Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ФМОП «Международных образовательных программ»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Метод классификации новостных текстов по тематикам с использованием опорных векторов»

Студент **ИУ7И-84Б** _____ **Динь Вьет Ань**
(Подпись, дата)

Руководитель _____ **Кострицкий А. С.**
(Подпись, дата)

Нормоконтролер _____ _____
(Подпись, дата)

2024 г.

РЕФЕРАТ

Расчетно–пояснительная записка содержит 67 с., 20 рис., 4 табл., 21 ист., 1 прил.

Ключевые слова: метод классификации, классификации текста, машинное обучение, метод опорных векторов.

В работе представлена разработка метода классификации новостных текстов по тематикам с использованием опорных векторов.

Рассмотрена задача классификации текста. Рассмотрены этапы решения задачи и основные методы классификации текста. Проведена формализация постановки задачи в виде IDEF0-диаграммы. Разработан метод классификации новостных текстов по тематикам с использованием опорных векторов. Представлена реализация разработанного метода, приведены результаты исследования качества классификатора в зависимости от различных параметров.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениям.

1. SVM (англ. support vector machine) — метод опорных векторов.
2. KNN (англ. k-nearest neighbors) — метод K-ближайших соседей.
3. TF-IDF (англ. term frequency-inverse document frequency) — Частота термина - обратная частота документа

ВВЕДЕНИЕ

В современном информационном обществе огромное количество новостей и статей публикуется каждый день. Чтобы эффективно обрабатывать и анализировать этот объем информации, важно иметь систему классификации новостей. Классификация позволяет автоматически определять тематику новости, ее важность, тональность и другие характеристики. Задача классификации новостей является актуальной и интересной задачей в области обработки естественного языка и машинного обучения. Она имеет широкий спектр применений, начиная от организации и ранжирования новостных потоков до автоматического анализа общественного мнения и выявления трендов.

Решение задачи классификации новостей может иметь большое практическое применение, включая создание интеллектуальных новостных агрегаторов, систем мониторинга общественного мнения, фильтрацию и организацию информации для пользователей. Это способствует улучшению доступа к информации, оптимизации процессов принятия решений и повышению качества новостных сервисов.

Однако, классификация новостей может быть сложной задачей из-за разнообразия тематик, стилей и тональностей новостных статей. Нередко встречаются случаи смешения нескольких тем в одной статье или субъективной интерпретации информации. Поэтому выбор правильного алгоритма и подхода, а также качество и разнообразие обучающих данных играют важную роль в достижении высокой точности классификации.

Цель работы — разработать метод классификации новостных текстов по тематикам с использованием опорных векторов. Для достижения поставленной цели необходимо выполнить следующие задачи:

— провести анализ предметной области, проанализировать предметную область и основные методы классификации текстов;

— разработать метод классификации новостных текстов по тематикам с помощью опорных векторов;

— разработать программное обеспечение, реализующее данный метод;

— оценить результаты работы метода в зависимости от различных параметров программного обеспечения.

1 Аналитический раздел

В данном разделе приводятся анализ предметной области и этапы очистки и предварительной обработки текста, также рассматривается обзор методов извлечения признаков из текста. Также анализируются основные методы классификации текстов и проведено сравнение методов классификации текстов по преимуществам и недостаткам. Также в этом разделе представляется формализованная постановка задачи в виде IDEF0-диаграммы.

1.1. Анализ предметной области

1.1.1. Задача классификации текстов

Классификация текста —– это процесс присвоения предопределенной категории или метки предложениям, абзацам, текстовым отчетам или неструктурированного текста.

За последние несколько десятилетий проблемы классификации текста широко изучались и решались во многих практических приложениях. Многие исследователи теперь заинтересованы в разработке приложений, использующих преимущества методов классификации текста, особенно в связи с недавними достижениями в области обработки естественного языка.

Некоторые задачи классификации текста в реальном [1]:

1. анализ настроений — задача понимания аффективных состояний и субъективной информации, содержащейся в фрагменте текста;

2. маркировка тем — задача распознавания одной или нескольких тем фрагмента текста (т. е. его тем);

3. классификация новостей — задача присвоения новостям категорий;

4. ответ на вопрос — задача выбора ответа на вопрос, выбора из потенциальных предложений - кандидатов (обычно извлекаемых из контекстного документа);

5. вывод на естественном языке — задача определения того, влекут ли два предложения друг друга (классификация, происходит ли следование в одном из двух направлений или ни в одном из них);

6. распознавание именованных объектов — задача поиска именованных объектов в неструктурированном тексте и маркировка их заранее определенными категориями;

7. синтаксический анализ — серия задач, связанных с прогнозированием морфо - синтаксических свойств слов.

1.1.2. Процесс классификации текста

Большинство процессов классификации текста, обычно, состоят из следующих трёх шагов: предобработка текста, извлечение признаков и классификации текста с помощью некоторого алгоритма.

Ниже, на рисунке 1.1, представлены этапы процесса классификации текста.

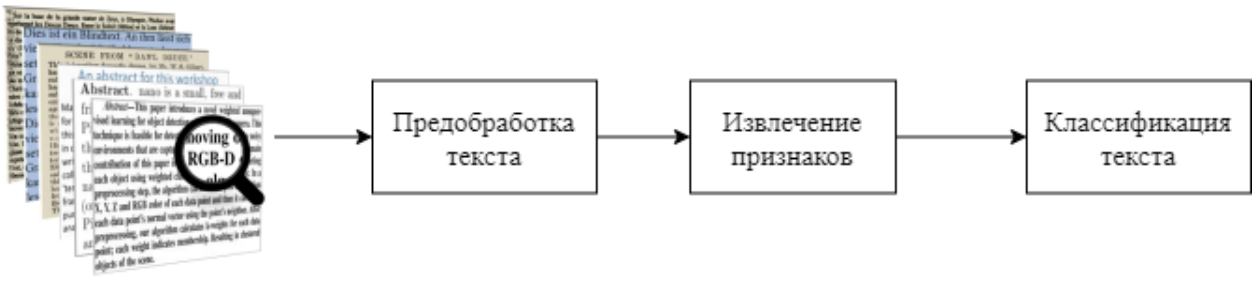

Рисунок 1.1 – Этапы процесса классификации текста.

Система классификации текстов содержит четыре различных уровня области применения, которые можно применять.

1. Уровень документа. На этом уровне документа алгоритм получает соответствующие категории полного документа.

2. Уровень абзаца. На этом уровне абзаца алгоритм получает соответствующие категории одного абзаца (части документа).

3. Уровень предложения. На этом уровне предложения получают соответствующие категории одного предложения (части абзаца).

4. Уровень подпредложения. На этом уровне подпредложения алгоритм получает соответствующие категории подвыражений внутри предложения.

1.2. Предобработка текста

1.2.1. Необходимость предобработки текста

Входные данные для задач на естественном языке состоят из необработанного неструктурированного текста. Текстовая информация, в отличие от других типов данных, таких как изображения или временные ряды, не обладает числовым представлением, поэтому перед подачей ее в какой-либо классификатор она должна быть спроецирована в соответствующее пространство признаков. Поэтому процедуры предварительной обработки имеют особое значение, поскольку без них не существует основы ни для процедур выделения признаков, ни для алгоритмов классификации.

1.2.2. Очистка и предобработка текста

Токенизация — самая базовая операция предварительной обработки, которую необходимо применить к тексту. Этот процесс определяет уровень детализации анализа текстовых данных и в целом может быть описан как процесс предварительной обработки, целью которого является разделение текстового потока на слова, фразы, символы или другие значимые элементы, называемые токенами. Разделение основано на правилах и может быть простым, как разделение пробелами или знаками препинания.

Например, предложение:

After eating, I decided to start working.

В данном случае токены следующие:

{“After”, “eating”, “I”, “decided”, “to”, “start”, “working”}.

Стоп-слова (текстовые шумы) —– это неинформативные слова, которые встречаются в большом количестве, но не имеют семантического значения. Например, слова “и”, “в”, “только” не несут никакой ценности и только добавляют шум в данные. Множество токенов, полученный после процесса токенизации, может содержать множество ненужных или бессмысленных элементов. Удаление стоп-слов необходимо, поскольку это уменьшит количество различных элементов в пространстве признаков.

Обычно тексты содержат разные грамматические формы одного и того же слова, а также могут встречаться однокоренные слова. Лемматизация и стемминг преследуют цель привести все встречающиеся словоформы к одной, нор-

мальной словарной форме.

Стемминг — это грубый эвристический процесс, который отрезает “лишнее” от корня слов, часто это приводит к потере словообразовательных суффиксов.

Лемматизация — это более тонкий процесс, который использует словарь и морфологический анализ, чтобы в итоге привести слово к его канонической форме — лемме.

Отличие в том, что стеммер (конкретная реализация алгоритма стемминга) действует без учёта контекста и, соответственно, не делает разницы между словами, которые имеют разный смысл в зависимости от части речи. Однако у стеммеров есть своё преимущество — они работают быстрее.

1.3. Извлечения признаков

1.3.1. Модель Bag of Words

Модель Bag of Words (модель BoW)[2] —– это уменьшенное и упрощенное представление текстового документа из выбранных частей текста на основе определенных критериев, таких как частота слов.

При всей простоте реализации данный подход имеет ряд недостатков:

— для больших наборов текстов размерность словаря, а, следовательно, и размерность вектора, представляющего текст, может исчисляться сотнями тысяч, а иногда и миллионами;

— не учитывается контекст слова в документе.

1.3.2. Метод Word2Vec

Word2Vec (Word to Vector)[2] —– это метод, используемый для преобразования слов в векторы, тем самым фиксируя их значение, семантическое сходство и взаимосвязь с окружающим текстом. Этот метод помогает компьютерам изучать контекст и значение выражений и ключевых слов из больших текстовых коллекций, таких как новостные статьи и книги.

Основная идея Word2Vec состоит в том, чтобы представить каждое слово как многомерный вектор, где положение вектора в этом многомерном пространстве отражает значение слова.

Word2Vec использует модель мелкой нейронной сети для изучения значения слов из большого массива текстов. В отличие от глубоких нейронных сетей, которые имеют несколько скрытых слоев, мелкие нейронные сети имеют только один или два скрытых слоя между входом и выходом. Это делает обработку быстрой и прозрачной. Неглубокая нейронная сеть Word2Vec может быстро распознавать семантические сходства и идентифицировать слова-синонимы, что делает ее быстрее глубоких нейронных сетей.

1.3.3. Алгоритм GloVe

GloVe (Global Vector)[2] — алгоритм обучения без учителя для получения векторных представлений слов. Обучение проводится на основе агрегированной глобальной статистики частоты совпадения слов из корпуса, и полученные представления демонстрируют интересные линейные подструктуры векторного пространства слов.

Преимущества GloVe:

— простая архитектура без нейронной сети;

— модель быстрая, и этого может быть достаточно для простых приложений;

— GloVe улучшает Word2Vec. Она добавляет частоту встречаемости слов и опережает Word2Vec в большинстве приложений;

— осмысленные эмбеддинги.

Недостатки алгоритма:

— хотя матрица совместной встречаемости предоставляет глобальную информацию, GloVe остаётся обученной на уровне слов и даёт немного данных о предложении и контексте, в котором слово используется;

— плохо обрабатывает неизвестные и редкие слова.

1.3.4. FastText

FastText[2] —— это созданная в Facebook библиотека, содержащая предобученные готовые векторные представления слов и классификатор, то есть алгоритм машинного обучения разбивающий тексты на классы.

К основной модели Word2Vec добавлена модель символьных n-грамм. Каждое слово представляется композицией нескольких последовательностей символов определённой длины. Например, слово they в зависимости от гиперпараметров, может состоять из “th”, “he”, “ey”, “the”, “hey”. По сути, вектор слова – это сумма всех его n-грамм.

Результаты работы классификатора хорошо подходят для слов с небольшой частотой встречаемости, так как они разделяются на n-граммы. В отличие от Word2Vec и Glove, модель способна генерировать эмбеддинги для неизвестных слов.

1.3.5. TF-IDF

TF-IDF (англ. term frequency-inverse document frequency) — статистическая мера, используемая для оценки важности слова в контексте документа. Большой вес в TF-IDF получают слова с высокой частотой в пределах конкретного документа и с низкой частотой употребления в других документах. Вычисление вес в TF-IDF состоит из трех этапов. Первым этапом выполняется подсчет доли вхождений каждого слова (TF) по формуле (1.1).

$$TF(word) = \frac{W(word)}{A}, \tag{1.1}$$

где W(word) — количество вхождений слова word в документ, A — количество всех слов в документе.

Вторым этапом выполняется измерение того, насколько важно это слово (IDF) по формуле (1.2).

$$IDF(word) = \log \frac{D}{DW(word)}, \tag{1.2}$$

где D — общее количество документов, DW(word) — количество документов, которые содержат слово word.

И третий этап совмещает первые две оценки.

$$TF - IDF(word) = TF(word) \cdot IDF(word). \tag{1.3}$$

TF-IDF предоставляет несколько ключевых преимуществ.

1. Учет важности слов: TF-IDF учитывает как частоту слова в документе,

так и его общую редкость по всей коллекции. Таким образом, он помогает выделять ключевые слова, которые часто встречаются в данном документе, но не слишком распространены в остальных.

2. Устранение шума: Слова, которые встречаются в большинстве документов (стоп-слова), имеют низкий IDF и, следовательно, низкий общий вес TF-IDF. Это позволяет устранить шум и фокусироваться на более важных словах.

1.4. Основные методы классификации текстов

1.4.1. Логистическая регрессия

При совместном обучении с учителями логистическая регрессия — это выбор наилучших параметров для маркировки для получения эффективных результатов классификации. Логистическая регрессия похожа на линейную регрессию, поскольку нужно найти значения коэффициентов входных переменных. Разница декоммутируется в том, что выходные значения преобразуются с использованием нелинейных или логистических функций.

В зависимости от того, как вы обучаете свою модель, можно использовать оценки логистической регрессии для отображения вероятностей выборок, относящихся к классу 0 или 1. Это полезно, когда нужно повысить точность прогнозирования.

Как и в случае с линейной регрессией, логистическая регрессия лучше справляется со своей задачей, если вы удаляете ненужные аналогичные переменные. Модели логистической регрессии быстро усваиваются и подходят для задач двоичной классификации.

1.4.2. Наивный байесовский классификатор

Это вероятностный классификатор, обычно используемый в машинном обучении, но также может использоваться в качестве статистического метода. В основном он используется для предварительной обработки данных, поскольку его легко вычислить. Для прогнозирования целевого класса используются байесовское рассуждение и вероятностный вывод. Поскольку атрибуты играют важную роль в классификации, производительность может быть улучшена путем присвоения атрибутам различных весовых коэффициентов [3].

Производительность наивного байесовского классификатора зависит от точности оценки условной вероятности. Без достаточного количества обучающих данных трудно точно оценить эти условия. Поэтому для оценки условных вероятностей используются некоторые методы метаэвристики, такие как генетические алгоритмы и дифференциальная эволюция. В некоторых случаях преимущества этого классификатора ставятся под сомнение из-за условного предположения о независимости между атрибутами, которые влияют на эффективность классификации [3]. Используются различные методы мета-обучения для повышения производительности, включая расширение структуры, выбор атрибутов, преобразование частоты, взвешивание атрибутов, взвешивание экземпляров и локальное обучение. Таким образом, наивный байесовский классификатор прост в реализации и в то же время эффективен с точки зрения надежности. Эти особенности делают классификаторы подходящими для решения задач обработки естественного языка.

1.4.3. Метод опорных векторов

Метод опорных векторов (англ. support vector machine, SVM) — один из наиболее популярных методов обучения, который применяется для решения задач классификации, регрессии и обнаружения выбросов. Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. В двумерном пространстве эту оптимальную гиперплоскость можно представить как линию, разделяющую пространство на две части: где одна часть пространства содержит точки данных, которые принадлежат одному классу, а другая часть пространства содержит точки данных, принадлежащие другому класс. Концепция линий, действующих как классификатор, верна только в том случае, если точки данных линейно разделимы. SVM также можно использовать для поиска оптимальной кривой, которая может использоваться для классификации точек данных, которые нельзя разделить линейно.

Преимущества метода:

— хорошо работает с пространством признаков большего размера;

— метод находит разделяющую полосу максимальной ширины, позволяет в дальнейшем осуществлять более уверенную классификацию;

— метод использует подмножество обучающих точек в функции принятия

решений (называемых опорными векторами), поэтому это также эффективно с точки зрения памяти.

Недостатки метода:

— долгое время обучения (для больших наборов данных);

— неустойчивость к шуму: выбросы в исходных данных становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости.

1.4.4. Decision Tree and Random Forest

Деревья решений (англ. Decision Tree) являются одними из самых ранних и популярных классификаторов[4]. Структура этого метода представляет собой иерархическую декомпозицию пространства данных[5]. Основная идея заключается в создании дерева на основе атрибута для категоризированных точек данных, но основная задача дерева решений заключается в том, какой атрибут или функция может находиться на родительском уровне, а какой должен быть на дочернем уровне[6].

Деревья решений работает в определенной последовательности, чтобы проверить соответствие решения определенному пороговому значению среди доступных значений. Тестирование выполняется в соответствии с определенными логическими правилами, аналогичными весам нейронной сети. На этапе роста дерева обучающий набор разбивается на части, а на этапе обрезки его данные суммируются. Деревья, основанные на ансамбле, используют методы расширения возможностей и упаковки для объединения нескольких классификаторов, которые используют разные правила принятия решений для разных наборов данных [7]. Эти ансамбли показали выдающуюся производительность по сравнению с традиционными деревьями решений, но вычислительные затраты увеличиваются по мере добавления входных запросов [7].

Дерева решений обладают низкой эффективностью при обработке многомерных данных. Для решения этой проблемы предлагаются кластерные деревья. Инкрементные деревья принятия решений лучше всего подходят для потоков данных, поскольку они обладают способностью стабилизироваться в соответствии с накапливающимися данными. Они использует несколько атрибутов для обучаемых функций.

Ниже, на рисунке 1.2, представлен принцип работы метода деревьев решений.

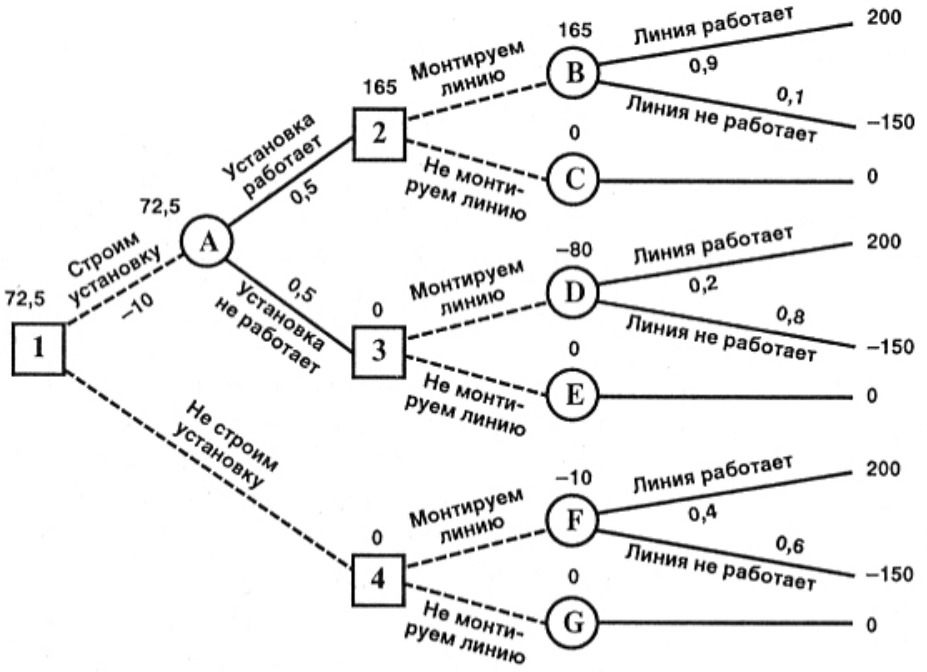

Рисунок 1.2 – Принцип работы метода деревьев решений

Метод случайных лесов (Random Forest) —— это метод обучения для классификации текста. Случайные леса представляют собой наборы деревьев решений, обученных с использованием случайных подмножеств признаков, которые достигли гораздо более высокой производительности и чаще используются на практике. Этот метод очень быстро обучается работе с наборами текстовых данных по сравнению с другими методами, такими как глубокое обучение, но довольно медленным для создания прогнозов после обучения[8]. Таким образом, чтобы добиться более быстрой структуры, количество деревьев в лесу необходимо уменьшить, поскольку большее количество деревьев в лесу увеличивает временную сложность на этапе прогнозирования.

1.4.5. Метод K-ближайших соседей

Метод K-ближайших соседей (англ. K-nearest neighbors, KNN) — это простой алгоритм машинного обучения с учителем, который можно использовать для решения задач классификации и регрессии.

Алгоритм K-NN сохраняет все доступные данные и классифицирует новую точку данных на основе сходства. Это означает, что когда появляются но-

вые данные, их можно легко классифицировать по категории наборов с помощью алгоритма K-NN.

Согласно принципу алгоритма KNN, структура классификатора включает в себя 4 параметра: данные для классификации, набор выборочных данных, набор выборочных меток и значение K. Затем вычислить расстояние между новыми данными и выборочными данными, упорядочить расстояния от наименьшего к наибольшему, возьмите первые K ближайших данных. Наиболее часто встречающаяся метка может быть идентифицирована как новая метка данных путем определения количества вхождений каждого введенного типа данных в K первых точках.

Ниже, на рисунке 1.3, представлен принцип работы метода K-ближайших соседей.

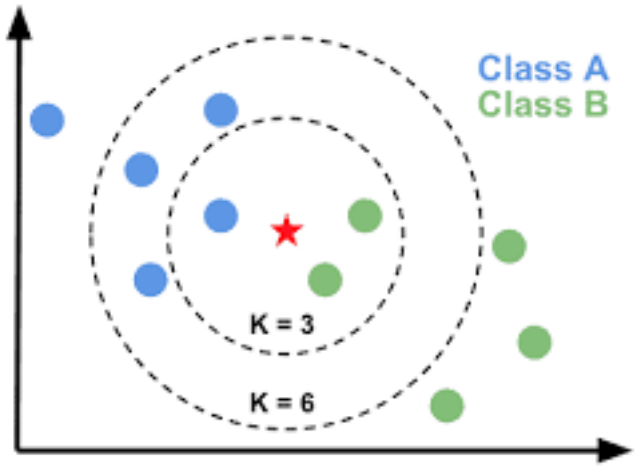

Рисунок 1.3 – Принцип работы метода K-ближайших соседей

Преимущества метода.

1. Алгоритм прост и легко реализуем.
2. Нет необходимости строить модель, настраивать несколько параметров или делать дополнительные допущения.
3. Алгоритм универсален. Его можно использовать для обоих типов задач: классификации и регрессии.

Недостатки метода.

1. Из аргумента выше следуют большие вычислительные затраты во время выполнения.

2. Алгоритм работает медленнее при увеличении объема выборки.

3. Всегда нужно определять оптимальное значение k.

1.4.6. Искусственные нейронные сети

Искусственные нейронные сети (англ. Artificial Neural Network, ANN) имитируют работу человеческого мозга при принятии решений. Они работают, обучаются и развиваются с минимальным вмешательством человека или без него. Для классификации данных был предложен конкурентный алгоритм коэволюции, основанный на модели нейронной сети. Радиальные базовые функции являются компонентом ANN, поскольку они используют более быстрые алгоритмы обучения. Эта функция имеет компактную сетевую архитектуру, повышающую точность классификации. Кроме того, развив октябрьские алгоритмы, как правило, хорошо работают в динамических средах, адаптируются на лету и адаптируются к ”нечетким”свойствам [9].

Нейронные сети также используются, когда требуется иерархический подход к классификации по нескольким тегам. Этот тип классификации означает, что каждая выборка может принадлежать более чем одному классу, и 1 уровень прогнозирования может использоваться в качестве входных данных для принятия окончательного решения на следующем уровне [10].

ANN обладает отличной прикладной ценностью и потенциалом для разработки, а поскольку он не требует обучения отдельных бинарных классификаторов для задач с несколькими наборами, он создает лучший базовый классификатор с помощью подхода сообщества.

Сверточная нейронная сеть (англ. Convolutional neural network, CNN) —— это архитектура глубокого обучения, которая обычно используется для иерархической классификации документов [11]. Хотя CNN изначально были созданы для обработки изображений, они также эффективно использовались для классификации текста[12].

В базовой CNN для обработки изображений тензор изображения свернут с набором ядер размера d × d. Эти слои свертки называются картами объектов и могут объединяться для предоставления нескольких входных фильтров[13]. Чтобы снизить сложность вычислений, CNN используют пуллинг для уменьшения размера выходных данных от одного уровня сети к другому. Различные методы объединения используются для уменьшения выходных данных при сохра-

нении важных функций[14]. Чтобы передать объединенные выходные данные составных избранных карт на следующий слой, карты сводятся в один столбец. Последние слои CNN обычно полностью связаны. В общем, на этапе обратного распространения ошибки сверточной нейронной сети корректируются как веса, так и фильтры детектора признаков. Потенциальная проблема, которая возникает при использовании CNN для классификации текста, это количество «каналов» S (размер пространства признаков). Хотя приложения классификации изображений обычно имеют мало каналов (например, только 3 канала RGB), S может быть очень большим (например, 50000) для приложений классификации текста, что приводит к очень высокой размерности[15].

Было предложено множество подходов, одним из самых популярных является TextCNN[16], сравнительно простая модель на основе CNN с однослойной структурой свертки, которая размещается поверх вложений слов.

1.5. Сравнение основные методов классификации текста

Различные алгоритмы, обсуждавшиеся в этом разделе, суммированы в соответствии с их преимуществами и недостатками ниже, в таблице 1.1.

Таблица 1.1 – Обзор различных методов классификации текста

Метод	*Преимущества*	*Недостатки*
Логистическая регрессия	Простая оценка параметров, хорошо работает для категориальных прогнозов	Требуется большой размер выборки, не подходит для нелинейных задач
Наивный байесовский классификатор	Быстрый классификатор, требует меньшего времени обучения	Отсутствие взаимодействия между признаками
Метод опорных векторов (SVM)	Эффективность в многомерных пространствах, простая реализация	Выбор наилучшего ядра, а также время, затраченное на обучение и тестирование
Дерево решений	Простота интерпретации, легко визуализировать, быстрота обучения и прогнозирования	Чувствительность к шумам входных данных, сложен для неопределенных и многозначных атрибутов
Метод К-ближайших соседей	Более простая реализация, гибкий выбор функций, хорошо подходит для много классовых задачи	Поиск ближайших соседей и оценка оптимального значения k
Нейтронные сети	Простота использования, скорость реализации, приближены по возможностям к любым предыдущим алгоритмам	Требует больших обучающих и тестовых данных, большая часть операций скрыта и труднодоступна для повышения точности

1.6. Постановка задачи

В настоящей работе проектируется метод классификации новостных текстов с помощью машинного обучения, а именно – метод опорных векторов. Набор данных для создания и обучения должен быть подготовлен для использования классификатором — а именно, должно быть установлено уникальное соответствие между новостным текстом и тематикой согласно разметке, представленной в наборе. Набор данных должен быть предварительно обработан и извлечены признаки. Обученный классификатор используется для распознава-

ния тематика новостных текстов.

Ниже, на рисунке 1.4, представлена IDEF0-диаграмма нулевого уровня.

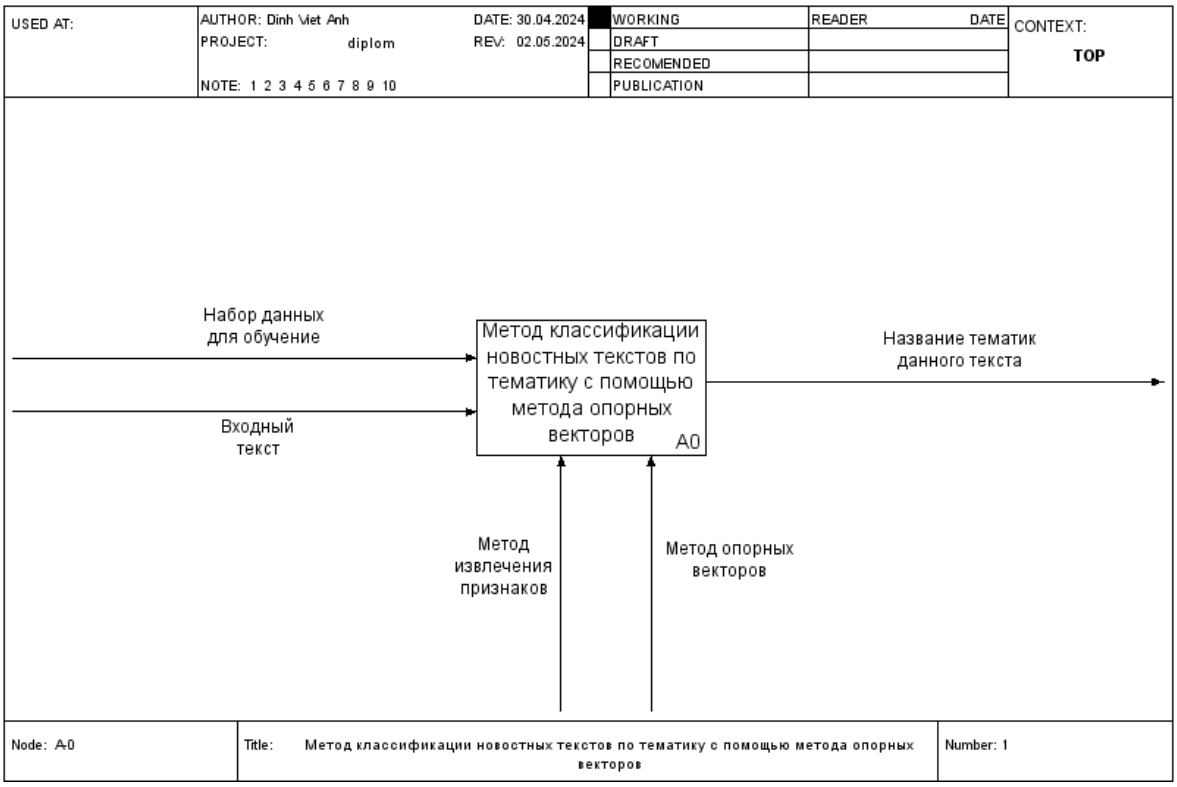

Рисунок 1.4 – IDEF0-диаграмма нулевого уровня

Вывод

В данном разделе был проведен анализ предметной области. Были рассмотрены методы очистки и предварительной обработки текста. Был проведен обзор методов извлечения признаков из текста. Был рассмотрены и проанализированы основные методы классификации текстов. Была представлена формализованная постановка задачи в виде IDEF0-диаграммы.

В разрабатываемом методе предлагается использовать подход, основанный на использовании TF-IDF для извлечения признаков, для классификации текстов использован метод опорных векторов.

2 Конструкторский раздел

В этом разделе описывается этапы работы алгоритма разрабатываемого метода, также рассматриваются функциональные схемы обучения модели и метода классификации. Также приводятся метрики, используемые для оценки качества работы обученной модели. Описывается используемый набор данных для обучения модели и рассматривается структура программного обеспечения.

2.1. Структура программного обеспечения

Разрабатываемый программный продукт состоит из следующих частей.

1. Модуль предобработки текстов — отвечает за очистку и предварительную обработку текстов.
2. Модуль классификации — отвечает за извлечение признаков из текстов, обучение модели и прогнозирование класса новых данных.
3. Модуль создания выборки данных — отвечает за создание файлов из скачанного набора данных.
4. Модуль пользовательского интерфейса — отвечает за предоставление пользовательского интерфейса.

Внутри модуля предобработки текстов реализованы токенизация, удаление стоп-слов и лемматизация.

2.2. Описание этапы работы алгоритма

2.2.1. Очистка и предобработка набора данных

Прежде всего, набор данных необходимо очистить и предварительно обработать, поскольку это необходимый этап, как упоминалось в предыдущем разделе. Очистка текста включает в себя преобразование текста в нижний регистр, удаление лишних пробелов и небуквенно-цифровых символов.

Предобработка текстовых данных состоит из трех этапов: токенизация, удаление стоп-слов и лемматизация.

После токенизации исходный текст преобразуется в массив, элементами которого являются отдельные слова этого текста (токены). Текст будет сохранен как массив слов в том же порядке, что и в исходном тексте.

После удаления стоп-слов массив токенов уменьшит количество элементов, но не сильно повлияет на результат обучения модели, поскольку стоп-слова при удалении не потеряют смысла исходного текста.

После процесса лемматизация массив токенов будет преобразован в массив, элементы которого являются словами в их начальной форме с соответствующим типом слова.

2.2.2. Формирование матрицы признаков

Для извлечения признаков из массива токенов используется матрица признаков с помощью метода TF-IDF.

Сначала из массива слов будет создан список уникальных слов (словарь). На основе этого словаря рассчитывается значение IDF каждого слова в словаре. Затем для каждого текста в наборе данных рассчитывается вектор значений TF-IDF. Значения IDF и TF-IDF рассчитываются по формулам, рассмотренным в предыдущем разделе. Из векторов признаков создается матрица признаков, в которой каждая строка соответствует одному вектору признака одного текста после нормализация.

Каждый вектор признака нормализуется по формуле:

$$\vec{v}_{norm} = \frac{\vec{v}}{\sqrt{\sum_{i=1}^{n} v_i^2}} = (\frac{v_1}{\sqrt{\sum_{i=1}^{n} v_i^2}}, \frac{v_2}{\sqrt{\sum_{i=1}^{n} v_i^2}}, ..., \frac{v_n}{\sqrt{\sum_{i=1}^{n} v_i^2}}), \qquad (2.1)$$

где $\vec{v} = (v_1, v_2, v_3, ..., v_n)$ — вектор признака, $\sqrt{\sum_{i=1}^{n} v_i^2}$ — длинна вектора $\vec{v}$, $\vec{v}_{norm}$ — нормализованное вектор признака $\vec{v}$.

2.2.3. Обучение классификатора

Для выполнения классификации был выбран метод опорных векторов, потому что этот метод является одним из применимых алгоритмов машинного обучения, который используется для различных задач классификации и особенно подходит для данных большого размера[17]. SVM — линейный классификатор, основан на разделении множества векторов из n-мерного пространства гиперплоскостью. Этот метод требует обучения перед классификацией.

Поскольку в размеченном наборе данных есть несколько тематик, необходимо использовать многоклассовый классификатор. Для создания многоклас-

сового классификатора используется подход ”один против всех”(англ. one and rest). Этот подход заключается в подборе одного классификатора на класс. Для каждого классификатора класс сопоставляется со всеми другими классами. Таким образом, задачи многоклассовой классификации преобразуется в бинарную кластеризацию.

Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки. То есть нужно найти гиперплоскость, которая задается следующим уравнением:

$$(w, x) - b = 0, \tag{2.2}$$

где w — вектор, b — число.

Тогда очевидно, что документы одного класса должны удовлетворять $(w, x_i) >= b$, а другого $(w, x_j) <= b$. Зафиксируем гиперплоскость, тогда параллельные гиперплоскости, содержащие опорные вектора, будут выглядеть таким образом:

$$\begin{cases} (w, x) = b + \epsilon, \\ (w, x) = b - \epsilon. \end{cases} \tag{2.3}$$

Ниже, на рисунке 2.1, представлен принцип работы метода SVM.

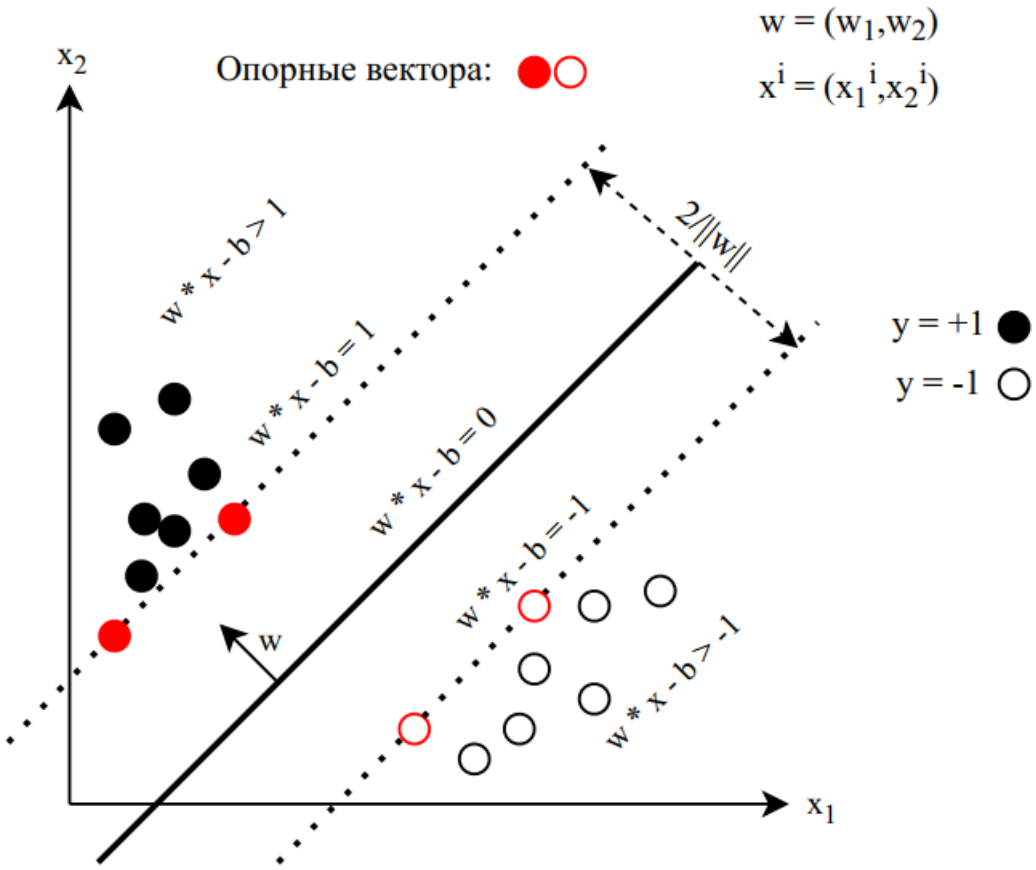

Рисунок 2.1 – Принцип работы метода опорных векторов

Отклонение ϵ одно и тоже в силу того, что расстояние до опорных векторов будет одинаковым, иначе такая гиперплоскость точно не будет оптимальной.

После нормировки, а именно поделим вектор с и число d на ϵ, получается:

$$\begin{cases} (w', x) - b' = 1, \\ (w', x) - b' = -1. \end{cases} \tag{2.4}$$

Расстояние между двумя параллельными гиперплоскостями можно вычислить следующим образом:

$$dist = \frac{|b_1 - b_2|}{||w||}, \tag{2.5}$$

где b_1 — свободный член первой гиперплоскости, b_2 — свободный член второй гиперплоскости, $||w|| = w \cdot w$ — одинаков, вследствие параллельности гиперплоскостей.

Тогда несложно убедиться, что расстояние от параллельных гиперплоскостей до оптимальной гиперплоскости равно $\frac{1}{||w||}$. Теперь, чтобы найти оптимальную гиперплоскость, нужно решить следующую задачу оптимизации:

$$\begin{cases} ||w|| \longrightarrow min, \\ (w, x_i) - b \geq 1, \text{если } y_i = 1, \\ (w, x_j) - b \leq -1, \text{если } y_j = -1. \end{cases} \tag{2.6}$$

Или в более удобной записи:

$$\begin{cases} ||w|| \longrightarrow min, \\ y_i \cdot [(w, x_i) - b] \geq 1, y_i \in \{-1; 1\}. \end{cases} \tag{2.7}$$

Эта задача сводится к двойственной задаче поиска седловой точки функции Лагранжа. Но если в случае отсутствия линейной разделимости требуется ввести набор дополнительных переменных δ_i, характеризующих величину ошибки на объектах $x_i \in \{x_1, ..., x_n\}$. Тогда задача оптимизации изменится следующим образом:

$$\begin{cases} ||w|| + \sum \delta_i \longrightarrow min, \\ y_i \cdot [(w, x_i) - b] \geq 1 - \delta_i, y_i \in \{-1; 1\}, \\ \delta_i \geq 0, i \in \{1, ..., n\}. \end{cases} \tag{2.8}$$

Здесь, δ_i равно нулю в том случае, если соответствующее неравенство выполняется и без него. То есть δ_i является минимально возможным значением, при котором неравенство будет выполнено. Решая эти задачи, можно получить вектор с, который и нужен для построения гиперплоскости.

Ниже, на рисунке 2.2, представлен вывод правил настройки весов.

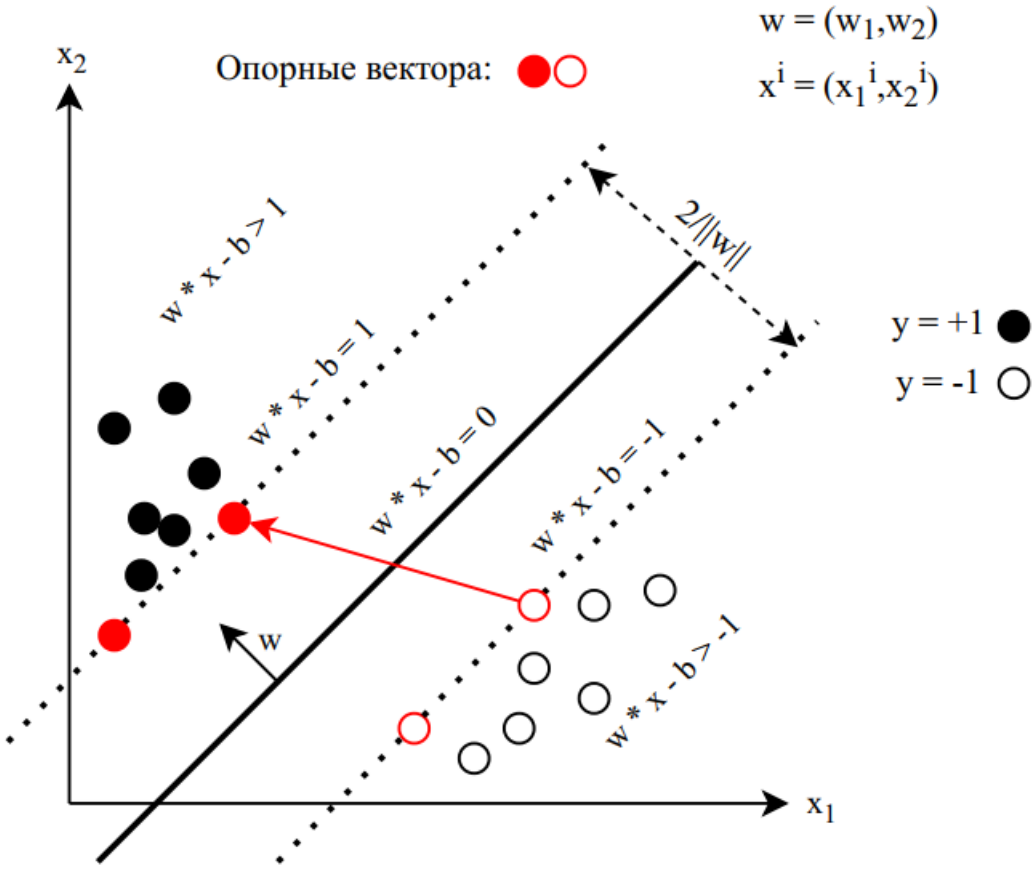

Рисунок 2.2 – Вывод правил настройки весов

2.2.4. Классификации текстов

На данном этапе обученная модель получает на вход новостной текст. Этот текст необходимо очистить, предварительно обработать и извлечь признаки, используя те же этапы, что и для набора обучающих данных, чтобы получить соответствующие признаки. Входной текст преобразуется в вектор признаков и с помощью обученной модели прогнозирует тематику исходного новостного текста.

2.3. Метрики оценки качества классификации текстов

Показатели критериев оценки качества классификации, будут базироваться на следующих основных метриках результатов классификации на тестовой выборке.

— TP (англ. True Positive) — истинный положительный результат. Верные данные, классифицированные как верные.

— TN (англ. True Negative) — истинный отрицательный результат. Неверные данные, классифицированные как неверные.

— FP (англ. False Positive) — ложный положительный результат. Верные данные, классифицированные как неверные.

— FN (англ. False Negative) — ложный отрицательный результат. Неверные данные, классифицированные как верные.

Для оценки качества модели, обученного на обучающей и тестовой выборках из набора данных, используются следующие две метрики: аккуратность и F1-мера.

Метрики аккуратности (англ. Accuracy) — один из наиболее простых, а поэтому и распространенной метрикой. Она показывает количество правильно проставленных меток класса (суммы истинно положительных и истинно отрицательных результатов) от общего количества данных и считается следующим образом:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.9}$$

F1-мера —– это показатель оценки, который измеряет качество работы модели классификации. Он сочетает в себе оценки точности и полноты модели.

Точность модели (англ. Precision) —– это показатель оценки модели и производительности, который соответствует доле значений, которые фактически принадлежат положительному классу, среди всех значений, которые, по прогнозам, принадлежат этому классу. Точность также известна как положительная прогностическая ценность (англ. positive predictive value, PPV). Оценка F1 использует точность, чтобы получить долю истинно положительных записей среди общего числа записей, классифицированных как положительные с помощью модели машинного обучения. Точность модели рассчитывается следующим образом:

$$Precision = \frac{TP}{TP + FP}. \tag{2.10}$$

Полноты модели (англ. Recall)— это показатель оценки модели и производительности, который соответствует доле значений, которые, по прогнозам, относятся к положительному классу, среди всех значений, которые действительно принадлежат к положительному классу (включая ложно отрицательные). F1-мера использует полноты, чтобы получить долю истинно положительных запи-

сей среди общего числа фактически положительных записей. Полноты модели рассчитывается следующим образом:

$$Recall = \frac{TP}{TP + FN}. \tag{2.11}$$

Значение F1-меры рассчитывается как гармоническое среднее значений точности и полноты модели, по следующей формуле:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}. \tag{2.12}$$

2.4. Функциональная схема обучения модели

Для обучения модели используется метод SVM, а для метода извлечения признаков из набора текстов используется метод TF-IDF. На входе поступает набор новостных текстов с соответствующей меткой для обучения, а на выходе — обученная модель на этом наборе.

Ниже, на рисунке 2.3 представлена функциональная схема обучения модели (с помощью SVM) в виде IDEF0-диаграммы. На этой схеме отображены этапы создания обучающей и тестовой выборок, очистки и предобработки текстов, извлечения признаков и выполнения многоклассовой классификации для обучающей выборки.

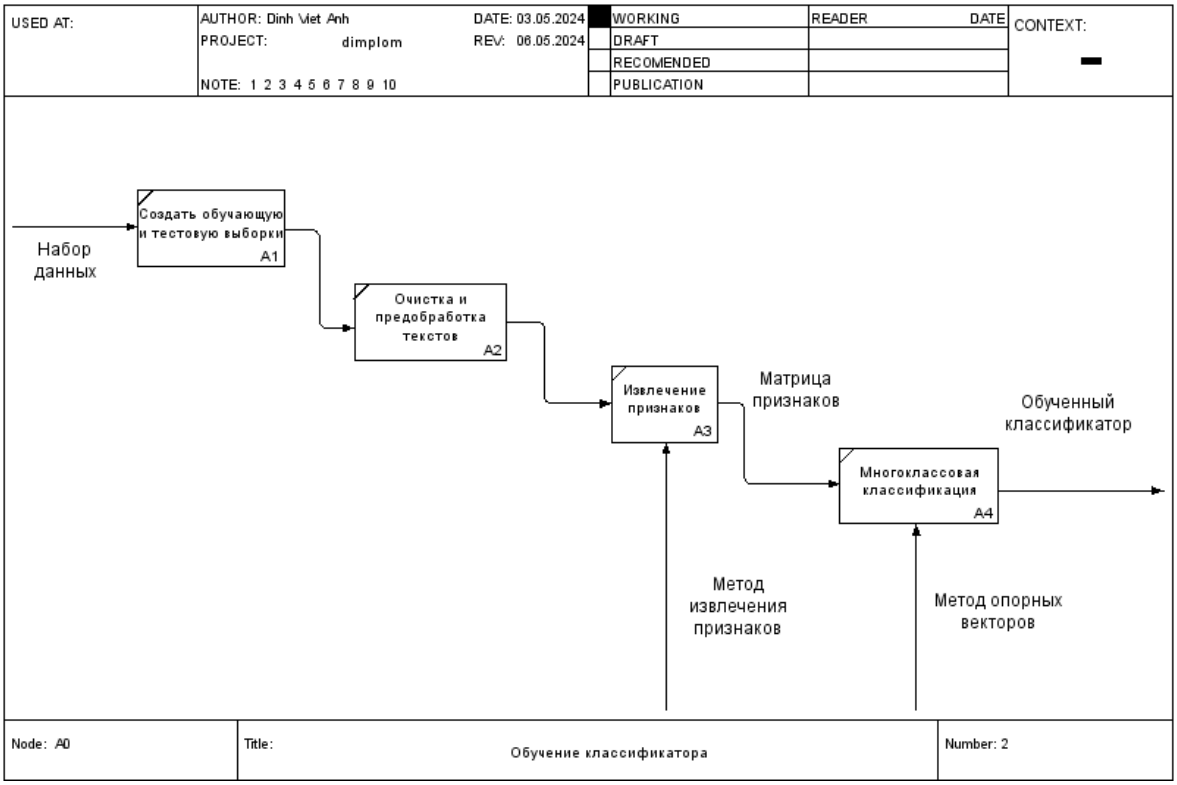

Рисунок 2.3 – Функциональная схема обучения модели

2.5. Функциональная схема метода классификации

Ниже, на рисунке 2.4 представлена функциональная схема метода классификации в виде IDEF0-диаграммы. На входе поступает новостной текст для классификации, а выходе — прогнозируемое название тематика этого текста.

На этой схеме отображены этапы очистки и предобработки входного текста, извлечения признаков и выполнения распознавания тематика текста, использующего обученную модель.

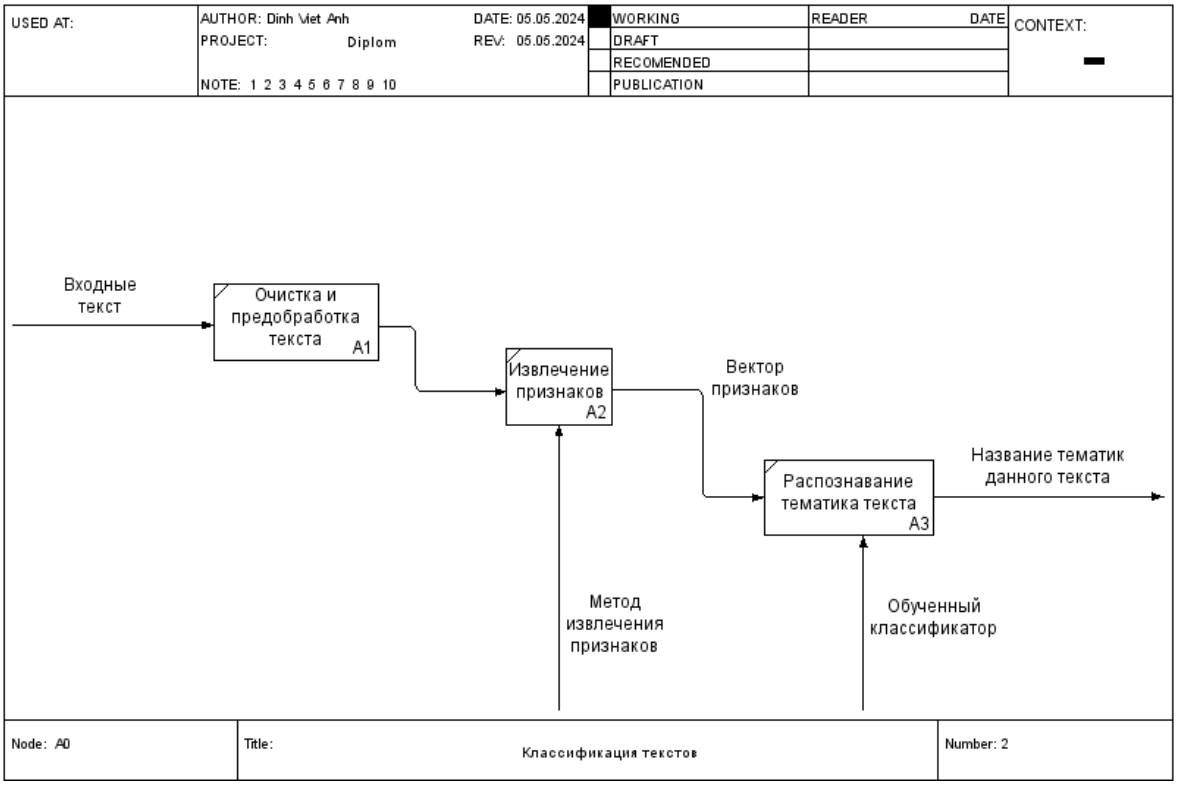

Рисунок 2.4 – Функциональная схема метода классификации

2.6. Описание используемого набора данных

Для обучения модели было решено использовать набор данных News dataset from Lenta.Ru [18]. Этот набор данных содержит более 800 тысяч новостей на русском языке, соответствующих более чем 20 тематикам, с сентября 1999 года по декабрь 2019 года. Источник новостей —– сайт lenta.ru [19], российское новостное интернет-издание, основанное в 1999 году. В наборе данных для каждой новости сохраняются текст, заголовок, тематик, дата публикации и ссылка на источник. Этот набор данных можно скачать с сайта Kaggle [20] — система организации конкурсов по исследованию данных, а также социальная сеть специалистов по обработке данных и машинному обучению.

После скачивания набора данных необходимо выполнить этап фильтрации данных, чтобы сохранить набор данных в виде файла CSV и создать наборы обучения для будущего использования.

Ниже, на рисунке 2.5, представлены примеры некоторых строк в наборе данных.

url	title	text	topic	tags	date
https://lenta.ru/news/1914/09/16/hungarnn/	1914. Русские войска вступили в пределы Венгрии	Бои у Сопоцкина и Друскеник закончились отступлением германцев. Неприятель, приблизившись с севера к...	Библиотека	Первая мировая	1914/09/16
https://lenta.ru/news/1914/09/16/lermontov/	1914. Празднование столетия М.Ю. Лермонтова отложено	Министерство народного просвещения, в виду происходящих чрезвычайных событий, признало соответственн...	Библиотека	Первая мировая	1914/09/16
https://lenta.ru/news/1914/09/17/nesteroff/	1914. Das ist Nesteroff!	Штабс-капитан П. Н. Нестеров на днях, увидев в районе Желтиева, в Галиции, летящий над нашим располо...	Библиотека	Первая мировая	1914/09/17
https://lenta.ru/news/1914/09/17/bulldogn/	1914. Бульдог-гонец под Льежем	Фотограф-корреспондент Daily Mirror рассказывает случай, который порадует всех друзей животных. Лейт...	Библиотека	Первая мировая	1914/09/17
https://lenta.ru/news/1914/09/18/zver/	1914. Под Люблином пойман швабский зверь	Лица, приехавшие в Варшаву из Люблина, передают, что туда доставлен	Библиотека	Первая мировая	1914/09/18

Рисунок 2.5 – Примеры некоторых строк в наборе данных

Вывод

В этом разделе были описаны этапы работы алгоритма разрабатываемого метода, также были рассмотрены функциональные схемы обучения модели и метод классификации. Также были приведены метрики, используемые для оценки качества работы обученного модели. Был описан используемый набор данных для обучения модели и была рассмотрена структура программного обеспечения.

3 Технологический раздел

В этом разделе приводится выбор языка программирования и средства программирования и рассматриваются необходимые библиотеки, которые используются для разработки программного обеспечения. Также описывается формат входных и выходных данных. Также приводятся описание пользовательского интерфейса и руководство пользователя для установки и использования программного обеспечения.

3.1. Выбор средств реализации программного обеспечения

3.1.1. Выбор языка программирования

Для реализации программного обеспечения был использован язык программирования Python[21]. Этот выбор обусловлен следующими причинами:

— это язык программирования высокого уровня, имеет простой синтаксис;

— наличие библиотек с открытым исходным кодом, позволяющих работать с естественным языком, выпонять классификации методом SVM и визуализировать данные;

— имеется навыки использования данного языка программирования, что сократить время написания программы.

3.1.2. Выбор среды программирования

Для разработки программы использовалась среда Visual Studio Code [22] в качестве среды разработки по следующим причинам:

— доступна бесплатная версия;

— имеет множество утилит, упрощающих написание кода;

— имеется навыки программирования при помощью данной среды, что сократить время написания программы.

Библиотека scikit-learn используется для работы с машинным обучением, потому что она предоставляет множество различных инструментов для разных

задач машинного обучения, включая кластеризацию, регрессию, кластеризацию, обучение и тестирование моделей, а также предварительную обработку данных, и так далее.

Для создания пользовательского интерфейса используется фреймворк Qt с помощью библиотеки PyQt5. Кроме того, в процессе разработки программного обеспечения библиотека pandas также используется для анализа набора данных, а библиотеки nltk и pymorphy2 — для очистки и предварительной обработки текстов.

3.2. Формат входных и выходных данных

Для модуля создания выборки данных входными данными является набор данных, скачанный с сайта kaggle.com, а выходные данные представляют собой файлы с указанным количеством строк для каждой темы.

На этапе обучения модели входными данными является файл, который создается из набора новостных текстов, в формате файла CSV. Каждая строка соответствует новости, состоящему из двух частей: текста и тематика новости. Каждый файл содержит определенное количество текста по каждой тематику.

На этапе классификации входными данными является текст или текстовый файл (в формате файла TXT), содержащий текст. Текст должен быть на русском языке, не менее 10 слов.

Результатом модуля классификации является прогнозируемое название тематика входного текста или содержимого файла и выводятся на пользовательский интерфейс программного обеспечения.

3.3. Руководство пользователя

Для запуска разработанного программного обеспечения требуется установить интерпретатор для Python и используемые библиотеки. Используемые в разработке библиотеки, которые необходимы для запуска ПО, приведены в файле requirements.txt, который находится в корневом каталоге проекта. С помощью пакетного менеджера pip все зависимости можно установить или обновить, запустив в терминале команду, приведенную в листинге 3.1.

Листинг 3.1 – Установка всех необходимых библиотек

```
pip install -r requirements.txt
```

Для работы библиотеки nltk необходимо скачать библиотеки. Для установки библиотеки nltk и скачивания библиотек нужно в коде программы выполнить команды, приведенные в листинге 3.2

Листинг 3.2 – Установка словарей nltk

```
import nltk
nltk.download()
```

Для создания выборки необходимо разместить папку с файлом исходного набора данных внутри корневого каталога проекта и запустить скрипт dataset.py. Скрипт может запущен из графического интерфейса среды разработки, либо командой из терминала, приведенную в листинге 3.3.

Листинг 3.3 – Команда для создания обучающих выборок

```
python dataset.py
```

Чтобы обучить модель на заранее созданных размеченных выборках и предсказать тематик входного текста или текстового файла, пользователь может воспользоваться графическим интерфейсом приложения. Для открытия приложения необходимо запустить скрипт main.py посредством интерфейса среды разработки или командой в терминале, приведенную в листинге 3.4.

Листинг 3.4 – Команда для запуска приложения

```
python main.py
```

В результате разработанное программное обеспечение может установить тематик текста или выдвинуть предположение, что может быть тематикой.

3.4. Интерфейс пользователя

Пользовательский интерфейс, который представлен на рисунке 3.1, был разработан с помощью программы Qt Designer. Интеграция с кодом на Python осуществлена посредством с помощью библиотеки PyQt5, которая предоставляет различные классы для работы с объектами пользовательского интерфейса.

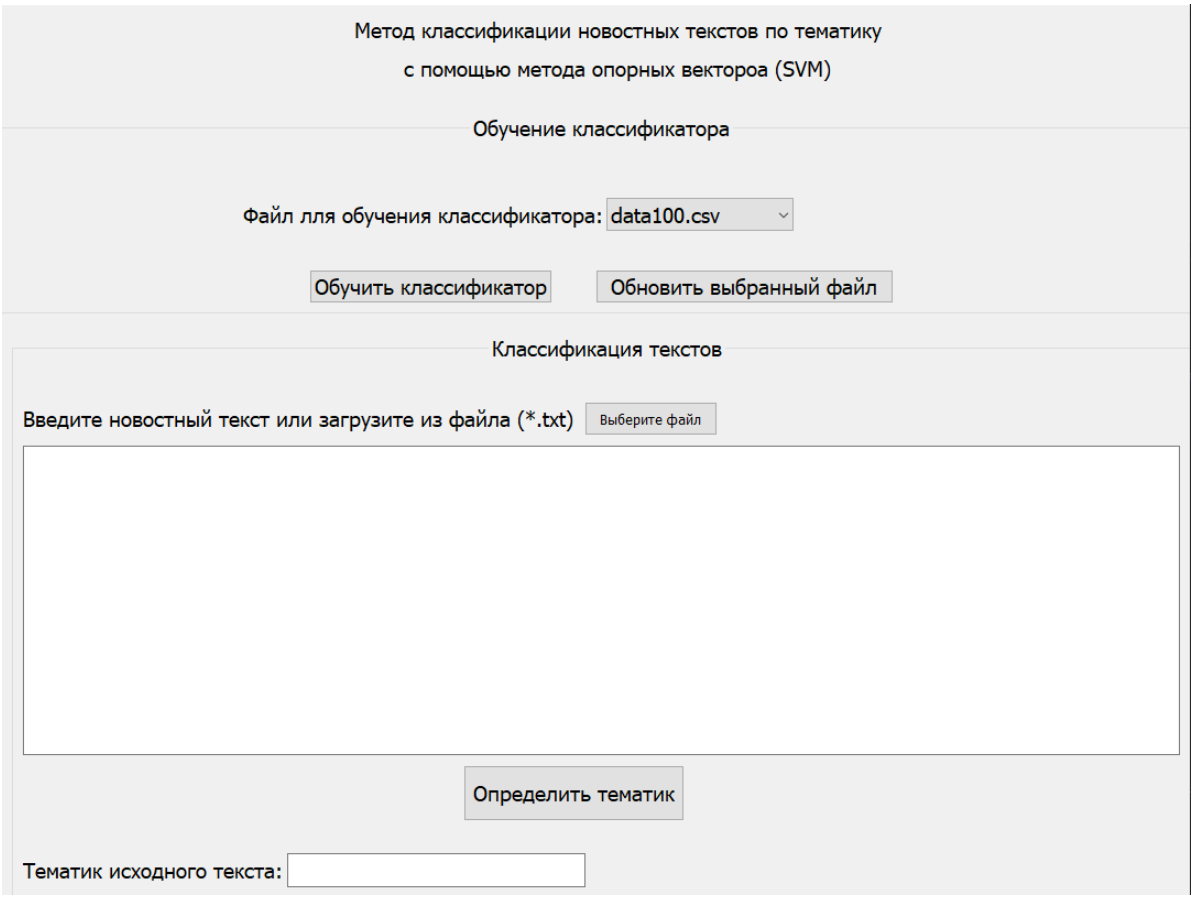

Рисунок 3.1 – Пользовательского интерфейса ПО

Пользовательский интерфейс программного обеспечения состоит из 2 частей. В верхней части находится функционал обучения модели. В этом окне пользователь может выбрать файл для обучения модели из поля выбора. Каждый файл содержит определенное количество текста по каждой тематику. Пользователь может обновить содержимое этих файлов с помощью кнопки «Обновить выбранного файла». При нажатии на кнопку «Обучить модель» выбранная выборка разбивается на две части: 80% корпуса — обучающая выборка, 20% — тестовая. Программа сообщит, когда модель завершит обучение , как показано на рисунке 3.2.

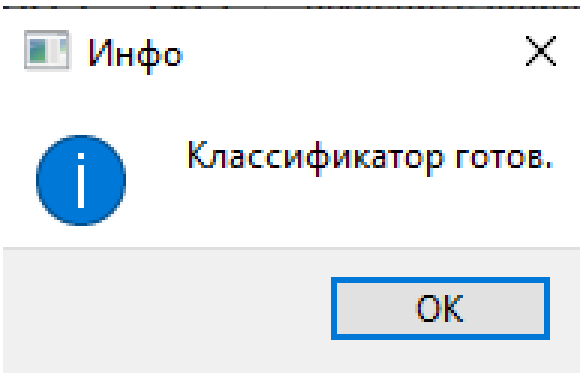

Рисунок 3.2 – Модель готова

В нижней части находится функционал предсказания тематика текста. Если пользователь попытается определить тематик, не обучив сначала модель, будет возвращена ошибка, как показано на рисунке 3.3.

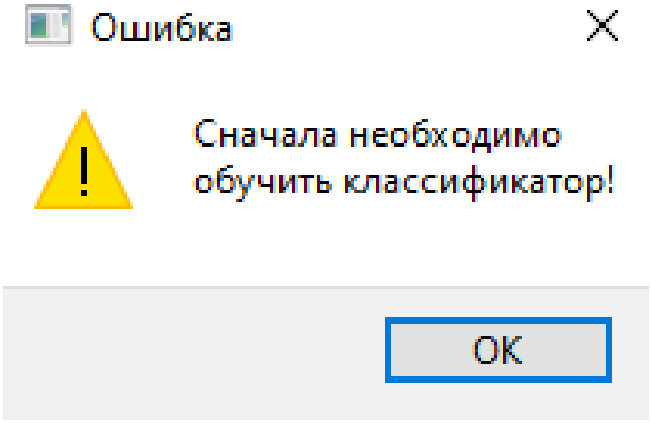

Рисунок 3.3 – Ошибка, возникающая в случае, если модель не обучена

Пользователь может ввести текст в поле ввода или выбрать файл. Программа поддерживает только txt файлы (текстовые файлы). При нажатии на кнопку «Выбрать файл» открывается файловая система компьютера, в которой необходимо выбрать интересующий текстовый файл. Когда пользователь выбирает файл, его содержимое отображается в текстовом поле. Если пользователь нажимает кнопку «Определить тематик», когда вводимый текст пуст или имеет неверный формат, будет возвращена ошибка, как показано на рисунке 3.4.

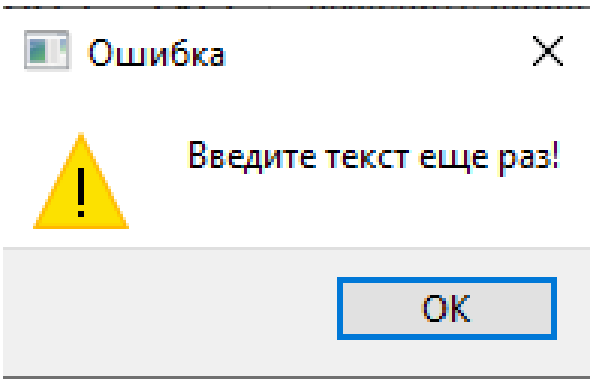

Рисунок 3.4 – Ошибка, возникающая в случае, если текстовое поле пусто

Если входной текст не пуст, то запускается алгоритм классификации. После того, как модель завершит работать, на экран будет выведено название наиболее вероятного тематика, как показано на рисунке 3.5.

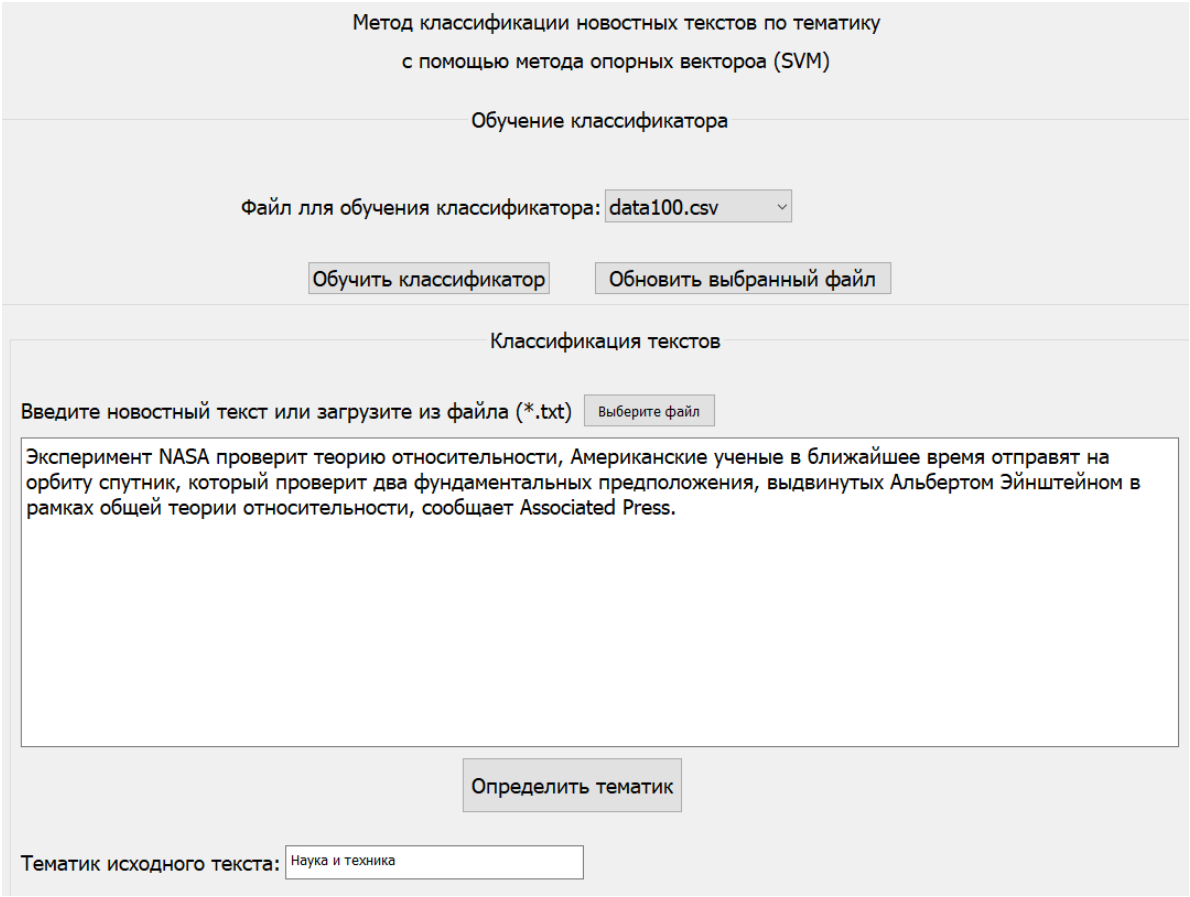

Рисунок 3.5 – Результат работы классификации

Вывод

В этом разделе были приведены выбор языка программирования и средства программирования и рассмотрены необходимые библиотеки, которые ис-

пользуются для разработки программного обеспечения. Также был описан формат входных и выходных данных. Также были приведены описание пользовательского интерфейса и руководство пользователя для установки и использования программного обеспечения.

4 Исследовательский раздел

4.1. Технические характеристики

Технические характеристики устройства, на котором выполнялись исследования разработанного метода:

— операционная система Window 10 Home Single Language [23];

— память 8 Гб;

— процессор 11th Gen Intel(R) Core(TM) i5-1135G7 2.42 ГГц, 4 ядра [24].

Во время выполнения исследований устройство было подключено к сети электропитания, нагружено приложениями окружения и самой системой замера.

Для проведения исследований разработанного метода используются выборки, созданные на основе текстов по 5 темам: Наука и техника, Спорт, Культура, Экономика и Мир. Количество текста по каждой тематике в каждой выборке используется одинаковое. Исходная выборка разбивается на обучающую (80% объема) и тестовую (20%) выборки.

Как отмечалось раньше, метрики аккуратности и F1-меры используются в качестве метрики для оценки качества классификатора.

4.2. Влияние размера обучающей выборки

Обучающая выборка — это набор данных, который используется для обучения модели машинного обучения. Она представляет собой подмножество общего набора данных, которое содержит примеры, сопоставленные с соответствующими целевыми значениями или метками. Размер обучающей выборки — фактор, влияющий на качество классификатора.

Для проведения данного исследования было проведено сравнение зависимости времени обучения и качества классификатора от количества текстов на обучающей выборке.

Результаты проведенного исследования приведены в таблицах 4.1 и 4.2.

Таблица 4.1 – Замеры времени обучения от размера выборки

Количество текстов в выборке	*Время обучения классификатора (с.)*
50	0.03
100	0.08
200	0.23
500	1.59
1000	5.67
2000	21.29
3000	47.39
4000	82.76
5000	135.10

Таблица 4.2 – Зависимость качества классификатора от размера выборки

Количество текстов в выборке	*Точность на тестовой выборке*	*F1-мера*
50	0.2	0.27
100	0.9	0.89
200	0.9	0.90
500	0.91	0.92
1000	0.917	0.93
2000	0.92	0.935
3000	0.94	0.937
4000	0.942	0.94
5000	0.943	0.943

Ниже, на рисунках 4.1 и 4.2, представлены данные из таблиц 4.1 и 4.2 соответственно в виде графика.

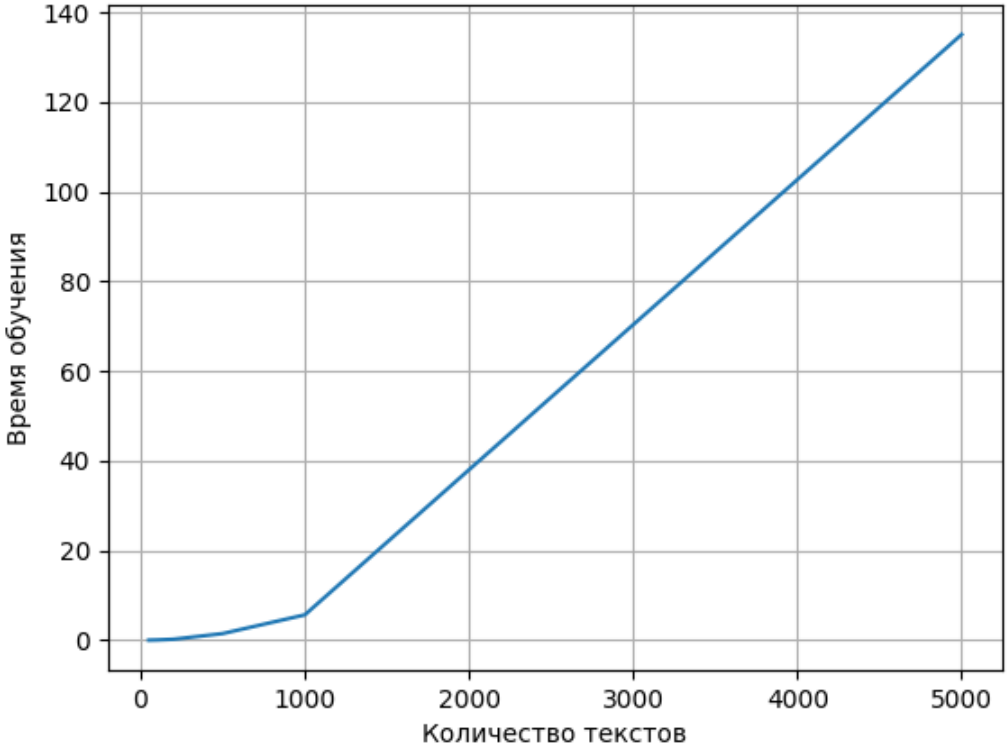

Рисунок 4.1 – График зависимости времени обучения от размера выборки

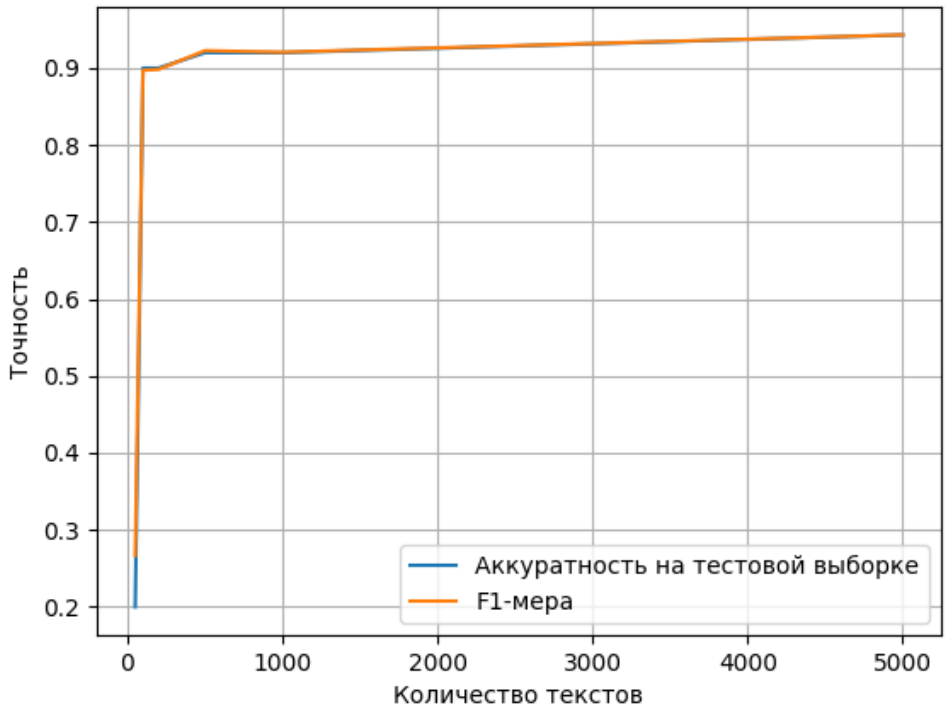

Рисунок 4.2 – График зависимости качества классификатора от размера выборки

На основании полученных результатов можно сделать вывод, что количество текстов в обучающей выборке влияет на время обучения и качество классификатора. При использовании выборок с меньшим количеством текста классификатор будет обучаться за меньшее время, но с низкой точностью. По мере увеличения количества текстов в обучающей выборке точность классификатора также увеличивается. Стоит отметить, что увеличение количества текстов увеличивает время обучения, но существенно не увеличивает точность классификатора после того, как количество текстов достигнет 1000.

Таким образом, можно обучить классификатор на выборке из 1000 текстов для оптимизации времени обучения и качества классификатора.

4.3. Влияние наличия стоп-слов в тексте

На этапе предварительной обработки текста происходит процесс удаления стоп-слов. Как отмечалось ранее, стоп-слова — это текстовые шумы или неинформативные слова, которые встречаются в большом количестве, но не имеют семантического значения и при игнорировании этих слов исходное предложение не теряет своего смысла.

Для проведения данного исследования было проведено сравнение зависимости качества классификатора от наличия процесса удаления стоп-слов на этапе предварительной обработки текстов.

Результаты проведенного исследования приведены в таблице 4.3.

Таблица 4.3 – Зависимость качества классификатора от наличия стоп-слов

Количество текстов	*Точность на тестовой выборке (Без)*	*Точность на тестовой выборке (С)*	*F1-мера (Без)*	*F1-мера (С)*
50	0.2	0.2	0.27	0.27
100	0.9	0.95	0.89	0.94
200	0.9	0.90	0.90	0.90
500	0.92	0.93	0.92	0.93
1000	0.92	0.91	0.92	0.91
5000	0.943	0.933	0.94	0.93

Ниже, на рисунках 4.3 и 4.4, представлены данные из таблицы 4.3 в виде графика.

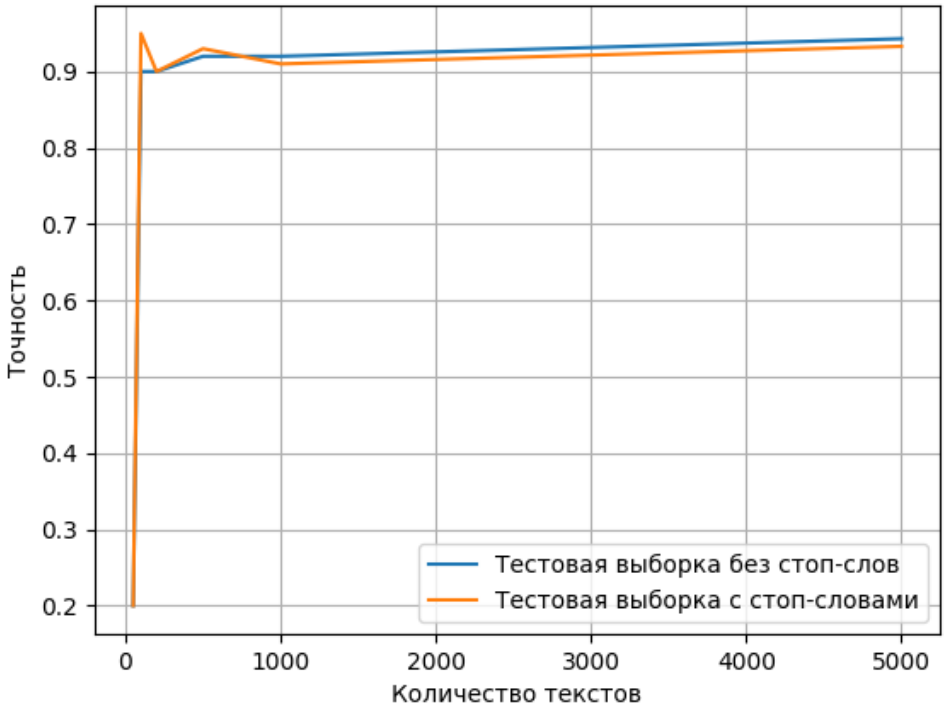

Рисунок 4.3 – График зависимости точности классификатора от наличия стоп-слов

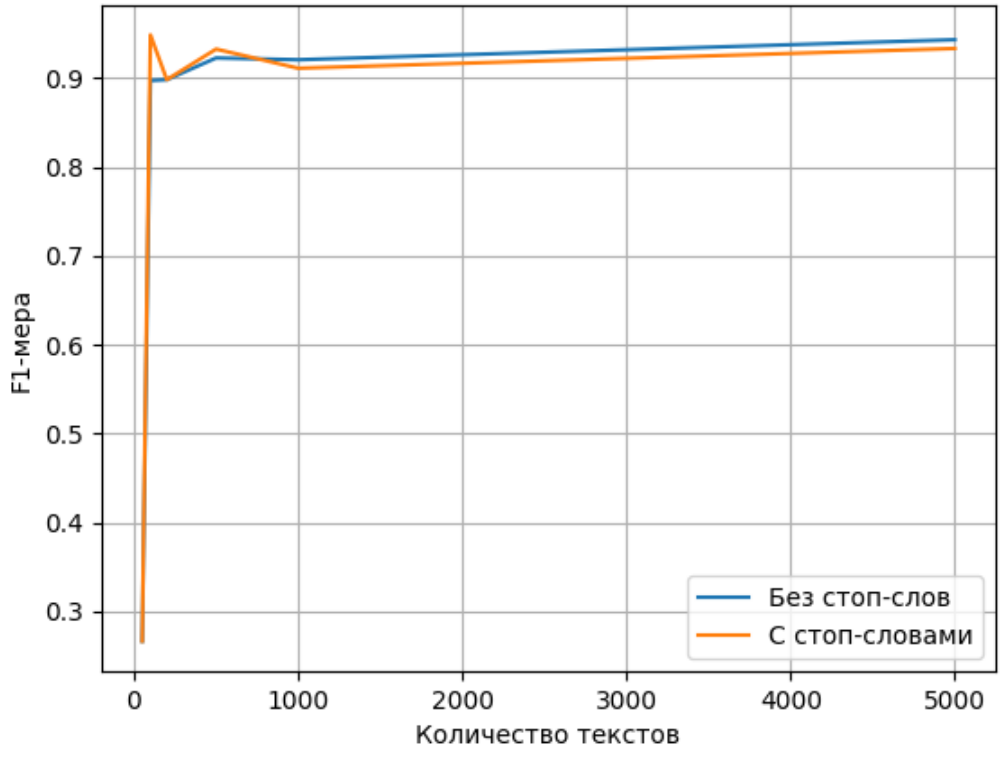

Рисунок 4.4 – График зависимости F1-меры от наличия стоп-слов

На основании полученных результатов можно сделать вывод, что наличие стоп-слов в тексте обучающей выборки не сильно влияет на качество классификатора. В обучающих выборках с небольшим количеством текста (с 50 до 500 текстов) с стоп-словами обученные на них классификаторы имеют несколько более высокую точность, чем классификаторы, обученныы на обучающих выборках без стоп-слов (на 1-2%). Но в обучающих выборках с большим количеством текстов (1000 или 5000 текстов) классификаторы, обученные на обучающих выборках без стоп-слов, имеют более высокую точность (на 1%).

4.4. Влияние ядра метода опорных векторов

Ядро (kernel) в методе опорных векторов определяет функцию, которая вычисляет скалярное произведение двух векторов в пространстве признаков. Оно позволяет проецировать данные в более высокомерное пространство, где они могут быть линейно разделимыми, даже если исходное пространство не является линейно разделимым.

Для проведения этого исследования было проведено сравнение зависимости качества классификатора от ядра метода опорных векторов при использовании следующих ядер: линейное, полиномиальное, Гауссово RBF и сигмоидальное.

Ниже, на рисунках 4.5 и 4.6, представлены результаты проведенного исследования

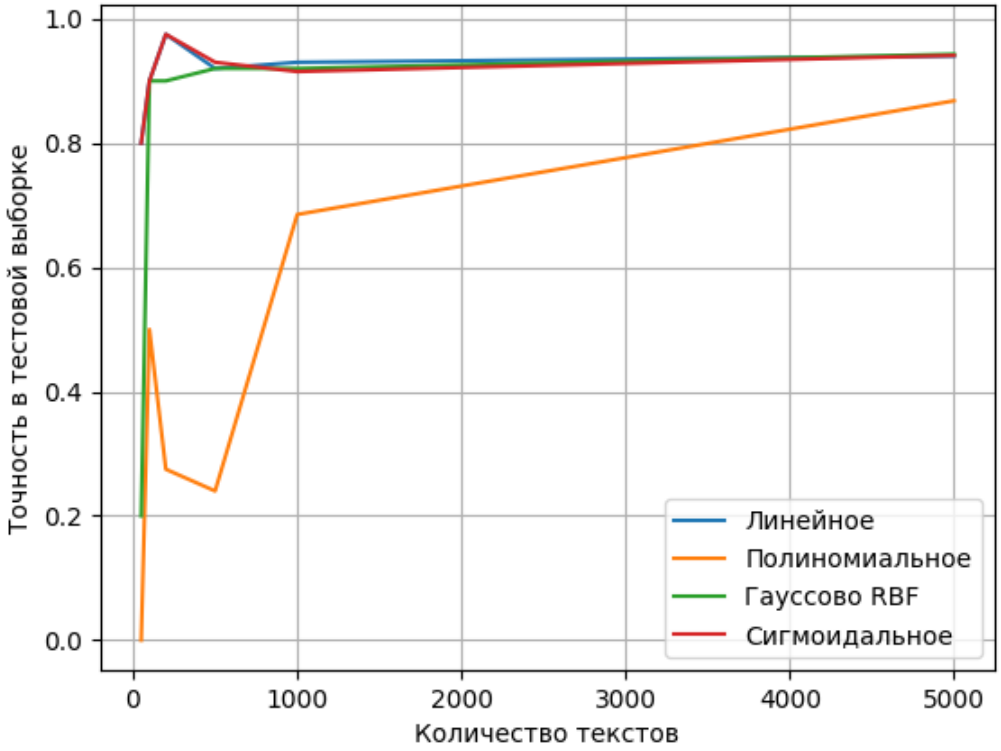

Рисунок 4.5 – График зависимости точности классификатора от ядер

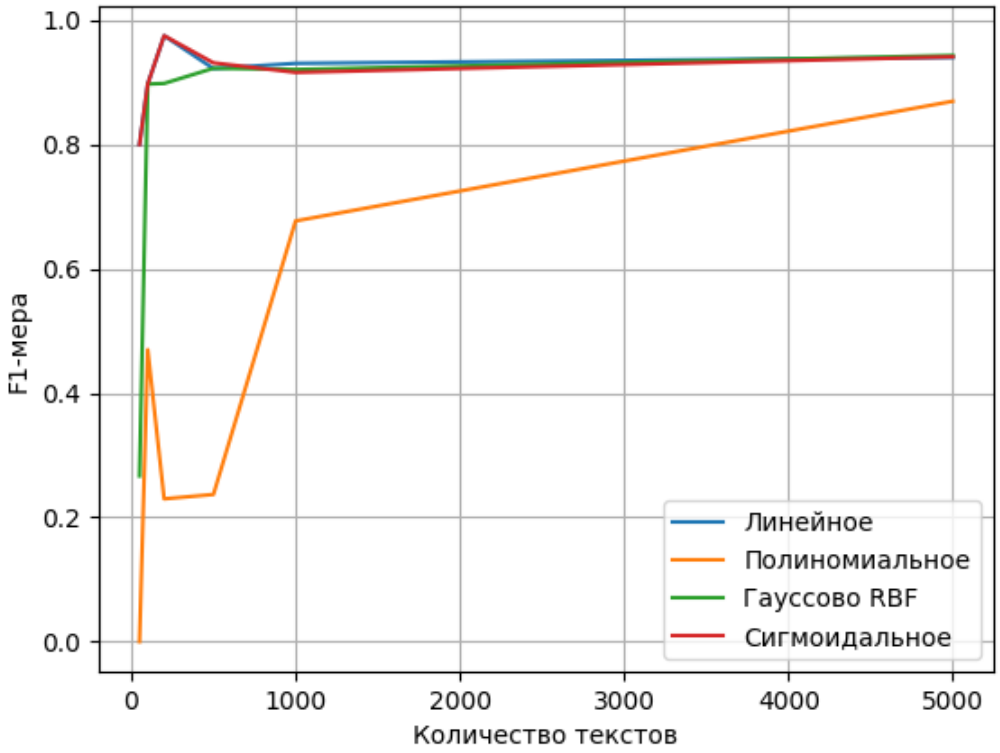

Рисунок 4.6 – График зависимости F1-меры от ядер

Из полученных результатов можно сделать вывод, что зависимости точности и F1-меры классификатора от различных ядер совпадают. Классификатор с полиномиальным ядром имеет наименьшее качество, а классификаторы с другими ядрами имеет примерно одинаковую точность.

Вывод

В этом разделе были описаны технические характеристики устройства для проведения исследований и приведены исследования разработанного метода. Из полученных результатов исследований можно сделать вывод, что при увеличении количества текстов в обучающей выборке время обучения классификатора будет быстро увеличиваться, но точность классификатора возрастает ненамного для обучающих выборок с большим количеством текстов. Результаты также показывают, что процесс удаления стоп-слов на этапе предварительной обработки текста не сильно влияет на качество классификатора. Использование различных ядер для классификатора показывает, что использование линейного ядра, гауссовского ядра RBF и сигмоидального ядра приводит к получению высокоточного классификатора.

ЗАКЛЮЧЕНИЕ

В рамках настоящей работы был разработан, реализован и исследован метод классификации новостных текстов по тематикам с использованием опорных векторов. Все поставленные задачи были выполнены.

Был проведен анализ предметной области. Были рассмотрены методы очистки и предварительной обработки текста. Был проведен обзор методов извлечения признаков из текста.

Был рассмотрены и проанализированы основные методы классификации текстов. Была представлена формализованная постановка задачи в виде IDEF0-диаграммы.

Были описаны этапы работы алгоритма разрабатываемого метода, также были рассмотрены функциональные схемы обучения классификатора и метод классификации.

Были приведены метрики, используемые для оценки качества работы обученного классификатора. Был описан используемый набор данных для обучения классификатора и была рассмотрена структура программного обеспечения.

Были приведены выбор языка программирования и средства программирования и рассмотрены необходимые библиотеки, которые используются для разработки программного обеспечения. Также был описан формат входных и выходных данных. Также были приведены описание пользовательского интерфейса и руководство пользователя для установки и использования программного обеспечения.

Были описаны технические характеристики устройства для проведения исследований и приведены исследования разработанного метода

Для реализованного метода можно предложить следующие развития:

— ускорение работы метода.

— добавление возможности работы с различными языками.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. A Survey on Text Classification: From Traditional to Deep Learning / Q. Li [и др.] // ACM Transactions on Intelligent Systems and Technology (TIST). — 2020. — Т. 13. — С. 1—41.
2. Text classification algorithms: A survey / K. Kowsari [и др.] // Information. — 2019. — Т. 10, № 4. — С. 150.
3. *Tsangaratos P.*, *Ilia I.* Comparison of a logistic regression and Naïve Bayes classifier in landslide susceptibility assessments: The influence of models complexity and training dataset size // CATENA. — 2016. — Т. 145. — С. 164—179.
4. *Whitney D. L.*, *Evans B. W.* Abbreviations for names of rock-forming minerals // American Mineralogist. — 2010. — Т. 95. — С. 185—187.
5. *C.C. A.* Mining Text Data. — Springer New York, 2012.
6. *Morgan J. N.*, *Sonquist J. A.* Problems in the Analysis of Survey Data, and a Proposal // Journal of the American Statistical Association. — 1963. — Т. 58. — С. 415—434.
7. *Nidhi V. G.* Recent Trends in Text Classification Techniques // International Journal of Computer Applications. — 2011. — Т. 35, № 6. — С. 45—51.
8. Social network analytics for contemporary business organizations / H. Bansal [и др.]. — IGI Global, 2018.
9. Statistical Topic Models for Multi-Label Document Classification / T. Rubin [и др.] // Mach Learn. — 2011. — Т. 88.
10. *Staš J.*, *Juhár J.*, *Hládek D.* Classification of heterogeneous text data for robust domain-specific language modeling // EURASIP Journal on Audio, Speech, and Music Processing. — 2014. — Т. 2014. — С. 1—12.
11. Recurrent Convolutional Neural Networks for Text Classification / S. Lai [и др.] // AAAI Conference on Artificial Intelligence. — 2015.
12. *Cun L. Y.* Deep Learning // Nature. — 2015. — Т. 521. — С. 436—444.
13. *Jasim D. S.* Data mining approach and its application to dresses sales recommendation // Researchgate. Net. — 2016.

14. *Scherer D.*, *Müller A. C.*, *Behnke S.* Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition // International Conference on Artificial Neural Networks. — 2010.

15. Gradient-based learning applied to document recognition / Y. LeCun [и др.] // Proc. IEEE. — 1998. — Т. 86. — С. 2278—2324.

16. *Kim Y.* Convolutional Neural Networks for Sentence Classification // Conference on Empirical Methods in Natural Language Processing. — 2014.

17. *Mirzamomen Z.*, *Kangavari M. R.* Evolving fuzzy min–max neural network based decision trees for data stream classification // Neural Processing Letters. — 2017. — Т. 45. — С. 341—363.

18. New dataset from Lenta.RU [Электронный ресурс]. — Режим доступа URL: https://www.kaggle.com/datasets/yutkin/corpus-of-russian-news-articles-from-lenta/data (дата обращения: 02.05.2024).

19. Lenta.ru - Новости России и мира сегодня [Электронный ресурс]. — Режим доступа URL: https://lenta.ru/ (дата обращения: 02.05.2024).

20. New dataset from Lenta.RU [Электронный ресурс]. — Режим доступа URL: https://www.kaggle.com/datasets/yutkin/corpus-of-russian-news-articles-from-lenta/data (дата обращения: 02.05.2024).

21. Welcome to Python.org [Электронный ресурс]. — Режим доступа URL: https://www.python.org/ (дата обращения: 23.04.2024).

22. Visual Studio Code. — Режим доступа URL: https://code.visualstudio.com/ (дата обращения: 23.04.2024).

23. Windows 10 «Домашняя для одного языка». — Режим доступа URL: https://tehnichka.pro/windows-10-home-for-one-language/ (дата обращения: 23.04.2024).

24. Intel Core i5-1135G7. — Режим доступа URL: https://www.notebookcheck-ru.com/Intel-Core-i5-1135G7-Testirovanie-processora.507554.0.html (дата обращения: 23.04.2024).

ПРИЛОЖЕНИЕ А

Презентация