**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Научный руководитель, старший преподаватель департамента  программной инженерии  факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В. Пантюхин  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. | УТВЕРЖДАЮ  Академический руководитель  ОП «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Шилов «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. |

|  |  |
| --- | --- |
| *Подп. и дата* |  |
| *Инв. № дубл.* |  |
| *Взам. инв. №* |  |
| *Подп. и дата* |  |
| *Инв. № подл* |  |

**Программа прогнозирования интереса к тексту по анализу его содержания**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.07-01 12 01-1**-**ЛУ**

Исполнитель:  
студент группы БПИ198  
\_\_\_\_\_\_\_\_\_\_ / Мелехин Д.A /  
«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**Москва 2020**

УТВЕРЖДЕН

RU.17701729.04.07-01 12 01-1-ЛУ

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

**ПРОГРАММА ПРОГНОЗИРОВАНИЯ ИНТЕРЕСА К ТЕКСТУ ПО АНАЛИЗУ ЕГО СОДЕРЖАНИЯ**

**Текст программы**

**RU.17701729.04.07-01 12 01-1**-**ЛУ**

**Листов 33**

**Москва 2020**

Содержание

[1. ТЕКСТ ПРОГРАММЫ 3](#_Toc40971470)

[1.1. Reader.py 3](#_Toc40971471)

[1.2. NaiveBayes.py 5](#_Toc40971472)

[1.3. LSTM.py 9](#_Toc40971473)

[1.4. SpamClassifier.py 18](#_Toc40971474)

[1.5. Form1.cs 22](#_Toc40971475)

[1.6. Program.cs 28](#_Toc40971476)

[1.7. Classifier.cs 29](#_Toc40971477)

# 1. ТЕКСТ ПРОГРАММЫ

# 1.1. Reader.py

import os

import random

class Data:

def \_\_init\_\_(self):

self.data = None

def Read(self, percent\_test = 0.15):

#1ый элемент - данные для обучения, 2ой элемент - для тестирования

length = 0

answer = []

cant = 0

for part in range(1, 7):

cur\_path\_ham = f'dataset2/enron{part}/ham/'

cur\_path\_spam = f'dataset2/enron{part}/spam/'

for \_, \_, files in os.walk(cur\_path\_ham):

for file\_name in files:

file\_type = 'fam'

with open(cur\_path\_ham + file\_name) as f:

try:

lst = f.read().splitlines()

item = tuple((self.list\_to\_tokens(lst), file\_type))

answer.append(item)

length += 1

except Exception:

cant += 1

for \_, \_, files in os.walk(cur\_path\_spam):

for file\_name in files:

file\_type = 'spam'

with open(cur\_path\_spam + file\_name) as f:

try:

lst = f.read().splitlines()

item = tuple((self.list\_to\_tokens(lst), file\_type))

answer.append(item)

length += 1

except Exception:

cant += 1

random.shuffle(answer)

answer2 = ([], [])

already\_in = 0

percent\_train = 1 - percent\_test

for elem in answer:

if already\_in/length <= percent\_train:

answer2[0].append(elem)

else:

answer2[1].append(elem)

already\_in += 1

len\_train = len(answer2[0])

len\_test = len(answer2[1])

answer = answer2

print(f'Ошибка при чтении {cant} файлов!')

print(f"Просмотрено {length} файлов")

print(f"Добавлено {len(answer[0]) + len(answer[1])} файлов")

self.data = answer

# Оставляет только буквы, приводит к нижнему регистру и возвращает список слов

def list\_to\_tokens(self, lst):

answer = []

for line in lst:

cur\_line = line.lower()

for elem in cur\_line.split(' '):

if elem == 'subject:' or elem == '':

continue

is\_okey = True

for symbol in elem:

if symbol < 'a' or symbol > 'z':

is\_okey = False

break

if is\_okey:

answer.append(elem)

return answer

# 1.2. NaiveBayes.py

from Reader import \*

import math

import json

class Bayes:

def \_\_init\_\_(self, input):

self.input = input

def CheckBayes(self, answers, test\_ind = 1):

if test\_ind != 0 and test\_ind != 1:

raise AttributeError("0 - выборка для тренировки, 1 - для тестирования")

samples = 0

right\_spam = 0

right\_fam = 0

right\_answers = 0

cnt\_spam = 0

cnt\_fam = 0

for index, text in enumerate(self.input.data[test\_ind]):

samples += 1

if answers[index] == text[1]:

right\_answers += 1

if answers[index] == 'spam':

right\_spam += 1

else:

right\_fam += 1

if text[1] == 'spam':

cnt\_spam += 1

else:

cnt\_fam += 1

print(f"Общий результат: {(right\_answers / samples) \* 100 : .2f}%")

print(f"Верно определено информативных текстов: {(right\_spam / cnt\_spam) \* 100 : .2f}%")

print(f"Верно определено не информативных текстов: {(right\_fam / cnt\_fam) \* 100 : .2f}%")

with open('results/Bayes/result.txt', 'w') as f:

f.write(f"Общий результат: {(right\_answers / samples) \* 100 : .2f}%\n")

f.write(f"Верно определено информативных текстов: {(right\_spam / cnt\_spam) \* 100 : .2f}%\n")

f.write(f"Верно определено не информативных текстов: {(right\_fam / cnt\_fam) \* 100 : .2f}%\n")

return right\_answers / samples

def bayes\_vocab(self):

print('Начинаю инициализацию bayes\_vocab.json')

vocab = {}

current = self.input

for row in current.data[0]:

for word in row[0]:

if row[1] == 'spam':

if word in vocab:

vocab[word][2] += 1

vocab[word][0] += 1

else:

vocab[word] = [1, 0, 1]

else:

if word in vocab:

vocab[word][1] += 1

vocab[word][0] += 1

else:

vocab[word] = [1, 1, 0]

print('Записываю bayes\_vocab!')

json.dump(vocab, open('inputs/bayes\_vocab.json', 'w'))

@staticmethod

def train\_bayes():

input = Data()

input.Read()

bayes = Bayes(input)

words = None

try:

# 0 - сколько слово встречалось везде, 1 - сколько слово встречалось в спаме, 2 - сколько слово встречалось не в спаме

words = json.load(open('inputs/bayes\_vocab.json'))

except:

bayes.bayes\_vocab()

words = json.load(open('inputs/bayes\_vocab.json'))

spam\_textes = 0

ham\_textes = 0

all\_textes = 0

# Проходимся по данным для обучения

for text in input.data[0]:

all\_textes += 1

if text[1] == 'spam':

spam\_textes += 1

else:

ham\_textes += 1

answers = []

# 0 - тестирование по выборке для обучение, 1 - для тестов

test\_ind = 1

# Проходимся по тестовым данным

for text in input.data[test\_ind]:

p\_spam = 1

p\_fam = 1

for word in text[0]:

if word in words and words[word][1] != 0 and words[word][2] != 0:

p\_spam += math.log(words[word][2] / words[word][0])

p\_fam += math.log(words[word][1] / words[word][0])

p\_spam \*= spam\_textes / all\_textes

p\_fam \*= ham\_textes / all\_textes

cur\_ans = 'fam'

if p\_spam > p\_fam:

cur\_ans = 'spam'

answers.append(cur\_ans)

bayes.CheckBayes(answers, test\_ind=test\_ind)

if \_\_name\_\_ == '\_\_main\_\_':

Bayes.train\_bayes()

# 1.3. LSTM.py

from Reader import \*

import torch

import torch.nn as nn

from collections import Counter

import numpy as np

import json

class RNN(nn.Module):

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

min\_words\_in\_sentence = 25

def InitTrain\_x(self):

print('Инициализирую train\_x')

temp\_x = []

cnt\_batch = 0

cur2 = []

for row in self.input.data[0]:

cur = []

cnt = 0

for word in row[0]:

cur.append(self.vocab\_to\_int[word])

cnt += 1

if cnt == RNN.min\_words\_in\_sentence:

break

for \_ in range(cnt, RNN.min\_words\_in\_sentence):

cur.append(0)

cur2.append(cur)

cnt\_batch += 1

if cnt\_batch % self.batch\_size == 0:

temp\_x.append(cur2)

cur2 = []

# print(temp\_x.shape)

mem = np.shape(temp\_x)

self.train\_x = torch.LongTensor(temp\_x)

print('inputs/train\_x.pt инициализирован!')

torch.save(self.train\_x, 'inputs/train\_x.pt')

print('inputs/train\_x.pt сохранен в файл!')

def InitTrain\_y(self):

print('Инициализирую train\_y')

temp\_y = []

cnt\_bacth = 0

cur = []

for row in self.input.data[0]:

if row[1] == 'spam':

cur.append(1)

else:

cur.append(0)

cnt\_bacth += 1

if cnt\_bacth % self.batch\_size == 0:

temp\_y.append(cur)

cur = []

self.train\_y = torch.Tensor(temp\_y)

print('inputs/train\_y.pt инициализирован!')

torch.save(self.train\_y, 'inputs/train\_y.pt')

print('inputs/train\_y.pt сохранен в файл!')

def InitTest\_x(self):

print('Инициализирую test\_x')

temp\_x = []

cnt\_batch = 0

cur2 = []

for row in self.input.data[1]:

cur = []

cnt = 0

for word in row[0]:

cur.append(self.vocab\_to\_int[word])

cnt += 1

if cnt == RNN.min\_words\_in\_sentence:

break

for \_ in range(cnt, RNN.min\_words\_in\_sentence):

cur.append(0)

cur2.append(cur)

cnt\_batch += 1

temp\_x.append(cur2)

cur2 = []

self.test\_x = torch.LongTensor(temp\_x)

print('inputs/test\_x.pt инициализирован!')

torch.save(self.test\_x, 'inputs/test\_x.pt')

print('inputs/test\_x.pt сохранен в файл!')

def InitTest\_y(self):

print('Инициализирую test\_y')

temp\_y = []

for row in self.input.data[1]:

if row[1] == 'spam':

temp\_y.append(1)

else:

temp\_y.append(0)

self.test\_y = torch.Tensor(temp\_y)

print('inputs/test\_y.pt инициализирован!')

torch.save(self.test\_y, 'inputs/test\_y.pt')

print('inputs/test\_y.pt сохранен в файл!')

def big\_vocab(self):

print('Начинаю инициализацию словаря')

vocabs = []

for row in self.input.data[0]:

for word in row[0]:

vocabs.append(word)

for row in self.input.data[1]:

for word in row[0]:

vocabs.append(word)

vocab\_count = Counter(vocabs)

vocab\_count = vocab\_count.most\_common(len(vocab\_count))

vocab\_to\_int = {word: index + 2 for index, (word, count) in enumerate(vocab\_count)}

vocab\_to\_int.update({'\_\_PADDING\_\_': 0}) # index 0 for padding

vocab\_to\_int.update({'\_\_UNKNOWN\_\_': 1}) # index 1 for unknown word such as broken character

vocab\_path = 'inputs/big\_vocab.json'

print(f'Сохраняю словарь в файл {vocab\_path}')

json.dump(vocab\_to\_int, open(vocab\_path, 'w'))

def preprocess(self):

vocab\_path = 'inputs/big\_vocab.json'

try:

self.vocab\_to\_int = json.load(open(vocab\_path))

self.vocab\_size = len(self.vocab\_to\_int)

except Exception as ex:

print(f'Ошибка при считывании словаря {vocab\_path} {type(ex)}')

if self.input.data == None:

self.input.Read()

self.big\_vocab()

self.vocab\_to\_int = json.load(open(vocab\_path))

self.vocab\_size = len(self.vocab\_to\_int)

def InitAll(self):

self.preprocess()

try:

self.train\_x = torch.load('inputs/train\_x.pt')

print('Файл inputs/train\_x.pt загружен!')

except Exception as ex:

print('При загрузке файла inputs/train\_x.pt произошла ошибка ', type(ex))

if self.input.data is None:

self.input.Read()

self.InitTrain\_x()

try:

self.train\_y = torch.load('inputs/train\_y.pt')

print('Файл inputs/train\_y.pt загружен!')

except Exception as ex:

print('При загрузке файла inputs/train\_y.pt произошла ошибка ', type(ex))

if self.input.data is None:

self.input.Read()

self.InitTrain\_y()

try:

self.test\_x = torch.load('inputs/test\_x.pt')

print('Файл test\_x.pt2 загружен!')

except Exception as ex:

print('При загрузке файла inputs/test\_x.pt произошла ошибка ', type(ex))

if self.input.data is None:

self.input.Read()

self.InitTest\_x()

try:

self.test\_y = torch.load('inputs/test\_y.pt')

print('Файл inputs/test\_y.pt загружен!')

except Exception as ex:

print('При загрузке файла inputs/test\_y.pt произошла ошибка ', type(ex))

if self.input.data is None:

self.input.Read()

self.InitTest\_y()

def \_\_init\_\_(self, input, test = False):

super().\_\_init\_\_()

self.input = input

self.train\_x = None

self.train\_y = None

self.test\_x = None

self.test\_y = None

self.batch\_size = 80

if test:

self.preprocess()

else:

self.InitAll()

self.hidden\_size = 25

self.embedding\_dim = 100

self.num\_classes = 1

self.embedding = nn.Embedding(self.vocab\_size, self.embedding\_dim)

self.lstm = nn.LSTM(self.embedding\_dim, self.hidden\_size, self.num\_classes, batch\_first=True)

self.linear\_layer = nn.Linear(self.min\_words\_in\_sentence, 1)

self.sigmoid = nn.Sigmoid()

def forward(self, x):

embedded\_seq\_tensor = self.embedding(x)

lstm\_out, hidden = self.lstm(embedded\_seq\_tensor, None)

lstm\_out = self.linear\_layer(lstm\_out[:, :, -1])

lstm\_out = self.sigmoid(lstm\_out)

return lstm\_out, hidden

def test\_on\_test(self, epoch):

print('Начинаю тестирование на тестовом датасете!')

right\_answers = 0

right\_spam = 0

right\_fam = 0

all\_spam = 0

all\_fam = 0

self.test\_x = self.test\_x.to(RNN.device)

self.test\_y = self.test\_y.to(RNN.device)

cnt = 0

for inputs in self.test\_x:

outputs, hidden = self(inputs)

has = 0

if outputs[0].item() >= 0.5:

has = 1

if self.test\_y[cnt].item() == 1:

all\_spam += 1

else:

all\_fam += 1

if has == self.test\_y[cnt].item():

right\_answers += 1

if has == 1:

right\_spam += 1

else:

right\_fam += 1

cnt += 1

with open(f'results/LSTM/epoch{epoch}.txt', 'w') as f:

f.write(f'На тестовом датасете распознано верно {(right\_answers / cnt) \* 100}%\n')

f.write(f'На тестовом датасете информативных текстов верно распознано {(right\_spam / all\_spam) \* 100}%\n')

f.write(f'На тестовом датасете не информативных текстов верно распознано {(right\_fam / all\_fam) \* 100}%\n')

print(f'На тестовом датасете распознано верно {(right\_answers / cnt) \* 100}%')

print(f'На тестовом датасете информативных текстов верно распознано {(right\_spam / all\_spam) \* 100}%')

print(f'На тестовом датасете не информативных текстов верно распознано {(right\_fam / all\_fam) \* 100}%')

@staticmethod

def train\_lstm():

input = Data()

neuro = RNN(input).to(RNN.device)

criterion = nn.BCELoss()

optimizer = torch.optim.Adam(neuro.parameters(), lr=0.03)

neuro.train\_x = neuro.train\_x.to(RNN.device)

neuro.train\_y = neuro.train\_y.to(RNN.device)

print('Starting to train')

neuro.train(True)

scheduler = torch.optim.lr\_scheduler.ReduceLROnPlateau(optimizer, \

mode='min', \

factor=0.5, \

patience=2)

for epoch in range(4):

cnt = 0

all = len(neuro.train\_x)

print(f'Epoch: {epoch}')

scheduler.step(epoch)

for inputs in neuro.train\_x:

outputs, hidden = neuro(inputs)

loss = criterion(outputs, neuro.train\_y[cnt])

optimizer.zero\_grad()

loss.backward()

nn.utils.clip\_grad\_norm\_(neuro.parameters(), 5)

optimizer.step()

cnt += 1

if cnt % 100 == 0:

print(cnt)

neuro.test\_on\_test(epoch)

torch.save(neuro.state\_dict(), f'models/lstm\_model\_epoch{epoch}.pt')

if \_\_name\_\_ == '\_\_main\_\_':

RNN.train\_lstm()

# 1.4. SpamClassifier.py

#!/usr/bin/env python3

import sys

import codecs

import cgi

import argparse

import json

from LSTM import RNN

import torch

import os

import math

sys.stdout = codecs.getwriter("utf-8")(sys.stdout.detach())

print('Content-type: text/plain; charset=utf-8\n\n')

def list\_to\_tokens(lst):

answer = []

for line in lst:

cur\_line = line.lower()

for elem in cur\_line.split(' '):

if elem == 'subject:' or elem == '':

continue

is\_okey = True

for ind, symbol in enumerate(elem):

if symbol < 'a' or symbol > 'z':

if ind == len(elem)-1:

continue

is\_okey = False

break

if is\_okey:

answer.append(elem)

return answer

def LSTM(cur\_input):

inp = None

lst = cur\_input.splitlines()

inp = list\_to\_tokens(lst)

vocab\_to\_int = None

try:

vocab\_to\_int = json.load(open('inputs/big\_vocab.json'))

except Exception:

wasExcept = 'Ошибка при чтении словаря inputs/big\_vocab.json'

print(wasExcept)

return wasExcept

def InitTest\_x(inp):

temp\_x = []

cur = []

cnt = 0

for word in inp:

if word in vocab\_to\_int:

cur.append(vocab\_to\_int[word])

cnt += 1

if cnt == RNN.min\_words\_in\_sentence: # min\_words\_in\_sentence

break

for \_ in range(cnt, RNN.min\_words\_in\_sentence):

cur.append(0)

temp\_x.append(cur)

# print(temp\_x.shape)

test\_x = torch.LongTensor(temp\_x)

return test\_x

test\_x = InitTest\_x(inp)

BIG\_MACHINE = None

try:

BIG\_MACHINE = RNN(inp, test = True)

BIG\_MACHINE.load\_state\_dict(torch.load('model/lstm\_model.pt'))

except Exception as ex:

wasExcept = 'Ошибка при чтении модели model/lstm\_model.pt'

print(wasExcept)

return wasExcept

machine\_ans = BIG\_MACHINE(test\_x)

if machine\_ans[0] >= 0.5:

machine\_ans = 'Похоже на не информативный текст'

else:

machine\_ans = 'Похоже на информативный текст!'

return machine\_ans

def bayes\_predict\_single(cur\_input):

lst = cur\_input.splitlines()

text = list\_to\_tokens(lst)

words = None

try:

words = json.load(open('inputs/bayes\_vocab.json'))

except Exception as ex:

return 'Ошибка при чтении словаря для Байеса'

p\_spam = 0

p\_ham = 0

for word in text:

if word in words and words[word][1] != 0 and words[word][2] != 0:

p\_spam += math.log(words[word][2] / words[word][0])

p\_ham += math.log(words[word][1] / words[word][0])

ans = 'Похоже на информативный текст!'

if p\_spam > p\_ham:

ans = 'Похоже на не информативный текст'

return ans

def run():

output\_path1 = 'UserOutput/LstmOutput.txt'

output\_path2 = 'UserOutput/BayesOutput.txt'

cur\_input = None

form = cgi.FieldStorage()

if "input" in form:

cur\_input = form["input"].value

else:

print("Did not get input, try again!")

return

print(f"LSTM: {cur\_input[:10]}...: {LSTM(cur\_input)}")

cur\_ans = bayes\_predict\_single(cur\_input)

print(f"Bayes: {cur\_input[:10]}...: {cur\_ans}")

run()

# 1.5. Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using ClassifierLibrary;

using System.Security;

using System.Diagnostics.Tracing;

namespace MyCoursework

{

public partial class Form1 : Form

{

private TextBox outputBox;

private TextBox inputBox;

private Button recognize;

private Classifier bot;

private OpenFileDialog openFileDialog1;

private Button selectButton;

private TextBox fileBox;

private ToolStripContainer toolStripContainer;

private ToolStrip toolStrip;

private ToolStripButton aboutButton = new ToolStripButton();

private ToolStripButton information = new ToolStripButton();

private void RecognizeClicked(object sender, System.EventArgs e)

{

this.outputBox.Text = "Выполнение...";

string input = this.inputBox.Text;

bot = new Classifier();

string answer = bot.GetAnswer(input);

this.outputBox.Text = answer;

}

private void SelectClicked(object sender, System.EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

string path = null;

try

{

this.fileBox.Text = openFileDialog1.FileName;

path = openFileDialog1.FileName;

this.inputBox.Text = File.ReadAllText(path);

}

catch (SecurityException ex)

{

MessageBox.Show($"Security error.\n\nError message: {ex.Message}\n\n" +

$"Details:\n\n{ex.StackTrace}");

} catch(Exception ex)

{

MessageBox.Show($"При считывании файла {path} произошла ошибка: " +

Environment.NewLine + ex.Message);

}

}

}

private void AboutClicked(object sender, System.EventArgs e)

{

MessageBox.Show("Данное приложение было создано Мелехиным Денисом в 2020 году и" +

" предназначено для демонстрации работы " +

"модели LSTM сети и наивного Байесовского класификатора на примере распознования " +

"информативности текста.");

}

private void InformationClicked(object sender, System.EventArgs e)

{

MessageBox.Show("Вы можете задать программе входные данные двумя способами:" + Environment.NewLine

+ "1) Ввести текст на английском в поле для входных данных и нажать кнопку \"Проверить\""

+ Environment.NewLine + "2) Выбрать доступный файл с помощью кнопки \"Выбрать файл\""

+ " и нажать на кнопку \"Проверить\""

+ Environment.NewLine + "Результат будет выведен в поле для выходных данных. "

+ Environment.NewLine + "Анализ информативности текста проводился двумя методами: "

+ Environment.NewLine + "1) LSTM сетью"

+ Environment.NewLine + "2) Наивным Байесовским классификатором");

}

private void inputBox\_Enter(object sender, EventArgs e) //происходит когда элемент стает активным

{

this.inputBox.Text = null;

this.inputBox.ForeColor = Color.Black;

}

public void InitializeComponent2()

{

this.outputBox = new System.Windows.Forms.TextBox();

this.inputBox = new System.Windows.Forms.TextBox();

this.recognize = new Button();

this.selectButton = new Button();

this.fileBox = new System.Windows.Forms.TextBox();

this.SuspendLayout();

//

// outputBox

//

this.outputBox.ReadOnly = true;

this.outputBox.Multiline = true;

this.outputBox.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;

this.outputBox.Location = new Point(650, 120);

this.outputBox.Size = new System.Drawing.Size(350, 500);

this.outputBox.Text = "Результат:";

//

// inputBox

//

this.inputBox.ReadOnly = false;

this.inputBox.Multiline = true;

this.inputBox.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;

this.inputBox.Location = new Point(50, 120);

this.inputBox.Size = new System.Drawing.Size(550, 500);

this.inputBox.Text = "Введите текст для распознавания его информативности";

this.inputBox.ForeColor = Color.Gray;

this.inputBox.Enter += inputBox\_Enter;

//

// recognizeButton

//

this.recognize.Text = "Проверить";

this.recognize.Location = new Point(50, 70);

this.recognize.Size = new System.Drawing.Size(100, 22);

this.recognize.Click += RecognizeClicked;

//

//fileDialog

//

this.openFileDialog1 = new OpenFileDialog();

this.openFileDialog1.ValidateNames = false;

this.openFileDialog1.CheckFileExists = true;

this.openFileDialog1.CheckPathExists = true;

this.openFileDialog1.Filter = "Text files (\*.txt)|\*.txt";

//

//selectButton

//

this.selectButton.Text = "Загрузить из файла";

this.selectButton.Location = new Point(460, 39);

this.selectButton.Size = new System.Drawing.Size(140, 22);

this.selectButton.Click += SelectClicked;

//

//fileBox

//

this.fileBox.ReadOnly = true;

this.fileBox.Location = new Point(50, 40);

this.fileBox.Size = new System.Drawing.Size(390, 45);

//

//aboutButton

//

this.aboutButton = new ToolStripButton();

aboutButton.Text = "О программе";

aboutButton.Size = new System.Drawing.Size(40, 15);

aboutButton.Click += AboutClicked;

//

//informationButton

//

this.information = new ToolStripButton();

information.Text = "Справка";

information.Size = new System.Drawing.Size(40, 15);

information.Click += InformationClicked;

//

//ToolStrip

//

this.toolStripContainer = new System.Windows.Forms.ToolStripContainer();

this.toolStrip = new System.Windows.Forms.ToolStrip();

// Add items to the ToolStrip.

this.toolStrip.Items.Add(aboutButton);

this.toolStrip.Items.Add(information);

this.toolStrip.CanOverflow = false;

// Add the ToolStrip to the top panel of the ToolStripContainer.

this.toolStripContainer.Size = new System.Drawing.Size(160, 25);

this.toolStripContainer.TopToolStripPanel.Controls.Add(this.toolStrip);

// Add the ToolStripContainer to the form.

Controls.Add(this.toolStripContainer);

//

// Form1

//

this.ClientSize = new System.Drawing.Size(1050, 650);

this.Controls.Add(this.outputBox);

this.Controls.Add(this.inputBox);

this.Controls.Add(this.recognize);

this.Controls.Add(this.selectButton);

this.Controls.Add(this.fileBox);

this.Text = "Coursework";

this.ResumeLayout(false);

this.PerformLayout();

}

public Form1()

{

InitializeComponent2();

}

}

}

# 1.6. Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MyCoursework

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

# 1.7. Classifier.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Diagnostics;

using System.Net;

namespace ClassifierLibrary

{

public class Classifier

{

public string GetInput(string path)

{

try

{

string answer = File.ReadAllText(path);

return answer;

} catch(FileNotFoundException)

{

throw new FileNotFoundException("Не найден файл с выходными данными!");

} catch(Exception exc)

{

throw exc;

}

}

public string GetAnswer(string input)

{

try

{

WebRequest request = WebRequest.Create("http://denis.hiweb.ru/cgi-bin/SpamClassifier.py");

var data = Encoding.ASCII.GetBytes($"input={input}");

request.ContentType = "application/x-www-form-urlencoded";

request.ContentLength = data.Length;

request.Method = "POST";

using (var stream = request.GetRequestStream())

{

stream.Write(data, 0, data.Length);

}

WebResponse response = request.GetResponse();

using (Stream dataStream = response.GetResponseStream())

{

// Open the stream using a StreamReader for easy access.

StreamReader reader = new StreamReader(dataStream);

// Read the content.

string responseFromServer = reader.ReadToEnd();

// Display the content.

Console.WriteLine(responseFromServer);

string response2 = "";

for(int i = 0; i < responseFromServer.Length; i++)

{

if(responseFromServer[i] == '\n')

{

response2 += Environment.NewLine;

} else

{

response2 += responseFromServer[i];

}

}

return response2;

}

} catch(Exception ex)

{

return $"Ошибка при обращении к серверу: {ex.Message}";

}

}

}

}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц) в докум. | № документа | Входящий номер сопроводи-тельного докум. и дата | Под-пись | Дата |
| Изм. | изменен-ных | заменен-ных | новых | аннулиро-ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**