|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Ю.В. Строганов\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2021 г.*

# **РЕФЕРАТ**

Расчётно-пояснительная записка, ?? с., ?? рисунков, ?? табл., ?? источников.

# **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 словоформ представлен как кортеж из частей речи этих словоформ

**Лемма** — этоначальная 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.

СОДЕРЖАНИЕ

[РЕФЕРАТ 2](#_Toc73182064)

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 3](#_Toc73182065)

[ВВЕДЕНИЕ 6](#_Toc73182066)

[1 Аналитический раздел 8](#_Toc73182067)

[1.1 Описание предметной области 8](#_Toc73182068)

[1.2 Этапы обработки текста 9](#_Toc73182069)

[1.2.1 Токенизация 9](#_Toc73182070)

[1.2.2 Нормализация 10](#_Toc73182071)

[1.3 Этапы анализа текста 11](#_Toc73182072)

[1.4 Методы и решения определения авторского инварианта 13](#_Toc73182073)

[1.4.1 Метод полного синтаксического анализа 13](#_Toc73182074)

[1.4.2 Метод энтропийной классификации 15](#_Toc73182075)

[1.4.3 Рекуррентные нейронные сети 15](#_Toc73182076)

[1.4.4 Метод анализа длин слов 17](#_Toc73182077)

[1.4.5 Применения методов из теории вероятности и математической статистики 18](#_Toc73182078)

[1.4.6 Метод выделения N-грамм 18](#_Toc73182079)

[1.5 Извлечение признаков из текста 20](#_Toc73182080)

[1.5.1 Счётчики слов 20](#_Toc73182081)

[1.5.2 TF-IDF 20](#_Toc73182082)

[1.6 Классификация текстов 21](#_Toc73182083)

[1.6.1 Метод «наивной» байесовской классификации 22](#_Toc73182084)

[1.6.2 Метод k ближайших соседей 22](#_Toc73182085)

[1.6.3 Метод деревьев решений 23](#_Toc73182086)

[1.6.4 Метод опорных векторов 24](#_Toc73182087)

[1.7 Выводы 25](#_Toc73182088)

[2 Конструкторский раздел 26](#_Toc73182089)

[2.1 Структура разрабатываемого программного обеспечения 26](#_Toc73182090)

[2.2 Функциональная модель метода определения признаков авторского стиля 26](#_Toc73182091)

[2.3 Функциональная модель обучения метода классификации 28](#_Toc73182092)

[2.4 Функциональная модель метода классификации 28](#_Toc73182093)

[2.5 Описание этапов работы разрабатываемого алгоритма 30](#_Toc73182094)

[2.5.1 Предобработка текстовых документов 30](#_Toc73182095)

[2.5.2 Выделение N-грамм 30](#_Toc73182096)

[2.5.3 Получение частеречных N-грамм 31](#_Toc73182097)

[2.5.4 Формирование матрицы признаков 31](#_Toc73182098)

[2.5.5 Нормализация матрицы признаков 32](#_Toc73182099)

[2.5.6 Обучение классификатора 33](#_Toc73182100)

[2.5.7 Классификация 37](#_Toc73182101)

[2.5.8 Метрики оценки качества классификации 38](#_Toc73182102)

[2.6 Выводы 39](#_Toc73182103)

[3 Технологический раздел 40](#_Toc73182104)

[3.1 Выбор средств программной реализации 40](#_Toc73182105)

[3.1.1 Выбор языка программирования 40](#_Toc73182106)

[3.1.2 Выбор среды программирования и отладки 41](#_Toc73182107)

[3.1.3 Используемые библиотеки 41](#_Toc73182108)

[3.2 Система контроля версий 42](#_Toc73182109)

[3.3 Требования к вычислительной системе 43](#_Toc73182110)

[3.4 Формат входных данных 43](#_Toc73182111)

[3.5 Формат хранения данных 43](#_Toc73182112)

[3.6 Формат выходных данных 44](#_Toc73182113)

[3.7 Схема разработанного программного обеспечения 44](#_Toc73182114)

[3.8 Описание основных моментов программной реализации 45](#_Toc73182115)

[3.8.1 Частеречные N-граммы 46](#_Toc73182116)

[3.9 Установка программного обеспечения 47](#_Toc73182117)

[3.10 Интерфейс пользователя 47](#_Toc73182118)

[3.11 Руководство пользователя 49](#_Toc73182119)

[3.12 Модульное тестирование 49](#_Toc73182120)

[3.13 Выводы 51](#_Toc73182121)

[4 Экспериментальный раздел 52](#_Toc73182122)

[ЗАКЛЮЧЕНИЕ 53](#_Toc73182123)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 54](#_Toc73182124)

# **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:

* изучение предметной области;
* анализ существующих решений определения авторства текста;
* анализ алгоритмов классификации;
* разработка метода определения признаков авторского стиля;
* разработка метода классификации текстов, использующего предложенный метод классификации текстов;
* проектирование структуры и реализация программного обеспечения, реализующего разработанный метод;
* подготовка входных данных и обучение классификатора;
* апробация предложенного метода.

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.

# **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.

## 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, характеризующими параметрами текста называют статистические характеристики: количество предложений разного типа и их средняя длина, число знаков препинания, определенных частей речи, количество служебных слов в тексте и многое другое.

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 разделить на две большие группы – экспертные и формальные. Экспертные методы подразумевают, что анализ и исследование текста провидится профессиональным лингвистом. Формальные методы предполагают машинную обработку текста и включат в себя алгоритмы классификации, кластеризации, нейронные сети, приемы из теории вероятности и математической статистики.

## Этапы обработки текста

В данном разделе будет выполнен обзор этапов обработки текстов на естественном языке.

### Токенизация

Токенизация — это базовый этап в автоматической обработке текстов, суть которого 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].

### 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.

## Этапы анализа текста

Процесс полного анализа текста может быть поделен на несколько этапов [3]. Выбор того, как 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 тексте определить его начальную форму. Помимо самой леммы на данном этапе так же 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.

## Методы и решения определения авторского инварианта

В данном разделе проанализированы существующие методы и решения для определения признаков авторского стиля.

### Метод полного синтаксического анализа

Под синтаксическим анализом в области обработки текстов на естественном языке понимается построение такой структуры, которая позволяет приблизиться к некоторому формализованному представлению смысла текста и установить связи между словами в предложении.

Большинство моделей синтаксической структуры предложения опираются либо на грамматику составляющих, предложенной в работах Ноама Хомского [5], либо на грамматику зависимостей, для которой основополагающими считаются работы Люсьена Теньера [6] и Игоря Мельчука [7].

Грамматика составляющих предполагает, что предложение на естественном языке может быть представлено в виде иерархии составляющих. Данная иерархия предполагает выделение синтаксических групп, которые не должны пересекаться между собой, но которые в свою очередь состоят из более мелких групп, вплоть до атомарных единиц – слов предложения. Такую иерархическую структуру называют деревом составляющих. Дерево составляющих удовлетворяет свойству проективности, согласно которому при представлении дерева графически связи между структурными единицами не должны пересекаться.

Грамматика зависимостей предполагает, что предложения текста представляют собой деревья зависимостей, в которых слова связаны ориентированными дугами, обозначающими синтаксическое подчинение. Считается, что подход с деревом зависимостей более точно отражает специфику языков с произвольным порядком слов. Грамматика зависимостей допускает наличие непроективных связей, нарушающих свойство проективности дерева

Синтаксический анализ разделяют на глубокий (полный) анализ и поверхностный [8, 9]. Задачей глубокого синтаксического анализа является построение полного синтаксического дерева предложения с максимальной связанностью с учетом дальних связей, а также определение грамматических функций слов предложения (подлежащее, сказуемое, обстоятельства места, времени и т.д.). Понятие поверхностного синтаксического анализа объединяет в себе различные подходы, направленные на построение неполной (частично связанной) синтаксической структуры текста разной сложности. Поверхностный синтаксический анализ охватывает такие задачи как разделение предложения на рекурсивно невложенные синтаксические группы, сегментацию (выделение в предложении различных оборотов и простых предложений в составе сложного), построение поверхностного синтаксического 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.

### Метод энтропийной классификации

Метод классификации текстов для задачи определения авторства, предложенный в работе [10], основан на применении алгоритмов сжатия данных.

Суть рассматриваемого метода заключается в том, чтобы добавить текст, автор которого неизвестен, к корпусу текстов, характеризующему конкретного автора, проанализировать, насколько хорошо сжимается полученная после добавления выборка текстов, и сравнить полученные результаты для различных авторов. В рамках энтропийного подхода правильным исходным классом текста является тот класс, на котором получены наилучшие результаты сжатия.

Рассмотрим на примере задачу классификации текста относительно текстов , характеризующих классов. Выбор источника текста должен осуществляться согласно оценке, рассчитываемой по формуле:

где ­– характеристика энтропии текста по отношению к тексту

Существенным преимуществом метода энтропийной классификации является отсутствие необходимости в предварительной обработке текста.

### Рекуррентные нейронные сети

Для решения задачи классификации текстов используют в том числе и нейронные сеть. Рекуррентные нейронные сети – подкласс нейронных сетей с обратными связями, которые используют предыдущие состояния сети для вычисления текущего. Пример такой сети представлен на рисунке 1. Обычно данный класс нейронных сетей используется в задачах обработки последовательностей нефиксированной длины. В частности, рекуррентные нейронные сети показывают результаты лучше других методов в задачах классификации текста [11].

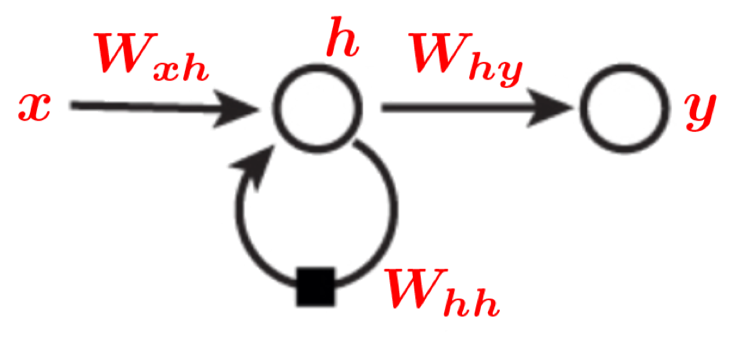


Рисунок 1 – Структура рекуррентной нейронной сети с одним входным нейронном, одним скрытым нейроном и одним выходным нейроном

Рекуррентные нейронные сети позволяют использовать текст в его исходном виде, то есть учитывается порядок слов. Сеть последовательно получается на вход слова. При этом в рекуррентной сети есть скрытый слой , который обновляется после появления каждого нового слова или токена (например, буквы). Слой обновляется, учитывая и своё предыдущее состояние, и полученное слово. После того как скрытое состояние обновлено, на его основе генерируется выход для данного токена. Процесс повторяется до тех пор, пока через сеть не пройдут все слова в тексте.

Данная сеть описывается двумя формулами. Первая — это обновление скрытого состояния. Оно обновляется путём некого усреднения предыдущего значения и нового входа:

где – состояние слоя в предыдущий момент времени;

– слово, которое было подано на вход;

– некая нелинейная функция активации.

Вторая формула, которой описывается сеть, — это значение на выходе:

где – обновлённое состояние скрытого слоя;

– функция активации.

Рекуррентная нейронная сеть представляет собой сеть прямого распространения, длина которой зависит от числа токенов, которые подаются на вход. При этом параметры, матрицы являются общими и не меняются от шага к шагу. Изменяются только скрытые состояния , входы , выходы .

Для обучения рекуррентных нейронных сетей используется алгоритм обратного распространения ошибки по времени (backpropagation through time) [12], который является вариантом алгоритма обратного распространения ошибки (backpropagation), используемого для нейронных сетей прямого распространения сигнала.

### Метод анализа длин слов

Для поставленной задачи определения авторства текста существует также решение анализа стилистики текста, основанное на длине слов. Данное решение легло в основу программы «Худломер» [13].

«Худломер» — это метод автоматической классификации функционального стиля текста на основе спектров длин слов. Программа позволяет определять следующие стили: разговорный стиль, стиль художественной литературы, газетно-информационный стиль, научно-деловой стиль.

### Применения методов из теории вероятности и математической статистики

Предлагаемый в работе [14] метод основывается на учете статистики употребления пар элементов любой природы, идущих друг за другом в тексте (букв, морфем, словоформ и т. п.), т. е. на формальной математической модели последовательности букв (и любых других элементов) текста как реализации цепи Маркова. По произведениям, для которых достоверно определен автор, вычисляется матрица переходных частот употребления пар элементов (букв, грамматических классов слов и т. п.). Для каждого автора строится матрица переходных частот и оценивается вероятность того, что именно он написал исследуемый текст. Автором неизвестного текста считается тот, для кого вычисленная оценка вероятности больше.

Примеры того, в виде каких пар можно представить текст:

а) пары букв в словах (в той форме, в которой они употреблены в тексте) и пробелах между ними;

б) пары букв в словах, приведенных к начальной форме;

в) пары наиболее обобщенных грамматических классов слов в их последовательностях в предложениях текста. К таким классам слов относят части речи и некоторые условные категории вроде «конец предложения», «сокращение» и др.;

г) пары менее обобщенных грамматических классов слов. К ним относятся такие семантико-грамматические разряды, как одушевленные и неодушевленные существительные, прилагательные качественные, относительные, притяжательные и т. п.

### Метод выделения N-грамм

N-грамма — это последовательность из N идущих подряд слов в тексте. Использование N-грамм характерно тем, что позволяет учитывать порядок слов.

Для предложения «Наборы подряд идущих токенов» существуют следующие N-граммы:

* униграммы (N = 1): наборы, подряд, идущих, токенов;
* биграммы (N = 2): наборы подряд, подряд идущих, идущих токенов;
* триграммы (N = 3): наборы подряд идущих, подряд идущих токенов.

N-грамма имеет в основе своего использования математическую модель и определяется 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» [15].

Чем больше N, до которого будут найдены N-граммы, тем больше будет получено признаков. При этом увеличение параметра N может привести к переобучению. Например, если N — это длина текста, то у каждого документа будет уникальный признак, и алгоритм сможет переучиться под обучающую выборку.

N-граммы можно использовать не только на словах. В качестве токенов можно рассматривать отдельные символы в предложении. После нахождения таких токенов для них можно также вычислять N-граммы (буквенные N-граммы). Данный подход позволяет учитывать известные слова в незнакомых формах. Часто буквенные N-граммы используют вместе с N-граммами по словам.

Skip-граммы — это чуть более расширенный подход к использованию N-грамм. Более точно они называются k-skip-n-граммы, это наборы из N токенов, причём расстояние между соседними должно составлять не более K токенов [16]. Данный метод позволяет учитывать больше различных N-грамм в предложении, и, в связи с этим, как правило, используется в языковом моделировании в сочетании с другими подходами.

## Извлечение признаков из текста

В данном разделе рассматриваются различные подходы к извлечению признаков из текста и определению значимости полученных признаков.

### Счётчики слов

Данный подход подразумевает, что текст рассматривается как неупорядоченный набор слов. Более формально описать его можно следующим образом. Пусть всего в выборке различных слов: . В этом случае каждый текст кодируется с помощью признаков, причём признак — это доля вхождений слова среди всех вхождений слов в документе. Таким образом, текст кодируется вектором признаков, а сумма значений всех признаков составляет единицу.

Используя такой подход, имеет смысл обращать внимание на два нюанса. Первый — это стоп-слова. Это популярные слова, встречающиеся в каждом тексте (например, предлоги или союзы) и не несущие в себе никакой информации, которая могла бы выделить определенный текст среди всех остальных. Их стоит удалять ещё при предобработке, например, на этапе токенизации. Кроме того, имеет смысл удалять редкие слова. Если какое-то слово входит только в 1 или 2 текста, то, скорее всего, не получится его значимо учесть в модели и оценить, какой вклад оно вносит в целевую переменную.

### TF-IDF

TF-IDF — статистическая мера, используемая для оценки важности слова в контексте текста, являющегося частью корпуса текстов. Как и в подходе с использованием счетчика слов, считается, что если слово часто встречается в тексте, и оно не является стоп-словом, то, скорее всего, оно важно. Второе утверждение, которое лежит в основе TF-IDF: если слово встречается в других документах реже, чем в данном, то и в этом случае, скорее всего, оно важно для текста. По этому слову можно отличить этот текст от остальных. Если учесть описанные выше соображения, в результате получится подход TF-IDF (TF — term frequency, IDF — inverse document frequency) [17].

TF — частотность термина, которая измеряет, насколько часто термин встречается в документе. TF измеряется как отношение числа вхождений некоторого слова к общему числу слов документа. IDF — инверсия частоты, с которой некоторое слово встречается в документах коллекции. Учёт IDF уменьшает вес широкоупотребительных слов. Таким образом, мера TF-IDF является произведением двух сомножителей TF и IDF.

Значение признака для слова и текста вычисляется по следующей формуле:

где – доля вхождений слова в документ ;

– общее количество документов;

– число документов в выборке, в которых слово встречается хотя бы раз.

Если отношение велико, слово редко встречается в других документах, значение признака будет увеличиваться. Если слово встречается в каждом тексте, то значение признака будет нулевым .

## Классификация текстов

В данном разделе будут рассмотрены алгоритмы классификации текстов.

### Метод «наивной» байесовской классификации

Наивный байесовский классификатор [18] ﻿основан на применении 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 том, что входящие в текст слова не зависят друг от друга.

Для построение наивного байесовского классификатора необходимо выбрать закон, по которому распределены данные. Обучение данного классификатора заключается в вычислении параметров распределения по примерам из тестового набора данных.

Если предположить, что данные распределены по закону Бернулли, то класс , к которому относится текст , вычисляется по следующей формуле:

где – характеристики, по которым оцениваются тексты;

– количество текстов;

– величины, показывающие наличие -ой характеристики в тексте;

– метка класса;

– параметры, вычисленные при обучении классификатора.

### Метод k ближайших соседей

В основе метода k ближайших соседей [19] лежит гипотеза компактности векторного пространства, которая гласит, что: документы одного класса образуют в пространстве терминов компактную область, причём области разных классов не пересекаются [4]. Согласно рассматриваемому методу, для нахождения категории, соответствующей документу , классификатор должен сравнить со всеми документами из обучающей выборки , то есть для каждого вычисляется расстояние . Далее из обучающей выборки выбираются документов, ближайших к . Считается, что документ принадлежит тому классу, который является наиболее распространенным среди соседей данного документа, то есть для каждого класса вычисляется функция ранжирования:

где – документов из ближайшие к ;

– известные величины, уже расклассифицированные по категориям документы из обучающей выборки.

### Метод деревьев решений

Подход, использующий деревья решений, относится к символьным (то есть нечисловым) алгоритмам.

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.

Процессе классификации заключается в последовательных переходах от одного узла к другому в соответствии со значениями признаков объекта. Классификация считается завершенной, когда достигнут один из листьев дерева. Значение этого листа определяет тот класс, которому принадлежит рассматриваемый документ. На практике как правило используют бинарные деревья решений, в которых принятие решения перехода по ребрам осуществляется в зависимости от результата проверки наличия признака в документе [19].

Лесом решений [20] называют комитет (ансамбль) из нескольких деревьев решений. Использование нескольких деревьев приводит к улучшению качества прогнозирования и к лучшему пониманию закономерностей исследуемого явления. Разные деревья могут быть получены различными методами (или одним методом, но с различными параметрами работы), по разным выборкам наблюдений за одним и тем же явлением, путем привлечения различных характеристик. Для задачи классификации решение принимается по большинству результатов, выданных деревьями решений, а в задаче регрессии — по их среднему значению.

### Метод опорных векторов

Метод опорных векторов [21] основан на разбиении векторного пространства документов разделяющей поверхностью на подпространства, где каждому подпространству соответствует только один класс. Алгоритм работает в предположении, что чем больше зазор классификации (расстояние между найденной поверхностью и ближайшей точкой данных), тем меньше будет средняя ошибка классификатора.

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. Операторы ядра подразделяются на линейные и нелинейные. 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.

На рисунке 2 показано разбиение данных при бинарной классификации при помощи линейнеого 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 [22]. 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.

## 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.

# **Конструкторский раздел**

В данном разделе рассматривается проектирование структуры программного обеспечения и требования к нему. Так же рассматриваюется последовательность действия для метода определения признаков авторского и метода классификации, описываются структуры данных, используемые на каждом этапе работы алгоритма.

## Структура разрабатываемого программного обеспечения

Разрабатываемое программное обеспечение состоит из следующих частей:

* модуль считывания входных данных;
* модуль предобработки текстов;
* модуль обучения классификатора;
* модуль классификации;
* модуль сбора статистических данных;
* модуль формирования выходных данных.

## Функциональная модель метода определения признаков авторского стиля

На рисунке 3 представлена функциональная схема предлагаемого метода определения признаков авторского стиля в нотации IDEF0:

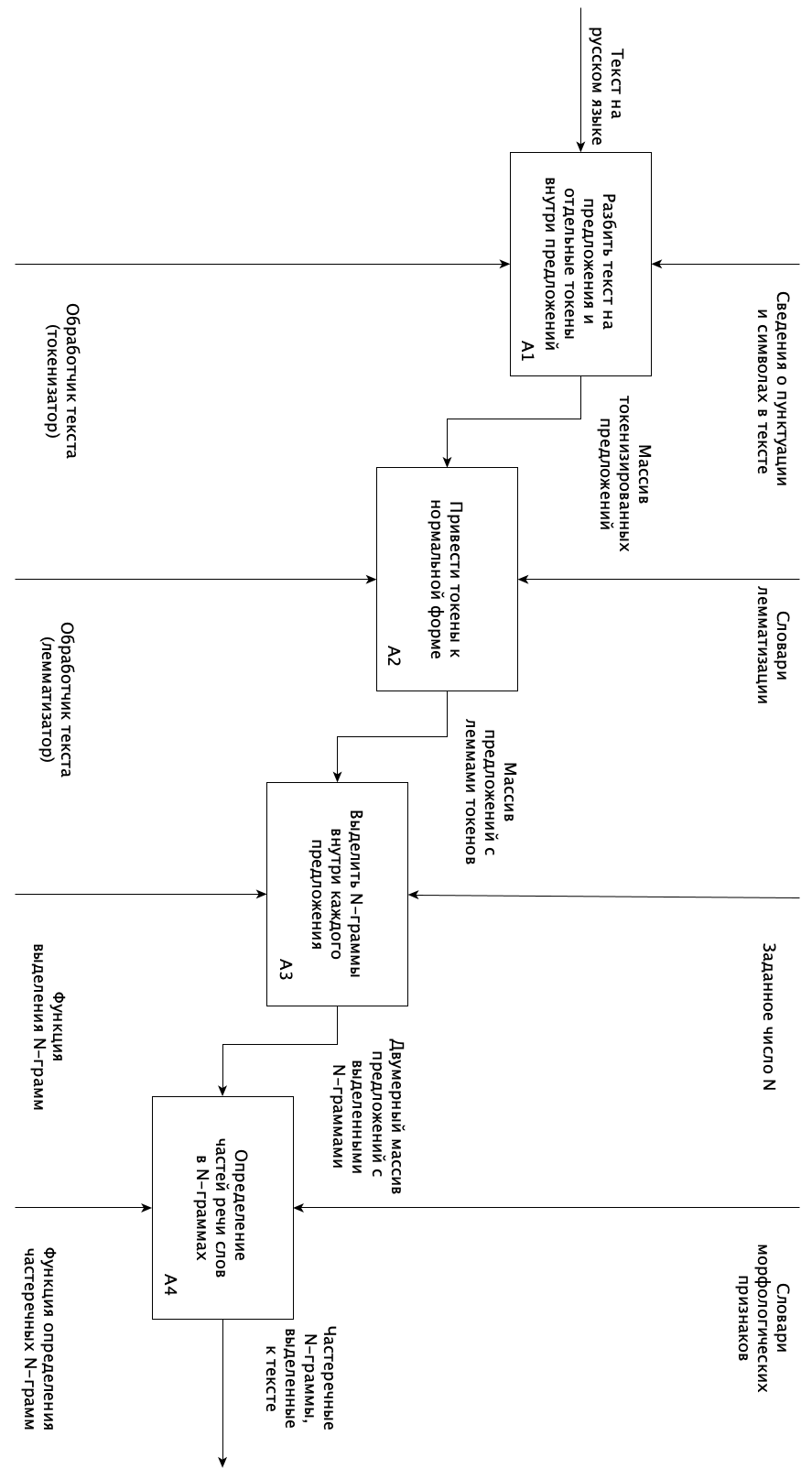


Рисунок 3 – Функциональная схема IDEF0 работы метода

## Функциональная модель 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.

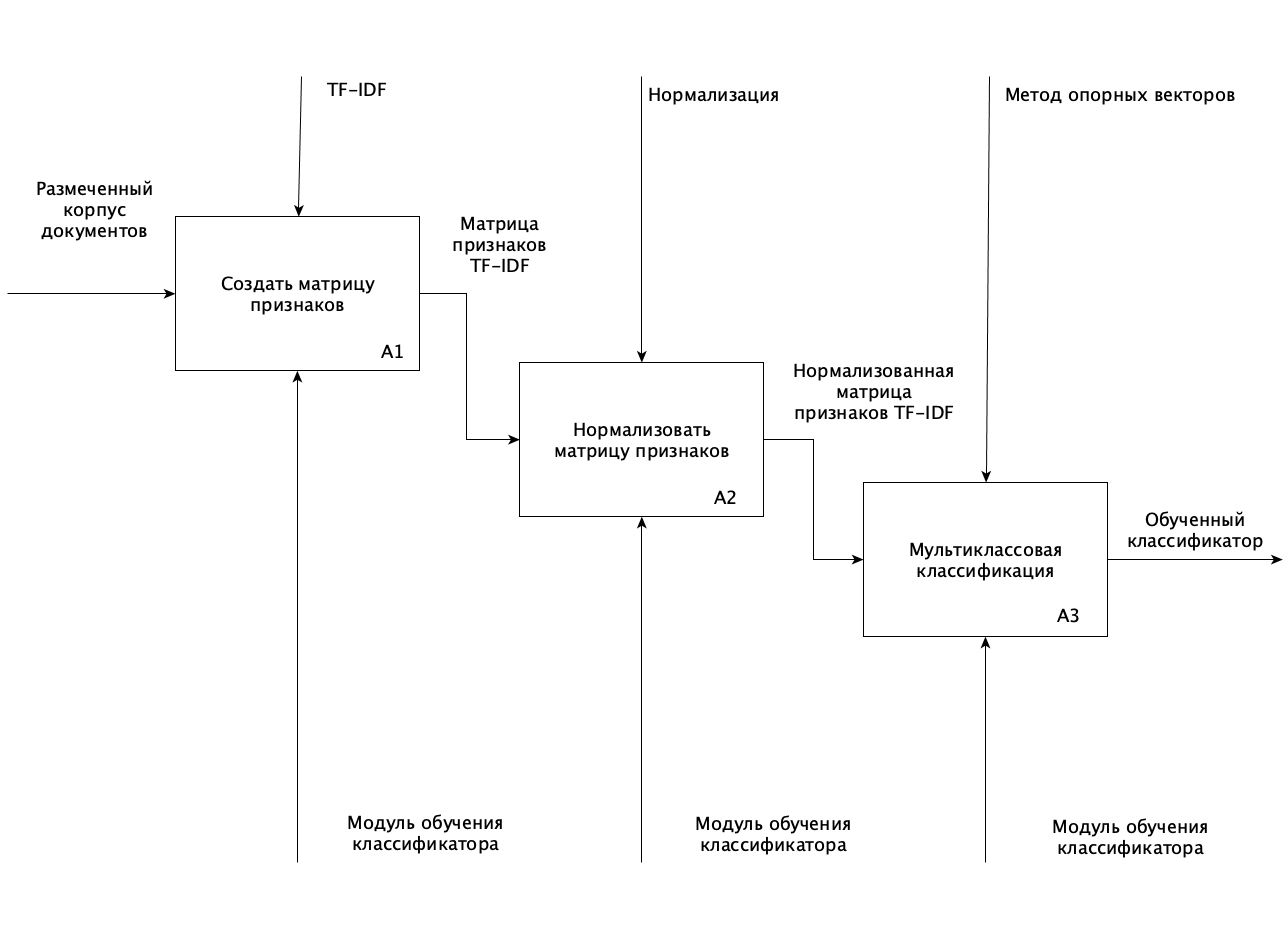


Рисунок 4 – Функциональная схема IDEF0 обучения метода классификации

## Функциональная модель 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.

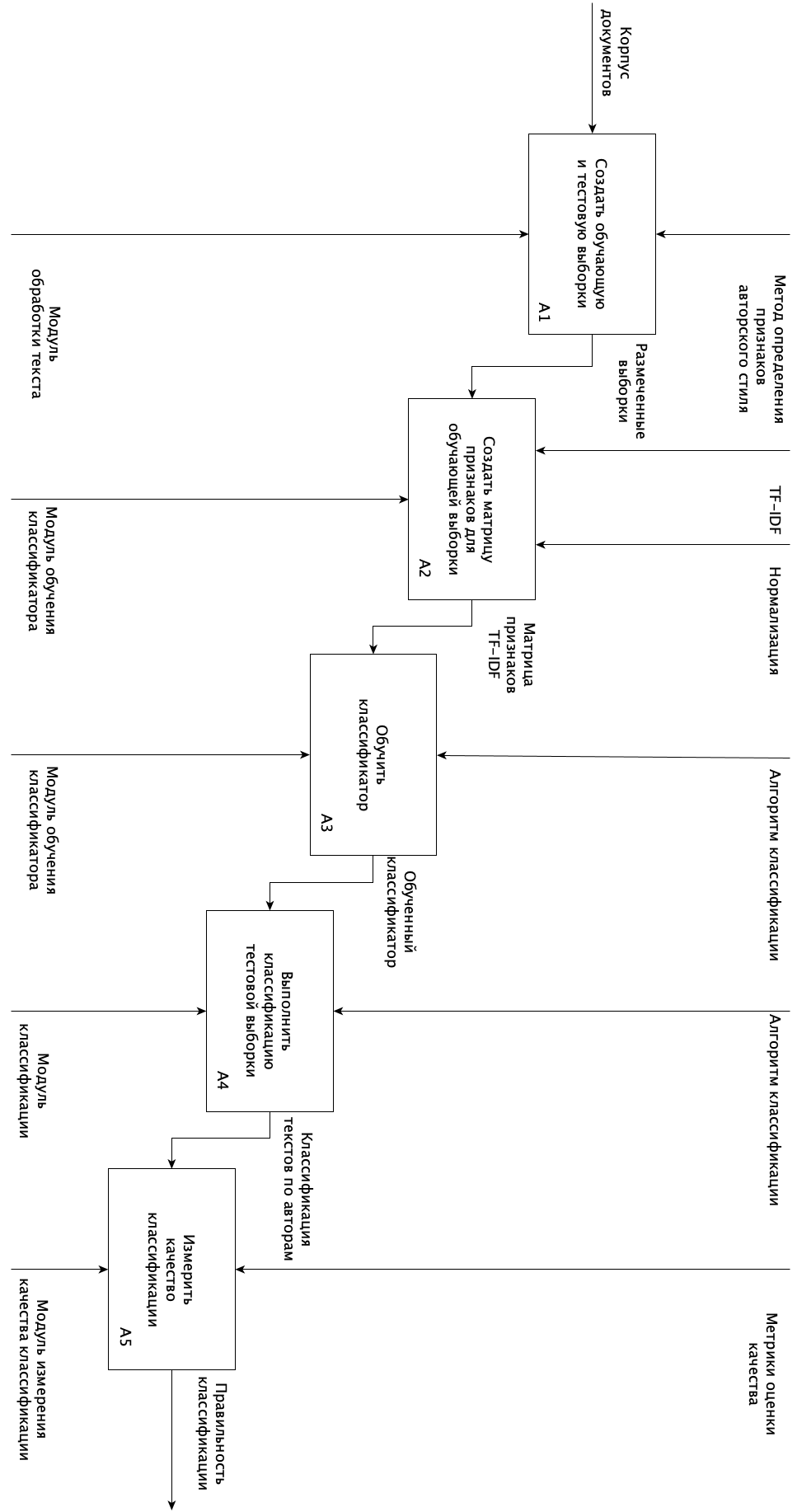


Рисунок 5 – Функциональная схема IDEF0 метода классификации

## Описание этапов работы разрабатываемого алгоритма

Основные этапы работы алгоритма могут быть представлены в виде схемы (рисунок 6).

### 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).

### 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.

### 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.

### 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:

где – число вхождений слова в документ ;

– общее число слов в документе .

IDF — инверсия частоты – рассчитывается следующим образом:

где – корпус документов;

– количество документов в корпусе;

– количество документов в корпусе , в которых присутствует слово .

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:

### 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.

Каждое значение в матрице признаков нормализуется по следующей формуле:

где – значение признака;

– нормализованное значение признака.

### 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, используется подход «один против всех». При данном подходе обучающая выборка делится на рассматриваемый класс и все остальные классы и создается бинарный классификатор, который различает метку 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 и число такие, что для некоторого выполняется:

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

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

Полоса, разделяющая классы, задается условием . 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 , . Нужно найти такие и , чтобы выполнялись все линейные ограничения, и при этом норма вектора была как можно меньше (следовательно, шире разделяющая полоса), то есть необходимо минимизировать:

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 , характеризующих величину ошибки на объектах . Тогда ограничения принимают следующий вид:

Если в документе отсутствуют ошибки, то . Если , то в документе допускается наличие ошибки. В случае объект попадает внутрь разделительной полосы, но относится алгоритмом к своему классу.

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

где – 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 приминимизации 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:

где – уравнения ограничений;

– множители Лагранжа (дополнительные переменные).

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

В рассматриваемой задаче необходимо при ограничениях и минимизировать квадратичную функцию:

Необходимо при условиях найти минимум по , , и максимум по для данной функции:

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

Если , то документ обучающей коллекции называется опорным вектором. Теперь уравнение разделяющей гиперплоскости выглядит следующим образом:

где – документ, поступающий в классификатор.

После взятия производную целевой функции по :

При условиях и после подставления , выраженное через , в Лагранжиан, задачу можно сформулировать следующим образом: найти такие значения множителей, при которых достигается максимум:

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

### Классификация

На данном этапе обученный классификатор получает на вход данные из тестовой выборки. Для каждого документа формируется предсказание наиболее вероятного автора, после чего производится оценка качества классификация согласно различным метрикам.

### Метрики оценки качества классификации

Для оценки работы классификатора на обучающей и тестовой выборках используются различные метрики.

Точность – показатель качества работы классификатора, который определяется отношением правильных прогнозов ко всем (число правильных прогнозов, делённое на число всех прогнозов). Точность рассчитывается по следующей формуле:

где – истинно-положительное решение;

– ложноположительное решение;

Полнота – доля найденных классификатором документов, принадлежащих классу относительно всех документов этого класса в тестовой выборке. Находится по следующей формуле:

где – ложноотрицательное решение.

F-мера – среднее гармоническое точности и полноты, стремится к нулю, если точность или полнота стремится к нулю. Находится по следующей формуле:

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:

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

## Выводы

В данном разделе была рассмотрена структура программного обеспечения. Была приведена последовательность действия для метода определения признаков авторского и метода классификации, описаны структуры данных, используемые на каждом этапе работы алгоритма.

# **Технологический раздел**

В данном разделе описываются средства, используемые для разработки программного обеспечения, требования к вычислительной системе для функционирования ПО, формат выходных и выходных данных, а также описывается интерфейс пользователя.

## Выбор средств программной реализации

В данном разделе приводится обоснования выбранных средств программной реализации: языка и среды программирования, средств отладки и используемых библиотек.

### 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.

### Выбор среды программирования и отладки

В качестве среды разработки и отладки была выбрана кроссплатформенная IDE PyCharm, предоставляющая средства для анализа кода и графический отладчик. Использовалась IDE PyCharm Community Edition, которая является бесплатной для проектов с открытым исходным кодом.

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.

### Используемые библиотеки

В процессе реализации ПО были использованы следующие библиотеки:

* NumPy [23] – библиотека реализаций вычислительных алгоритмов, оптимизированных для работы с многомерными массивами;
* pandas [24] – библиотека для анализа данных, используемая для представления данных в формате таблиц;
* nltk [25] – библиотека для работы с естественным языком, используемая для токенизации текстов по предложениям и выделения стоп-слов;
* pymorphy2 [26] – библиотека для морфологического анализа для русского языка, используемая для лемматизации слов и определения их частей речи;
* scikit-learn [27] – 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 [28] – библиотека, предоставлящая набор расширений (биндингов) графического фреймворка Qt для языка программирования Python

## 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, 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 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.

## 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 так же требуется наличие свободной оперативной памяти на вычислительной системе, следовательно, объем объём доступной оперативной памяти ЭВМ должен составлять не менее 5 ГБ.

## 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).

## 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.

## Формат выходных данных

Типы выходных данных разработанного программного продукта:

* метрики качества классификатора, полученные на обучающей и тестовой выборках;
* предсказание автора для нового классифицируемого текста.

Выходные данные выводятся на графический интерфейс ПО.

## Схема разработанного программного обеспечения

На рисунке 6 представлена UML-диаграмма разработанного ПО, реализующего предлагаемый метод:

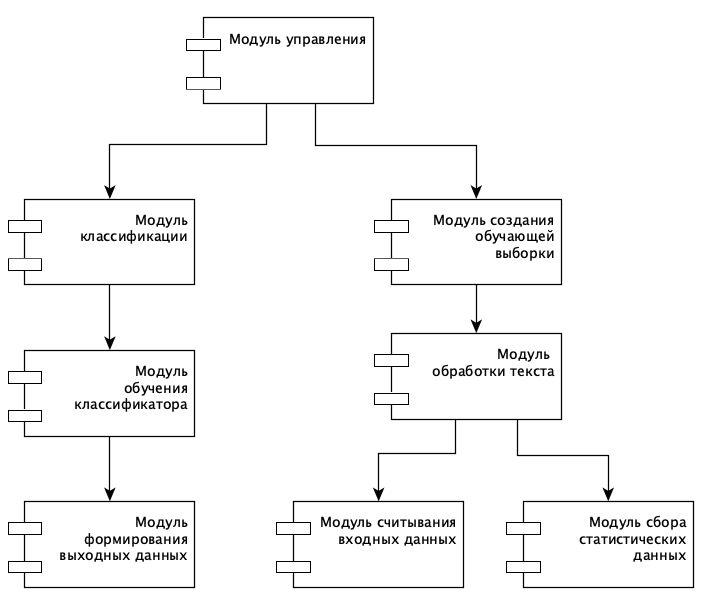


Рисунок 6 – Схема разработанного ПО

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).

## Описание основных моментов программной реализации

В данном разделе описываются технологии, используемые для реализации метода.

### Частеречные N-граммы

Части речи слов определяются при помощи морфологического анализа. Для проведения морфологического была использована библиотека pymorphy2 языка Python. В pymorphy2 для русского языка используются словари OpenCorpora и граммемы, принятые в OpenCorpora (с небольшими изменениями) [29]. 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 |

## Установка программного обеспечения

Для запуска разработанного программного продукта требуется установить на ПК интерпретатор для Python 3.9. 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 следующую команду в терминале:

**$ pip3 install -r requirements.txt**

## Интерфейс пользователя

Пользовательский интерфейс (рисунок 6) разрабатывался в программе QtDesigner. Интеграция с кодом на Python осуществлена посредством использования библиотеки PyQt5, предоставляющей различные классы для работы с объектами интерфейса.



Рисунок 7 – 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, равно 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 2 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. При нажатии на кнопку «выбрать файл» открываются файловая система компьютера, внутри которой необходимо выбрать интересующий текстовый файл. ПО поддерживает работу исключительно с файлами в формате TXT, поэтому в случае, если пользователь выберет файл с другим расширением, ему будет возвращена ошибка с сообщением о необходимости выбрать TXT файл. Далее, запускается алгоритм классификации. На экран выводится название выбранного текста для лучшей наглядности. После того, как классификатор закончит работать, на экран так же будет выведена фамилия наиболее вероятного автора.

## 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:

**$ 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:

**$ 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).

## Модульное тестирование

Из размеченных данных была выбрана тестовая выборка, состоящая из 24 объектов, на которой проводилась классификация. Представлены тексты 4 авторов: Горький, Достоевский, Толстой, 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 |
| Достоевский | Достоевский |
| Горький | Горький |
| Чехов | Чехов |
| Толстой | Толстой |
| Горький | Горький |
| Чехов | Чехов |
| Чехов | Чехов |
| Горький | Горький |
| Горький | Горький |
| Толстой | Толстой |
| Чехов | Чехов |
| Чехов | Толстой |
| Горький | Горький |
| Горький | Горький |
| Толстой | Толстой |
| Достоевский | Чехов |
| Чехов | Чехов |
| Толстой | Толстой |
| Достоевский | Толстой |
| Достоевский | Достоевский |
| Горький | Толстой |
| Толстой | Толстой |
| Толстой | Толстой |
| Чехов | Чехов |

Полученная точность составила 83%. Классификатор дважды совершил ошибку на классе Достоевского и по одному разу на классах Горького и Чехова.

## Выводы

В данном разделе были рассмотрены средства, используемые для разработки программного обеспечения, требования к вычислительной системе для функционирования ПО, формат выходных и выходных данных, а также приведено описание интерфейса пользователя.

# **Экспериментальный раздел**

# **ЗАКЛЮЧЕНИЕ**

# 

# **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. Federici S., Montemagni S., Pirrelli V. Shallow parsing and text chunking: a view on underspecification in syn- tax // Cognitive science research paper-university of Sus- sex CSRP. — 1996. — P. 35–44.
10. ﻿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: 11.12.2020).
11. Lai S., Xu L., Liu K., Zhao J. Recurrent Convolutional Neural Networks for Text Classification. // AAAI, 2015. P. 2267 – 2273.
12. 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.
13. ﻿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).
14. Кукушкина О. В., Поликарпов А. А., Хмелев Д. В. Определение авторства текста с использованием буквенной и грамматической информации // Проблемы передачи информации. М.: Наука, 2001. Т. 37, № 2. С. 96–108.
15. ﻿Гудков В. Ю., Гудкова Е. Ф. N-граммы в лингвистике // Вестник Челябинского государственного университета. 2011. № 24 (239). Филология. Искусствоведение. Вып. 57. С. 69–71.
16. ﻿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.
17. 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
18. Murphy K.P. Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series). The MIT Press, 2012. ISBN: 0262018020.
19. Батура Т.В. Методы автоматической классификации текстов // Программные продукты и системы. 2017. №1.
20. 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.
21. 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.
22. 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)