosa

import tensorflow as tf

import os

from IPython.display import Audio, display

import numpy as np

CONTENT\_FILENAME = "uploads/music.wav"

STYLE\_FILENAME = "uploads/style.wav"

OUTPUT\_FILENAME = 'uploads/out.wav'

def calc\_new\_music():

N\_FFT = 2048

def read\_audio\_spectum(filename):

x, fs = librosa.load(filename)

S = librosa.stft(x, N\_FFT)

p = np.angle(S)

S = np.log1p(np.abs(S[:,:430]))

return S, fs

a\_content, fs = read\_audio\_spectum(CONTENT\_FILENAME)

a\_style, fs = read\_audio\_spectum(STYLE\_FILENAME)

N\_SAMPLES = a\_content.shape[1]

N\_CHANNELS = a\_content.shape[0]

a\_style = a\_style[:N\_CHANNELS, :N\_SAMPLES]

N\_FILTERS = 4096

a\_content\_tf = np.ascontiguousarray(a\_content.T[None,None,:,:])

a\_style\_tf = np.ascontiguousarray(a\_style.T[None,None,:,:])

# filter shape is "[filter\_height, filter\_width, in\_channels, out\_channels]"

std = np.sqrt(2) \* np.sqrt(2.0 / ((N\_CHANNELS + N\_FILTERS) \* 11))

kernel = np.random.randn(1, 11, N\_CHANNELS, N\_FILTERS)\*std

g = tf.Graph()

with g.as\_default(), g.device('/cpu:0'), tf.Session() as sess:

# data shape is "[batch, in\_height, in\_width, in\_channels]",

x = tf.placeholder('float32', [1,1,N\_SAMPLES,N\_CHANNELS], name="x")

kernel\_tf = tf.constant(kernel, name="kernel", dtype='float32')

conv = tf.nn.conv2d(

x,

kernel\_tf,

strides=[1, 1, 1, 1],

padding="VALID",

name="conv")

net = tf.nn.relu(conv)

content\_features = net.eval(feed\_dict={x: a\_content\_tf})

style\_features = net.eval(feed\_dict={x: a\_style\_tf})

features = np.reshape(style\_features, (-1, N\_FILTERS))

style\_gram = np.matmul(features.T, features) / N\_SAMPLES

from sys import stderr

ALPHA= 1e-2

learning\_rate= 1e-3

iterations = 100

result = None

with tf.Graph().as\_default():

# Build graph with variable input

#x = tf.Variable(np.zeros([1,1,N\_SAMPLES,N\_CHANNELS], dtype=np.float32), name="x")

x = tf.Variable(np.random.randn(1,1,N\_SAMPLES,N\_CHANNELS).astype(np.float32)\*1e-3, name="x")

kernel\_tf = tf.constant(kernel, name="kernel", dtype='float32')

conv = tf.nn.conv2d(

x,

kernel\_tf,

strides=[1, 1, 1, 1],

padding="VALID",

name="conv")

net = tf.nn.relu(conv)

content\_loss = ALPHA \* 2 \* tf.nn.l2\_loss(

net - content\_features)

style\_loss = 0

\_, height, width, number = map(lambda i: i.value, net.get\_shape())

size = height \* width \* number

feats = tf.reshape(net, (-1, number))

gram = tf.matmul(tf.transpose(feats), feats) / N\_SAMPLES

style\_loss = 2 \* tf.nn.l2\_loss(gram - style\_gram)

# Overall loss

loss = content\_loss + style\_loss

opt = tf.contrib.opt.ScipyOptimizerInterface(

loss, method='L-BFGS-B', options={'maxiter': 300})

# Optimization

with tf.Session() as sess:

sess.run(tf.initialize\_all\_variables())

print('Started optimization.')

opt.minimize(sess)

print ('Final loss:', loss.eval())

result = x.eval()

a = np.zeros\_like(a\_content)

a[:N\_CHANNELS,:] = np.exp(result[0,0].T) - 1

# This code is supposed to do phase reconstruction

p = 2 \* np.pi \* np.random.random\_sample(a.shape) - np.pi

for i in range(500):

S = a \* np.exp(1j\*p)

x = librosa.istft(S)

p = np.angle(librosa.stft(x, N\_FFT))

librosa.output.write\_wav(OUTPUT\_FILENAME, x, fs)

**Приложение 1**

**Список используемой литературы**

* 1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  2. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  3. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  4. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  5. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  6. ГОСТ 19.201-78 Текст программы. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  7. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  8. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |