**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**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.07-01 81 01-1**-**ЛУ**

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

**Москва 2020**

УТВЕРЖДЕН

RU.17701729.04.07-01 81 01-1-ЛУ

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

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

**Пояснительная записка**

**RU.17701729.04.07-01 81 01-1-ЛУ**

**Листов 17**

**Москва 2020**

Содержание

[1. ВВЕДЕНИЕ 3](#_Toc40971548)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 3](#_Toc40971549)

[2.1. Назначение программы 3](#_Toc40971550)

[2.2. Краткая характеристика области применения программы 3](#_Toc40971551)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 4](#_Toc40971552)

[3.1. Постановка задачи на разработку программы 4](#_Toc40971553)

[3.2. Описание алгоритма функционирования программы 4](#_Toc40971554)

[3.2.1. Описание алгоритма функционирования программы создания и обучения модели 4](#_Toc40971555)

[3.2.1.1. Описание алгоритма функционирования наивного байесовского классификатора 5](#_Toc40971556)

[3.2.1.2. Описание алгоритма функционирования LSTM сети 6](#_Toc40971557)

[3.2.2. Описание алгоритма функционирования программы, демонстрирующей работу модели 9](#_Toc40971558)

[3.3. Описание и обоснование метода организации входных и выходных данных 11](#_Toc40971559)

[3.4. Описание и обоснование метода выбора технических и программных средств 11](#_Toc40971560)

[4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 11](#_Toc40971561)

[5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 11](#_Toc40971562)

[Приложение 1 13](#_Toc40971563)

[Описание классов, полей и методов программы создания и обучения модели 13](#_Toc40971564)

[Приложение 2: описание классов, полей и методов программы, демонстрирующей работу модели 15](#_Toc40971565)

# 1. ВВЕДЕНИЕ

Наименование программы: «Программа прогнозирования интереса к тексту по анализу его содержания».

Основание для разработки: Приказ декана факультета компьютерных наук И.В. Аржанцева "Об утверждении тем, руководителей курсовых работ студентов образовательной программы «Программная инженерия» факультета компьютерных наук" № 2.3-02/1112-04 от 11.12.2019

# 2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

# 2.1. Назначение программы

Основным назначением программы является прогнозирование интереса к тексту по анализу его содержания.

# 2.2. Краткая характеристика области применения программы

Данная программа предназначена для прогнозирования интереса к тексту по анализу его содержания. Данная программа будет применяться в образовательных целях: демонстрация обучающимся возможностей машинного обучения.

# 3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

# 3.1. Постановка задачи на разработку программы

Разработать программу в рамках курсовой работы. Тема курсовой работы: «Программа прогнозирования интереса к тексту по анализу его содержания».

Условное обозначение темы разработки –«Software for Text Likability Prediction Based on Its Content».

# 3.2. Описание алгоритма функционирования программы

Программа, состоит из двух подпрограмм:

1) Программа создания и обучения модели;

2) Программа, демонстрирующая работу модели.

# 3.2.1. Описание алгоритма функционирования программы создания и обучения модели

Данная программа реализована на языке python. Цель программы: присуждение входному тексту одной из двух категорий: “Информативный текст”, “Не информативный текст”. Программа была реализована двумя различными алгоритмами:

1. Наивный байесовский классификатор;
2. Нейронная LSTM (Long short-term memory, долгая краткосрочная память) сеть.

Для реализации каждого из алгоритмов необходимы данные для обучения. Для обучения использован открытый набор данных (датасет) доступный по ссылке: <http://nlp.cs.aueb.gr/software_and_datasets/Enron-Spam/index.html>

Датасет представляет собой набор из 6 папок, в каждой из которых находится 2 папки: с информативным текстом и не информативным текстом. Данный датасет уже имел одно преобразование: знаки препинания выделены пробелами.

После считывания программа разделяет датасет на 2 части: для обучения и для тестирования. Для обучения выбирается 85% входных данных, а для тестирования – 15%. При считывании каждый текст разбивается на массив отдельных слов для передачи в модуль обучения.

# 3.2.1.1. Описание алгоритма функционирования наивного байесовского классификатора

Наивный байесовский классификатор изначально производит считывание входных данных, после чего для каждого слова считает количество его вхождений в информативные и не информативные тексты. После чего данная статистика записывается в файл bayes\_vocab.json. На этом процесс обучения наивного байесовского классификаторы завершается. Опишем процесс тестирования:

Используя формулу Байеса:

**A close up of a logo

Description automatically generated**

<https://studfile.net/preview/6294750/>

Программа считает вероятность того, что данный текст информативный.

B = набор слов bi, 1 <= i <= n

P(A|B) = вероятность, что данный текст информативный;

P(bi|A) = вероятность, что текст содержащий слова bi информативный

P(B|A) = P(b1|A) \* P(b2|A) \* … \* P(bn|A)

P(A) = вероятность что текст информативный (вне зависимости от его текста). Т.е. кол-во информативных текстов / кол-во всех текстов.

Вероятность, что данный текст не информативный считается аналогично.

Если вероятность того, что данный текст информативный, больше, то программа определяет данный текст информативным, иначе не информативным.

Значение P(B|A), при наличии большого количества слов в тексте, становится настолько близким к нулю, что при вычислениях теряется точность. А так как для определения является текст информативным или не информативным программа только сравнивает вероятности, то можно P(B|A) считать по такой формуле:

P(B|A) = ln(P(b1|A)) + ln(P(b2|A)) + … + ln(P(bn|A))

Где ln – натуральный логарифм. Так как для логарифма выполняется свойство ln(a\*b) = ln(a) + ln(b) и, так как логарифм является непрерывной функцией, то для сравнения мы можем использовать формулу с логарифмом.

Далее программа для каждого текста из тестового набора данных определяет, является ли текст информативным и подсчитывает статистику правильных ответов. Затем выводит статистику в консоль и в файл result.txt.

При обучении и тестировании на оговоренном раннее датасете наивный байесовский классификатор достиг результата в 88.34% правильных ответов на тестовой части датасета. Более детальный результат:

Общий результат: 85.74%

Верно определено информативных текстов: 71.69%

Верно определено не информативных текстов: 99.71%

# 3.2.1.2. Описание алгоритма функционирования LSTM сети

Изначально программа считывает входные данные, после чего, преобразует входные данные в тензоры. Это нужно для работы нейронной сети на библиотеке “pytorch”. Для преобразования в тензоры необходимо, чтобы размерность каждого измерения массива была фиксированной. Опишем преобразование в тензор части датасета для обучения:

Программа получает на вход двумерный массив (так как при считывании каждый текст сразу разбивается на список слов). После чего мы из каждого текста оставляем только первые min\_words\_in\_sentence = 25 слов (каждый константный параметр был подобран практическим путем и с заданными параметрами нейронная сеть выдавала наилучшие результаты). После этих преобразований набор данных разбивается на пакеты (батчи, по batch\_size = 80 текстов в каждом батче). Далее программа составляет словарь всех слов из всего датасета. В словаре ключом будет само слово, а значением – количество упоминаний данного слова во всем датасете. Так как датасет большой, то почти каждому слову сопоставляется уникальное значение. После создания словаря программа заменяет каждое слово в входных данных на его значение в словаре. На этом преобразования входных данных заканчиваются.

#### Устройство LSTM сети в программе:

1. Входной Embedding слой:

Слой реализован в модуле nn.Embedding библиотеки pytorch. Данный слой необходим для преобразования каждого слова в вектор. Он принимает на вход вектор, размером с словарь, где на месте значения слова в словаре стоит 1, а на остальных местах нули. Выход этого слоя состоит из 100 нейронов (embedding\_dim), в каждом из которых будет некоторое значение. Вектор из этих выходных значений – это новое векторное представление слова в нейросети. Данный вектор получается для каждого слова и затем каждое слово заменяется этим вектором.

Более детальное описание структуры слоя в [7].

1. LSTM слой:

Данный слой является основным в LSTM сети. Он реализован в модуле nn.lstm библиотеки “pytorch”. В программе данный слой представляет собой блок нейронов с рекуррентными связями. Структура нейрона в LSTM слое представлена на схеме

A picture containing clock, game

Description automatically generated

Рисунок [1][[1]](#footnote-1)

Всего таких нейронов 25 в блоке (hidden\_size).

На вход lstm слой получает выход из Embedding слоя размерностью <batch\_size, min\_words\_in\_sentence, embedding\_dim>. В нашем случае это трехмерный вектор размером <80, 25, 100>.

Выход из lstm слоя имеет размерность <batch\_size, min\_words\_in\_sentence, hidden\_size>. В нашем случае это трехмерный вектор размером <80, 25, 25>.

Более подробное описание принципов работы lstm слоя находится в [8].

1. Слой линейного преобразования:

Данный слой необходим для преобразования выхода lstm слоя. Он принимает выход последнего нейрона lstm слоя размерностью min\_words\_in\_sentence, после чего преобразует данный вектор в скаляр.

Более подробное описание линейного слоя в [9].

1. Сигмоидальное преобразование:

Данный слой необходим для представления выходного скаляра из предыдущего слоя в вероятность принадлежности информативному или не информативному тексту. Функция, применяемая к каждому итоговому значению каждого батча:

A close up of a logo

Description automatically generated

Более подробное описание в [10].

#### Процесс обучения сети:

Обучение производилось оптимизатором из модуля torch.optim.Adam. Обучение на входных данных в программе происходило 4 раза (количество эпох). Для подсчета функции ошибки использовалась бинарная кросс-энтропия (модуль nn.BCELoss). Для регулировки скорости использовался планировщик torch.optim.lr\_scheduler.ReduceLROnPlateau.

Адам – это современный алгоритм, используемый в области глубокого обучения, который быстро достигает отличных результатов. Подробное описание работы оптимайзера в [11].

Функция бинарной кросс-энтропии используется для подсчета ошибки в задачах, в которых необходимо распределить входные данные на 2 категории. Данная функция принимает на вход два числа из отрезка [0;1] и возвращает число из этого же отрезка. Подробное описание функции ошибки в [12].

Планировщик в нейронной сети необходим для установки более подходящей скорости обучения для каждой эпохи. Он позволяет нейросети обучаться медленнее в конце обучения. Подробное описание torch.optim.lr\_scheduler.ReduceLROnPlateau в [13].

Результаты тестирования на тестовом датасете LSTM нейросети:

На тестовом датасете распознано верно 97.81%

На тестовом датасете информативных текстов верно распознано 97.74%

На тестовом датасете не информативных текстов верно распознано 97.87%

# 3.2.2. Описание алгоритма функционирования программы, демонстрирующей работу модели

Данная программа была реализована на языке C#. Интерфейс программы был реализован на windows forms. Модуль анализа данных размещен на сервере.

Алгоритм функционирования программы:

1. Получение входных данных от пользователя

Программа позволяет пользователю ввести входные данные двумя способами: в текстовом поле или выбрать файл с разрешением .txt. Текст следует вводить на английском языке, так как модели были обучены на тренировочных данных, тексты которых на английском языке.

1. Отправка входных данных на сервер и получение ответа

После получения входных данных программа отправляет данные на сервер. Модуль, принимающий POST запрос, находится по ссылке:

<http://denis.hiweb.ru/cgi-bin/SpamClassifier.py>

Реализация POST запроса производилась с помощью библиотеки WebRequest, а получение ответ с помощью библиотеки WebResponse.

1. Отображение результатов

После получения ответа от сервера в поле для выходных данных выводится ответ от сервера или сообщение об ошибке, если такая произошла при обращении к серверу.

Рисунки, демонстрирующие работу программы:

A screenshot of a cell phone

Description automatically generatedРисунок [2]

A screenshot of a cell phone

Description automatically generatedРисунок [3]

# 3.3. Описание и обоснование метода организации входных и выходных данных

Программа на вход получает либо текст, напечатанный в приложении пользователем, либо файл формата .txt, с разрешенным доступом к этому файлу.

# 3.4. Описание и обоснование метода выбора технических и программных средств

Для корректной работы программы рекомендуется компьютер, имеющий следующие   
технические характеристики:

1. Процессор c частотой 1 ГГц или более;
2. Оперативная память 1 Гб или более;
3. Дисковое пространство размером 32 Гб или более;
4. Видеокарта с поддержкой Microsoft DirectX 9 с драйвером WDDM
5. Монитор с разрешением 1920\*1080 или более высоким;
6. Функционирующая мышь и клавиатура или их аналоги;
7. Свободный доступ в интернет.

Требования к программным средствам, используемым программой:

1. операционная система Windows 10 Fall Creators Update (версия 1709);
2. установленный Microsoft .NET Framework 4.7.2 .

# 4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

В рамках данной работы расчет экономической эффективности не предусмотрен.

# 5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Модуль Pytorch [Электронный ресурс]// URL: <https://pytorch.org> (Дата обращения: 09.05.2020, режим доступа: свободный).
2. Word embeddings tutorial [Электронный ресурс]// URL: <https://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html> (Дата обращения: 09.05.2020, режим доступа: свободный).
3. Spam-Ham Classification [Электронный ресурс]// URL: <https://medium.com/analytics-vidhya/spam-ham-classification-using-lstm-in-pytorch-950daec94a7c> (Дата обращения: 09.05.2020, режим доступа: свободный).
4. Документация языка C# [Электронный ресурс]// URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> (Дата обращения: 09.05.2020, режим доступа: свободный).
5. Модуль json [Электронный ресурс]// URL: <https://docs.python.org/3/library/json.html> (Дата обращения: 09.05.2020, режим доступа: свободный).
6. Python [Электронный ресурс]// URL: <https://www.python.org> (Дата обращения: 09.05.2020, режим доступа: свободный).
7. Модель Embeddings [Электронный ресурс]// URL: <https://pytorch.org/docs/stable/nn.html#torch.nn.Embedding> (Дата обращения: 09.05.2020, режим доступа: свободный).
8. Модель LSTM [Электронный ресурс]// URL: https://habr.com/ru/company/wunderfund/blog/331310/ (Дата обращения: 09.05.2020, режим доступа: свободный).
9. Описание линейного слоя [Электронный ресурс]// URL: <https://pytorch.org/docs/stable/nn.html#linear> (Дата обращения: 09.05.2020, режим доступа: свободный).
10. Описание сигмоидального слоя [Электронный ресурс]// URL: <https://pytorch.org/docs/stable/nn.html#sigmoid> (Дата обращения: 09.05.2020, режим доступа: свободный).
11. Описание оптимизатора [Электронный ресурс]// URL: <https://pytorch.org/docs/stable/optim.html#torch.optim.Adam> (Дата обращения: 09.05.2020, режим доступа: свободный).
12. Описание функции ошибки BCELoss [Электронный ресурс]// URL: <https://pytorch.org/docs/stable/nn.html#bceloss> (Дата обращения: 09.05.2020, режим доступа: свободный).
13. Описание sheduler [Электронный ресурс]// URL: <https://pytorch.org/docs/stable/optim.html#torch.optim.lr_scheduler.ReduceLROnPlateau> (Дата обращения: 09.05.2020, режим доступа: свободный).
14. Требования .Net Framework [Электронный ресурс]// URL: <https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/system-requirements> (Дата обращения: 09.05.2020, режим доступа: свободный).

# Приложение 1

# Описание классов, полей и методов программы создания и обучения модели

**Описание классов**

|  |  |
| --- | --- |
| **Класс** | **Описание класса** |
| [Data](file:///D:\STUDY\%D0%9B%D0%B5%D1%88%D0%B0\Help\Help\html\3eea08d3-38a7-9cc8-6ae6-45d47bd47c1b.htm) | Класс, считывающий датасет и хранящий его в удобном формате |
| RNN | Класс, реализующий lstm нейросеть |
| Bayes | Класс, реализующий наивный Байесовский классификатор |

# 

**Описание методов класса Data**

|  |  |
| --- | --- |
| **Метод** | **Описание метода** |
| \_\_init\_\_ | Конструктор класса, инициализирующий нейтральными значениями поля класса |
| Read | Метод, реализующий считывание датасета |
| list\_to\_tokens | Метод, разбивающий текст на отдельные слова |

**Описание методов класса RNN**

|  |  |
| --- | --- |
| **Метод** | **Описание метода** |
| \_\_init\_\_ | Конструктор класса, инициализирующий поля класса |
| big\_vocab | Метод, создающий словарь на основе тренировочной части датасета.  Ключ словаря – любое слово из тренировочной части датасета  Значение словаря – количество упоминаний данного слова в тренировочной части датасета |
| InitTrain\_x | Метод выполняет следующую предобработку тренировочной части датасета:   1. Оставляет у каждого текста только первые 25 слов. 2. Заменяет слова на их значения в словаре 3. Распределяет входные данные на блоки, в каждом из которых 80 текстов (создает батчи) 4. Записывает полученный тензор в файл |
| InitTest\_x | Метод выполняет следующую предобработку тестовой части датасета:   1. Оставляет у каждого текста только первые 25 слов. 2. Заменяет слова на их значения в словаре 3. Записывает полученный тензор в файл |
| InitTrain\_y | Метод выполняет следующую предобработку ответов тренировочной части датасета:   1. Преобразует ответы в числа 1 и 0. Где 1 – не информативный текст, 0 – информативный текст 2. Распределяет входные данные на блоки, в каждом из которых 80 ответов (создает батчи) 3. Записывает полученный тензор в файл |
| InitTest\_y | Метод выполняет следующую предобработку ответов тестовой части датасета:   1. Преобразует ответы в числа 1 и 0. Где 1 – не информативный текст, 0 – информативный текст 2. Записывает полученный тензор в файл |
| preprocess | Метод загружает словарь из файла и в случае неудачи вызывает метод big\_vocab для создания словаря |
| InitAll | Метод пытается считать тензоры из файла и в случае неудачи вызывает метод, соответствующий не найденному тензору, для его инициализации |
| test\_on\_test | Метод выполняет тестирование модели класса на тестовой части датасета, выводит результат на экран и записывает результат в файл. |
| forward | Метод, вызывающийся при пропуске входных данных через нейросеть |
| train\_lstm | Статический метод класса, в котором обучается нейросеть. После обучения нейросеть тестируется на тестовой части датасета. |

**Описание методов класса Bayes**

|  |  |
| --- | --- |
| **Метод** | **Описание метода** |
| \_\_init\_\_ | Конструктор класса, инициализирующий поля класса |
| bayes\_vocab | Метод, создающий и записывающий в файл словарь для наивного Байесовского классификатора |
| CheckBayes | Метод, считающий статистику верных ответов наивного Байесовского классификатора |
| train\_bayes | Метод, тренирующий наивный Байесовский классификатор и затем тестирующий его |

**Описание полей класса Data**

|  |  |
| --- | --- |
| **Поле** | **Описание поля** |
| data | В данном поле хранится датасет |

**Описание полей класса RNN**

|  |  |
| --- | --- |
| **Поле** | **Описание поля** |
| input | Поле для хранения датасета |
| train\_x | Поле для хранения тренировочной части датасета |
| train\_y | Поле для хранения ответов тренировочной части датасета |
| test\_x | Поле для хранения тестовой части датасета |
| test\_y | Поле для хранения ответов тестовой части датасета |
| device | Название устройство, на котором будет происходить обучение нейросети |
| min\_words\_in\_sentence | Константа – количество слов в каждом тексте после преобразований |
| batch\_size | Константа – размер батча |
| hidden\_size | Константа – количество нейронов в LSTM слое |
| num\_classes | Константа – на сколько классов мы хотим разделить входной текст |
| embeddin\_dim | Константа – размерность вектора для представления слова |
| embedding | Слой нейронной сети для представления слов вектором |
| lstm | LSTM слой нейронной сети |
| linear\_layer | Линейный слой нейронный сети для преобразования выхода LSTM слоя |
| criterion | Поле, содержащее метод для подсчета ошибки нейросети |
| optimizer | Поле, содержащее метод для выполняющий изменение весов в нейросети |
| list\_to\_tokens | Метод, разбивающий текст на отдельные слова |
| scheduler | Поле, содержащий метод регулировки скорости обучения |

**Описание полей класса Bayes**

|  |  |
| --- | --- |
| **Поле** | **Описание поля** |
| input | В данном поле хранится датасет |
| words | В данном поле хранится словарь для наивного Байесовского классификатора |
| answers | В данном поле хранятся ответа наивного Байесовского классификатора для тестовой части датасета |

# Приложение 2: описание классов, полей и методов программы, демонстрирующей работу модели

**Описание классов**

|  |  |
| --- | --- |
| **Класс** | **Описание класса** |
| Classifier | Класс, реализующий считывание входных данных из файла и получающий результат нейросети для этих входных данных с сервера |
| Form1 | Класс, описывающий и реализующий интерфейс приложения |

**Описание методов класса Classifier**

|  |  |
| --- | --- |
| **Метод** | **Описание метода** |
| GetInput | Считывает входные данные из файла |
| GetAnswer | Передает серверу входные данные, получает ответ от сервера и возвращает это значение |

**Описание методов класса Form1**

|  |  |
| --- | --- |
| **Метод** | **Описание метода** |
| RecognizeClicked | Метод, вызывающийся при нажатии на кнопку “проверить”. После чего вызывает метод GetAnswer класса Classifier и выводит в поле для вывод значение, что вернул GetAnswer |
| SelectClicked | Метод, вызывающийся при выборе файла. Загружает данные из файла в поле для ввода. |
| AboutClicked | Метод, вызывающийся при нажатии кнопки “О программе”. Выводит информацию о программе |
| InformationClicked | Метод, вызывающийся при нажатии кнопки “Справка”. Выводит справку о программе |
| InitializeComponent2 | Инициализирует компоненты и добавляет их в форму |
| Form1 | Конструктор, запускающий InitializeComponent2 |

**Описание полей класса Form1**

|  |  |
| --- | --- |
| **Поле** | **Описание поля** |
| outputBox | TextBox для отображения результата работы программы |
| inputBox | TextBox для отображения ввода пользователя |
| recognize | Кнопка, при нажатии на которую программа определяет тип входных данных (информативный или не информативный текст) |
| bot | Экземпляр класса Classifier. Через это поле происходит обработка входных данных |
| openFileDialog1 | Поле, с помощью которого реализуется окно для выбора файла пользователем |
| selectButton | Кнопка, открывающее окно для выбора файла пользователем |
| fileBox | Поле, в котором хранится путь до файла, выбранного пользователем |
| aboutButton | Кнопка, при нажатии на которую отображается информация о программе |
| information | Кнопка, при нажатии на которую отображается справка о программе |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

1. Источник рисунка [1]: https://en.wikipedia.org/wiki/Long\_short-term\_memory [↑](#footnote-ref-1)