МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«КУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет физики, математики, информатики

Кафедра программного обеспечения и администрирования информационных систем

КУРСОВОЙ ПРОЕКТ

по дисциплине

«Основы проектирования информационных систем и баз данных»

на тему: ИНФОРМАЦИОННАЯ СИСТЕМА ДЛЯ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ОПРЕДЕЛЕНИЯ ОТНОШЕНИЙ МЕЖДУ ПОЛЬЗОВАТЕЛЯМИ СОЦИАНЫХ СЕТЕЙ НА ОСНОВЕ АНАЛИЗА КОРОТКИХ ТЕКСТОВЫХ СООБЩЕНИЙ

Обучающегося 4 курса

очной формы обучения

направления подготовки

02.03.03 Математическое обеспечение и администрирование информационных систем

направленности (профиля)

Проектирование информационных систем и баз данных

Золотарева Ивана Юрьевича

Руководитель: к.т.н., профессор кафедры ПОиАИС

Бабкин Евгений Александрович

Допустить к защите:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Бабкин Е.А./

«24» декабря 2021 г.

Курск, 2021

содержание

[ВВЕДЕНИЕ 4](#_Toc102175359)

[1 Анализ требований к информационной системе 6](#_Toc102175360)

[1.1 Описание и анализ предметной области 6](#_Toc102175361)

[1.2 Обзор и анализ возможных альтернатив 7](#_Toc102175362)

[1.3 Анализ функциональных и эксплуатационных требований 8](#_Toc102175363)

[1.3.1 Стандарты 8](#_Toc102175364)

[1.3.2 Функциональные требования пользователя 9](#_Toc102175365)

[1.3.3 Входные данные 9](#_Toc102175366)

[1.3.4 Выходные данные 9](#_Toc102175367)

[1.3.5 Требования к интерфейсу 10](#_Toc102175368)

[1.3.6 Требования к надежности 10](#_Toc102175369)

[1.3.7 Требования к программной документации 10](#_Toc102175370)

[1.3.8 Требования к составу и параметрам технических средств 11](#_Toc102175371)

[1.3.9 Модель вариантов использования 11](#_Toc102175372)

[1.3.10 Глоссарий проекта 15](#_Toc102175373)

[1.3.11 Проверка модели на полноту 16](#_Toc102175374)

[2 Проектирование информационной системы 19](#_Toc102175375)

[2.1 Разработка архитектуры системы 19](#_Toc102175376)

[2.2 Разработка модели предметной области 20](#_Toc102175377)

[2.3 Разработка алгоритма функционирования системы 23](#_Toc102175378)

[2.4 Проектирование интерфейса пользователя 23](#_Toc102175379)

[2.5 Реляционная модель данных 24](#_Toc102175380)

[2.6 Проектирование классов предметной области 25](#_Toc102175381)

[2.6.1 Построение диаграмм последовательностей для варианта использования «Выбор операции над документом» 25](#_Toc102175382)

[2.6.2 Построение диаграммы кооперации 26](#_Toc102175383)

[2.6.3 Построение диаграммы классов 27](#_Toc102175384)

[2.6.4 Уточнение структуры классов предметной области и разработка алгоритмов методов 28](#_Toc102175385)

[2.7 Модель нейронной сети 29](#_Toc102175386)

[3 Реализация системы 30](#_Toc102175387)

[3.1 Реализация программного обеспечения системы 30](#_Toc102175388)

[3.1.1 Разработка диаграммы компонентов 30](#_Toc102175389)

[3.1.2 Объекты интерфейса пользователя 31](#_Toc102175390)

[3.2 Реализация технического обеспечения 33](#_Toc102175391)

[ЗАКЛЮЧЕНИЕ 34](#_Toc102175392)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc102175393)

[ПРИЛОЖЕНИЕ А 36](#_Toc102175394)

[ПРИЛОЖЕНИЕ Б 45](#_Toc102175395)

# ВВЕДЕНИЕ

Социальные сети пользуются большим успехом у огромного количества людей. Ими пользуются как для поддержания связи с лично знакомыми людьми, так и для заведения новых знакомств. Но социальные сети не могут передать всю полноту живых отношений, и зачастую из-за этого возникает проблема недопонимая участников по поводу того, в каких отношениях они состоят друг с другом. Взглянув под другим углом можно отметить, что социальные сети позволяют то, чего нельзя делать вживую, а именно общаться одновременно лично с большим количеством людей. В свою очередь этот фактор создает колоссальный пласт информации, который можно изучить.

В связи с этим встает вопрос об исследовании отношений между людьми с помощью интеллектуальных систем, для решения проблемы недопонимая пользователей социальных сетей между собой и их (отношений) классификации, а так же в не меньшей степени мотивирует исследовательский интерес этого аспекта социальных сетей.

Целью разработки приложения является создание интеллектуальной системы, способной определить эмоциональный окрас текстового сообщения.

Задачи проекта:

1. Выбрать наиболее подходящую для выполнения поставленной задачи модель обучения системы.
2. Реализовать механизм рационального хранения текстов для обучения модели.
3. Предоставить пользователю возможность вводить текстовое сообщение.
4. Реализовать пользовательский интерфейс, предоставляющий результат обработки сообщения.

Для разработки программного продукта применяется среда визуального объектно-ориентированного программирования Microsoft Visual Studio 2019. Для создания информационной системы используются CASE-средства Drow IO, The Createlu Blog.

# Анализ требований к информационной системе

## Описание и анализ предметной области

Алгоритм бинарной классификации текстовых сообщений является подходящим для решения поставленной задачи, не смотря на свою простоту, он довольно эффективен там, где есть потребность в строгом разделении на полярные группы данных.В данной работе предполагается разбиение выборки на два класса:

- позитивный,

- негативный;

Типичные задачи анализа текста включают:

* изучение частотных распределений слов;
* категоризацию;
* поиск по ключевым словам;
* тематическое индексирование;
* извлечение концептов (сущностей);
* моделирование отношений между сущностями;
* кластеризацию;
* обобщение документов;
* аннотирование.

Решение задачи классификации состоит из четырех последовательных этапов:

* предобработка данных;
* уменьшение размерности пространства признаков;
* построение и обучение классификатора с помощью методов машинного обучения;
* оценка качества классификации.

DFD-диаграмма система, построенная в CASE-средстве Drow IO представлена в приложении Б.2. Описание бизнес-процесса в Visual Paradigm представлено в приложении Б.3.

## Обзор и анализ возможных альтернатив

При поиске альтернатив разрабатываемой информационной системы был найден онлайн сервис тонального анализа текста «Tone Analyzer» [2]

«Tone Analyzer» – это сторонний проект IBM, который направлен на расшифровку эмоционального тона в переписке. Сервис анализирует текст, а точнее каждое слово в нём, и ассоциирует его с какой-либо эмоцией. Внешний вид сервиса представлен на рисунке 1.

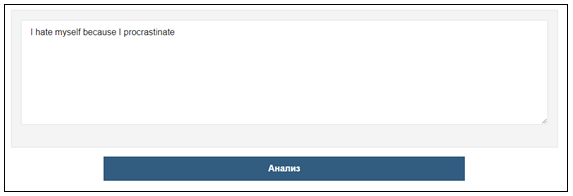


Рисунок 1 – Внешний вид сервиса «Tone Analyzer»

Для работы сервиса необходимо ввести текстовое сообщение в окно ввода и нажать «Анализ». После анализа выводится результат обработки сообщения с указанием тона написанного и его уровнем в процентах. Результаты анализа представлены на рисунках 2 и 3..

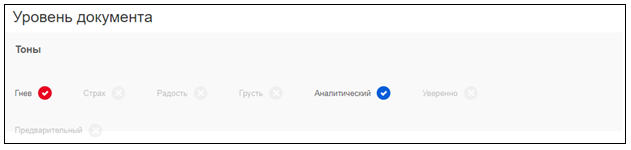


Рисунок 2 – Раздел «Уровень документа»

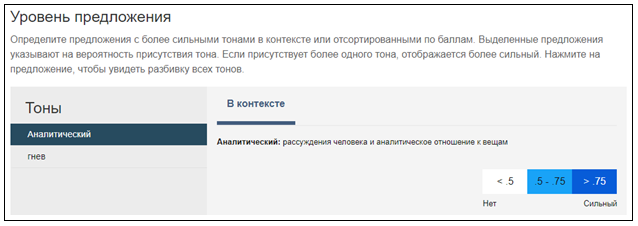


Рисунок 3 – Раздел «Уровень предложения»

Главным недостатком данного сервиса является отсутствие поддержки русского языка и вероятность отключения сервиса ввиду политической обстановки.

Опираясь на приведенный пример следует вывод, что существует необходимость в создании аналогичного продукта способного поддерживать русский язык, а так же неспособного поддерживать связь с сетью интернет.

## Анализ функциональных и эксплуатационных требований

### Стандарты

Программный продукт разрабатывается на основании следующих государственных стандартов:

1. Межгосударственный стандарт ГОСТ 7.32-2017 «Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно – исследовательской работе. Структура и правила оформления».
2. Международный стандарт ISO/IEC 12207. Информационные технологии. Процессы жизненного цикла программного обеспечения.
3. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания.
4. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
5. ГОСТ 34.603-92. Информационная технология. Виды испытаний автоматизированных систем.

### Функциональные требования пользователя

Программный продукт, разрабатываемый в рамках выпускной квалификационной работы должен удовлетворять следующему перечню функциональных требований:

* предоставить возможность ввода сообщения;
* обеспечить точный анализ;
* предоставить информацию об обработке введенного сообщения;
* сохранить сообщение с результатом его обработки.

### Входные данные

На вход программе поступает вводимое пользователем текстовое сообщение на русском языке.

### Выходные данные

Результатом работы программы является информация о сообщений, выводимая на экран. В частности процентная оценка того, на сколько текст является позитивным, а так же негативным.

### Требования к интерфейсу

Программный продукт должен иметь поля для ввода требующего обработки текстового сообщения и вывода информации, полученной в ходе работы программы.

Интерфейс должен быть простым, удобным и интуитивно понятным для пользователей, не имеющих навыков работы с программой.

### Требования к надежности

При работе с программным продуктом необходимо предусмотреть:

* Исключить возможность пользователя совершения ошибок приводящих к поломке программного продукта;
* Исключить возможность отображения результата без введения данных;
* Исключить возможность сохранить результат работы программы и введенные данные, если таковых нет.

### Требования к программной документации

В состав сопровождающей документации программного продукта должны входить следующие компоненты:

1. Пояснительная записка на 30 – 50 листах, содержащая описание разработки.
2. В приложении к пояснительной записке исходные тексты основных модулей на языке C#.
3. Пояснительная записка, задание на курсовую работу, справка проверки на плагиат, отчет о проверке, исходные тексты модулей на языке C# на CD-диске.

### Требования к составу и параметрам технических средств

Система должна работать на персональных компьютерах. Минимальная конфигурация:

* тип процессора – Pentium;
* объем оперативного запоминающего устройства – 512 Мб;
* система – Windows 7, 10.

### Модель вариантов использования

Действующие лица для диаграммы вариантов использования приведены в таблице 1.

Для изображения варианта использования применялся Rational Rose [4].

Таблица 1 **–** Действующие лица

|  |  |
| --- | --- |
| Действующее лицо | Описание |
| Пользователь | Человек, использующий функционал программы |

На основании анализа требований пользователя были выделены следующие варианты использования, представленные в таблице 2.

Таблица 2 – Описание вариантов использования

|  |  |
| --- | --- |
| Вариант использования | Описание |
| Ввод исходного текста | Пользователь вводит текст в систему для дальнейший обработки |
| Запуск анализа текста | Пользователь запускает анализ текста для определения его эмоциональной окраски |
| Вывод результата | Вывод на экран информации об эмоциональной окраске текста |
| Сохранение результата | Сохранение информации о данных и результата работы программы |

На основании всех выше рассмотренных вариантов использования была составлена диаграмма вариантов использования, представленная на рисунке 6.

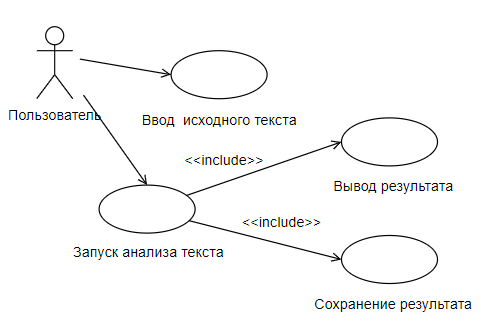


Рисунок 4 – Диаграмма вариантов использования

Рассмотрим сценарий варианта использования «Ввод исходного текста».

Краткое описание**:** данный вариант использования позволяет пользователю осуществить ввод текстового сообщения в программу для его дальнейшей обработки.

Рассмотрим основной поток событий.

Данный вариант использования начинает выполняться, когда пользователь начинает вводить в поле ввода текстовое сообщение. Сценарий основного потока представлен в таблице 3.

Таблица 3 – Сценарий основного потока

|  |  |
| --- | --- |
| Действия актера | Действия (Отклик) системы |
| 1. Пользователь выбирает операцию ввода текста | 2. Система запрашивает ввод текста |
| 3. Пользователь вводит текст | 4. Система сохраняет текст во временную память программы |
|  | 5. Вариант использования завершается |

Предусловия: перед началом выполнения данного варианта использования пользователь должен открыть систему.

Постусловия: если вариант использования завершится успешно, произойдёт сохранение текстового сообщение во временную память программы.

Рассмотрим сценарий варианта использования «Запуск анализа текста».

Краткое описание**:** данный вариант использования позволяет пользователю запустить обработку введенного текстового сообщения для последующего вывода результата.

Рассмотрим основной поток событий.

Данный вариант использования начинает выполняться, когда пользователь запускает анализ текста. Сценарий основного потока представлен в таблице 4.

Таблица 4 – Сценарий основного потока

|  |  |
| --- | --- |
| Действия актера | Действия (Отклик) системы |
| 1. Пользователь запускает операцию анализа текста | 1. Система обрабатывает введенный текст |
|  | 1. Вариант использования завершается |

Предусловия: перед началом выполнения данного варианта использования пользователь должен ввести данные для анализа.

Постусловия: результат обработки текста хранится во временной памяти программы.

Рассмотрим сценарий варианта использования «Вывод результата».

Краткое описание**:** данный вариант использования позволяет пользователю увидеть результат работы системы.

Рассмотрим основной поток событий.

Данный вариант использования начинает выполняться, когда пользователь запускает операцию отображения результата. Сценарий основного потока представлен в таблице 5.

Таблица 5 – Сценарий основного потока

|  |  |
| --- | --- |
| Действия актера | Действия (Отклик) системы |
| 1. Пользователь запускает операцию отображения результата | 2. Система выводит результат работы на экран |
|  | 3. Вариант использования завершается |

Рассмотрим сценарий варианта использования «Сохранение результата».

Краткое описание**:** данный вариант использования позволяет пользователю осуществить сохранение введенных данных вместе с результатом работы системы по его анализу.

Рассмотрим основной поток событий.

Данный вариант использования начинает выполняться, когда пользователь запускает операцию сохранения. Сценарий основного потока представлен в таблице 6.

Таблица 6 – Сценарий основного потока

|  |  |
| --- | --- |
| Действия актера | Действия (Отклик) системы |
| 1. Пользователь запускает операцию сохранения результата | 2. Система сохраняет данные |
|  | 3. Вариант использования завершается |

Предусловия: перед началом выполнения данного варианта использования пользователь должен ввести данные и запустить анализ.

Постусловия: если вариант использования завершится успешно, произойдёт сохранение текстового сообщение и результата работы системы над ним в данные.

### Глоссарий проекта

В таблице 7 приведены термины предметной области и их значения.

Таблица 7 – Термины и их значения

|  |  |
| --- | --- |
| Термин | Значение |
| Кластеризация | Объединение объектов или наблюдений в непересекающиеся группы, называемые кластерами, на основе близости значений их признаков |
| Машинное обучение | Это направление искусственного интеллекта, связанное с разработкой и построением аналитических моделей, которые способны автоматически обнаружить в данных скрытые и ранее неизвестные закономерности, а также самостоятельно приобретать свойства, необходимые для распознавания этих закономерностей |
| Интеллектуальный анализ данных | Это направление информационных технологий, охватывающее всю область проблем, связанных с извлечением знаний из массивов данных |
| Текст (сообщение) | Краткое текстовое сообщение, подлежащее тональному анализу |
| Результат | Процентная оценка тональности краткого текстового сообщения |
| Данные | Краткое текстовое сообщение, его тональный анализ и вероятность каждой из возможных тональностей |
| Тональность | Эмоциональная окраска текста |

### Проверка модели на полноту

Проверка на полноту диаграммы вариантов использования производится по операциям, выполняемым над основными объектами (табл. 7).

Ключевыми объектами в модели вариантов использования, являются: Сообщение, Результат, Данные.

В таблице 8 обозначены виды операций:

1 – создание;

2 – просмотр;

3 – изменение;

4 – удаление.

Результаты анализа полноты выполнения функциональных требований пользователя в модели вариантов использования приведены в таблице 8.

Таблица 7 – Анализ полноты выполнения требований пользователя

|  |  |  |  |
| --- | --- | --- | --- |
| Варианты использования | Объекты | | |
| Сообщение | Результат | Данные | |
| Ввод исходного текста | 1,2,3.4 |  |  | |
| Запуск анализа текста |  | 1,2 |  | |
| Вывод результата |  | 2 | 1,2,4 | |

Таблица 8 – Анализ полноты выполнения требований пользователя

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требования пользователя | Варианты использования | | | |
| Ввод исходного текста | Запуск анализа текста | Вывод результата | Сохранение результата |
| Предоставить возможность ввода сообщения | + |  |  |  |
| Обеспечить точный анализ |  | + |  |  |
| Предоставить информацию об обработке введенного сообщения |  |  | + |  |
| Сохранить сообщение с результатом его обработки |  |  |  | + |

# Проектирование информационной системы

## Разработка архитектуры системы

Разрабатываемое приложение разрабатывается под персональные компьютеры на платформе Windows. Для его работы необходим Windows 7 или 10, процессор архитектуры Pentium, такие технические средства как экран, мышь и клавиатура.

На рисунке 5 представлена архитектура технических средств системы.

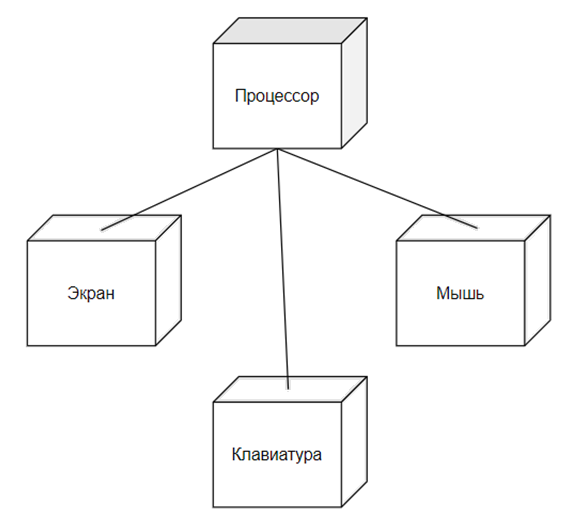


Рисунок 5 – Архитектура технических средств системы

## Разработка модели предметной области

Пользуясь списком категорий и методом анализа словесного описания вариантов использования, составлен список кандидатур на роль концептуальных классов для предметной области. Он соответствует требованиям и принятым упрощениям для всей предметной области.

Список концептуальных классов:

* Текст;
* Обработанный текст;
* База данных;
* Тональность;
* Результат.

На основании анализа словесного описания варианта использования, составлен список ассоциаций для предметной области, представленный в таблице 9.

Таблица 9 – Ассоциации для модели предметной области

|  |  |
| --- | --- |
| Ассоциация | Описание ассоциации |
| Загружает | Пользователь загружает текст |
| Относится к | Обработанный документ относится к тональности |
| Помогает определить | Результат помогает определить тональность |
| Преобразуется | Обработанный текст преобразуется в тональность текста |
| Включается в | Тональность текста включается в результат |
| Сохраняется | Обработанный текст сохраняется в базу данных |

На основании анализа технического задания и описания вариантов использования выделены атрибуты классов для модели предметной области, представленные в таблице 10.

В результате объединения концептуальных классов, ассоциаций и атрибутов классов концептуальная модель предметной области имеет вид, показанный на рисунке 6.

Таблица 10 – Атрибуты классов для модели предметной области

|  |  |
| --- | --- |
| Название класса | Атрибуты класса |
| Текст | Содержание Длина |
| Обработанный текст | Вероятность тональности в процентах |
| Данные |  |
| Тональность | Название |
| Тональность текста | Содержание  Процент точности |
| Результат | Содержание Вероятность тональности в процентах Название тональности |

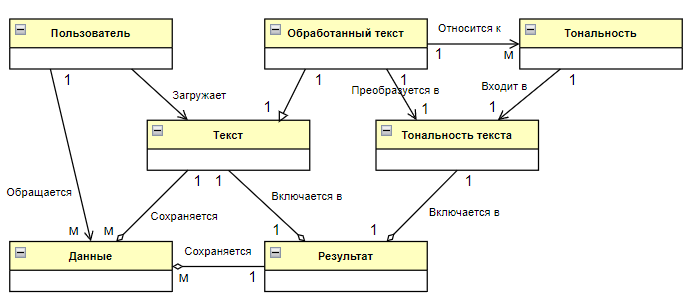


Рисунок 6 – Концептуальная модель предметной области

## Разработка алгоритма функционирования системы

Приложение не требует авторизации, пользователь запускает приложение и может работать с ним. Алгоритм работы приложения представлен в виде диаграммы деятельностей на рисунке 7.

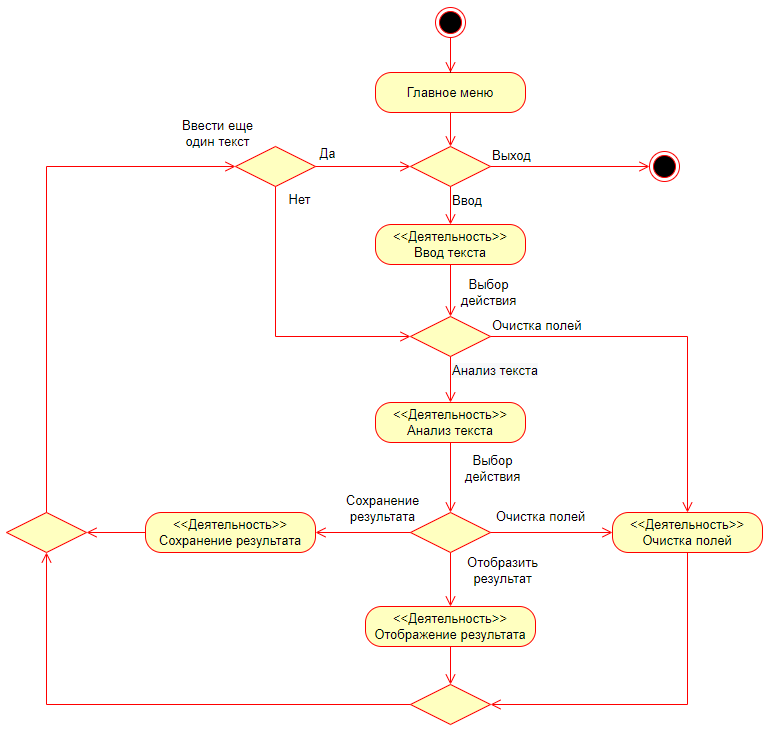


Рисунок 7 – Алгоритм работы системы

## Проектирование интерфейса пользователя

На основании алгоритма функционирования и требований к интерфейсу (раздел 1) разработана диаграмма состояний (рис. 8).

После запуска приложения на экране появляется форма главного меню.

Введено суперсостояние интерфейса «Работа приложения» объединяющее все состояния кроме состояния «Главное меню». Из суперсостояния по событию «Возврат» (нажатие кнопки «Возврат» на экране) осуществляется переход в состояние «Главное меню». Этот переход наследуется всеми подсостояниями суперсостояния.

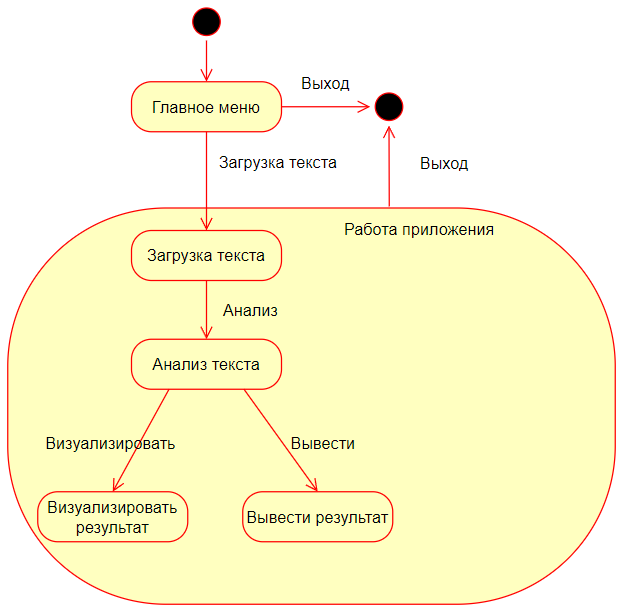


Рисунок 8 – Диаграмма состояний работы приложения

## Реляционная модель данных

На рисунке 9 изображена реляционная модель данных.

Данная модель реализуется в СУБД DB Browser for SQLite [6].

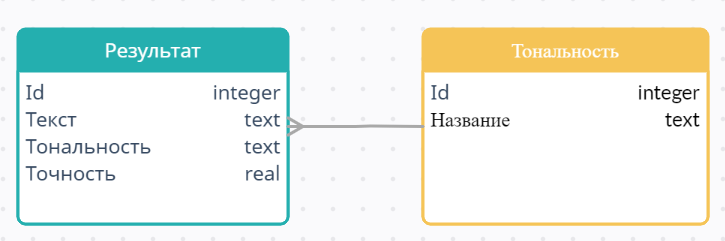


Рисунок 9 – Реляционная модель данных

## Проектирование классов предметной области

### Построение диаграмм последовательностей для варианта использования «Выбор операции над документом»

Диаграмма последовательности, описывающая основной поток событий и его подчиненные потоки изображена на рисунке 10

Инициатором является пользователь, всё взаимодействие идёт через класс «Окно».

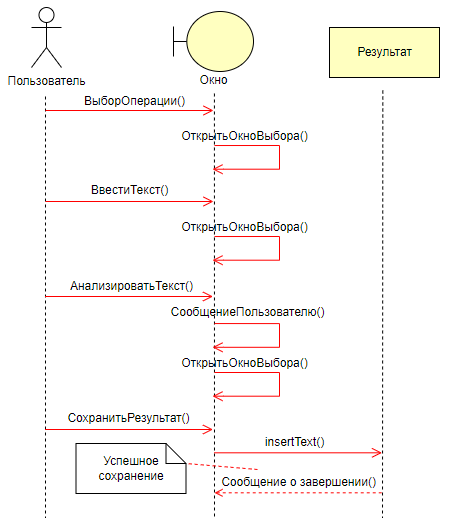


Рисунок 10 – Диаграмма последовательности работы системы

### Построение диаграммы кооперации

Структурные особенности передачи и приема сообщений между объектами представлены на диаграмме кооперации на рисунке 11.

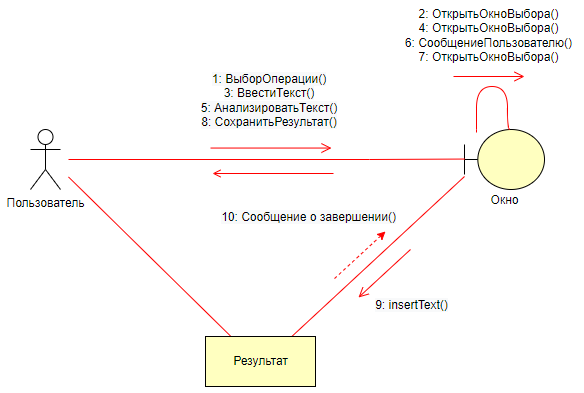


Рисунок 11 – Диаграмма кооперации

### Построение диаграммы классов

Диаграмма классов представлена на рисунке 12.

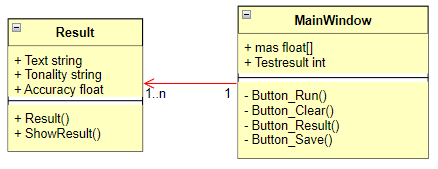


Рисунок 12 – Диаграмма классов

Result – класс, содержащий текст, тональность вероятность полученной тональности.

MainWindow – класс окна, через этот класс пользователь взаимодействует с остальными классами.

### Уточнение структуры классов предметной области и разработка алгоритмов методов

Класс Result содержит информацию о полученном результате анализа. Атрибуты класса представлены в таблице 11. Операции класса представлены в таблице 12.

Таблица 11 – Атрибуты класса Result

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Описание |
| Text | string | Содержание текстового сообщения |
| Tonality | string | Название тональности сообщения |
| Accuracy | float | Вероятность тональности |

Таблица 12 – Операции класса Result

|  |  |
| --- | --- |
| Операция | Описание |
| Result() | Конструктор класса |
| ShowResult() | Выводит атрибута класса на экран |

Класс MainWindow класс окна, через которое пользователь взаимодействует с программой. Атрибуты класса представлены в таблице 13. Операции класса представлены в таблице 14.

Таблица 13 – Атрибуты класса MainWindow

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Описание |
| mas | float[] | Массив предназначенный для приведения результата анализа к удобному виду |
| Testresult | int | Переменная хранящая результат анализа тональности |

Таблица 14 – Операции класса MainWindow

|  |  |
| --- | --- |
| Операция | Описание |
| Button\_Run() | Анализирует введенное текстовое сообщение |
| Button\_Clear() | Очищает все поля окна |
| Button\_Result() | Выводит результат анализа |
| Button\_Save() | Сохраняет результат анализа вместе с текстом |

## Модель нейронной сети

В выпускной квалификационной работе для обучения распознавания жанров литературных произведений использовалась нейронная сеть для бинарной классификации.

Классификация по двум классам, или бинарная классификация, является едва ли не самой распространенной задачей машинного обучения. Для решения задачи был использован размеченный датасет содержащий немногим более 200тыс. твиттов. Набор состоит на 50% из положительных и на 50% из отрицательных твиттов. Таким образом достигнута максимальная точность обучения модели.

В качестве бинарного классификатора в системе используется усредненный перцептрон. Метод усредненного перцептрона является ранней и простой версией нейронной сети. В этом подходе входные данные делятся на несколько возможных значений на основе линейной функции в сочетании с набором весовых коэффициентов, производным от вектора компонента.

# Реализация системы

## Реализация программного обеспечения системы

### Разработка диаграммы компонентов

Для реализации системы использовалась платформа Microsoft Visual Studio 2019, Windows Presentation Foundation и язык программирования C#.

Приложение было реализовано на Windows Presentation Foundation.

Для обучения модели использовалась библиотека Microsoft.ML, модель классификации текстов.

Для работы с базой данных использовалась библиотека Dapper.

Программа состоит из 3 проектов: WpfApp1, WpfApp1ML.ConsoleApp, WpfApp1ML.Model. Главным проектом является WpfApp1, в нем реализован пользовательский интерфейс и взаимодействие с обученной моделью.

GenreDefinerCore состоит из двух классов и базы данных. ProjectDb.db является базой sqlite и хранится как один файл.

WpfApp1ML.ConsoleApp используется для переобучения модели машинного обучения. ModelBuilder – класс создания и обучения модели, Program – класс вызова консольного приложения для запуска переобучения.

WpfApp1ML.Model содержит классы для описания самой модели и определения тональности. ModelInput – входные параметры, ModelOutput – выходные параметры, ConsumeModel – класс для определения тональности.

Исходный текст программы представлен в приложении А.

Диаграмма компонентов представлена на рисунке 13.

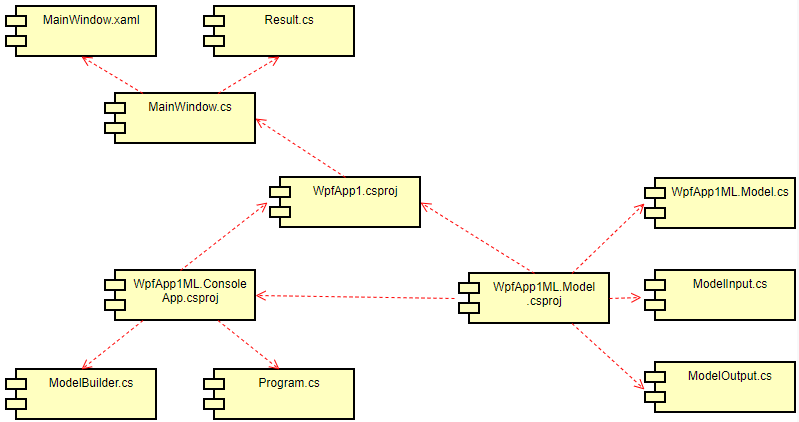


Рисунок 13 – Диаграмма компонентов приложения

### Объекты интерфейса пользователя

Система содержит представляет собой окно MainWindow.

Внешний вид окна MainWindow представлен на рисунке 14.

В таблице 15 представлены расположенные на окне MainWindow компоненты.

Таблица 15 – Компоненты окна MainWindow

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование компонента | Тип компонента | Назначение |
| 1 | Запустить | Button | Запуск анализа введенного текста |
| 2 | textBox\_Text | TextBox | Поле ввода текстового сообщения |
| 3 | textBox\_Result | TextBox | Поле вывода названия тональности |

Продолжение таблицы 15

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование компонента | Тип компонента | Назначение |
| 4 | textBox\_otr | TextBox | Поле вывода вероятности отрицательной тональности |
| 5 | textBox\_pol | TextBox | Поле вывода вероятности положительной тональности |
| 6 | Положительно | Label | Для вывода вероятности положительной тональности |
| 7 | Отрицательно | Label | Для вывода вероятности отрицательной тональности |
| 8 | Введите сообщение | Label | Подпись для ввода текстового сообщения |
| 9 | Результат | Label | Для вывода результата |
| 10 | Очистить поля | Button | Очистка всех полей окна |
| 11 | Результат | Button | Вывод результата анализа |

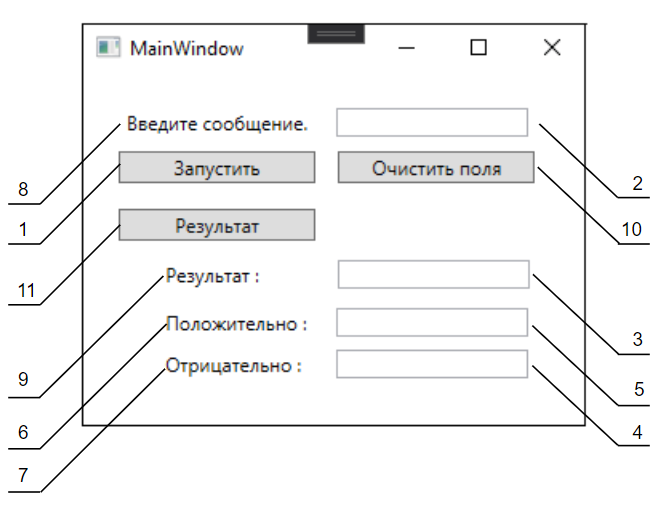


Рисунок 14 – Внешний вид окна MainWindow

## Реализация технического обеспечения

Полная диаграмма развертывания приложения приведена на рисунке 14.

Разрабатываемое приложение разрабатывается под персональные компьютеры на платформе Windows. Для его работы необходим Windows 7, 10, процессор архитектуры Pentium, такие технические средства как экран, мышь и клавиатура.

На рисунке 15 представлена архитектура технических средств системы.

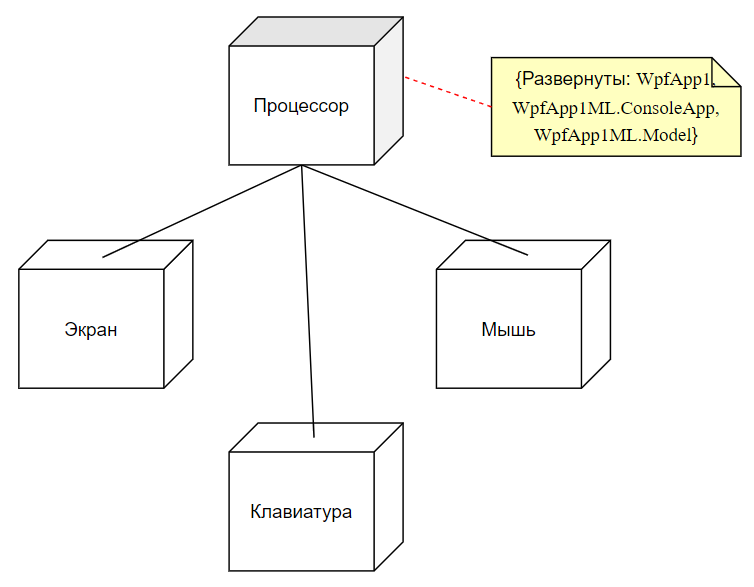


Рисунок 15 – Архитектура технических средств системы

# ЗАКЛЮЧЕНИЕ

Разработанный программный продукт позволяет с высокой точностью определить эмоциональную окраску текста короткого сообщения, а так же предоставляет информацию о вероятности конкретной окраски.

В процессе создания системы в соответствии с заданием были разработаны: модель вариантов использования, концептуальная модель предметной области, диаграммы деятельности, реляционная модель данных, диаграмма состояний приложения, диаграмма развертывания. Была выполнена частично проверка и отладка приложения.

Приложение позволяет:

* анализировать эмоциональную окраску сообщений;
* выводить результат анализа сообщений на экран;
* сохранять полученный результат анализа вместе с сообщением;

В данной выпускной квалификационной работе было создано настольное приложение на языке программирования C#. Приложение имеет удобный и понятный интерфейс, способно в короткие сроки определить эмоциональную окраску тестового сообщения на русском языке.

Таким образом, в выпускной квалификационной работе удалось выполнить все представленные требования, реализовав анализатор эмоциональной окраски коротких текстовых сообщений.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Tекст.рф - SEO-анализ текста: [Электронный ресурс]. URL: https://text.ru/seo (дата обращения 07.12.2020).
2. ER-диаграммы в нотациях Баркера и Мартина. CASE-средства.: [Электронный ресурс] URL: https://studme.org/93801/informatika/erdiagrammy\_notatsiyah\_barkera\_martina\_case-sredstva (дата обращения 15.12.2020).
3. Модуль «Двухклассовый усредненный перцептрон»: Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/azure/machine-learning/algorithm-module-reference/two-class-averaged-perceptron

(дата обращения 16.01.2021).

1. Краткий обзор языка C#.: [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/

(дата обращения 04.02.2021).

1. Общие сведения о Windows Forms: [Электронный ресурс]. URL: https://metanit.com/sharp/wpf/ (дата обращения 09.12.2020).
2. Что такое ML.NET и принципы работы этой системы: [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/machine-learning/how-does-mldotnet-work. (дата обращения 20.02.2021).
3. iTextSharp: [Электронный ресурс] URL: https://www.nuget.org/packages/iTextSharp/ (дата обращения 18.03.2021).
4. WPF. Меню [Электронный ресурс] URL: https://metanit.com/sharp/wpf/5.11.php (дата обращения 05.05.2021).
5. Пошаговое руководство. Создание и запуск модульных тестов для управляемого кода: [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2019 (дата обращения 14.05.2021).

# ПРИЛОЖЕНИЕ А

**Текст программы**

В приложении представлен программный код на языке C#.

Приведен текст формы MainWindow

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.IO;

using Microsoft.Win32;

using ToneOfMessagesML.Model;

namespace ToneOfMessages

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

double[] mas = new double[2];

int Testresult;

private void Button\_Run(object sender, RoutedEventArgs e)

{

string text = textBox\_Text.Text;

text = Result.Clear\_Text(text);

if (textBox\_Text.Text != "")

{

var input = new ModelInput { Col3 = text };

var result = ConsumeModel.Predict(input);

mas[0] = result.Score[0];

mas[1] = result.Score[1];

}

Clear\_All();

textBox\_Text.Text = text;

}

private void Button\_Clear(object sender, RoutedEventArgs e)

{

Clear\_All();

}

private void Button\_Result(object sender, RoutedEventArgs e)

{

textBox\_Result.Text = Result.Tonality\_Result(mas[0], mas[1]);

textBox\_otr.Text = string.Format("{0:F}", (float)Math.Round(Result.Result\_Score(mas[0]), 2));

textBox\_pol.Text = string.Format("{0:F}", (float)Math.Round(Result.Result\_Score(mas[1]), 2));

}

private void Button\_Save(object sender, RoutedEventArgs e)

{

if (textBox\_Text.Text != "" && textBox\_Result.Text != "" && textBox\_otr.Text != "" && textBox\_pol.Text != "")

{

Double Accuracy;

if (Convert.ToDouble(textBox\_otr.Text) > Convert.ToDouble(textBox\_pol.Text))

Accuracy = Convert.ToDouble(textBox\_otr.Text);

else

Accuracy = Convert.ToDouble(textBox\_pol.Text);

Result elem = new Result(textBox\_Text.Text, textBox\_Result.Text, Accuracy);

Result.Save(elem);

}

}

private void Show\_Result()

{

textBox\_otr.Text = Result.ShowResult(mas[0]);

textBox\_pol.Text = Result.ShowResult(mas[1]);

}

private void Clear\_All()

{

textBox\_Text.Text = "";

textBox\_Result.Text = "";

textBox\_otr.Text = "";

textBox\_pol.Text = "";

}

}

}

Приведен текст класса Result

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

using System.Windows;

using ToneOfMessagesML.Model;

namespace ToneOfMessages

{

public class Result

{

public string Text { get; set; }

public string Tonality { get; set; }

public double Accuracy { get; set; }

public Result()

{

Text = "\_";

Tonality = "unknown";

Accuracy = 0;

}

public Result(string text, string tonality, double accuracy)

{

Text = text;

Tonality = tonality;

Accuracy = accuracy;

}

public static string ShowResult(double mas)

{

double val = (float)Math.Round(mas, 2);

return string.Format("{0:F}", val);

}

public static double Result\_Score(double score)

{

return score \* 100;

}

public static async void Save(Result element)

{

string path = "Result.txt";

using (StreamWriter writer = new StreamWriter(path, true, System.Text.Encoding.UTF8))

{

await writer.WriteLineAsync(

"Текст: " + element.Text + "; " +

"Тональность: " + element.Tonality + "; " +

"Точность: " + element.Accuracy + "; ");

writer.Close();

}

}

public static string Tonality\_Result(double score0, double score1)

{

string result = "";

if (score0 > 0.501)

{

result = "Негативно";

}

else if (score1 > 0.501)

{

result = "Позитивно";

}

return result;

}

public static string Result\_Analysis(string text)

{

double[] mas = new double[2];

var input = new ModelInput { Col3 = text };

var result = ConsumeModel.Predict(input);

mas[0] = result.Score[0];

mas[1] = result.Score[1];

return Tonality\_Result(mas[0], mas[1]);

}

public static string Clear\_Text(string text)

{

text = text.Replace(",", "");

text = text.Replace(".", "");

text = text.Replace(":", "");

text = text.Replace(";", "");

text = text.Replace("!", "");

text = text.Replace("?", "");

return text;

}

}

}

Модульные тесты проекта Tests:

using Microsoft.VisualStudio.TestTools.UnitTesting;

using ToneOfMessages;

using System.Linq;

using System.Data;

namespace Tests

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestShowResult()

{

var a = Result.ShowResult(2);

Assert.AreEqual(a, "2,00");

}

[TestMethod]

public void TestResult\_Score()

{

var a = Result.Result\_Score(0.05);

Assert.AreEqual(a, 5);

}

[TestMethod]

public void TestTonalityNeg()

{

double score1 = 0.8, score2 = 0.2;

var a = Result.Tonality\_Result(score1,score2);

Assert.AreEqual(a, "Негативно");

}

[TestMethod]

public void TestTonalityPos()

{

double score1 = 0.2, score2 = 0.8;

var a = Result.Tonality\_Result(score1, score2);

Assert.AreEqual(a, "Позитивно");

}

[TestMethod]

public void TestResult\_AnalysisNeg()

{

var a = Result.Result\_Analysis("Смерть, гнев, тоска, отчаяние");

Assert.AreEqual(a, "Негативно");

}

[TestMethod]

public void TestResult\_AnalysisPos()

{

var a = Result.Result\_Analysis("Радость, счастье, удача, веселье");

Assert.AreEqual(a, "Позитивно");

}

[TestMethod]

public void TestClear\_Text()

{

var a = Result.Clear\_Text("Да, Пожалуй. Здорово! Все; Они: Кто?");

Assert.AreEqual(a, "Да Пожалуй Здорово Все Они Кто");

}

}

}

Приведен текст проекта Program.cs

using System;

using ToneOfMessagesML.Model;

namespace ToneOfMessagesML.ConsoleApp

{

class Program

{

static void Main(string[] args)

{

// Create single instance of sample data from first line of dataset for model input

ModelInput sampleData = new ModelInput()

{

Col3 = @"А теперь еду домой информатика меня ждет",

};

// Make a single prediction on the sample data and print results

var predictionResult = ConsumeModel.Predict(sampleData);

Console.WriteLine("Using model to make single prediction -- Comparing actual Col4 with predicted Col4 from sample data...\n\n");

Console.WriteLine($"Col3: {sampleData.Col3}");

Console.WriteLine($"\n\nPredicted Col4 value {predictionResult.Prediction} \nPredicted Col4 scores: [{String.Join(",", predictionResult.Score)}]\n\n");

Console.WriteLine("=============== End of process, hit any key to finish ===============");

Console.ReadKey();

}

}

}

Приведен текст класса ModelInput.cs

using Microsoft.ML.Data;

namespace ToneOfMessagesML.Model

{

public class ModelInput

{

[ColumnName("col0"), LoadColumn(0)]

public float Col0 { get; set; }

[ColumnName("col1"), LoadColumn(1)]

public float Col1 { get; set; }

[ColumnName("col2"), LoadColumn(2)]

public string Col2 { get; set; }

[ColumnName("col3"), LoadColumn(3)]

public string Col3 { get; set; }

[ColumnName("col4"), LoadColumn(4)]

public string Col4 { get; set; }

[ColumnName("col5"), LoadColumn(5)]

public float Col5 { get; set; }

[ColumnName("col6"), LoadColumn(6)]

public float Col6 { get; set; }

[ColumnName("col7"), LoadColumn(7)]

public float Col7 { get; set; }

[ColumnName("col8"), LoadColumn(8)]

public float Col8 { get; set; }

[ColumnName("col9"), LoadColumn(9)]

public float Col9 { get; set; }

[ColumnName("col10"), LoadColumn(10)]

public float Col10 { get; set; }

[ColumnName("col11"), LoadColumn(11)]

public float Col11 { get; set; }

}

}

Приведен текст класса ModelOutput.cs

using System;

using Microsoft.ML.Data;

namespace ToneOfMessagesML.Model

{

public class ModelOutput

{

// ColumnName attribute is used to change the column name from

// its default value, which is the name of the field.

[ColumnName("PredictedLabel")]

public String Prediction { get; set; }

public float[] Score { get; set; }

}

}

# ПРИЛОЖЕНИЕ Б

**Внешний вид графического материала**

Внешний вид графического материала, выполненного на отдельных листах представлен на рисунках Б.1 – Б.5.

Перечень графического материала:

лист 1: Постановка задачи (рис. Б.1),

лист 2: Контекстная DFD-диаграмма системы (рис. Б.2),

лист 3: Описание бизнес-процесса в BPMN (рис. Б.3),

лист 4: Диаграмма вариантов использования (рис. Б.4),

лист 5: Алгоритм работы системы (рис. Б.5).

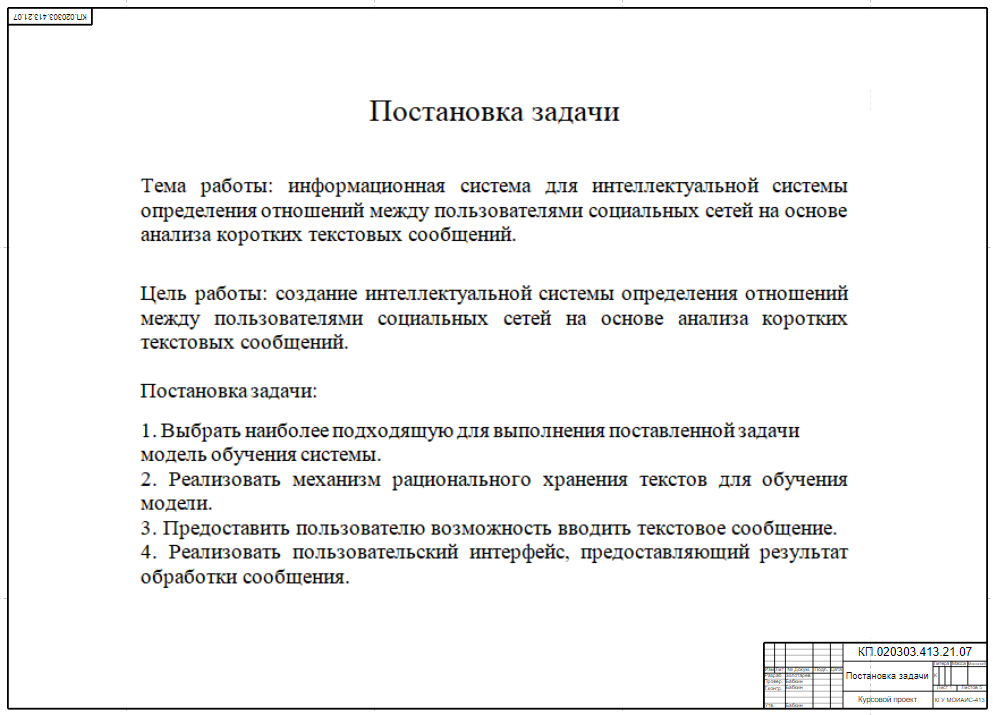


Рисунок Б.1 – Постановка задачи



Рисунок Б.2 – Контекстная DFD-диаграмма системы

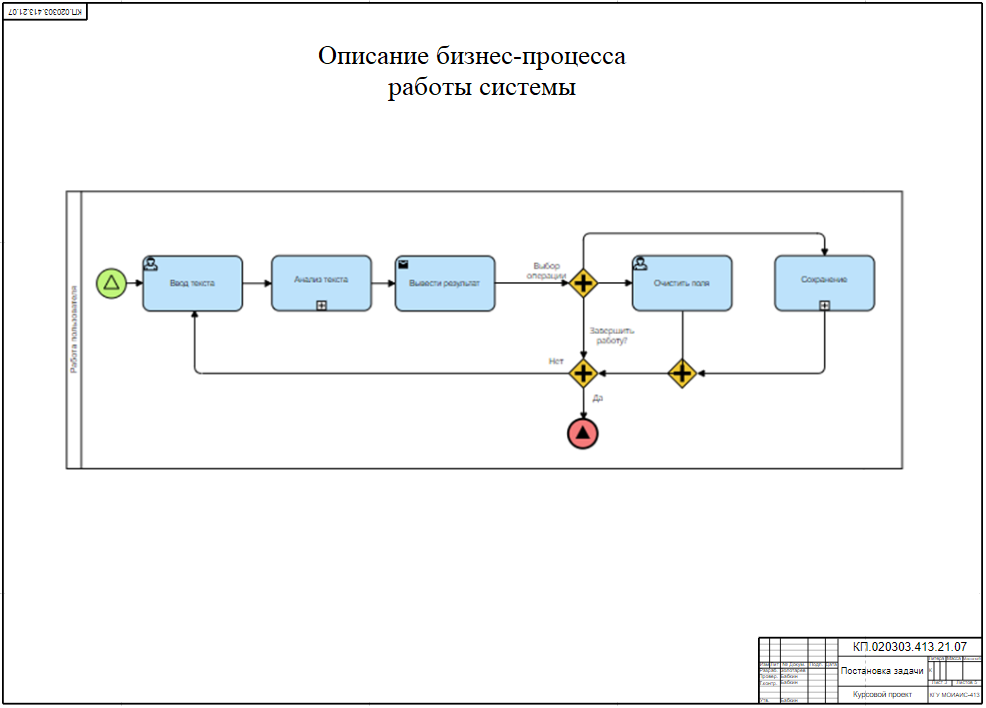


Рисунок Б.3 – Описание бизнес-процесса в BPMN

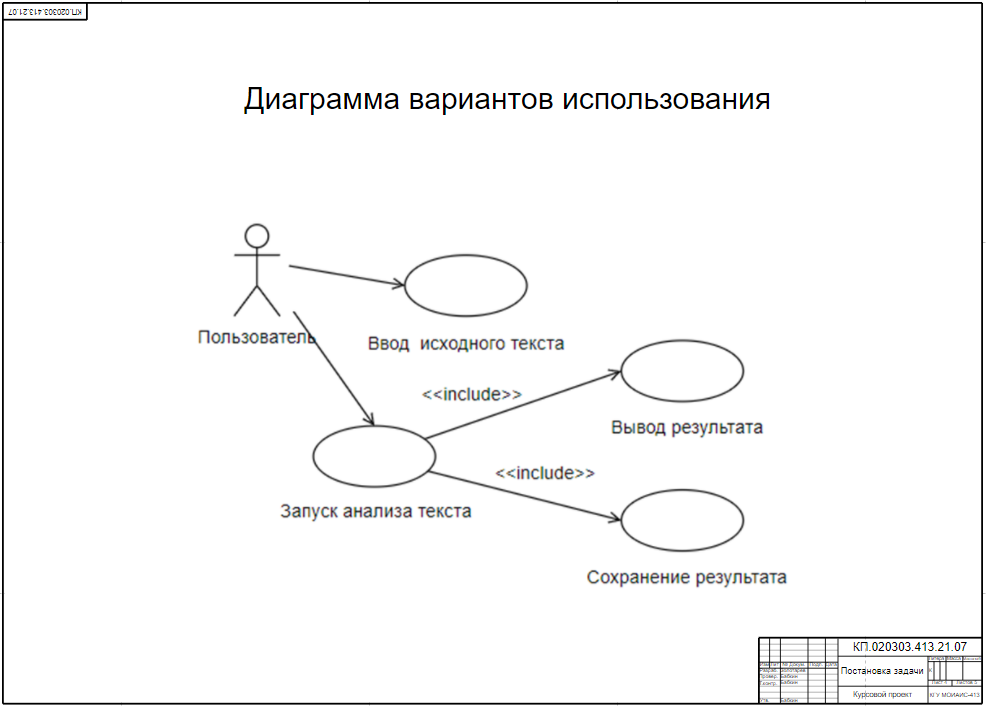


Рисунок Б.4 – Диаграмма вариантов использования



Рисунок Б.5 – Алгоритм работы системы