**Министерство образования и науки 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 Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ*

## *НА ТЕМУ:*

## ***Метод определения признаков авторского стиля для текстов на русском языке***

Студент ИУ7-85Б _____ **О.П. Кондрашова**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР _____ **А.П. Ковтушенко**
(Подпись, дата) (И.О.Фамилия)

Консультант _____ **Л.Л. Волкова**
(Подпись, дата) (И.О.Фамилия)

Нормоконтролер _____ **Ю.В. Строганов**
(Подпись, дата) (И.О.Фамилия)

*2021 г.*

# РЕФЕРАТ

Расчётно-пояснительная записка, 80 страниц, 20 рисунков, 6 таблиц, 28 источников.

ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА, АНАЛИЗ ТЕКСТА, КЛАССИФИКАЦИЯ ТЕКСТОВ, МОРФОЛОГИЧЕСКИЙ АНАЛИЗ

Объектом разработки является метод определения признаков авторского стиля для текстов на русском языке и использующий его метод классификации.

Цель работы - разработка определения признаков авторского стиля для текстов на русском и реализация программного обеспечения, использующего предложенный метод.

В данной работе решаются следующие задачи:

- анализ предметной области и существующих методов;
- формулирование основных положений разрабатываемого метода определения признаков авторского стиля;
- описание метода классификации, использующего предлагаемый метод определения признаков авторского стиля;
- разработка ПО, реализующего оба метода;
- оценка точности классификации текстов по предложенным признакам.

Область применения – электронные библиотечные системы.

В первой части работы анализируются существующие подходы к решению поставленных задачи. Во второй части рассматривается спроектированная структура программного обеспечения. В третьей части описываются технологии, используемые при реализации программного обеспечения. В четвертой части проводятся исследования разработанного метода.

Направления для дальнейшего развития:

- улучшение качества работы морфологического анализа текста;

- создание обучающей выборки, обеспечивающей получение более высоких значений метрик качества;
- ускорение работы метода.

# ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

**N-грамма** — это последовательность из N идущих подряд слов в тексте.

**Частеречная N-грамма** — связка из частей речи, то есть кортеж из частей речи, представляющих кортеж словоформ.

**Лемма** — это начальная форма слова.

**Граммема** — это грамматическое значение, понимаемое как один из элементов грамматической категории; различные граммемы одной категории исключают друг друга и не могут быть выражены вместе.

**Стоп-слова** — это популярные слова, встречающиеся в каждом тексте (например, предлоги или союзы) и не несущие в себе никакой важной информации с точки зрения решаемой задачи.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

По мере роста объёма текстовой информации растет и потребность в разработке методов обработки и анализа документов, написанных на естественных языках. Проблемы и задачи, связанные с анализом текстовых данных, относятся к области обработки текстов на естественных языках. В настоящее время увеличение количества доступных источников текстовой информации способствует активному развитию методов интеллектуального анализа текстов. Одной из важнейших проблем в области обработки естественных языков является установление авторства текста, основываясь на его стилистических признаках.

Выявление признаков авторского стиля позволяет установить принадлежность текста определенному человеку. Актуальность данного направления в компьютерной лингвистике обусловлена необходимостью в выявлении плагиата, установлении авторства анонимного текста или в создании рекомендательной системы для нахождения похожих текстов. Для определения стиля автора необходимо выделить характерные признаки из принадлежащих ему текстов.

Целью данной работы является разработка метода определения признаков авторского стиля для текстов на русском языке.

Для достижения поставленной цели ставятся следующие задачи:

- проанализировать предметную область и существующие методы;
- сформулировать основные положения разрабатываемого метода определения признаков авторского стиля;
- описать использующий его метод классификации;
- разработать ПО, реализующее оба метода;
- провести оценку точности классификации текстов по предложенным признакам.

Разрабатываемый метод позволит проводить классификацию текстов на русском языке по авторскому стилю на основании результатов

морфологического анализа, не задействуя более трудоемкий и менее доступный синтаксический анализ.

# 1 Аналитический раздел

В данном разделе рассматриваются понятия авторского стиля и стилеметрических характеристик, этапы обработки и анализа текста, исследуются алгоритмы классификации, также приводится обзор существующих методов и решений для определения авторского стиля, анализируются их достоинства и недостатки. Также в этом разделе приводится описание последовательного решения задачи.

## 1.1 Описание предметной области

Под авторским стилем понимается совокупность характеристик, позволяющих установить авторство текста или выдвинуть предположение, кем может являться автор или к какой группе авторов он может принадлежать. К особенностям авторского стиля можно отнести грамматические конструкции, стилистические приемы, способы построения фраз и абзацев или любой другой набор признаков, который отличает конкретного автора от всех других.

Определение характеристик личного стиля автора с целью систематизации текстов занимается раздел лингвистики под названием стилеметрия [1]. Объект стилеметрии – текст, написанный определенным установленным автором. Предмет исследования стилеметрии – элементы стиля текста, которые присущи конкретному автору. Стиль текста представляет собой набор выделенных в ходе анализа характеристик. Как правило, параметрами текста, определяющими конкретного автора, называют статистические характеристики: количество предложений разного типа и их средняя длина, число знаков препинания, определенных частей речи и т.д.

Авторским инвариантом называют набор стилеметрических характеристик, который присущ данному автору или небольшой группе похожих «близких авторов» [2]. При классификации текстов по авторам выбираются и анализируются те стилеметрические характеристики, которые будут различаться

у рассматриваемых: для произведений разных групп авторов рассматриваемые характеристики должны принимать разные значения. Для успешного установления авторства нужно, чтобы число «разных групп» было достаточно велико, и при этом каждая группа должна включать в себя небольшое количество близких по стилю произведений, для которых точно определен автор.

Методы анализа стиля произведения можно на две основные группы – формальные и экспертные. Экспертные методы основаны на исследовании текстов квалифицированным лингвистом. Формальные методы подразумевают машинную обработку и анализ текста и включат в себя алгоритмы классификации, кластеризации и нейронные сети.

## 1.2 Этапы обработки текста

В данном разделе будет выполнен обзор этапов обработки текстов на естественном языке. Предварительная обработка требуется для того, чтобы в дальнейшем было возможным выделить из текста его признаки и провести на их основании классификацию документов. Рассмотрены два основных этапа: токенизация и нормализация. В том числе, два основных подхода к нормализации: стемминг и лемматизация,

### 1.2.1 Токенизация

Токенизация — это базовый этап в машинной обработке текстов, который заключается в разбиении непрерывной строки на отдельные «слова» (токены) [3].

На этапе токенизации также решается задача разбиения текста на предложения. При разделении текста на отдельные предложения недостаточно учитывать только наличие точки и большой буквы после нее. Например, под данное правило не попадают сокращения и инициалы («И.И. Иванов» — три разных предложения).

Токенизация текста выполняется в несколько этапов. Вначале текст полностью переводится в нижний регистр. На данном этапе часть информации может быть утеряна. Например, «ООО» представляет собой аббревиатуру (общество с ограниченной ответственностью), в то время как «ооо» — способ выражения эмоций.

На следующем этапе токенизации требуется все знаки препинания и прочие символы, присутствующие в тексте и не являющиеся символами алфавита рассматриваемого языка, заменить на пробелы. После замены на пробелы всех символов, не являющихся буквами или строками, оставшиеся слова, разделённые пробелами, можно объявить отдельными токенами. Однако при наличии сложных составных слов замена разделителя между частями слова (например, дефиса) на пробел может привести к потере исходного смысла предложения. Например, из названия города «Санкт-Петербург» при замене дефиса на пробел получатся два отдельных слова «Санкт» и «Петербург». Чтобы не возникло такой ситуации, при которой исходные составные слова могут потерять свое первоначальное значение, на данном этапе токенизации стоит учитывать, что некоторые наборы слов должны рассматриваться как одно целое. Например, названия городов («Нижний Новгород»), или сокращения («м.н.с.», младший научный сотрудник). Если рассматривать слова «Нижний» и «Новгород» независимо друг от друг, важная информация, определяющая контекст текста, может быть утеряна, а разделенные на буквы сокращения полностью теряют изначально вложенный в них смысл [4].

### 1.2.2 Нормализация

После этапа токенизации для каждого слова выполняется нормализация. Нормализация слова заключается в приведении данного слова к его начальной форме (лемме).

Нормализация может быть выполнена одни из двух способов: стемминг и лемматизация [4].

Подход стемминга заключается в отделении окончания от каждого токена. Данный подход может работать некорректно в том случае, если при изменении формы само слово полностью меняется (например, «был», «есть», «будет»).

Суть подхода лемматизации заключается в использовании предварительно созданного словаря, в котором содержится большое число слов и их возможных форм. В первую очередь встретившаяся в тексте словоформа ищется по словарю. Если словоформа найдена в словаре, то с его помощью можно найти и его лемму, и все остальные известные формы данного слова. В противном случае по определённому алгоритму выводится способ изменения слова, и на его основании делаются выводы о первоначальной форме слова.

Лемматизация дает более точный результат при работе с незнакомыми словами, но в связи с поиском в словаре и работой алгоритма приведения к нормальной форме, является более медленным подходом по сравнению со стеммингом.

## 1.3 Этапы анализа текста

Процесс полного анализа текста может быть поделен на несколько этапов [3]. Выбор того, каким образом необходимо обработать и проанализировать текст и сколько этапов для этого понадобится, зависит от поставленной задачи и выбранного пути ее решения.

Начальный этап анализа текста на естественном языке называется графематическим и его суть заключается в выделении синтаксических или структурных единиц. В общем случае подаваемый на вход текст имеет нелинейную структуру, т.е. помимо основного текста встречаются комментарии, заголовки и т.д. При обработке текстов ставится задача сохранить их исходную структуру. Текст может иметь и линейную структуру, но и в этом случае важно правильно выделить его синтаксические единицы: отдельные слова, предложения, абзацы.

Далее следует морфологический анализ, цель которого – провести нормализацию, т.е. для каждого слова в тексте определить его начальную форму. Помимо самой леммы на данном этапе также составляется набор морфологических признаков слова (часть речи, число, род и т.д.). Данный этап позволяет уменьшить количество различных форм одного слова, заменив их все на его лемму, что позволяет сократить время машинной обработки текста в дальнейшем.

На этапе предсинтаксического анализа происходит объединение или разбиение лексических единиц. Лексические единицы из составных словосочетаний соединяются в единую синтаксическую единицу, т.к. в данном случае при использовании слов по отдельности теряется первоначальный смысл, заложенный в конкретное словосочетание. Также на данном этапе происходит разметка линейного текста на структурные единицы, что предшествует этапу синтаксического анализа.

Синтаксический анализ является самым трудоемким этапом, его цель определить роли слов в предложении и их связи между собой. На данном этапе решается задача сопоставления лексических единиц текста с формальной грамматикой языка. В результате проведения синтаксического анализа образуется набор синтаксических деревьев, наглядно показывающих соотношения между единицами текста.

Постсинтаксический анализ проводится для определения того смысла, который заложен в выражения из текста с учетом использованиям выразительных средств языка. Также на данном этапе выполняется нормализация синтаксических деревьев, полученных в результате синтаксического анализа.

Семантический анализ проводится для определения смысла, заложенного в слова и предложения. Для выполнения семантического анализа используется специальный словарь, который оперирует смыслами и, следовательно, описывает свойства и отношения понятий, а не слов.

## 1.4 Методы и решения определения авторского инварианта

В данном разделе проанализированы существующие методы для определения признаков авторского стиля. В том числе рассмотрены некоторые программные продукты, реализующие предложенные методы. Рассмотрены преимущества и недостатки методов, а также применимость подходов конкретно определению авторства текстов, написанных на русском языке, и зависимость качества работы метода от длины текстов.

### 1.4.1 Метод полного синтаксического анализа

Метод полного синтаксического анализа заключается в построении полного синтаксического дерева [8], позволяющего установить связи между структурными единицами текста и представить их наглядно, а также определить грамматические функции слов предложения.

Наиболее распространенными являются две модели синтаксической структуры: грамматика составляющих, предложенная в работах Ноама Хомского [5] и грамматика зависимостей, описанная в работах Люсьена Теньера [6] и Игоря Мельчука [7].

В грамматике составляющих предложение на естественном языке представляется и рассматривается как иерархия составляющих. Данная иерархия предполагает выделение непересекающихся между собой синтаксических групп, которые делятся на более мелкие группы. Атомарной единицей в такой структуре является слово в предложении. Описанная иерархическая структура называется деревом составляющих. Дерево составляющих удовлетворяет свойству проективности, согласно которому при графическом представлении дерева выделенные между структурными единицами связи не должны пересекаться.

В грамматике зависимостей предложения в тексте рассматриваются в виде деревьев зависимостей, где слова связаны ориентированными дугами,

показывающими синтаксическое подчинение. Грамматика зависимостей допускает наличие непроективных связей, нарушающих свойство проективности дерева

Полный синтаксический разбор гарантирует высокое качество анализа, поскольку устанавливает все существующие зависимости при разборе предложения. Но т.к. в русском языке в предложении зачастую встречается свободный порядок слов (в отличие, например, от английского языка), данный метод зачастую требует корректирующего вмешательства эксперта вдобавок к машинному синтаксическому разбору. В силу этих особенностей языка метод синтаксического разбора является очень затратным по времени. К тому же большая часть программного обеспечения, выполняющего синтаксический анализ, является коммерческими проектами и закрыта для использования.

### 1.4.2 Метод энтропийной классификации

В работе [9] для задачи установления авторства неисследованного текста предлагается использовать метод классификации, основанный на использовании алгоритмов сжатия.

Подход рассматриваемого метода заключается в том, что новый текст, для которого не определен автор, поочерёдно добавляется к корпусам текстов, характеризующим конкретных известных авторов. Далее, проанализировав степень полученного после добавления нового текста сжатия, и сравнив найденные значения для каждого из авторских корпусов, выдвигается предположение о том, к какому автору отнести рассматриваемый текст. В рамках энтропийного подхода классом текста является тот, на котором получены наилучшие результаты сжатия.

Рассмотрим на примере классификацию текста $T$ на текстах $S_1, \dots, S_n$, характеризующих $n$ классов. Выбор наиболее вероятного класса текста $T$ осуществляется в соответствии со значением оценки, рассчитываемой по формуле:

$$q(T) = argmin_i\ H(T|S_i), \quad (1)$$

где $H(T|S)$ – характеристика энтропии для текстов $T$ и $S$.

Главное преимущество энтропийной классификации – отсутствие необходимости в предобработке текста перед выполнением сжатия.

Проводимые с данным методом исследования показывают, что на текстах разной длины лучше работают разные алгоритмы сжатия.

### 1.4.3 Рекуррентные нейронные сети

Для решения задачи классификации текстов используют также и нейронные сети. Для рекуррентных нейронных сетей характерно наличие обратных связей, использующих для нахождения нового состояния сети его предыдущее состояние. Пример данной нейронной сети показан на рисунке 1. Как правило, данный подкласс нейронных сетей используется в задачах, где на вход поступает последовательность нефиксированной длины [10]. Примером такой последовательности является текст на естественном языке.

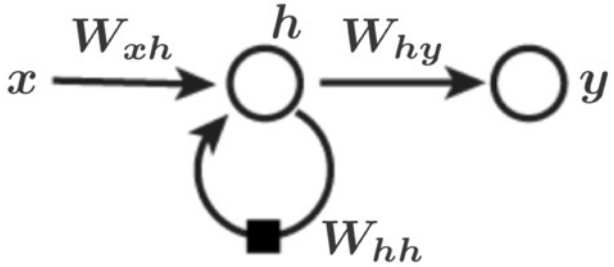



*Рисунок 1 – Структура рекуррентной нейронной сети с одним входным нейронном, одним скрытым нейроном и одним выходным нейроном*

Рекуррентные нейронные сети позволяют использовать текст в его исходном виде, то есть учитывая порядок слов. Сеть последовательно получает на вход слова одно за другим. В рекуррентной сети присутствует скрытый слой $h$, изменяющийся в связи с появлением нового токена на входе. Слой обновляется, учитывая и своё предыдущее состояние, и полученное слово. После

обновления скрытого состояния на его основе генерируется выход $y$ для данного токена. Обработка токенов заканчивается в тот момент, когда каждый из них пройдёт через нейронную сеть.

Рекуррентная нейронная сеть может быть описана с помощью двух формул. Первая формула определяет обновление скрытого слоя, происходящее путём нахождения среднего значения предыдущего и текущего значений:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1}), \quad (2)$$

где $h_{t-1}$ – предыдущее состояние слоя;

$x_t$ – слово, которое было подано на вход;

$f$ – некая нелинейная функция для активации нейрона.

Вторая формула, описывающая нейронную сеть, определяет значение, получаемое на выходе:

$$y_t = g(W_{hy}h_t), \quad (3)$$

где $h_t$ – обновлённое состояние скрытого слоя;

$g$ – функция активации.

Рекуррентная нейронная сеть является сетью прямого распространения. Длина такой нейронной сети зависит от числа токенов, поступающих на вход.

В данном классе нейронных сетей изменяются скрытые входы $x_t$, выходы $y_t$ состояния $h_t$. В то время как параметры матрицы $W_{hh}, W_{xh}, W_{hy}$ являются общими и не изменяются при поступлении нового токена.

Для обучения сети применяется алгоритм обратного распространения ошибки по времени (backpropagation through time) [11], который является вариантом алгоритма обратного распространения ошибки (backpropagation).

К недостаткам использования рекуррентных нейронных сетей можно отнести их трудоемкость, а также тот факт, что не каждая функция активации

позволяет обрабатывать очень длинные последовательности.

### 1.4.4 Метод анализа длин слов

Одним из готовых решений для отнесения неизвестного текста к работам определенного автора является программа «Худломер» [12]. В основе этой программы лежит метод определения стилистика текста в зависимости от длины слов. «Худломер» обучен определять разговорный, художественный, научно-деловой и газетно-информационный стили. Определение стиля текста зависит от спектра длин слов в нем.

### 1.4.5 Метод выделения N-грамм

N-грамма — это последовательность из N подряд идущих токенов. Использование N-грамм характерно тем, что при данном подходе учитывается порядок слов в тексте.

Для предложения «Метод определения признаков авторского стиля» существуют следующие N-граммы:

- униграммы (N = 1): метод, определения, признаков, авторского стиля;
- биграммы (N = 2): метод определения, определения признаков, признаков авторского, авторского стиля;
- триграммы (N = 3): метод определения признаков, определения признаков авторского, признаков авторского стиля.

В основе N-граммы лежит математическая модель, которую можно сформулировать следующим образом: «N-граммой на алфавите V называют произвольную цепочку длиной N, например последовательность из N букв алфавита V одного слова, одной фразы, одного текста или, в более интересном случае, последовательность из грамматически допустимых описаний N подряд стоящих слов» [13].

При увеличении размера N, до которого будут найдены N-граммы, будет увеличено и количество найденных признаков. При этом увеличение параметра

N может привести к переобучению. Например, если N равен длине рассматриваемых текстов, то для каждого документа будет получен единственный и неповторяющийся признак. Тогда, алгоритм сможет переучиться под обучающую выборку.

N-граммы могут быть использованы не только на словах, но и на символах. В таком случае N-граммы будут выделяться посимвольно, что позволит распознавать знакомые слова в неизвестных ранее формах. Часто два подхода комбинируются, и N-граммы по словам рассматриваются вместе с N-граммами по символам.

Skip-граммы — это один из вариантов развития использования N-грамм. Более точно данный подход называется k-skip-n-граммы и использует наборы из N токенов, причем расстояние между соседними токенами должно составлять не более K токенов [14]. Благодаря данному метода становится возможным учитывать большее количество различных N-грамм в предложении. В связи с этим, данный подход, как правило, используется в языковом моделировании в сочетании с другими.

## 1.5 Извлечение признаков из текста

В данном разделе рассматриваются некоторые из существующих подходов к извлечению из предварительно подготовленного и обработанного текста его признаков. Для каждого из выделенного признаков должна быть определена его значимость в контексте данного текста. Полученные признаки в дальнейшем становятся основой для составления вектора признаков документа и перевода его в векторное пространство.

### 1.5.1 Счётчики слов

При использовании данного подхода текст рассматривается как неупорядоченный набор токенов.

Пусть всего в выборке $N$ присутствуют различные слова $w_1, \dots, w_N$. Каждый документ кодируется с помощью $N$ признаков, где признак $j$ — доля вхождений слова $w_j$ среди всех слов в тексте. В таком случае получается, что текст представляется в виде вектора значений признаков. При сложении значений всех признаков должна получиться единица.

При использовании данного подхода, как правило, из документов в процессе предобработки удаляются стоп-слова – популярные слова, часто используемые в каждом из текстов и не несущие в себе никакой информации, которая могла бы выделить определенный текст или класс текстов. Как правило, удаление стоп-слов проводится на этапе токенизации. Также стоит удалять и редкие слова. Если определённое слово встречается только в одном документе из корпуса, то, скорее всего, вклад данного слова будет несоизмеримо мал по сравнению со всеми остальными, и оно никак не повлияет на качество классификации.

### 1.5.2 TF-IDF

TF-IDF — статистическая мера, которая позволяет оценить важности конкретного слова в контексте документа, входящего в состав корпуса документов. В данном подходе, аналогично подходу с использованием счетчика слов, считается, что если слово часто встречается в тексте и не является стоп-словом, то, скорее всего, данное слово является значимым для определения стилистики текста. Второе утверждение, которое лежит в основе TF-IDF: если слово встречается в других документах корпуса реже, чем в рассматриваемом тексте, то, скорее всего, оно является значимым для данного конкретного текста. Исходя из данных утверждений, получается подход TF-IDF (TF — term frequency, IDF — inverse document frequency) [15].

TF — частотность термина, которая измеряет, насколько часто термин встречается в документе. TF измеряется как отношение того, сколько раз конкретное слово встречается в документе к тому, сколько всего слов присутствует в данном тексте. IDF характеризует инверсию частотности употребления конкретного слова в анализируемом корпусе документов. Статистическая мера TF-IDF находится путем перемножения значений TF и IDF.

Значение признака TF-IDF слова $w$ в тексте $x$ вычисляется по следующей формуле:

$$TF\text{–}IDF(x,w) = n_{dw} \log \frac{l}{n_w}, \quad (4)$$

где $n_{dw}$ – доля вхождений слова $w$ в документ $d$;

$l$ – количество всех документов;

$n_w$ – количество документов, в которых хотя бы один раз используется слово $w$.

Чем выше значение $\frac{l}{n_w}$, тем реже слово используется в других текстах, следовательно, для рассматриваемого признака увеличится значение. Для слова, встречающего в каждом тексте, значение признака является нулевым ($\log \frac{l}{l} = 0$).

## 1.6 Классификация текстов

В данном разделе будут рассмотрены алгоритмы классификации текстов, а также проанализированы их преимущества и недостатки. Применительно к области компьютерной лингвистики, классификацией называется задача, цель которой заключается в отнесении документа к одной из нескольких категорий на основании содержания документа. Классификация относится к разделу «обучения с учителем» и работает с размеченным корпусом документов.

### 1.6.1 Метод «наивной» байесовской классификации

Наивный байесовский классификатор [16] основывается на использовании теоремы Байеса со строгими (наивными) предположениями о независимости и оперирует условными вероятностями. Данный алгоритм называют наивным по причине того, что он использует наивное допущение о независимости входящих в текст слов друг от друга.

В основе наивного байесовского классификатора лежит закон, по которому распределены данные. Соответственно, обучение данного классификатора сводится к вычислению параметров распределения, используя документы из тестового набора данных [17].

Если предположить, что данные распределены по закону Бернулли, то класс $c^*$, к которому относится текст $t$, вычисляется по следующей формуле:

$$c^* = argmaxP(c)\sum_{i=1}^{m} P(x|c)^{x_i(t)}, \quad (5)$$

где $x$ – характеристики, по которым оцениваются тексты;

$m$ – общее число текстов;

$x_i(t)$ – величины, показывающие наличие $i$-ой характеристики в тексте;

$c$ – метка класса;

$P(c), P(x|c)$ – параметры, вычисленные в процессе обучения классификатора.

Главными недостатками данного метода являются невозможность учитывать влияние сочетания признаков на результат классификации и низкие показатели метрик качества классификации по сравнению с другими методами.

### 1.6.2 Метод k ближайших соседей

Метод k ближайших соседей [18] основан на гипотезе компактности векторного пространства, которая гласит, что: объекты, принадлежащие одному

классу, образуют в пространстве терминов компактную область, причём области разных классов не пересекаются [4]. Согласно рассматриваемому методу, для нахождения класса, который соответствует документу $d$, классификатор должен вычислить расстояние $\rho(d_z, d)$ от документа $d$ до всех документов $d_z \in L$ из обучающей выборки $L$. Рассчитав расстояние до каждого документа из обучающей выборки, можно установить $k$ самых близких к $d$ документов. Считается, что документ $d$ является членом того класса, к которому принадлежит наибольшее число соседей данного документа. Для определения класса документа для каждого класса $c_i$ находится значение функции ранжирования:

$$CSV(d) = \sum_{d_z \in L_k(d)} \rho(d_z, d) \cdot \Phi(d_z, c_i), \qquad (6)$$

где $L_k(d) - k \in L$ ближайших к $d$ документов;

$\Phi(d_z, c_i)$ – расклассифицированные по классам документы из обучающей выборки.

Преимущества данного метода заключаются в возможности добавления новых объектов в обучающую выборку без необходимости переобучения, устойчивость классификатора к выбросам в данных, высокое качество обучения на линейно разделимых выборках.

Недостатком является медленная работа алгоритма в связи с полным перебором обучающей выборки.

### 1.6.3 Метод деревьев принятия решений

Метод деревьев принятия решений является символьным или, другими словами, нечисловым.

Для каждого дерева принятия решений основе обучающего множества узлами являются термины документов, листьями – метки классов, а на ребрах

отмечены веса терминов [4]. Каждое дерево представляет собой ациклический граф.

Процесс классификации заключается в поочередных перемещениях между узлами дерева в зависимости от определенных признаков данного документа. Процесс классификации подходит к концу в тот момент, когда достигнут некий конечный лист дерева. Лист, до которого дошел процесс классификации, определяет определенный класс и, следовательно, рассматриваемый объект является объектом данного класса. В задачах классификации чаще всего встречаются бинарные деревья решений, и результат перехода по ребрам определяется наличием или отсутствием некого признака в документе [19].

Лесом решений [18] называют набор (ансамбль) из нескольких деревьев решений. С помощью комбинирования нескольких деревьев решений можно повысить качество классификации: решение о принадлежности объекта какому-либо классу принимается по большинству результатов, полученных у деревьев решений по отдельности.

Преимуществом решающих деревьев является простая интерпретируемость полученных результатов классификации, недостатки метода – требуется большой объем данных для высокой точности классификации и отсутствует устойчивость к выбросам в данных.

### 1.6.4 Метод опорных векторов

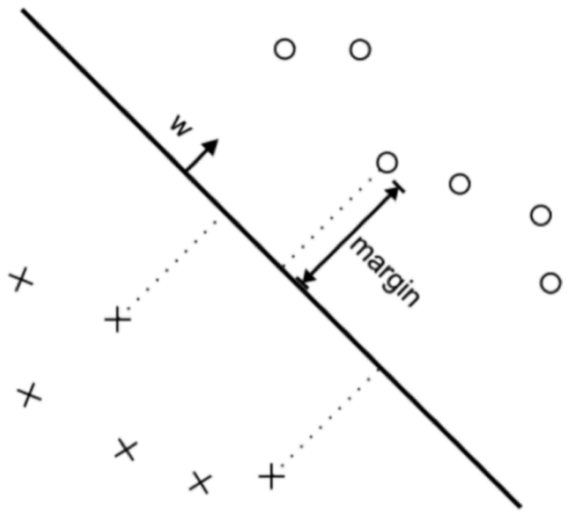
Метод опорных векторов [19] основан на разбиении векторного пространства документов разделяющей поверхностью на подпространства, где каждому подпространству соответствует только один класс. В данном алгоритме считается, что размер зазора классификации (расстояние между найденной поверхностью и ближайшей точкой данных) влияет на среднюю ошибку классификатора: чем больше зазор, тем меньше ошибка.

Для данного метода обязателен этап предобработки данных для преобразования текстовые данных в числовые векторы признаков. В процессе

обучения осуществляются преобразования над пространствами, которые производятся с помощью оператора ядра. Такие преобразования приводят данные к такому виду, чтобы их можно было разделить гиперплоскостями на подпространства таким образом, чтобы большая часть объектов из обучающей выборки соответствовала своему классу, располагающемуся в конкретном подпространстве. Далее при классификации нового текста его также необходимо перевести в векторное представление, после чего присвоить ему класс, найденный в зависимости от того, в каком подпространстве находится полученный вектор.

Метод опорных векторов использует ядро – функцию, которая способна вычислять скалярное произведение $\phi(a)^T \cdot \phi(b)$, оперируя только исходными векторами $a$ и $b$, не выполняя трансформацию $\phi$. Наиболее распространенными ядрами являются линейное: $K(a, b) = a^T \cdot b$; полиномиальное: $K(a, b) = (\gamma a^T \cdot b + r)^d$; Гауссово RBF: $K(a, b) = \exp(-\gamma||a - b||^2)$; сигмоидальное: $K(a, b) = \tanh(\gamma a^T \cdot b + r)$.

На рисунке 2 показано разбиение данных при бинарной классификации при помощи линейного оператора ядра.




*margin – расстояние до гиперплоскости; w – классифицируемый вектор*

*Рисунок 2 – Классификация двух классов методом опорных векторов с линейным оператором ядра.*

На рисунке 2 показан случай бинарной классификации, когда все данные принадлежат двум классам, но метод опорных векторов подходит и для многоклассовой классификации [20]. Подход к решению поставленной проблемы заключается в разделение одной многоклассовой задачи на несколько бинарных. Наиболее часто встречаются подходы создания бинарных классификаторов, которые различают одну из меток и все остальные («один против всех») или попарно между классами («один против одного»). При подходе «один против всех» классификация выполняется по принципу «победитель против всех»: класс присваивает себе тот классификатор, у которого функция достигла наивысшего результата. При подходе «один против одного» классификация выполняется по стратегии голосования с максимальным количеством побед: каждый классификатор назначает экземпляр одному из двух классов, сумма голосов за назначенный класс увеличивается на один и, в итоге, класс с наибольшим количеством голосов определяет классификацию экземпляра.

К преимуществам данного метода относится то, что он позволяет работать с малым количеством данных для обучения и сводит классификацию к задаче выпуклой оптимизации с единственным решением. Исследования показывают, что метод опорных векторов является одним из наиболее точных методов [17].

Недостатком является неустойчивость к выбросам данных в обучающей выборке.

## 1.7 Выводы

В данном разделе были рассмотрены этапы обработки и анализа текстов на естественном языке. Проведен обзор существующих методов и решений определения авторского стиля. Были проанализированы алгоритмы классификации текстов. В разрабатываемом методе предлагается использовать подход, основанный на использовании N-грамм, в качестве метрики использовать TF-IDF, для классификации текстов выбран метод опорных векторов, позволяющий работать с небольшой выборкой документов.

# 2 Конструкторский раздел

В данном разделе рассматривается структура программного обеспечения. Приводятся функциональные схемы предлагаемого метода определения признаков авторского стиля, обучения классификатора, метода классификации, использующего предложенный метод определения признаков авторского стиля. Также рассматривается последовательность действия для метода определения признаков авторского, процесс обучения классификатора и классификации. Приводятся метрики, используемые для оценки качества классификации.

## 2.1 Структура разрабатываемого программного обеспечения

Разрабатываемый программный продукт состоит из следующих частей:

- модуль считывания входных данных;
- модуль предобработки текстов;
- модуль создания выборки;
- модуль классификации;
- модуль формирования выходных данных;
- модуль управления.

В модуле классификации осуществляется обучение классификатора и прогнозирование класса новых данных. Модуль создания выборки отвечает за создание и обучающего, и тестового набора данных.

## 2.2 Функциональная схема метода выделения признаков

На рисунке 3 представлена схема предлагаемого метода определения признаков авторского стиля в нотации IDEF0. На данной схеме отображены этапы обработки текста (токенизация и лемматизация), выделение N-грамм и определения частей речи слов в выделенных N-граммах.

На вход поступает необработанный текст на русском языке, на выходе получаются частеречные N-граммы.

Пример последовательного преобразования текста к виду частеречных N-грамм приведён в приложении А.

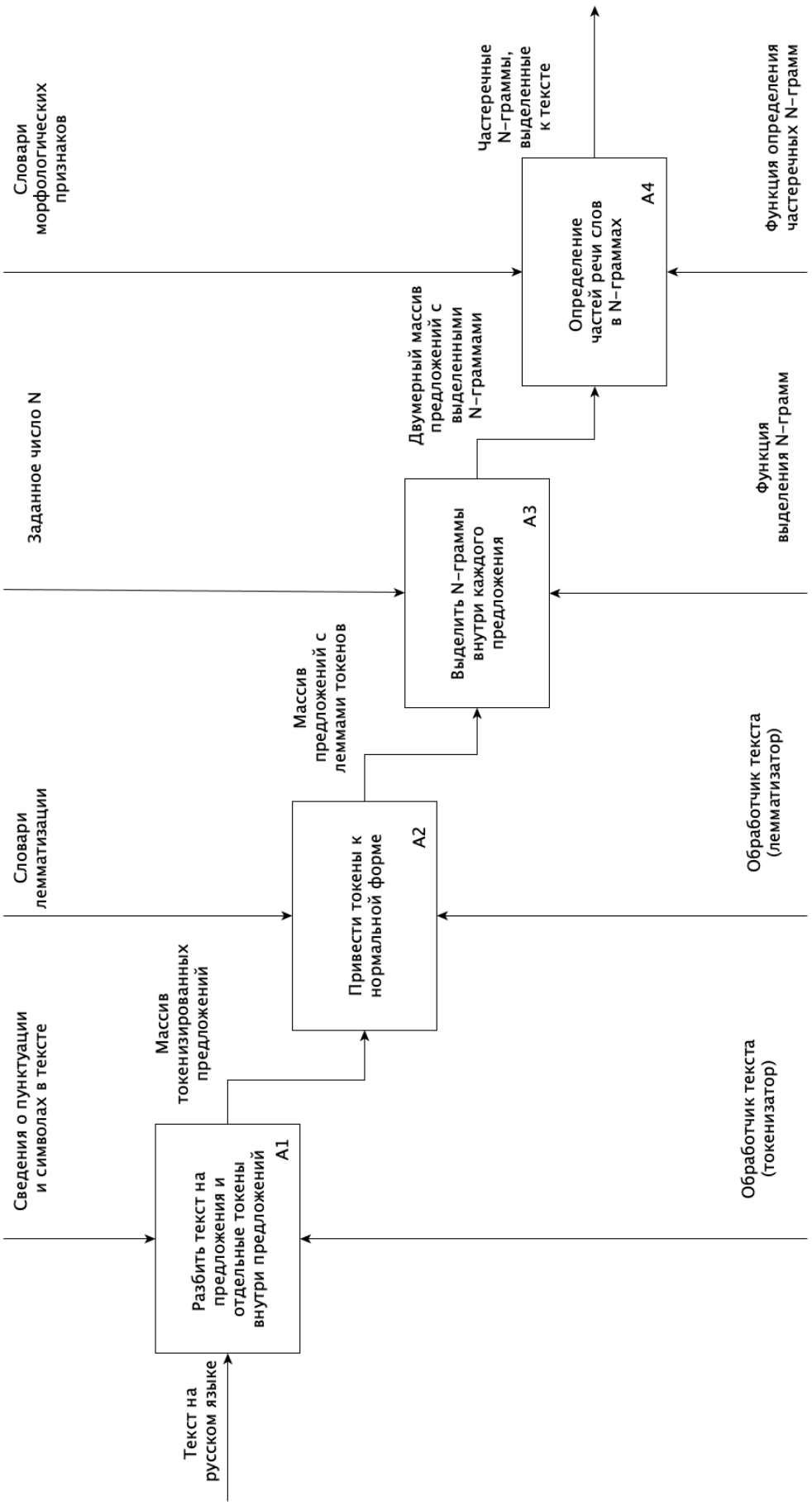



*Рисунок 3 – Функциональная схема работы метода выделения признаков: образования частеречных N-грамм*

## 2.3 Функциональная схема обучения метода классификации

На рисунке 4 представлена функциональная схема обучения метода классификации (метод опорных векторов) в нотации IDEF0. Метод классификации использует метод определения признаков авторского стиля, показанный на рисунке 3.

На данной схеме отображены этапы создания и нормализации матрицы признаков и проведения многоклассовой классификации для обучающей выборки. На вход поступает размеченный корпус документов, на выходе – обученный классификатор.

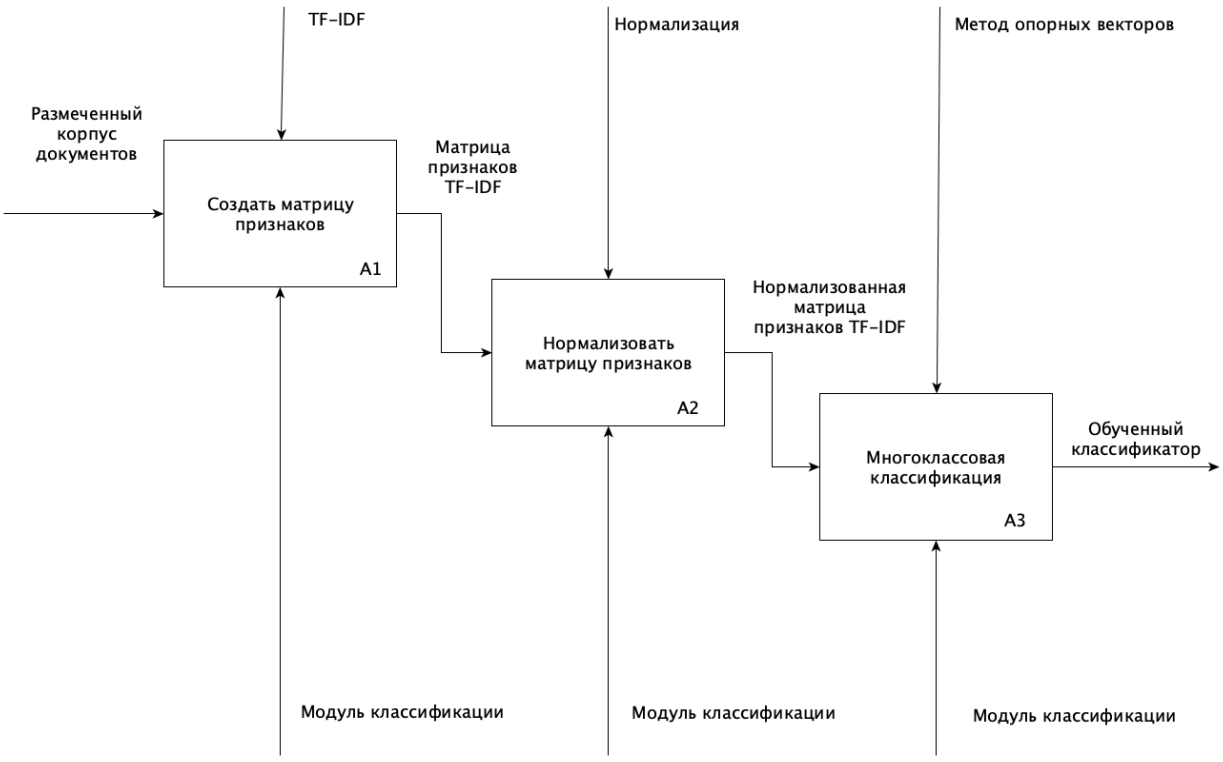



*Рисунок 4 – Функциональная схема обучения метода классификации*

## 2.4 Функциональная схема метода классификации

На рисунке 5 представлена функциональная схема метода классификации в нотации IDEF0. Данный метод включает в себя обучение метода классификации, показанный на рисунке 4.

На данной схеме отображены этапы создания обучающей и тестовой выборок, образования матрицы признаков, обучения классификатора, проведения классификации на тестовой выборке и получения метрик качества.

На вход поступает размеченный корпус документов, на выходе – полученные результаты для метрик качества классификации.

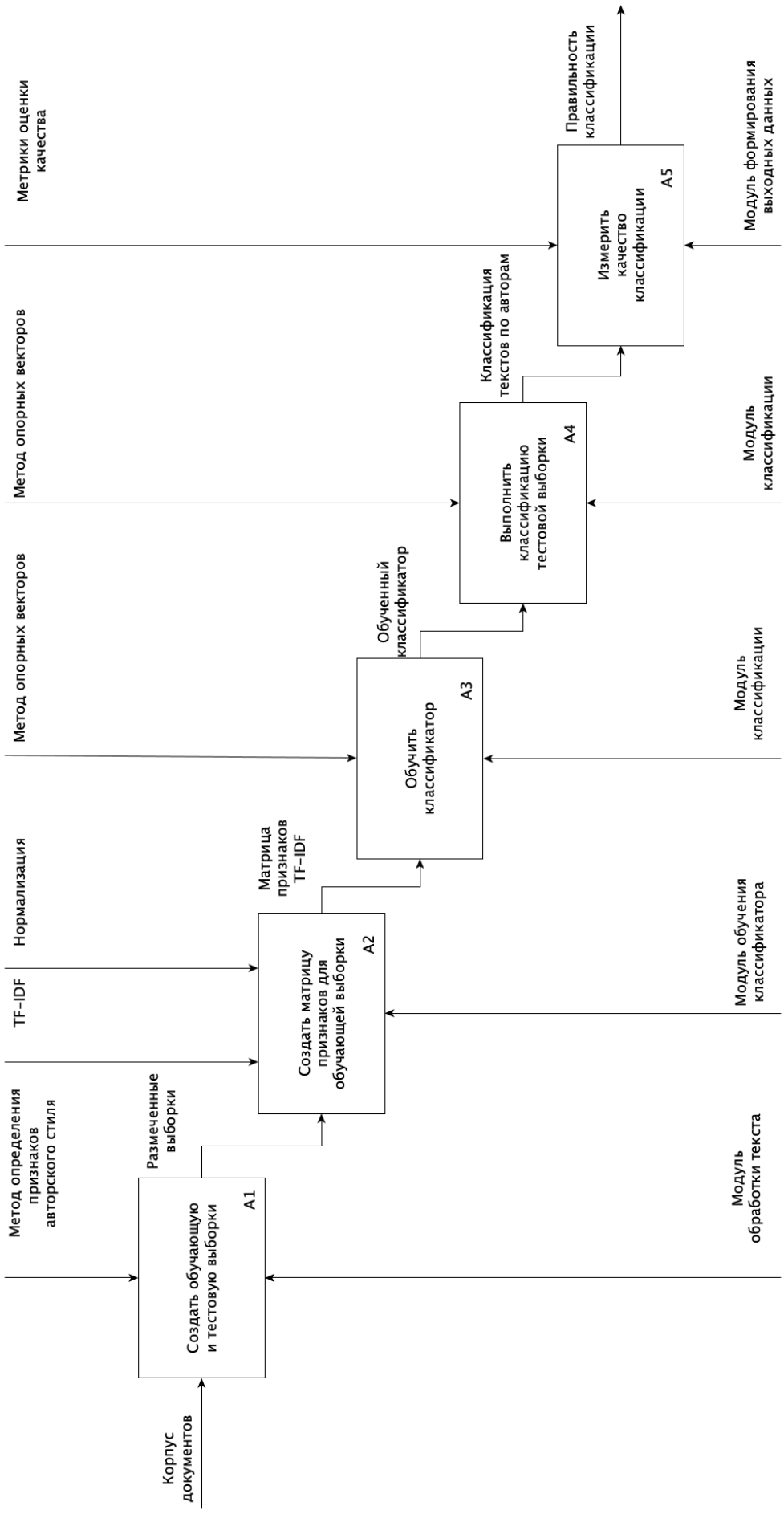



*Рисунок 5 – Функциональная схема метода классификации*

## 2.5 Диаграмма вариантов использования

На рисунке 6 представлена Use-Case диаграмма разрабатываемого программного обеспечения:


*Рисунок 6 – Use-Case диаграмма*

Для обучения классификатора пользователь должен обработать текстовые документы и привести к виду частеречных N-грамм, либо воспользоваться заранее подготовленными корпусами. После обучения классификатора пользователь может просмотреть значения метрик качества для обучающей и тестовой выборок или определить автора для нового текста

## 2.6 Описание этапов работы разрабатываемого алгоритма

### 2.6.1 Предобработка текстовых документов

В модуль обработки данных на вход подаются текстовые данные, полученные в модуле считывания входных данных. Для представления входных данных используется тип данных строка. Один текст представляет собой одну строку.

Предобработка текстовых данных состоит из двух этапов: токенизация и нормализация (лемматизация).

После токенизации исходная строка преобразуется в массив, элементы которого – предложения, состоящие из отдельных токенов. Предложение хранится в виде строки. Каждый токен внутри предложения приведен к нижнему регистру, а также отсутствуют все знаки препинания.

После лемматизации структура данных остается неизменной – массив строковых предложений. Токены предложений приведены к своей нормальной форме (лемме).

### 2.6.2 Выделение N-грамм

Функции выделения N-грамм на вход подается массив сток, где каждый элемент – токенизированное предложение, в котором каждое слово заменено на его лемму. Помимо массива строк на вход также подается N – количество элементов в последовательностях, которые будут выделены на данном этапе.

N-выделяются в пределах предложений. Каждая N-грамма является массивом строк, где каждая строка – слово в полученной последовательности.

Таким образом, функция выделения N-грамм возвращает трехмерный массив: весь текст представлен в виде массива, внутрь которого вложены массивы предложений, а внутрь каждого массива предложения – массив выделенных в предложении N-грамм.

### 2.6.3 Получение частеречных N-грамм

Функции выделения частеречных N-грамм на вход передается трехмерный массив, полученный в результате выполнения функции выделения N-грамм. Для преобразования обычных N-грамм в частеречные, для каждого токена определяется его часть речи. В результате каждый токен заменяется его частью речи.

Частеречные N-граммы представляют собой связку из частей речи, то есть кортеж из словоформ будет представлен как кортеж из частей речи этих

словоформ (например, существительное-глагол-прилагательное-существительное).

Частеречная N-грамма представляет собой строку, в которой записаны названия частей речи, соответствующие оригинальным словам. Из строки удаляются пробелы между названиями частей речи, это нужно для того, чтобы на этапе классификации можно было представить каждую N-грамму как отдельный токен.

Выходные данные рассматриваемой функции – массив строк, где массив представляет собой один текст, а каждая строка – это частеречная N-грамма.

### 2.6.4 Формирование матрицы признаков

Для оценки важности определенной частеречной N-граммы для текста заданного автора предлагается использовать статистическую меру TF-IDF.

TF — частотность термина – находится по следующей формуле:

$$TF(t,d) = \frac{n_t}{\sum_k n_k}, \tag{7}$$

где $n_t$ – число вхождений слова $t$ в документ $d$;

$\sum_k n_k$ – общее число слов в документе $d$.

IDF — инверсия частоты – рассчитывается следующим образом:

$$IDF(t,D) = \log\left(\frac{1+|D|}{1+|\{d_i \in D \mid t \in d_i\}|}\right) + 1, \tag{8}$$

где $D$ – корпус документов;

$|D|$ – количество документов в корпусе;

$|\{d_i \in D \mid t \in d_i\}|$ – количество документов в корпусе $D$, в которых присутствует слово $t$.

Сложение с единицей равносильно добавлению в корпус документа, в котором каждый термин содержался бы ровно один раз. Такое добавление предотвращает деление на ноль.

В основании логарифма может стоять любое число, поскольку при изменении значения основания веса каждого слова изменяются на одинаковый множитель, тем самым соотношения весов сохраняются.

Искомая мера TF-IDF является произведением TF и IDF:

$$TF\text{–}IDF(t,d,D) = TF(t,d) \times IDF(t,D). \quad (9)$$

### 2.6.5 Нормализация матрицы признаков

Каждый текстовый документ представляется в виде вектора признаков, где признаками являются полученные значения TF-IDF. Соответственно, для корпуса документов из векторов признаков всех текстов составляется матрица признаков, в которой каждая строка соответствует одному документу, а столбец – одному из признаков.

Каждое значение в матрице признаков нормализуется согласно следующей формуле:

$$v_{norm} = \frac{v}{\sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}}, \quad (10)$$

где $v$ – значение признака;

$v_{norm}$ – нормализованное значение признака.

### 2.6.6 Обучение классификатора

Для выполнения классификации был выбран метод опорных векторов (SVM – Support Vector Machine). Так как в размеченном корпусе данных присутствуют несколько авторов, необходимо провести многоклассовую

классификацию, для которой используется подход «один против всех». При данном подходе обучающая выборка делится на рассматриваемый класс $x$ и все остальные классы и создается бинарный классификатор, который различает метку $x$ и остальные. Таким образом, задача многоклассовой классификации сводится к бинарной. Классификация новых экземпляров происходит по стратегии «победитель получает все», согласно которой классификатор с функцией наивысшего результата присваивает класс себе.

Метод опорных векторов заключается в построении разделяющей гиперплоскости, которая будет максимально удалена от ближайших к ней точек обоих классов. Необходимо найти вектор $w$ и число $b$ такие, что для некоторого $\varepsilon > 0$ выполняется:

$$w \cdot x_i \geq b + \varepsilon \Rightarrow y_i = 1, \quad (11)$$

$$w \cdot x_i \leq b - \varepsilon \Rightarrow y_i = -1. \quad (12)$$

Т.к. алгоритм классификации не изменится при умножении $w$ и $b$ на одну и ту же константу, можно выбрать константу таким образом, чтобы для всех пограничных (ближайших к разделяющей поверхности) точек выполнялись условия:

$$w \cdot x_i - b = y_i. \quad (13)$$

При оптимальном расположении гиперплоскости все пограничные объекты находятся от нее на одинаковом расстоянии. Неравенства 11 и 12 домножаются на $\frac{1}{\varepsilon}$, при этом $\varepsilon$ равняется единице. Тогда для всех векторов $x_i$ из обучающей выборки:

$$w \cdot x_i - b \geq 1, \qquad \text{если } y_i = 1, \quad (14)$$

$$w \cdot x_i - b \leq -1, \qquad \text{если } y_i = -1. \quad (15)$$

Полоса, разделяющая классы, задается условием $-1 < w \cdot x_i - b < 1$. Ни одна из точек обучающей выборки не может лежать внутри этой полосы.

Для случая с линейной разделимости данных на классы задачу поиска оптимальной разделяющей полосы можно сформулировать следующим образом: имеются ограничения $y_i(w \cdot x_i - b) \geq 1$, $y_i \in \{-1, 1\}$. Нужно найти такие $w$ и $b$, чтобы выполнялись все линейные ограничения, и при этом норма вектора $w$ была как можно меньше (следовательно, шире разделяющая полоса), то есть необходимо минимизировать:

$$\left||w|\right|^2 = w \cdot w. \tag{16}$$

Задача нахождения минимума квадратичной функции при заданных линейных ограничениях называется задачей квадратичной оптимизации.

В случае отсутствия линейной разделимости (например, классы линейно не разделимо или при составлении обучающей выборки была допущена ошибка) требуется ввести набор дополнительных переменных $\varepsilon_i \geq 0$, характеризующих величину ошибки на объектах $x_i \in [x_1, \dots, x_n]$. Тогда ограничения принимают следующий вид:

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i. \tag{17}$$

Если в документе $x$ отсутствуют ошибки, то $\xi_i = 0$. Если $\xi_i > 0$, то в документе $x$ допускается наличие ошибки. В случае $0 < \xi_i < 1$ объект попадает внутрь разделительной полосы, но относится алгоритмом к своему классу.

Задача поиска оптимальной разделяющей может быть переформулирована следующим образом: при заданных ограничениях минимизировать сумму:

$$\left||w|\right|^2 + C \sum \xi_i \,, \tag{18}$$

где $C$ – параметр настройки метода, который выбирается вручную и позволяет регулировать отношение между максимизацией ширины разделяющей полосы и минимизацией суммарной ошибки.

Для того, чтобы найти минимум функции, необходимо исследовать ее производную. Помимо самой функции, при минимизации необходимо учитывать и заданные линейные ограничения. Решить задачу минимизации можно с помощью метода Лагранжа. В данном методе задача поиска условного минимума (при наличии ограничений) сводится к задаче поиска безусловного минимума (без ограничений), чтобы затем воспользоваться стандартным методом поиска минимума функции. Для этого необходимо изменить целевую функцию, то есть ту функцию, которую необходимо минимизировать.

Пусть целевая функция:

$$F - \sum_i \lambda_i G_i, \tag{19}$$

где $G_i$ – уравнения ограничений;

$\lambda_i$ – множители Лагранжа (дополнительные переменные).

Для нахождения минимума необходимо взять все производные целевой функции по переменным $x$ и приравнять их к нулю. С помощью получившихся уравнений выразить $x$ через переменные $\lambda_1, \dots, \lambda_n$. После этого $x$ в ограничения.

В рассматриваемой задаче необходимо при ограничениях $y_i(w \cdot x_i - b) \geq 1 - \xi_i$ и $\xi_i \geq 0$ минимизировать квадратичную функцию:

$$\frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_i \xi_i\,. \tag{20}$$

Необходимо при условиях $\xi_i \geq 0, \lambda_i \geq 0$ найти минимум по $w$, $b$, $\xi_i$ и максимум по $\lambda_i$ для данной функции:

$$\frac{1}{2}w \cdot w + C\sum_i \xi_i - \sum_i \lambda_i \left(\xi_i + y_i(w \cdot x_i - b) - 1\right). \quad (21)$$

Целевая функция 21 называется Лагранжианом. Необходимым условием метода Лагранжа является равенство нулю производных Лагранжиана по переменным $w$ и $b$. После взятия производной целевой функции по $w$, вектор $w$ выражается через множители Лагранжа:

$$w = \sum_i \lambda_i \, y_i x_i. \quad (22)$$

Если $\lambda_i > 0$, то документ обучающей коллекции $x_i$ называется опорным вектором. Теперь уравнение разделяющей гиперплоскости выглядит следующим образом:

$$\sum_i \lambda_i \, y_i x_i \cdot x - b = 0, \quad (23)$$

где $x_i$ – документ, поступающий в классификатор.

После взятия производную целевой функции по $b$:

$$\sum_i \lambda_i \, y_i = 0. \quad (24)$$

При условиях $\sum_i \lambda_i \, y_i = 0$ и $0 \le \lambda_i \le C$ после подставления $w$, выраженное через $\lambda_i$, в Лагранжиан, задачу можно сформулировать следующим образом: найти такие значения множителей, при которых достигается максимум:

$$\sum_i \lambda_i - \frac{1}{2}\sum_i \lambda_i \, \lambda_j y_i y_j \left(x_i \cdot x_j\right). \quad (25)$$

В итоге, исходная задача квадратичной оптимизации была переформулирована в задачу для множителей Лагранжа. Целевая функция зависит не от самих $x_i$, а от скалярных произведений между ними. Скалярное произведение – метрика близости, которая для векторов может показать количество одинаковых признаков.

### 2.6.7 Классификация

На данном этапе обученный классификатор получает на вход данные из тестовой выборки. Данные из тестовой выборки передаются в классификатор в виде матрицы признаков. Также классификатор может получить на вход новый текст, который должен быть предварительно обработан, приведен к виду частеречных N-грамм и преобразован в вектор признаков. Для каждого документа формируется предсказание наиболее вероятного автора, после чего производится оценка качества классификация согласно метрикам оценки качества классификации.

### 2.6.8 Метрики оценки качества классификации

Для оценки работы классификатора на обучающей и тестовой выборках используются следующие метрики: точность (accuracy) и F-мера.

Точность (accuracy) представляет собой долю правильно предсказанных классификатором классов среди всех предсказанных значений. Если каждое предсказанное значение соответствует истинному классу объекту, то значение точно равняется 1.0 или 100%. Точность находится по следующей формуле:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (26)$$

где $TP$ – истинно положительное решение;

$TN$ – истинно отрицательное решение;

$FP$ – ложно положительное решение;

$FN$ – ложно отрицательное решение.

Для нахождения значения F-меры необходимо вычислить точность (precision) и полноту (recall).

Точность показывает качество работы классификатора и находится как отношение количества правильных прогнозов к числу всех прогнозов. Точность рассчитывается по следующей формуле:

$$Precision = \frac{TP}{TP + FP}. \quad (27)$$

Полнота долю найденных классификатором документов данного класса относительно всех документов, принадлежащих рассматриваемому классу. Полнота находится по следующей формуле:

$$Recall = \frac{TP}{TP + FN}. \quad (28)$$

F-мера представляет собой среднее гармоническое между точностью и полнотой. F-мера стремится к нулю, если точность или полнота стремится к нулю. Находится по следующей формуле:

$$F = 2\frac{Precision \times Recall}{Precision + Recall}. \quad (29)$$

Формула 28 придает одинаковый вес точности и полноте, поэтому F-мера будет падать одинаково при уменьшении и точности и полноты. При расчёте F-меры можно придать разный вес точности и полноте:

$$F = (\beta^2 + 1)\frac{Precision \times Recall}{\beta^2 Precision + Recall}. \quad (30)$$

Для того, чтобы присвоить больший вес точности, значение $\beta$ должно принимать значения в диапазоне $0 < \beta < 1$. При $\beta > 1$ приоритет отдается полноте. Если приравнять $\beta = 1$, то формула сведется к формуле нахождения сбалансированной F-меры (формула 28).

## 2.7 Выводы

В данном разделе была рассмотрена структура программного обеспечения. Также были рассмотрены функциональные схемы предлагаемого метода, обучения классификатора и метода классификации. Была приведена последовательность действия для метода определения признаков авторского и метода классификации, описаны структуры данных, используемые на каждом этапе работы алгоритма. Рассмотрен процесс обучения классификатора. Приведены метрики, используемые для оценки качества проведённой классификации.

# 3 Технологический раздел

В данном разделе описываются средства, используемые для разработки программного обеспечения, необходимые для функционирования ПО библиотеки, требования к вычислительной. Описывается формат входных и выходных данных, а также формат хранения данных. Приводится описание интерфейса пользователя и руководство для установки и использования ПО. Рассматривается программная реализация лемматизации и определении частей речи слов в тексте. Также в данном разделе проводится модульное тестирование.

## 3.1 Выбор средств программной реализации

В данном разделе приводится обоснования выбора средств для программной реализации предлагаемого метода. Рассматривается выбранный язык программирования и его преимущества для реализации предлагаемого метода, инструменты, предоставляемые выбранной средой разработки и отладки программного кода, а также приводятся используемые библиотеки и их краткое описание.

### 3.1.1 Выбор языка программирования

Для реализации программного продукта был выбран язык Python, данный язык обладает следующими преимуществами:

- наличие большой базы библиотек и готовых решений для обработки естественного языка и выполнения классификации;
- наличие библиотек, позволяющих работать с различными математическими структурами и визуализировать данные;
- кроссплатформенность и отсутствие необходимости компилировать исходный код в файлы с различным расширением в зависимости от ОС;

- автоматическое управление памятью;
- поддержка процедурной, объектно-ориентированной и функциональной парадигм программирования.

Таким образом, Python подходит как для научно-исследовательской работы, так и для разработки коммерческих программных продуктов.

### 3.1.2 Выбор среды программирования и отладки

Для разработки и отладки ПО была выбрана кроссплатформенная IDE PyCharm, предоставляющая средства для анализа кода и графический отладчик.

Данная среда разработки предоставляет встроенные инструменты для разработки:

- встроенный отладчик;
- профилировщик Python;
- полнофункциональный встроенный терминал;
- интеграция с системами контроля версий;
- навигация по проекту;
- рефакторинг кода.

### 3.1.3 Используемые библиотеки

В процессе реализации ПО были использованы следующие библиотеки:

- NumPy [21] – библиотека реализаций вычислительных алгоритмов, оптимизированных для работы с многомерными массивами;
- pandas [22] – библиотека для анализа данных, используемая для представления данных в формате таблиц;
- nltk [23] – библиотека для работы с естественным языком, используемая для токенизации текстов по предложениям и выделения стоп-слов;

- pymorphy2 [24] – библиотека для морфологического анализа для русского языка, используемая для лемматизации слов и определения их частей речи;
- scikit-learn [25] – библиотека, предоставляющая большой набор инструментов для машинного обучения, в частности подсчет метрик TF-IDF, обучение классификатора, получение предсказаний на его основ, оценка качества проведенной классификации;
- PyQt5 [26] – библиотека, предоставлящая набор расширений (биндингов) графического фреймворка Qt для языка программирования Python;
- Matplotlib [27] – библиотека для визуализации данных.

## 3.2 Система контроля версий

В процессе разработки для отслеживания вносимых изменений использовалась система контроля версий. Использование системы контроля версий позволяет вернуться к определенной версии проекта, например, в случае обнаружения ошибок в текущей версии или при изменении требований. Проект может быть подключен к удаленному репозиторию, что позволяет сохранять изменения не только локально, но и удалённо, тем самым предоставляя возможность доступа к коду с различных устройств и уменьшая риск утери кода проекта в случае возникновения различных неполадок.

В качестве системы контроля версий был выбран Git, так как она обладает следующими свойствами:

- является распределенной системой контроля версий – полностью копирует локальный репозиторий и позволяет иметь несколько независимых копий проекта;
- предоставляет широкие возможности для просмотра истории изменений в проекте и внесения корректировок в нее;
- транзакционный подход к управлению пакетами;

- поддержка ветвления;
- поддерживается средой разработки PyCharm;
- поддерживается хостингом репозиториев GitHub.

## 3.3 Требования к вычислительной системе

Для запуска программного обеспечения необходимо установить интерпретатор Python 3.9 и все библиотеки, приведенные в пункте 3.1.3.

Так как ПО реализовано на языке Python, который является кроссплатформенным, отсутствуют требования к использованию определенной операционной системы.

Алгоритм приведения текста к виду частеречных N-грамм работает с большим объемом данных, алгоритму классификации также требуется наличие свободной оперативной памяти на вычислительной системе, следовательно, объём доступной оперативной памяти ЭВМ должен составлять не менее 5 ГБ.

## 3.4 Формат входных данных

Входными данными являются текстовые файлы в формате TXT. Файлы TXT содержат текст без форматирования, информация содержится в виде строк. Файлы данного формата могут быть открыты в любой операционной системе в любой предоставляемой программе для работы с текстом.

Текст должен быть написан на русском языке (фрагменты на иностранных языках будут удалены при обработке текстовых данных). Тексты закодированы в соответствии со стандартом UTF-8, позволяющим компактно хранить и передавать символы Unicode, используя переменное число байт (от одного до четырех).

## 3.5 Формат хранения данных

Текстовые документы, поступающие на вход, располагаются в файловой системе, во вложенной директории внутри папки проекта с исходным кодом.

После приведения исходных TXT файлов к виду частеречных N-грамм, созданная выборка хранится в файлах формата CSV (Comma-Separated Values — значения, разделённые запятыми). Файл CSV – текстовый, поделенный на отдельные строки. Каждая строка – это отдельная строка таблицы, а столбцы отделены один от другого специальными символами-разделителями – запятыми.

Файлы, хранящие данные можно разделить на два типа: файл, в котором текст представлен в исходном виде и файлы, в которых текст преобразован к виду последовательности частеречных N-грамм. Каждый файл состоит из двух столбцов: первый – сам текст, второй – его автор.

## 3.6 Формат выходных данных

Выходные данные модуля классификации можно условно разделить на два типа:

- метрики качества классификатора (точность для обучающей и тестовой выборок и F-мера для тестовой выборки);
- прогнозирование автора и процент вероятности авторства для нового классифицируемого текста.

Полученные выходные данные выводятся на графический интерфейс программного обеспечения.

Модуль создания выборки формирует выходные данные в формате CSV, в которых хранятся полученные выборки в виде частеречных N-грамм.

## 3.7 Схема разработанного программного обеспечения

На рисунке 7 представлена UML-диаграмма разработанного ПО, реализующего предлагаемый метод.

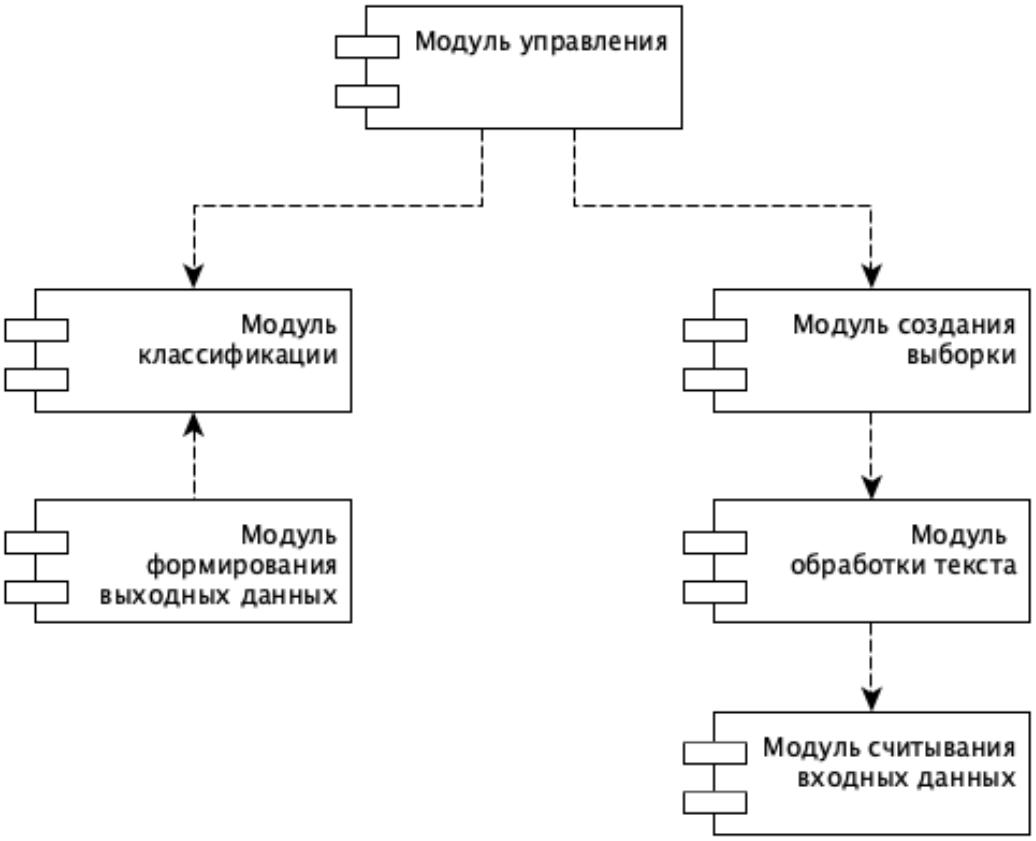



*Рисунок 7 – Схема разработанного ПО*

Пользователь имеет доступ к модулю классификации и модулю обучения классификатора через пользовательский интерфейс.

Внутри модуля обработки текста реализованы токенизатор и лемматизатор, которые проводят токенизацию и лемматизацию текста. Модуль обработки текста отвечает за выделение N-грамм и определение морфологических признаков слов (в частности, частей речи).

## 3.8 Нормальные формы слов

Нормальная форма слова определяются на этапе лемматизации. Для получения всех известных форм слов была использована библиотека pymorphy2 языка Python, в основе которой лежат словари, принятые в открытом корпусе

русского языка OpenCorpora [28]. В библиотеке OpenCorpora содержатся данные об известных формах каждого слова, в том числе и нормальная форма слова. Помимо этого, для каждой словоформы определен набор ее морфологических признаков.

## 3.9 Частеречные N-граммы

Части речи слов определяются при помощи морфологического анализа. Для проведения морфологического была использована библиотека pymorphy2. Библиотека pymorphy2 использует для русского языка словари и граммемы, определённые в открытом корпусе русского языка OpenCorpora. Библиотека pymorphy2 также позволяет выдвинуть предположение о части речи незнакомой словоформы на основе имеющихся в корпусе данных для обучения. В случае, если pymorphy2 не смог определить часть речи слова и вернул NONE, данное слово убирается из текста, поскольку может создать «выбросы» в обучающей выборки и понизить точность классификатора. В таблице 1 перечислены граммемы, принятые в словаре OpenCorpora, и их сокращенные названия.

*Таблица 1. Граммемы и обозначения*

| Обозначение | Граммема |
|---|---|
| NOUN | Имя существительное |
| ADJF | Имя прилагательное (полное) |
| ADJS | Имя прилагательное (краткое) |
| COMP | Компаратив |
| VERB | Глагол (личная форма) |
| INFN | Глагол (инфинитив) |
| PRTF | Причастие (полное) |
| PRTS | Причастие (краткое) |
| GRND | Деепричастие |

| | |
|---|---|
| NUMR | Числительное |
| ADVB | Наречие |
| NPRO | Местоимение-существительное |
| PRED | Предикатив |
| PREP | Предлог |
| CONJ | Союз |
| PRCL | Частица |
| INTJ | Междометие |

### 3.10 Установка программного обеспечения

Для запуска разработанного программного продукта требуется установить на ПК интерпретатор для Python 3.9. и создать виртуальное окружение для проекта. В листинге 1 приведены команды, которые необходимо выполнить для создания нового виртуального окружения.

Листинг 1 – Команды для установки виртуального окружения

```
python3 -m pip install —user virtualenv
python3 -m venv env
source env/bin/activate
```

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 2.

Листинг 2 – Команда для установки всех необходимых библиотек

```
pip3 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.

Листинг 3 – Установка словарей nltk

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

### 3.11 Интерфейс пользователя

Пользовательский интерфейс (рисунок 8) разрабатывался в программе QtDesigner. Интеграция с кодом на Python осуществлена посредством использования библиотеки PyQt5, предоставляющей различные классы для работы с объектами интерфейса.

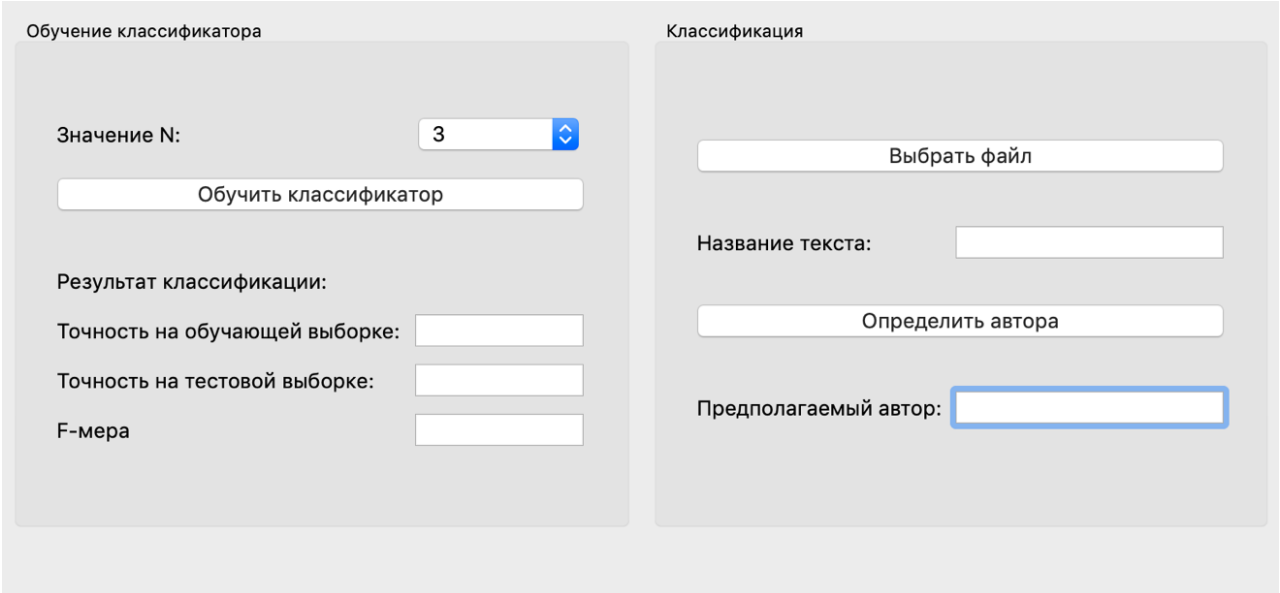



*Рисунок 8 – Интерфейс ПО*

Интерфейс ПО поделен на две области. В левой части окна расположен функционал обучения классификатора и получения метрик качества для обучающей и тестовой выборок. В Данном окне находит поле выбора значения N в диапазоне от 2 до 6 включительно. В зависимости от выбора пользователя

классификатор будет обучен на выборке с определенным размером N-грамм. Ручной ввод любого значения N отсутствует по причине того, что корпус документов полностью приводится к виду частеречных N-грамм за время, равное 6 часам. Поэтому создание выборок недоступно из интерфейса, и пользователь может работать только с заранее обработанными данными. При нажатии на кнопку «классифицировать» выборка из корпуса документов разбивается на две части: 80% корпуса – обучающая выборка, 20% – тестовая. Полученные в ходе классификации метрики выводятся в таблицу отдельно для обучающей и тестовой выборок.

В правой части окна расположен функционал предсказания автора неразмеченного текста. В данном блоке будет использовано то же значение N, которое выбрано в левой части окна. Если пользователь попытается выбрать файл, не обучив сначала классификатор, ему будет возвращена ошибка (рисунок 9).

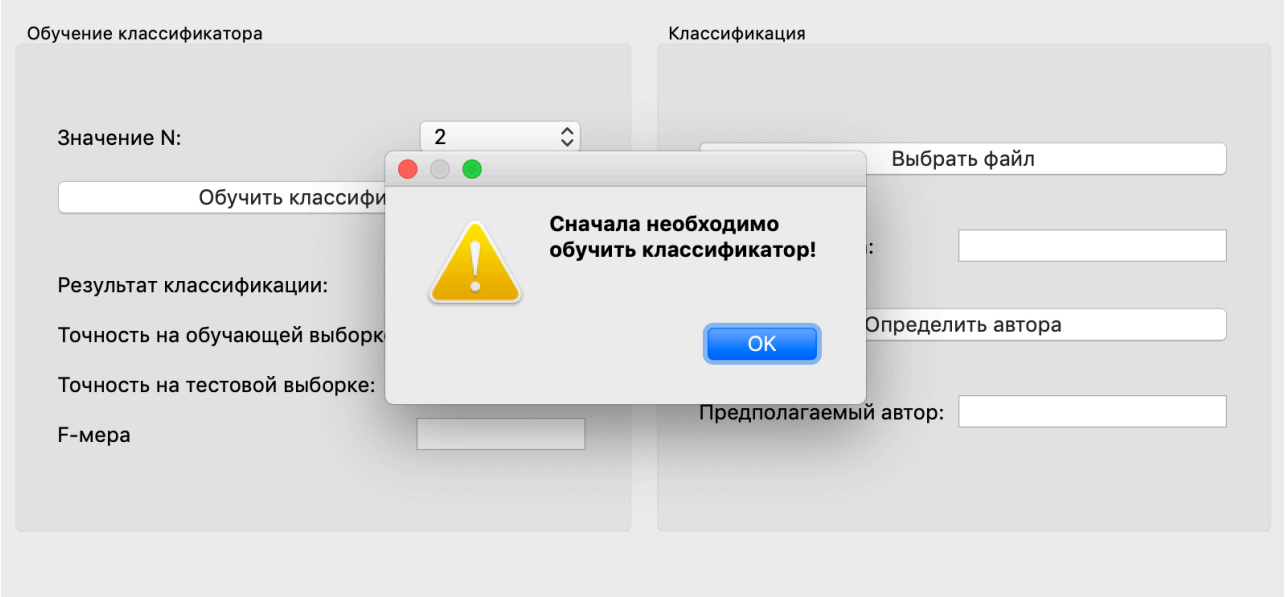



*Рисунок 9 – Ошибка, возникающая в случае, если классификатор не обучен*

При нажатии на кнопку «выбрать файл» открываются файловая система компьютера, внутри которой необходимо выбрать интересующий текстовый файл. После того, как пользователь выберет файл, название произведение

отобразится в поле «Название текста». Если пользователь нажмёт кнопку «Определить автора» до того, как он выберет файл, ему будет возвращена ошибка (рисунок 10).

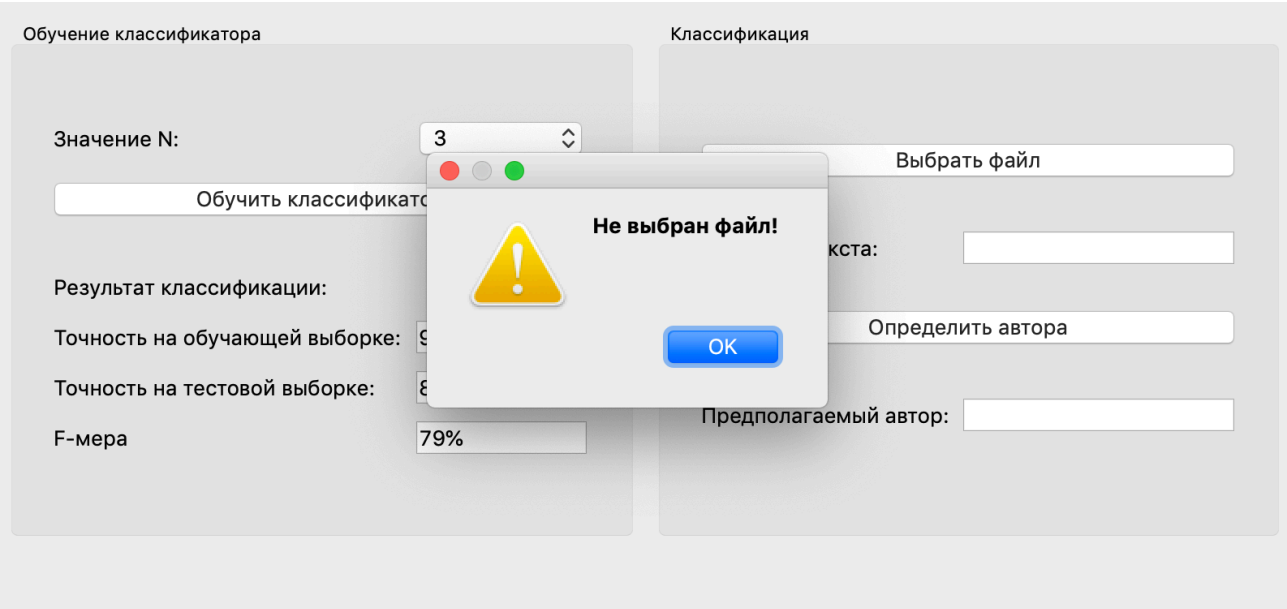



*Рисунок 10 – Ошибка, возникающая в случае, если файл не выбран*

Если файл выбран, то запускается алгоритм классификации. На экран выводится название выбранного текста для лучшей наглядности. После того, как классификатор закончит работать, на экран также будет выведена фамилия наиболее вероятного автора и процент вероятности того, что именно он является автором обработанного произведения (рисунок 11).

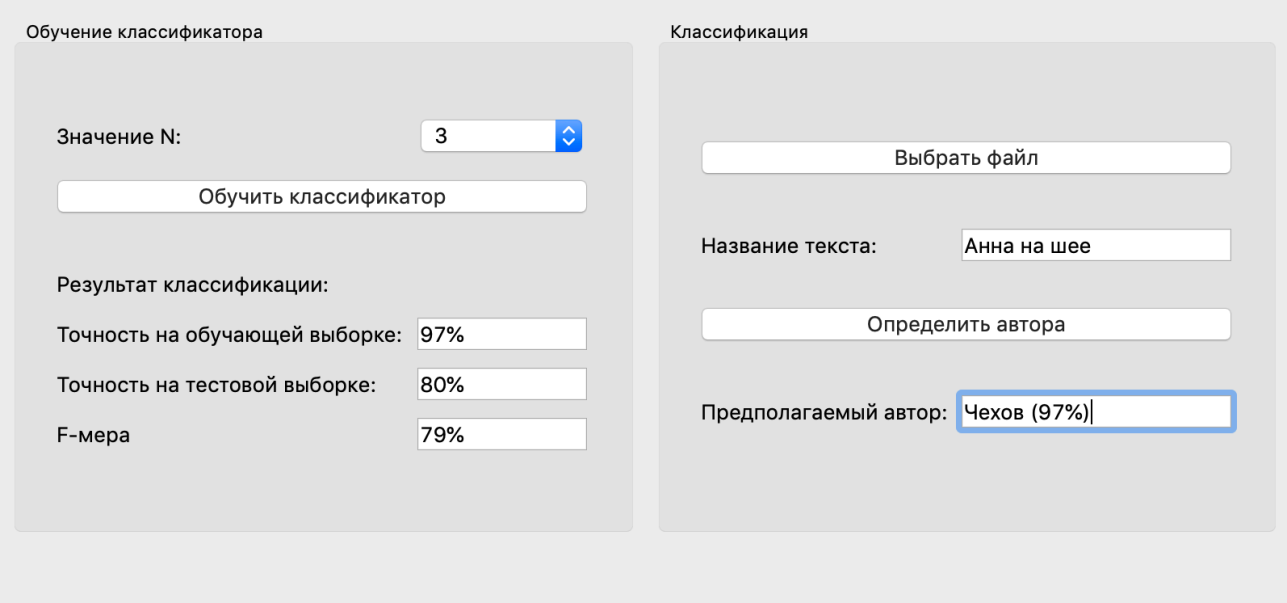



*Рисунок 11 – Результат работы классификации*

## 3.12 Руководство пользователя

Для создания выборки в виде последовательности частеречных N-грамм из имеющего у пользователя корпуса документов, необходимо разместить папку с документами внутри корневого каталога проекта и запустить скрипт dataset.py.

Скрипт может запущен из графического интерфейса среды разработки, либо командой из терминала (листинг 4).

Листинг 4 – Команда для создания обучающих выборок

```
python3 dataset.py
```

Чтобы обучить классификатор на заранее созданных размеченных выборках и предсказать автора неразмеченного текста, пользователь может воспользоваться графическим интерфейсом приложения.

Для открытия приложения необходимо запустить скрипт main.py посредством интерфейса среды разработки или командой в терминале (листинг 5).

Листинг 5 – Команда для запуска приложения

```
python3 main.py
```

В результате разработанное программное обеспечение может установить автора текста или выдвинуть предположение, кто может быть автором (в таком случае принятие решения должно быть предоставлено эксперту).

## 3.13 Модульное тестирование

Из размеченных данных была выбрана тестовая выборка, состоящая из 30 объектов, на которой проводилась классификация. Представлены тексты 5 авторов: Горький, Достоевский, Толстой, Тургенев, Чехов. Тестовая выборка состоит из частеречных триграмм. Тестирование работы классификатора заключается в проверке того, совпал ли предсказанный классификатором автор с настоящим автором текста. Ожидаемые и полученные в ходе тестирование значения приведены в таблице 2.

*Таблица 2. Тестирование классификатора*

| Ожидаемый результат | Полученный результат |
|---|---|
| Тургенев | Тургенев |
| Тургенев | Горький |
| Горький | Горький |
| Достоевский | Достоевский |
| Толстой | Толстой |
| Горький | Горький |
| Тургенев | Тургенев |
| Чехов | Чехов |
| Толстой | Толстой |
| Тургенев | Тургенев |

| | |
|---|---|
| Горький | Горький |
| Тургенев | Тургенев |
| Толстой | Толстой |
| Чехов | Чехов |
| Чехов | Толстой |
| Горький | Горький |
| Горький | Горький |
| Тургенев | Достоевский |
| Чехов | Чехов |
| Чехов | Толстой |
| Чехов | Достоевский |
| Чехов | Чехов |
| Достоевский | Достоевский |
| Достоевский | Достоевский |
| Достоевский | Достоевский |
| Тургенев | Тургенев |
| Толстой | Толстой |
| Тургенев | Тургенев |
| Чехов | Толстой |
| Горький | Горький |

Полученная точность составила 80%. Классификатор дважды совершил ошибку на классе Тургенева и три раза на классе Чехова.

### 3.14 Выводы

В данном разделе были рассмотрены средства, используемые для разработки программного обеспечения, необходимые для функционирования ПО библиотеки, требования к вычислительной системе. Описан формат входных и выходных данных, а также формат хранения данных. Приведено описание

интерфейса пользователя и руководство для установки и использования ПО. Рассмотрена программная реализация лемматизации и определении частей речи слов в тексте. Проведено модульное тестирование.

# 4 Экспериментальный раздел

Целью данной работы является создание метод определения признаков авторского стиля на основе выделения N-грамм в текстах, а также разработка использующего его метода классификации текстов. Основными оцениваемыми параметром метода является значение N – длина используемых при обработке текстов N-грамм. Помимо это исследуются зависимость качества предсказаний от наличия в текстах стоп-слов и влияние качества проведенного морфологического анализа. Также будет проведена апробация метода классификации.

## 4.1 Зависимость качества классификации от значения N

### 4.1.1 Описание исследования

Размеченный корпус для проведения данного исследования составил 150 произведений, написанных на русском языке: по 30 текстов от 5 авторов (Горький, Достоевский, Толстой, Тургенев, Чехов). Каждый текст был предобработан и представлен в виде частеречных N-грамм, таким образом, было составлено 5 корпусов с N равным от 2 до 6. При мере увеличения значения N увеличивается и объём корпуса.

Для каждой из 5 выборок было проведено обучение классификатора на обучающей выборке и классификация тестовой выборки. Весь корпус разбивается на обучающую и тестовую выборки согласно следующему соотношению: 80% исходного корпуса – обучающая выборка (120 текстов), 20% исходного корпуса – тестовая выборка (30 текстов).

В качестве метрик оценивания использовались точность (для обучающей и тестовой выборок) и F-мера (для тестовой выборки).

### 4.1.2 Результаты исследования

Результаты проведенного исследования приведены в таблице 3 и показаны на рисунках 12 и 13.

*Таблица 3. Результаты параметризации параметра N*

| **Значение N** | **Объём корпуса (количество N-грамм)** | **Точность на обучающей выборке** | **Точность на тестовой выборке** | **F-мера** |
|---|---|---|---|---|
| 2 | 2 972 956 | 83% | 76% | 76% |
| 3 | 2 789 053 | 96% | 80% | 79% |
| 4 | 2 606 454 | 100% | 76% | 76% |
| 5 | 2 429 961 | 100% | 70% | 69% |
| 6 | 2 264 315 | 100% | 56% | 51% |

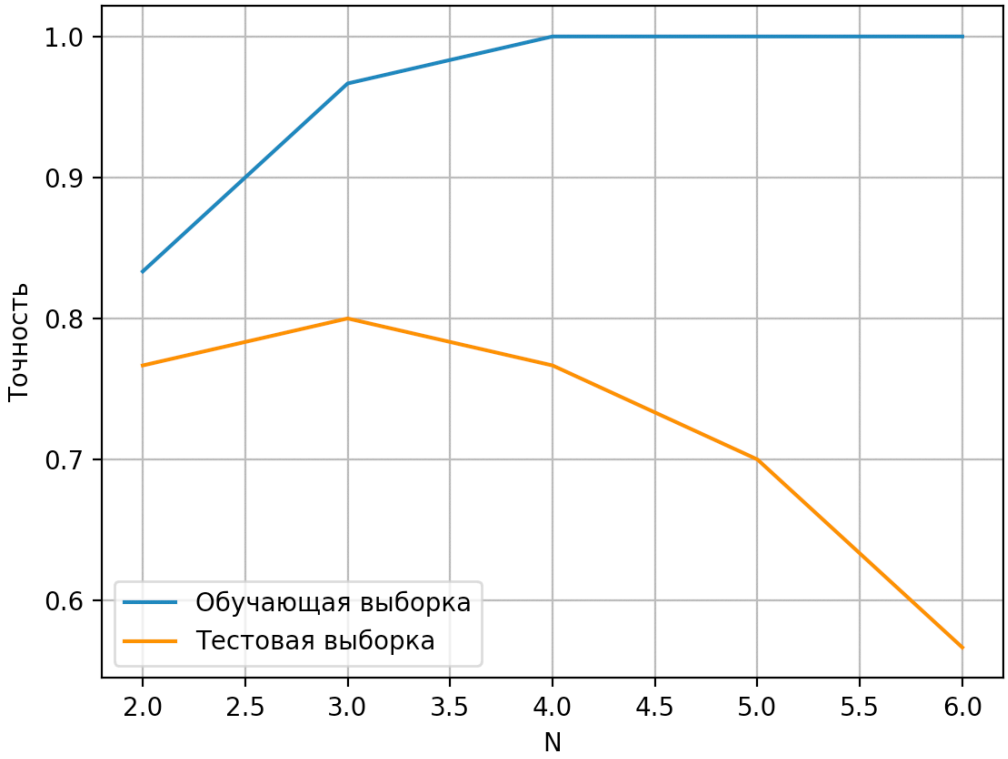



*Рисунок 12 – График зависимости точности классификации от значения N*

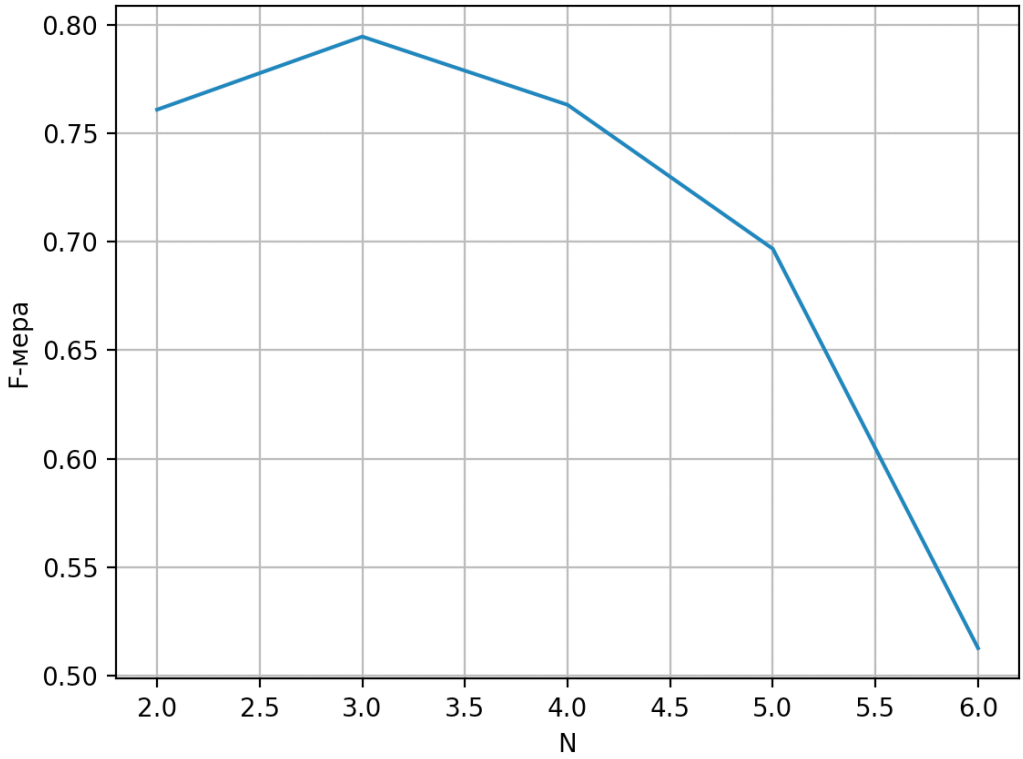



*Рисунок 13 – График зависимости F-меры от значения N*

Исходя из полученных результатов, можно сделать вывод, что самые высокие значения точности и F-меры достигаются при использовании триграмм. Значения точности на тестовой выборке при N равном двум или четырем приблизительно равны, при N равном 5 и 6 значение точно уменьшается. Аналогичная ситуация наблюдается и для метрики F-меры.

Также можно отметить, что при увеличении значения N до 4 точность на обучающей выборке возрастает до 100%.

## 4.2 Зависимость качества классификации от наличия стоп-слов

### 4.2.1 Описание исследования

На этапе предварительной обработки текстов из них часто убирают стоп-слова – такие слова, которые встречаются практически в каждом предложении, при игнорировании данных слов исходное предложение не теряет своего смысла. Для проведения данного исследования из текстов, полученных после этапа токенизации, также были удалены стоп-слова. Из полученных текстов было составлено 5 корпусов из частеречных N-грамм с N равным от 2 до 6.

Список русских стоп-слов взять из библиотеки NLTK и насчитывает 151 значение. Некоторые из встречающихся стоп-слов: и, в, во, не, что, он, на, я, с, со, как, а, то, все, чтоб, без и т.д.

### 4.2.2 Результаты исследования

Результаты проведенного исследования приведены в таблице 4 и показаны на рисунках 14 и 15.

*Таблица 4. Результаты, полученные при удалении стоп-слов*

| N | Объём корпуса | Объём корпуса без стоп-слов | Точность на обучающей выборке | Точность на обучающей выборке (без стоп-слов) | Точность на тестовой выборке | Точность на тестовой выборке (без стоп-слов) | F-мера | F-мера (без стоп-слов) |
|---|---|---|---|---|---|---|---|---|
| 2 | 2 972 956 | 1 644 007 | 83% | 48% | 76% | 33% | 76% | 30% |
| 3 | 2 789 053 | 1 464 728 | 96% | 82% | 80% | 57% | 79% | 56% |
| 4 | 2 606 454 | 1 300 859 | 100% | 90% | 76% | 67% | 76% | 67% |
| 5 | 2 429 961 | 1 156 769 | 100% | 98% | 70% | 63% | 69% | 62% |
| 6 | 2 264 315 | 1 030 297 | 100% | 100% | 56% | 56% | 51% | 54% |

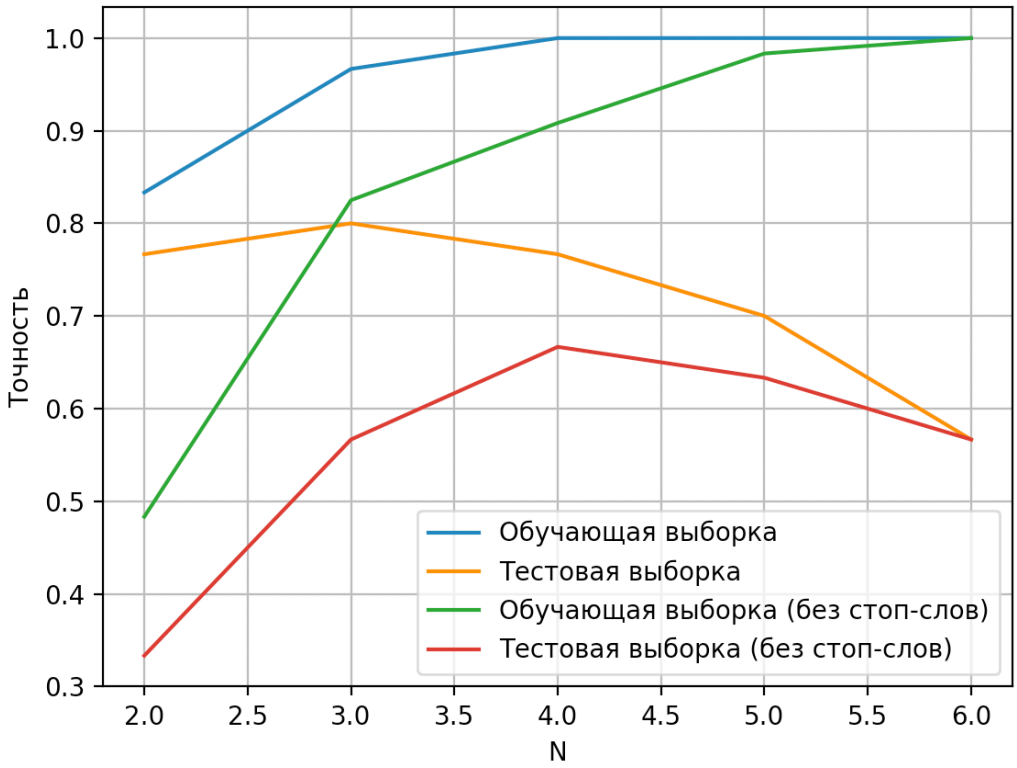



*Рисунок 14 – График зависимости точности классификации от значения N и стоп-слов*

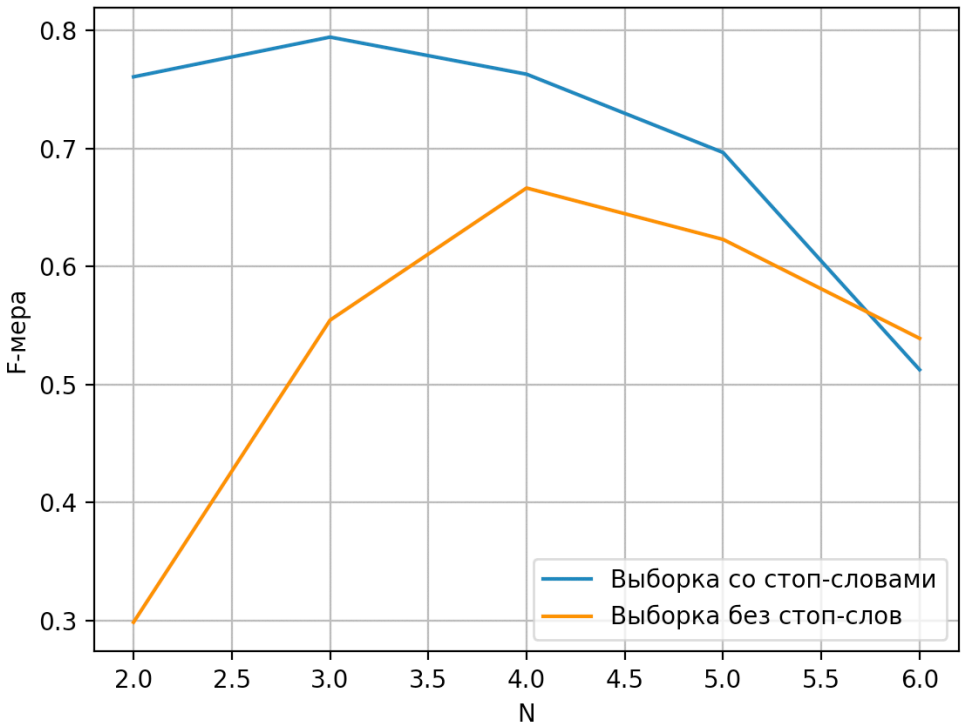



*Рисунок 15 – График зависимости F-меры от значения N и стоп-слов*

У корпусов без стоп-слов при использовании биграмм точность и значение F-меры значительно меньше, чем у корпусов со стоп-словами. По мере увеличение значения N значения метрик точности возрастают и приближаются к значениям, полученным на корпусах со стоп-словами. При N равном 6 значения точности на обучающей и тестовой выборках совпадают у корпусов со стоп-словами и без них. Значение F-меры при этом больше для корпуса с удаленными стоп-словами.

Исходя из этого можно сделать вывод, что удаление стоп-слов из текста имеет место быть только при использовании 6-грамм, т.к. иначе метрики качества будут ниже, чем при использовании корпусов со стоп-словами.

## 4.3 Зависимость качества классификации от морфологического анализа

### 4.3.1 Описание исследования

В данном исследование проверяется влияние качества морфологического анализа на качество классификации. В зависимости от проводимого морфологического анализа обработчик текста может как различать, так и не различать схожие части речи. Разница между двумя проведенными в данном исследованиями анализами заключается в том, что в первом случае инфинитивы глаголов (INFN) и личные глаголы (VERB) рассматривались как разные части речи, а во втором случае они объединены и рассматриваются как личные глаголы (VERB).

### 4.3.2 Результаты исследования

Результаты проведенного исследования приведены в таблице 5 и показаны на рисунках 16 и 17.

*Таблица 5. Результаты, полученные при различном морфологическом анализе*

| N | Объём корпуса | Точность на обучающей выборке | Точность на обучающей выборке (без инфинитивов) | Точность на тестовой выборке | Точность на тестовой выборке (без инфинитивов) | F-мера | F-мера (без инфинитивов) |
|---|---|---|---|---|---|---|---|
| 2 | 2 972 956 | 83% | 85% | 76% | 76% | 76% | 76% |
| 3 | 2 789 053 | 96% | 97% | 80% | 80% | 79% | 79% |
| 4 | 2 606 454 | 100% | 100% | 76% | 83% | 76% | 83% |
| 5 | 2 429 961 | 100% | 100% | 70% | 90% | 69% | 90% |
| 6 | 2 264 315 | 100% | 100% | 56% | 64% | 51% | 57% |

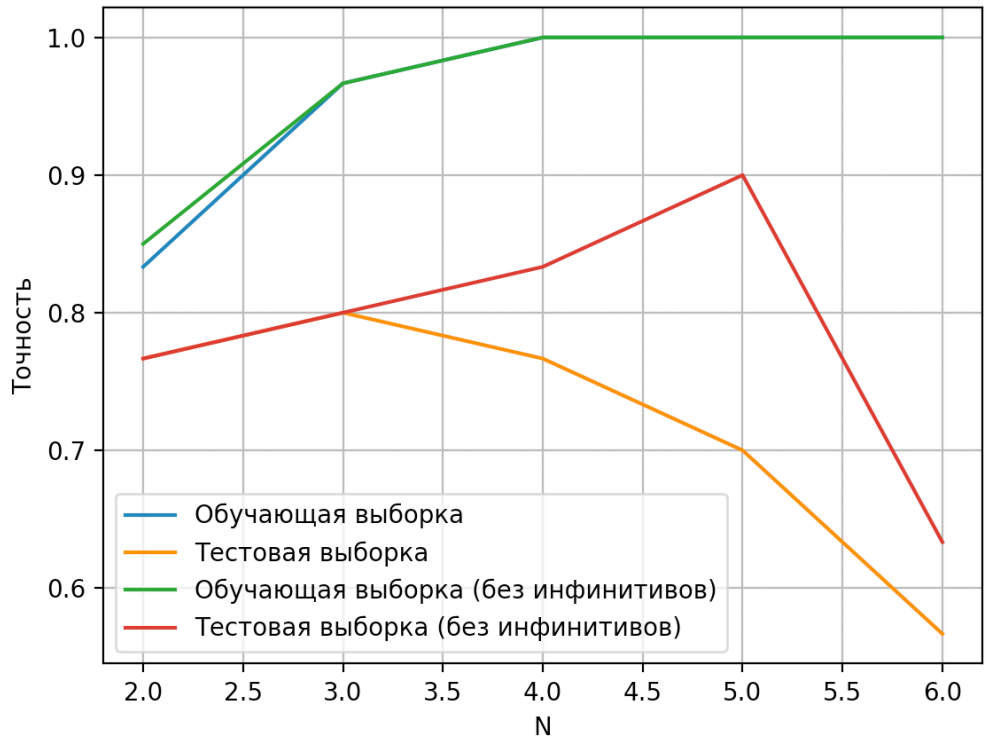



*Рисунок 16 – График зависимости точности классификации от значения N и морфологического анализа*

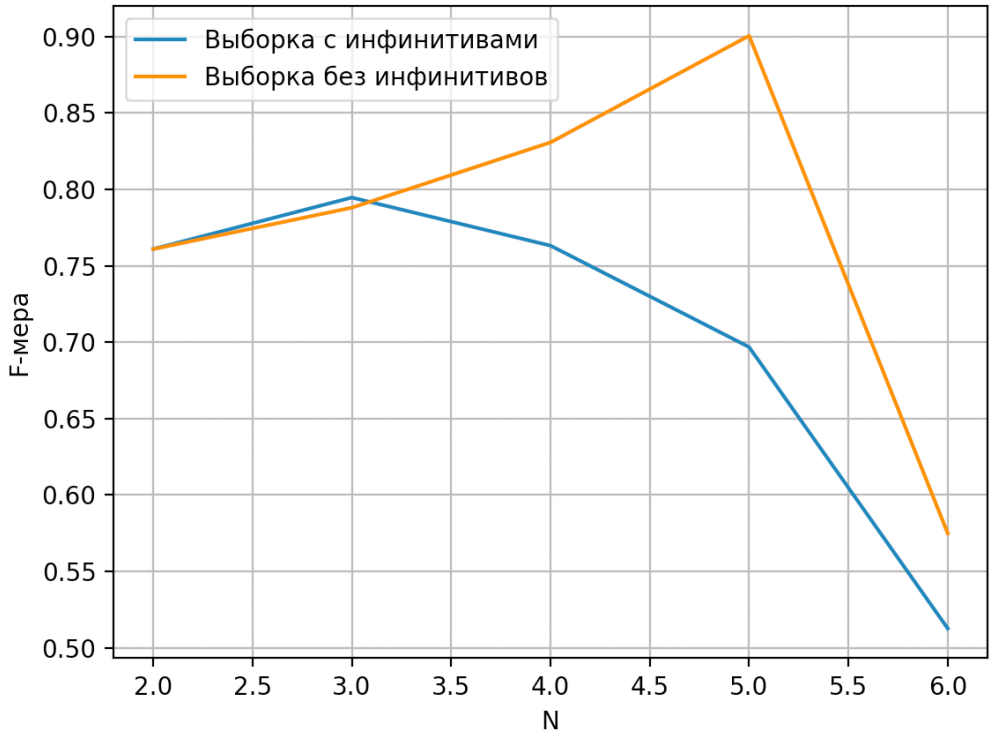



*Рисунок 17 – График зависимости F-меры от значения N и морфологического анализа*

Для обучающей выборки точность классификации обоих вариантов корпусов примерно совпала. Для тестовой выборки точность классификации корпусов с менее детальным морфологическим разбором, где глаголы в начальной форме не были выделены в качестве отдельной характеристики, оказалась выше при N равным четырем и значительно выше при N равным 5. Но при N равным шести точность резко упала и приблизилась к точности классификации, проведенной на корпусе с детальным морфологическим разбором. Аналогичная ситуация складывается и с измеренной F-мерой.

В результате явной зависимости от того, насколько точно морфологический анализ различает глаголы, выявлено не было: полученные на тестовых выборках результаты больше похожи на случайные выбросы, чем на закономерность.

## 4.4 Зависимость качества классификации от ядра в методе опорных векторов

### 4.4.1 Описание исследования

В предыдущих исследованиях в методе классификации использовалось Гауссово ядро RBF. В данном исследовании предлагается сравнить результаты метрик качества, полученные при использовании в методе опорных векторов следующих ядер: линейное, полиномиальное, Гауссово RBF, сигмоидальное.

Качества классификации с различными ядрами будет сравниваться на корпусах из 150 текстов, представленных виде частеречных N-грамм, где N принимает значение от до 6. Из текстов не были удалены стоп-слова, проведенный морфологический анализ различает инфинитивы и личные глаголы.

### 4.4.2 Результаты исследования

На рисунке 18 показан результат исследования точности классификации на обучающей выборке для различных N в зависимости от выбранного ядра:

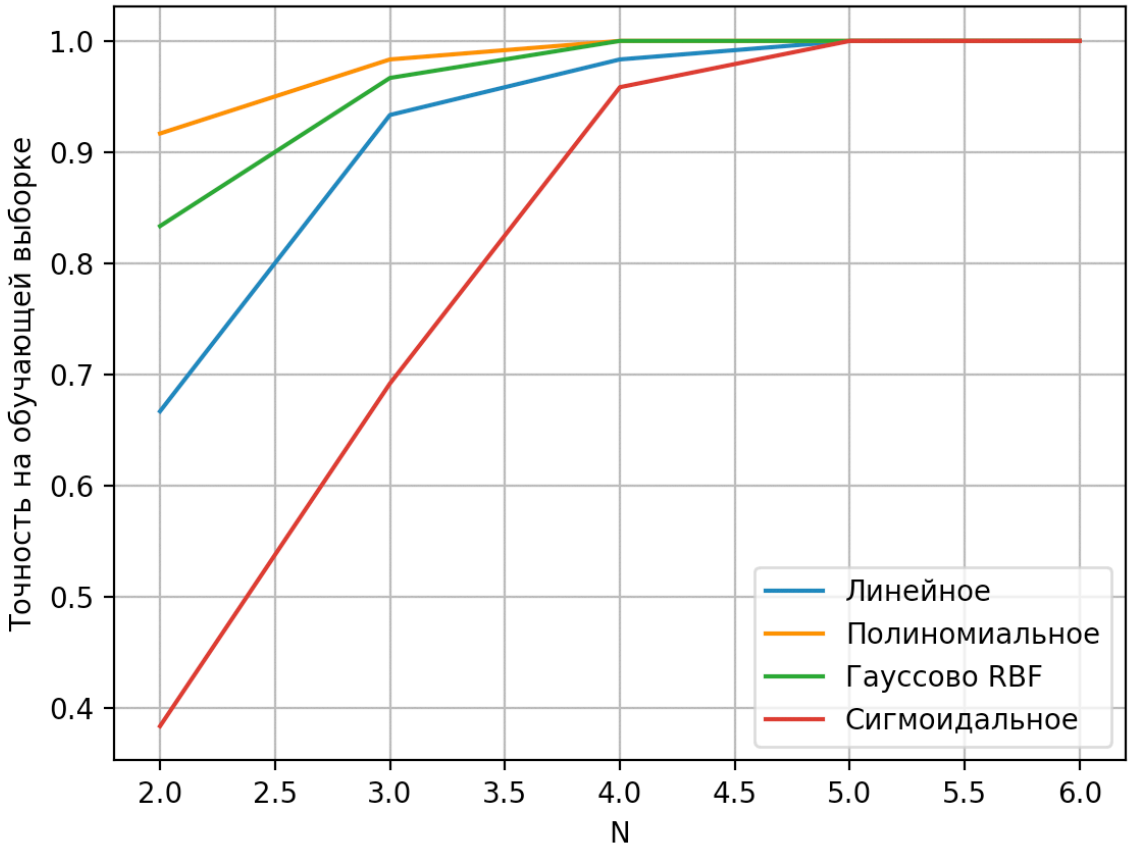



*Рисунок 18 – График зависимости точности классификации от значения N и ядра*

Из приведенного выше графика видно, что чем меньше N, тем больше разрыв между полученной точностью для различных ядер. Самый высокий результат получен с полиномиальным ядром, далее следуют Гауссово RBF и линейное, самая низка точность у сигмоидального ядра. При N равном 5 и выше результаты ядер становятся одинаковыми.

На рисунке 16 показан результат исследования точности классификации на тестовой выборке для различных N в зависимости от выбранного ядра:

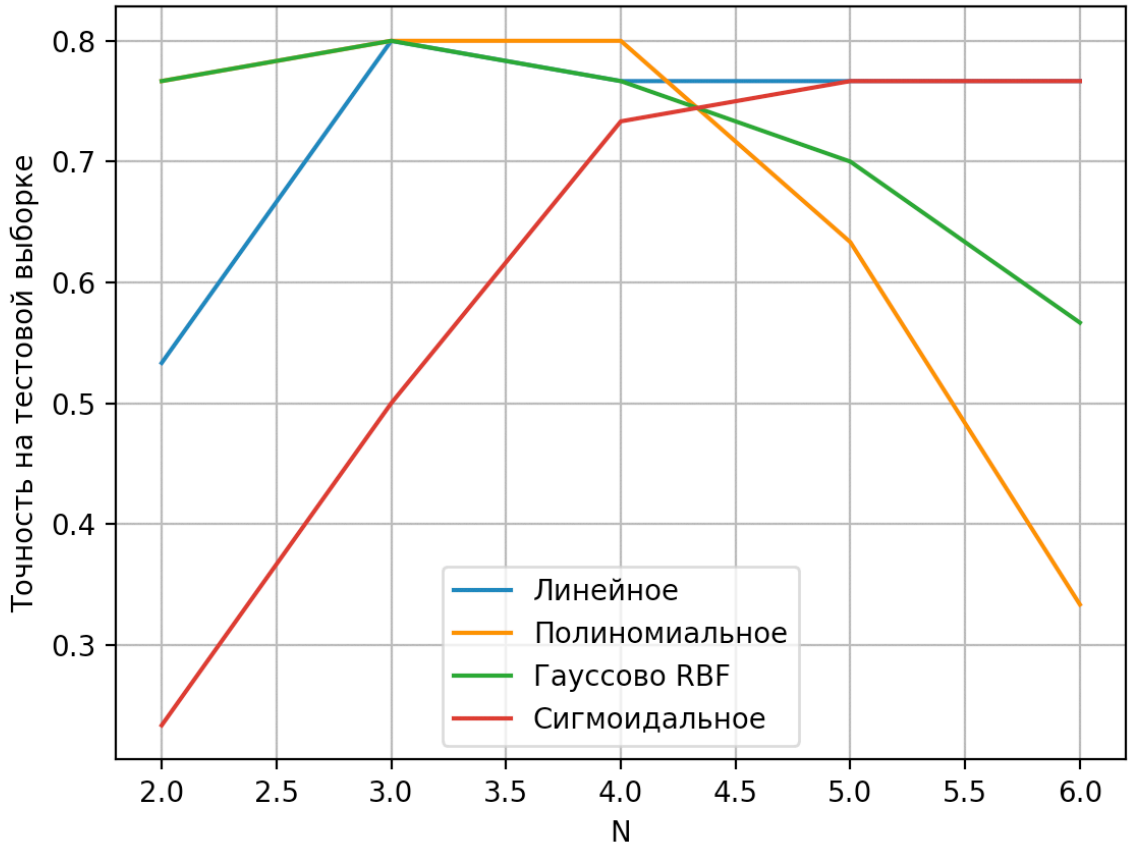



*Рисунок 19 – График зависимости точности классификации от значения N и ядра*

Как видно из рисунка 19, линейное ядро показывает наилучшую точность при N равном 3, полиномиальное при N равном 3 или 4, Гауссово RBF при N равном 3, сигмоидальное при N равном 5 и 6. Самая высокая точность у линейного, Гауссова и полиномиальное ядер при N=3, а также у полиномиального ядра при N=4.

На рисунке 20 показан результат исследования F-меры для различных N в зависимости от выбранного ядра:

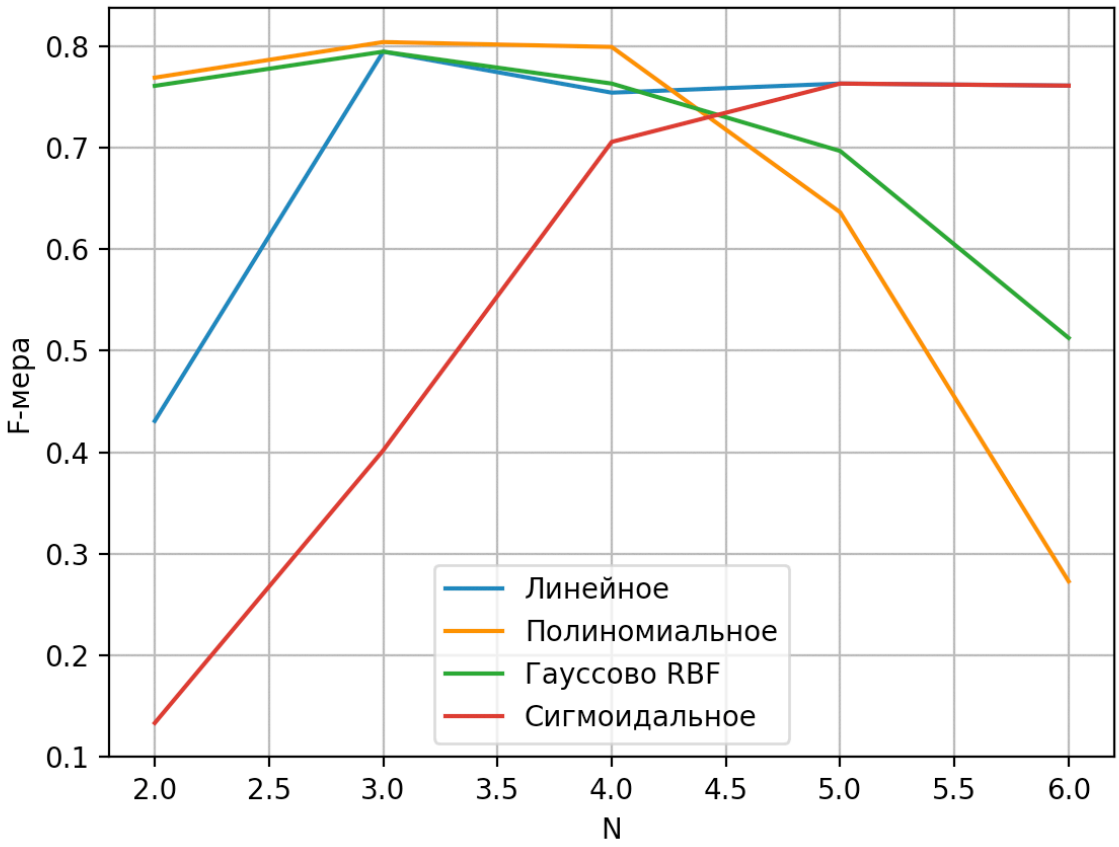



*Рисунок 20 – График зависимости F-меры от значения N и ядра*

Зависимости между результатами F-меры для различных ядер примерно совпадают с зависимости, полученными для точности на тестовой выборке. Но, в отличие от значения точности на тестовой выборке, для N равном 3 полиномиальное ядро показывает более высокую F-меру по сравнению с линейным и Гауссовым.

## 4.5 Выводы

В результате исследования разработанного метода определения признаков авторского стиля и использующего его алгоритма классификации было установлено, что наивысшие значения метрик качества (точность классификации и F-мера) достигаются при использовании частеречных триграмм.

Классификатор, обученный на корпусе, включающем стоп-слова, показывает лучшие значения метрик качества, чем классификатор, обученный на корпусе, из которого были удалены стоп-слова. Данная закономерность связана с тем, что классификатору необходимо учитывать порядок частей речи всех встречающихся слов, в том числе и стоп-слов. Поэтому для разработанного метода определения признаков авторского стиля важно учитывать часть речи каждого слова, для которого ее возможно определить.

Также классификатор показывает более высокую точность при более детальном морфологическом анализе, распознающем и разделяющем глаголы в начальной форме и глаголы в личной форме.

В ходе проведенных исследований была установлена зависимость между используемым в методе опорных векторов ядром и значением N: разные ядра показывают разную точность при разном N. Для каждого ядра наивысшие значения метрик качества достигаются при разном N. Чаще самый высокий показатель метрик встречается при N=3 (полиномиальное, Гауссово, линейное ядро). Но линейное ядро показывает более низкую точность на обучающей выборке, поэтому было установлено, что для метода классификации, использующего разработанный метод определения признаков авторского стиля, больше всего подходят Гауссово и полиномиальное ядра.

# ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы был разработан, реализован и исследован метод определения признаков авторского стиля для текстов на русском языке.

В результате проделанной работы были решены следующие задачи:

- Разработан метод определения признаков авторского стиля
- Проанализирована предметная область
- Сформулированы основные положения разрабатываемого метода определения признаков авторского стиля и использующего его метода классификации
- Разработано ПО, реализующее предложенные методы
- Проведена оценка точности классификации прозаических текстов по выделяемым признакам, достигнута точность 80%

Были выполнены все поставленные задачи.

Для реализованного метода можно предложить следующие пути развития:

- увеличение точности классификация за счет создания обучающей выборки, обеспечивающей получение более высоких значений метрик качества;
- проведение более детального исследования качества морфологического анализа;
- ускорение работы метода путем его распараллеливания.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Мартыненко Г. Я. Стилеметрия: Возникновение и становление в контексте междисциплинарного взаимодействия. Часть 2. Первая половина XX в.: Расширение междисциплинарных контактов стилеметрии. СТРУКТУРНАЯ И ПРИКЛАДНАЯ ЛИНГВИСТИКА, № 11, 2015, С. 9-28.

2. Фоменко В.П., Фоменко Т.Г. Авторский инвариант русских литературных текстов. Предисловие А.Т. Фоменко // Фоменко А.Т. Новая хронология Греции: Античность в средневековье. Т. 2. М.: Изд-во МГУ, 1996. С. 768-820.

3. Ермакович М.В. Автоматическое определение границ слова в русском тексте с помощью комплекса лингвистических правил. Компьютерная лингвистика и интеллектуальные технологии: по материалам международной конференции «Диалог 2017» Москва, 31 мая — 3 июня 2017.

4. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика: учеб. пособие / Большакова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В.

5. Chomsky N. Three models for the description of language // IRE Transactions on Information Theory. — 1956. — Vol. 2, no. 3. — P. 113–124.

6. Tesnière L. Elements de syntaxe structurale. — Editions Klincksieck, 1959.

7. Mel’cuk I. A. Dependency syntax: theory and practice. — ŠUNY Press, 1988. — P. 428.

8. Abney S. P. Parsing by chunks // Principle-Based Parsing. — Kluwer Academic Publishers, 1991. — P. 257–278.

9. Хмелев Д.В. Классификация и разметка текстов с использованием методов сжатия данных. Краткое введение [электронный ресурс] - режим доступа: URL: http://compression.ru/download/articles/classif/intro.html – Дата обращения: 11.12.2021

10. Lai S., Xu L., Liu K., Zhao J. Recurrent Convolutional Neural Networks for Text Classification. // AAAI, 2015. P. 2267 – 2273.

11. Werbos P.J. Backpropagation through time: what it does and how to do it. // Proceedings of the IEEE 78, Harvard, 1990. Issue 10. P. 1550 – 1560.

12. Программы анализа и лингвистической обработки текстов. Русская виртуальная библиотека [электронный ресурс] - режим доступа: URL: https://rvb.ru/soft/catalogue/c01.html (дата обращения: 11.12.2020).

13. Гудков В. Ю., Гудкова Е. Ф. N-граммы в лингвистике // Вестник Челябинского государственного университета. 2011. № 24 (239). Филология. Искусствоведение. Вып. 57. С. 69–71.

14. Guthrie David, Allison Ben, Liu Wei, Guthrie Louise, Wilks Yorick. (2006). A Closer Look at Skip-gram Modelling. Proc. of the Fifth International Conference on Language Resources and Evaluation.

15. Shahzad Qaiser, Ramsha Ali. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications (0975 – 8887) Volume 181 – No.1, July 2018

16. Murphy K.P. Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series). The MIT Press, 2012. ISBN: 0262018020.

17. Батура Т.В. Методы автоматической классификации текстов // Программные продукты и системы. 2017. №1.

18. Hastie, T., Tibshirani R., Friedman J. Chapter 15. Random Forests // The Elements of Statistical Learning: Data Mining, Inference, and Prediction. — 2nd ed. — Springer-Verlag, 2009. — 746 p. — ISBN 978-0-387-84857-0.

19. Tong S., Koller D. Support vector machine active learning with applications to text classification // The Journal of Machine Learning Research, 2002. Issue 2. P. 45 – 66.

20. Duan, K.-B., Keerthi, S.S.: Which is the best multiclass SVM method? An empirical study. Technical Report CD-03-12, Control Division, Department of Mechanical Engineering, National University of Singapore. (2003)

21. NumPy [электронный ресурс]. – режим доступа: URL: https://numpy.org – Дата обращения: 04.05.2021

22. pandas [электронный ресурс]. – режим доступа: URL: https://pandas.pydata.org – Дата обращения: 04.05.2021

23. NLTK [электронный ресурс]. – режим доступа: URL: https://www.nltk.org – Дата обращения: 04.05.2021

24. Pymorphy2 [электронный ресурс]. – режим доступа: URL: https://pymorphy2.readthedocs.io/en/stable/ – Дата обращения: 04.05.2021

25. scikit-learn [электронный ресурс]. – режим доступа: URL: https://scikit-learn.org/stable/ – Дата обращения: 04.05.2021

26. PyQt5 [электронный ресурс]. – режим доступа: URL: https://pypi.org/project/PyQt5/ – Дата обращения: 04.05.2021

27. Matplotlib [электронный ресурс]. – режим доступа: URL: https://matplotlib.org – Дата обращения: 04.05.2021

28. OpenCorpora [электронный ресурс]. – режим доступа: URL: http://opencorpora.org/dict.php?act=gram – Дата обращения: 06.05.2021

# ПРИЛОЖЕНИЕ А

Таблица 6. Пример преобразования текста к виду частеречных биграмм

| | |
|---|---|
| **Исходный текст** | У самой воды большими куртинами выглядывали из зарослей мяты невинные голубоглазые незабудки. А дальше, за свисающими петлями ежевики, цвела по откосу дикая рябина с тугими желтыми соцветиями. |
| **Токенизированный текст** | 'у самой воды большими куртинами выглядывали из зарослей мяты невинные голубоглазые незабудки', 'а дальше за свисающими петлями ежевики цвела по откосу дикая рябина с тугими желтыми соцветиями' |
| **Лемматизированный текст** | 'у сам вода больший куртина выглядывать из заросль мятый невинный голубоглазый незабудка', 'а далёкий за свисать петля ежевика цвести по откос дикий рябина с тугой жёлтый соцветие' |
| **Текст, разбитый на биграммы** | [['у', 'сам'], ['сам', 'вода'], ['вода', 'больший'], ['больший', 'куртина'], ['куртина', 'выглядывать'], ['выглядывать', 'из'], ['из', 'заросль'], ['заросль', 'мятый'], ['мятый', 'невинный'], ['невинный', 'голубоглазый'], ['голубоглазый', 'незабудка']], [['а', 'далёкий'], ['далёкий', 'за'], ['за', 'свисать'], ['свисать', 'петля'], ['петля', 'ежевика'], ['ежевика', 'цвести'], ['цвести', 'по'], ['по', 'откос'], ['откос', 'дикий'], ['дикий', 'рябина'], ['рябина', 'с'], ['с', 'тугой'], ['тугой', 'жёлтый'], ['жёлтый', 'соцветие']] |

| | |
|---|---|
| **Текст в виде частеречных биграмм** | PREPADJF ADJFNOUN NOUNADJF ADJFNOUN NOUNVERB VERBPREP PREPNOUN NOUNADJF ADJFADJF ADJFADJF ADJFNOUN CONJADJF ADJFPREP PREPVERB VERBGRND GRNDNOUN NOUNVERB VERBPREP PREPNOUN NOUNADJF ADJFNOUN NOUNPREP PREPADJF ADJFADJF ADJFNOUN |