

Реферат

РПЗ страниц, рисунков 24, источников 20.

В качестве объекта исследования данной работы выбраны электронные письма. Целью работы является разработка метода автоматизированного удаления подписей из текстов русскоязычных электронных писем.

Задачи работы:

- анализ предметной области;
- анализ существующих подходов и методов решения задач, связанных с обработкой текста на естественном языке;
- разработка собственного метода на основании проанализированных;
- разработка программного комплекса, реализующего выбранный метод;
- анализ эффективности работы программного продукта.

В **аналитическом** разделе был проведен анализ предметной области, рассмотрены особенности оформления подписей в электронных письмах, а также существующие методы и подходы, используемые для решения задач обработки текстов на естественном языке.

В **конструкторском** разделе описан разработанный метод, основанный на обучении классификатора на векторах признаков, характеризующих строки электронного письма. Также были приведены алгоритмы, используемые при разработке программного обеспечения.

В **технологическом** разделе был проведен обоснованный выбор средств программной реализации решения поставленной задачи, была приведена информация, необходимую для сборки и запуска программного обеспечения. А также описаны формат входных и выходных данных, структура ПО и его компоненты, а также интерфейса пользователя.

В **экспериментальном** разделе был проведен ряд исследований алгоритмов классификации с целью определения наиболее подходящего для решения поставленной задачи.

Способы применения программного продукта: программный продукт может применяться для предварительной обработки писем с целью преобразования текста в речь, автоматического формирования списка контактов и адресов, анонимизации отправителей, сортировки писем в соответствии с информацией, указанной в подписи.

Содержание

Введение	5
1 Аналитический раздел	7
1.1 Постановка задачи	7
1.2 Цель и решаемые задачи	7
1.3 Анализ предметной области	8
1.3.1 Электронное письмо	8
1.3.2 Подпись в электронных письмах	11
1.3.3 Выводы	12
1.4 Компьютерная лингвистика: задачи и основные направ- ления	13
1.5 Анализ существующих подходов и методов классифика- ции текстов на естественном языке	17
1.5.1 Предварительная обработка данных	18
1.5.2 Алгоритмы классификации	21
1.5.3 Параметры для оценки качества работы класси- фикатора	28
1.6 Выводы	32
2 Конструкторский раздел	34
2.1 Выполняемые задачи	34
2.2 Получение текста электронного письма	35
2.3 Преобразование строк письма в форму векторов признаков	37
2.4 Классификация векторов признаков	39
2.4.1 Описание обучающей выборки данных	39
2.4.2 Обучение классификатора	40
2.5 Структура программы	41
2.6 Выводы	42
3 Технологический раздел	43
3.1 Выбор средств и инструментов разработки	43
3.1.1 Выбор языка программирования	43
3.1.2 Выбор среды разработки	44
3.1.3 Выбор базы данных	44
3.1.4 Система контроля версий	44

3.2	Описание используемых библиотек	45
3.2.1	Библиотека для работы с базами данных	45
3.2.2	Библиотека для графического интерфейса	45
3.2.3	Библиотека для машинного обучения	46
3.3	Описание классов	47
3.4	Требования к системе пользователя	48
3.5	Тестирование ПО	49
3.6	Описание пользовательского приложения	49
3.7	Выводы	52
4	Экспериментальный раздел	53
4.1	Алгоритм опорных векторов	53
4.2	Алгоритм k-ближайших соседей	55
4.3	Сравнение алгоритмов классификации	56
4.4	Выводы	59
	Заключение	60
	Список использованных источников	61
A	Регулярные выражения, используемые для выделения признаков	64

Введение

С развитием интернета и информационных технологий электронная почта стала неотъемлемой частью жизни человека. Она используется для самых разных целей: личная и деловая переписка, обмен документами, информационные и рекламные рассылки. И если со стремительным ростом популярности социальных сетей электронная почта уже не играет важной роли в личной переписке, то деловое общение все еще плотно связано с использованием электронных писем. На почтовый ящик любого пользователя электронной почты, не говоря уже о крупной компании или корпорации, ежедневно могут поступать десятки или даже сотни писем. И соответственно, возникает необходимость обрабатывать эти письма, что является достаточно рутинной задачей, которую неплохо было бы автоматизировать.

Одной из важных частей электронного письма является подпись отправителя. Добавление подписи в текст письма является правилом хорошего тона, при этом подпись рекомендуется оформлять в соответствии с определенными правилами, которые зависят от характера переписки. Однако, будь то деловая, личная переписка, или обычная рекламная рассылка, подпись, помимо выполнения функции поддержания культуры общения, несет в себе важную и полезную для получателя информацию, такую как номера телефонов, адреса, должности и другую информацию об отправителе. Естественно в таком случае встает вопрос об извлечении данной информации, например, с целью поддержания базы данных контактов компании или сортировки писем. Также может возникнуть необходимость выделения какой-либо информации из той части письма, которая не относится к подписи, в таком случае ставится задача об автоматическом удалении текста подписи.

Таким образом, существует потребность в автоматическом отделении подписи от основного текста письма. В связи с чем в рамках дипломной работы предлагается разработать метод для автоматического удаления и выделения подписей из текстов электронных писем. Программная реализация данного метода может найти свое применение в системах, использующих удаление подписи, например, в системах преобразования

текста письма в речь. А также разработанное программное обеспечение может являться одной из частей систем, которые решают задачу по выделению какой-либо информации из текста письма, например, систем по автоматизации базы данных контактов.

1 Аналитический раздел

1.1 Постановка задачи

Необходимо разработать программный комплекс, который позволяет автоматически определить и удалить подпись в электронном письме, содержащем текст на русском языке. Программа должна принимать на входе полный текст электронного письма и по возможности имя отправителя. В результате работы программы в тексте письма должна быть найдены строки, относящиеся к подписи.

1.2 Цель и решаемые задачи

Целью работы является разработка метода автоматизированного удаления подписей из текстов русскоязычных электронных писем, а также его программная реализация.

Для достижения поставленной цели необходимо решить следующие задачи:

- анализ предметной области;
- анализ существующих подходов и методов решения задач, связанных с обработкой текста на естественном языке;
- разработка собственного метода на основании проанализированных;
- разработка программного комплекса, реализующего выбранный метод;
- анализ эффективности работы программного продукта.

1.3 Анализ предметной области

1.3.1 Электронное письмо

Так как объектом исследования данной работы является электронное письмо, и разрабатываемое программное обеспечение должно обрабатывать текст электронного письма, целесообразным будет рассмотреть структуру электронного письма, формат его представления, а также требования, предъявляемые к его оформлению. Данный анализ поможет конкретизировать постановку требований к разрабатываемому методу и программному обеспечению, а также поможет установить ограничения, в рамках которых будет реализован метод.

Структура электронного письма

Общепринятым и повсеместно используемым протоколом обмена электронной почтой является SMTP (англ. Simple mail transfer protocol — простой протокол передачи почты). Исходя из этого, следует рассмотреть структуру электронного письма, передаваемого по протоколу SMTP.

Согласно стандарту RFC 5322 [1], который описывает формат сообщений, передаваемых путем использования электронной почты, электронное письмо, передаваемое по протоколу SMTP, состоит из следующих частей:

- данные SMTP-конверта;
- заголовки письма;
- тело письма.

SMTP конверт отправляется как серии протокольных объектов. Включает в себя адрес отправителя (куда должна быть отправлена ошибка в случае неблагоприятного события); одного или более адресов получателей; дополнительный материал, расширяющий протокол [2]. В большинстве случаев данная информация недоступна конечному получателю. Для возможности контролировать работоспособность системы эта информация обычно сохраняется в журналах почтовых серверов.

Заголовки письма описываются стандартом RFC 2076 — Common Internet Message Headers (общепринятые стандарты заголовков сообщений) [3]. Заголовки используются для журналирования прохож-

дения письма и служебных пометок. В заголовка как правило содержится служебная информация о почтовых серверах, через которые прошло письмо, информация о проверке антивирусом и о том, похоже ли данное письмо на спам, а также другая служебная информация. Поскольку заголовки являются служебной информацией, то чаще всего почтовые клиенты скрывают их от пользователя при обычном чтении писем, но также предоставляют возможность увидеть эти заголовки, если возникает потребность в более детальном анализе письма. Также заголовки письма хранят и показываемую пользователю информацию, обычно это отправитель письма, получатель, тема и дата отправки.

Тело письма отделяется от заголовков пустой строкой и содержит в себе непосредственно сообщение письма. Первоначально в спецификации SMTP было определено использование только 7-битовых символов ASCII. Но с появлением стандарта многоцелевых почтовых расширений Internet (Multipurpose Internet Mail Extensions — MIME) и превращением Internet во всемирную сеть было предложено включить в дополнительные спецификации другие наборы символов. Благодаря этому в настоящее время электронное письмо может быть отправлено практически на любом национальном языке, а к письму могут прилагаться в закодированном виде данные почти любого типа, даже такие как изображения или исполняемые файлы [2].

В связи с этим, в настоящее время тело многих электронных писем представляется отправителем не в виде текста с сообщением (рисунок 1.1), а в виде текста на языке HTML, содержащем в себе текст самого сообщения (рисунок 1.2).

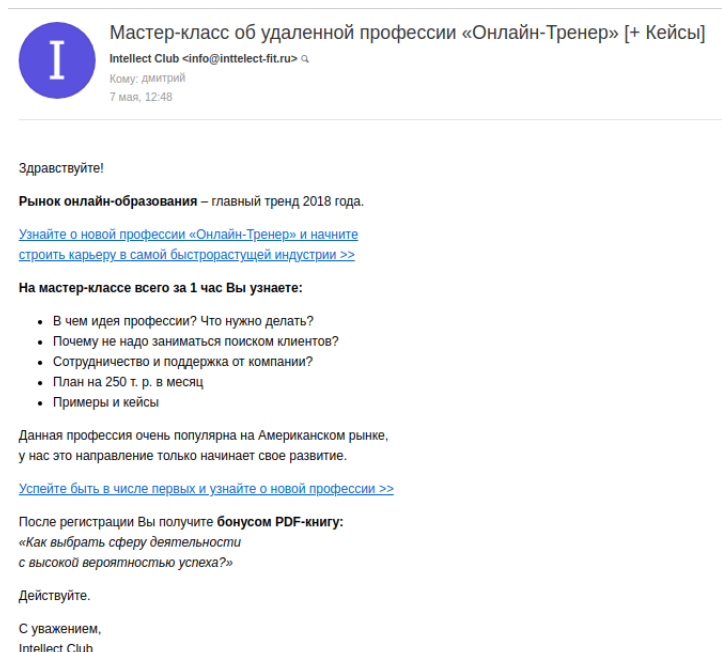


Рисунок 1.1 — Электронное письмо, содержащее тело в виде текста с сообщением

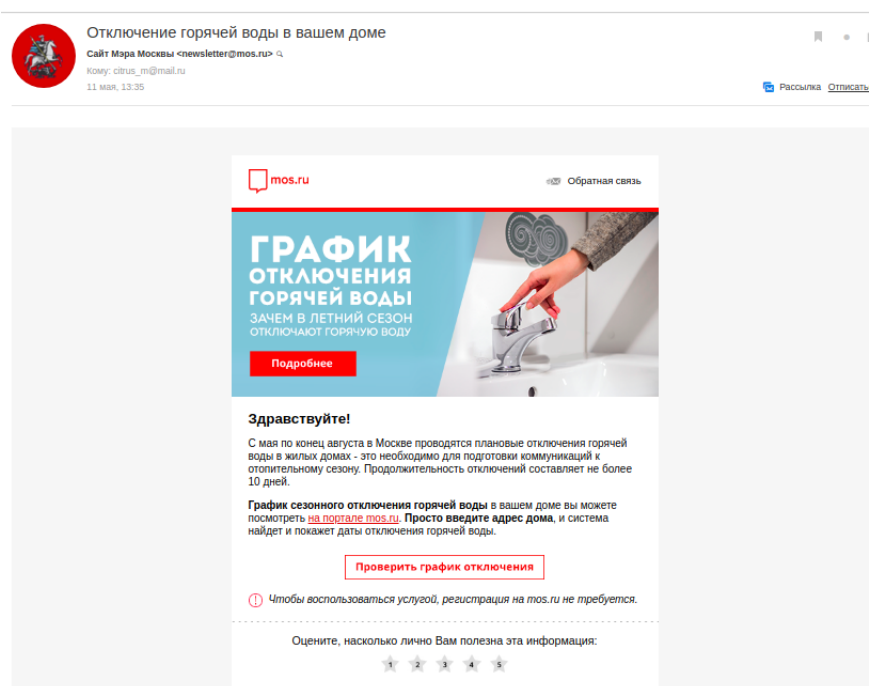


Рисунок 1.2 — Электронное письмо, содержащее тело в виде текста на языке HTML

1.3.2 Подпись в электронных письмах

Подпись представляет собой набор последовательных строк в конце текста электронного письма, который добавляется автоматически с помощью почтового клиента или дописывается автором(рисунок 1.3).

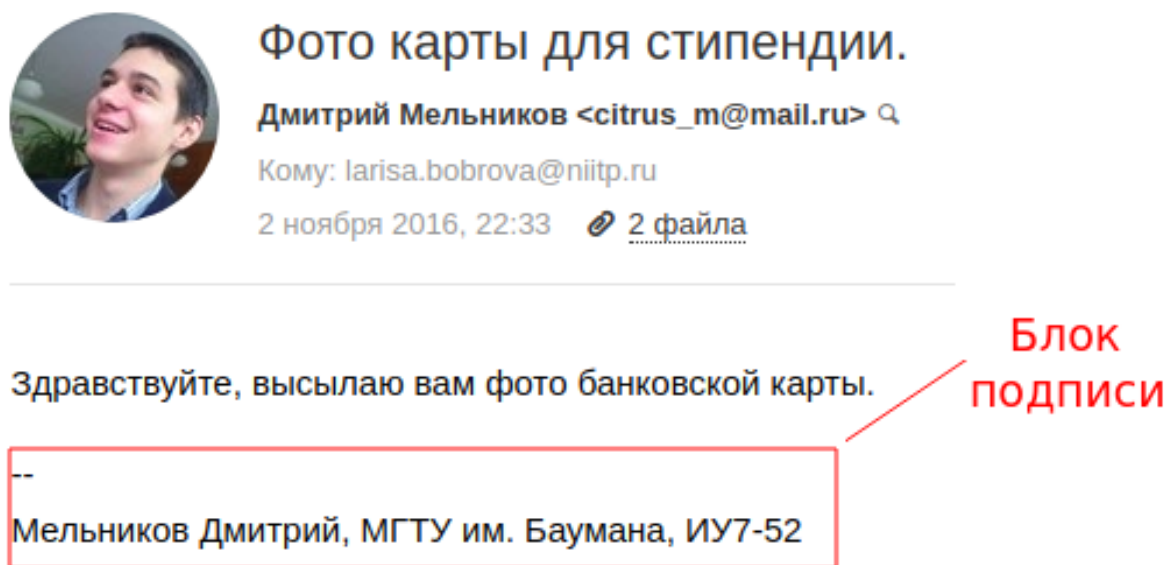


Рисунок 1.3 — Пример оформления подписи в электронном письме

Как правило, подпись содержит информацию об отправителе, например, имя отправителя, его должность, контактные данные и др.

Подпись в электронном письме является неотъемлемым атрибутом переписки, так как наличие подписи, при условии ее правильного оформления, производит благоприятное впечатление на партнеров по общению. В зависимости от характера письма(личная или деловая переписка, рекламная рассылка) можно выделить ряд основных функций, которые возлагаются на подпись:

- установление доверительных отношений с партнером по общению;
- доведение информации;
- ненавязчивая реклама;
- экономия временных затрат при автоматизации процедуры визуирования сообщений.

1.3.3 Выводы

На основе анализа структуры письма можно сделать вывод о том, что для текущей работы интерес будет представлять именно тело письма, а также, возможно, и некоторые заголовки.

Также следует заметить, что не существует единого стандарта для оформления подписи или ее обозначения почтовым клиентом. Как правило, оформление подписи в электронном письме может регламентироваться локальными правилами, например кодексом компании, или устоявшимися в обществе нормами. Однако, несмотря на это, подпись в электронном письме является частью текста сообщения и может иметь достаточно разные формы записи. Таким образом, можно сделать вывод о том, что задачу выделения подписи из текста электронного письма уместно будет отнести к одному из классов задач компьютерной лингвистики.

1.4 Компьютерная лингвистика: задачи и основные направления

Компьютерная лингвистика — научное направление в области математического и компьютерного моделирования интеллектуальных процессов у человека и животных при создании систем искусственного интеллекта, которое ставит своей целью использование математических моделей для описания естественных языков [4]. Полем деятельности компьютерных лингвистов является разработка алгоритмов и прикладных программ для обработки языковой информации.

Сложность задач компьютерной лингвистики связана с тем, что естественный язык — сложная многоуровневая система знаков, возникшая для обмена информацией между людьми, выработанная в процессе практической деятельности человека, и постоянно изменяющаяся в связи с этой деятельностью. Другая сложность разработки методов компьютерной лингвистики связана с многообразием естественных языков, существенными отличиями их лексики, морфологии, синтаксиса, разные языки предоставляют разные способы выражения одного и того же смысла [5].

Область приложений компьютерной лингвистики постоянно расширяется, однако можно выделить наиболее известные прикладные задачи, решаемые с помощью ее инструментов[5]:

- машинный перевод;
- реферирование текста;
- аннотирование текста;
- классификация текстов;
- кластеризация текстов;
- формирование ответов на вопросы;
- автоматизация подготовки и редактирования текстов на естественном языке;
- автоматическая генерация текстов на естественном языке;
- извлечение информации из текстов;
- распознавание и синтез звучащей речи.

Машинный перевод [6] является самым ранним приложением компьютерной лингвистики, вместе с которым развивалась и сама эта область. Первые программы, использующие машинный перевод, были построены более 50 лет назад и были основаны на простейшей стратегии пословного перевода. Однако сейчас, данное направление достаточно хорошо развито: программы машинного перевода используются повсеместно с применением самых последних технологий и методов.

Реферирование текста – сокращение его объема и получение его краткого изложения – реферата (свернутого содержания), что делает более быстрым поиск в коллекциях документов. Общий реферат может составляться также для нескольких близких по теме документов. Основным методом автоматического реферирования до сих пор является отбор наиболее значимых предложений реферируемого текста, для чего обычно сначала вычисляются ключевые слова текста и рассчитывается коэффициент значимости предложений текста.

Близкая к реферированию задача – **аннотирование текста** документа, т.е. составление его аннотации. В простейшей форме аннотация представляет собой перечень основных тем текста, для выделения которых могут использоваться процедуры индексирования.

При создании больших коллекций документов актуальны задачи **классификации** и **кластеризации** текстов с целью создания классов близких по теме документов [7]. Классификация означает отнесение каждого документа к определенному классу с заранее известными параметрами, а кластеризация – разбиение множества документов на кластеры, т.е. подмножества тематически близких документов. Для решения этих задач применяются методы машинного обучения, в связи с чем эти прикладные задачи называют Text Mining и относят к научному направлению, известному как Data Mining, или интеллектуальный анализ данных [8].

Еще одна относительно новая задача, связанная с информационным поиском – **формирование ответов на вопросы** (Question Answering) [9]. Эта задача решается путем определения типа вопроса,

поиском текстов, потенциально содержащих ответ на этот вопрос, и извлечением ответа из этих текстов.

Совершенно иное прикладное направление, которое развивается хотя и медленно, но устойчиво – это **автоматизация подготовки и редактирования текстов на ЕЯ**. Одним из первых приложений в этом направлении были программы автоматической определения переносов слов и программы орфографической проверки текста (спеллеры, или автокорректоры). Несмотря на кажущуюся простоту задачи переносов, ее корректное решение для многих языков (например, английского) требует знания морфемной структуры слов соответствующего языка, а значит, соответствующего словаря.

Все более актуальная прикладная задача, часто относимая к направлению Text Mining – это **извлечение информации** из текстов, или Information Extraction [10], что требуется при решении задач экономической и производственной аналитики. Для этого осуществляется выделение в тексте ЕЯ определенных объектов – именованных сущностей (имен, персоналий, географических названий), их отношений и связанных с ними событий. Как правило, это реализуется на основе частичного синтаксического анализа текста, позволяющего выполнять обработку потоков новостей от информационных агентств. Поскольку задача достаточно сложна не только теоретически, но и технологически, создание значимых систем извлечения информации из текстов осуществимо в рамках коммерческих компаний.

Как уже было сказано ранее, подпись в электронном письме содержится непосредственно в тексте письма, то есть является его частью. Полагаясь на данный факт и специфику задачи, можно сделать вывод о том, что строки текста электронного письма можно рассматривать с точки зрения принадлежности одному из двух классов: класс строк, содержащихся в подписи, и другие строки текста письма. На основании данного утверждения можно предположить, что данную задачу уместно рассматривать именно как задачу классификации множества строк текста электронного письма. Поэтому далее следует рассмотреть суще-

ствующие методы и подходы, применяемые для решения задачи классификации в области обработки текстов на естественном языке.

1.5 Анализ существующих подходов и методов классификации текстов на естественном языке

Для решения задач классификации, в том числе и в области обработки текстов на естественном языке, наиболее успешно применяются методы, использующие нейронные сети [11]. Их использование для решения задачи классификации состоит в указании принадлежности входного образа, представленного вектором входных признаков, одному или нескольким заранее определенным классам.

Построение эффективной нейросетевой модели, решающей задачи классификации, можно разделить на следующие основные этапы [11]:

- получение исходных данных из множества объектов, подлежащих классификации;
- предварительная обработка исходных данных и формирование обучающей выборки для обучения нейронной сети;
- разработка структуры нейронной сети: задание входов, выходов, числа слоев сети и нейронов в каждом слое;
- обучение нейронной сети для построения модели классификатора;
- тестирование и оценка качества работы нейросетевой модели.

При неудовлетворительных результатах оценки модели необходимо вернуться к одному из этапов и выполнить все последующие этапы в указанной последовательности.

Процесс получения исходных данных, которые необходимо использовать для обучения, определяется прежде всего спецификой задачи, а также требованиями к разрабатываемому программному обеспечению. Однако, для обучения классификатора недостаточно просто полученных исходных данных, так как модель классификатора, использующего методы машинного обучения, должна принимать на вход вектор числовых значений, который описывает точку в пространстве признаков. Используя множество таких векторов, можно обучить модель разделять все пространство признаков на определенное количество областей. В связи с этим возникает необходимость предварительной обработки исходных

данных с целью приведения их к форме, пригодной для передачи на вход классификатора.

1.5.1 Предварительная обработка данных

В зависимости о типа задачи и исходного формата входных данных можно выделить основные подходы, применяемые для предварительной обработки данных и приведения их к необходимому виду:

- использование словарей и частотного анализатора;
- использование векторного представления слов;
- выделения множества признаков, описывающих классифицируемые объекты;

Суть подхода с использованием **словарей и частотного анализатора** заключается в том, что перед обучением классификатора из множества учебных текстов определенным образом выделяются слова, т.е. формируется словарь. Затем на основании данного словаря с использованием частотного анализатора для текста строится вектор, который описывает частотные характеристики для всех слов, содержащихся в тексте. Схема частотного анализатора представлена на рисунке 1.4).

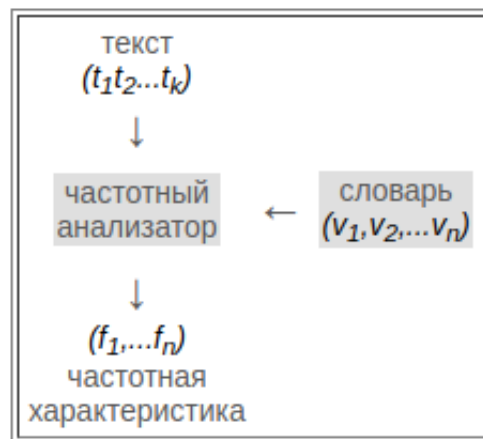


Рисунок 1.4 — Схема частотного анализатора

Частотный анализатор определяет для каждого слова v_i из словаря V его частоту вхождения f_i в данный текст t . Частотная характеристика это вектор $f = (f_1, \dots, f_n) \in F$, длина которого равна количеству

слов в словаре V , каждая компонента f_i это целое неотрицательное число:

$$f_i = \sum_{j=0}^k e(v_i, t_j); e(v_i, t_j) = \begin{cases} 0, & v_i \neq t_j \\ 1, & v_i = t_j \end{cases} \quad (1.1)$$

Другими словами - для каждого слова $v_i \in V$ определяется число его вхождений $f_i \geq 0$ в данный текст $t = t_1 t_2 t_3 \dots t_k$. Частотную характеристику f можно рассматривать как точку в пространстве признаков F , соответствующую тексту t . Таким образом, на вход частотного анализатора подается текст t и словарь V , на выход выдается точка в пространстве признаков F .

Словарь - важная часть модели классификатора, поэтому он должен удовлетворять следующим требованиям [12]:

- Словарь должен состоять из слов, имеющих четко определенное значение, чтобы результат частотного анализа содержал минимум шума;
- Словарь должен содержать достаточно слов, чтобы частотный анализ был информативен;
- Словарь должен быть небольшим, чтобы процедура частотного анализа могла быть выполнена за приемлемое время;

Составить словарь, удовлетворяющий этим требованиям, можно используя известные лингвистические закономерности. В качестве примера для такой закономерности можно привести закон Бредфорда [13].

Подход, использующий **векторные представления слов**, заключается в следующем: процесс предварительной обработки данных начинается с обучения нейронной сети, которая принимает на вход большой корпус текстовых данных и сопоставляет каждому слову вектор, выдавая координаты слов на выходе. Векторное представление основывается на контекстной близости: слова, встречающиеся в тексте рядом с одинаковыми словами, в векторном представлении будут иметь близкие координаты векторов-слов [14]. Минусом данного подхода является то,

что после получения векторного представления слов возникает задача объединения всех представлений, входящих в классифицируемые объекты(тексты, предложения), в единый вектор, пригодный для передачи на вход классификатора.

Также решить проблему предварительной обработки данных можно не прибегая к использованию словарей или больших корпусов текстовых данных. Для этого можно воспользоваться походом, который заключается в **выделении множества признаков**, которые будут пригодны для классификации объектов в рамках решаемой задачи.

В качестве примера можно рассмотреть применение данного подхода в статье А.С.Катасёва, Д.В.Катасёвой, А.П.Кирпичникова [15], в которой авторами описывается метод реализации спам-фильтра электронных сообщений на основе нейросетевого классификатора. Для классификации сообщений, в своей работе авторы статьи провели предварительную обработку исходных данных(текстов писем), представив их в качестве векторов признаков. Для этого они выделили следующие признаки, которые, по их мнению, влияют на результат классификации писем на категории "спам"/"не спам":

- частота встречаемости слов верхнего регистра;
- частота встречаемости цифр в письме;
- количество разных цветов в тексте письма;
- размер текста письма;
- количество пустых строк в тексте письма;

Стоит также заметить, что предварительная обработка данных включает в себя помимо представления исходных данных в формате числового вектора еще и **нормализацию** значений полученного вектора. Под нормализацией понимается такое преобразование вектора значений, в результате которого все значения вектора принимают значений в диапазоне от 0 до 1. Нормализация данных позволяет привести все используемые числовые значения переменных к одинаковой области их изменения,

благодаря чему появляется возможность свести их вместе в одной нейросетевой модели [16]. Чтобы выполнить нормализацию данных, нужно точно знать пределы изменения значений соответствующих переменных. Тогда им и будут соответствовать границы интервала нормализации. Когда точно установить пределы изменения переменных невозможно, они задаются с учетом минимальных и максимальных значений в имеющейся выборке данных. Методы нормализации принято разделять на две следующие группы [16]:

- линейная нормализация;
- нелинейная нормализация.

1.5.2 Алгоритмы классификации

Алгоритмы классификации можно разделить на две группы:

- алгоритмы классификация с учителем;
- алгоритмы классификации без учителя.

Алгоритмы **классификации с учителем** сортируют объекты классифицируемого множества по заранее известным категориям(классам). В роли учителя выступает выборка объектов, для которых заранее известна принадлежность той или иной категории, называемая обучающим множеством [5]. Алгоритмы классификации называют по имени метода обучения, положенного в его основу. К наиболее известным алгоритмам классификации с учителем относятся следующие [5]:

- Алгоритм "наивной"байесовской классификации;
- Алгоритм k-ближайших соседей;
- Алгоритм опорных векторов.

Алгоритмы **классификации с учителем** анализируют коллекцию объектов с целью разбиения их на группы так, чтобы внутри одной группы оказывались объекты наиболее родственные в некотором смысле, а различные - попадали в различные группы. При этом отсутствует «учитель» - обучающее подмножество документов и заранее известное множество категорий (классов). В общем случае алгоритм классификации без

учителя - алгоритм кластеризации - должен самостоятельно принимать решения о количестве и составе кластеров, то есть групп родственных объектов [5].

Так как в данной работе четко определены классы, согласно которым следует классифицировать строки электронного письма, то следует более детально рассмотреть алгоритмы классификации с учителем и не останавливаться на подробном разборе алгоритмов классификации без учителя, так как задача кластеризации в данной работе не стоит.

Алгоритм "наивной"байесовской классификации Алгоритм "наивной"байесовской классификации(НБА) - это алгоритм классификации, основанный на теорема Байеса [17] с допущением о независимости признаков. То есть алгоритм НБА не предполагает, что наличие какого-либо признака в классе не зависит от наличия какого-либо другого признака. Даже если какие-то признаки и зависят друг от друга, то считается, что они вносят независимый вклад в вероятность принадлежности классифицируемого объекта определенному классу. В связи с данным допущением алгоритм и называется "наивным".

Рассмотрим объект O . А также множество классов C , к одному из которых следует отнести объект O . Необходимо найти такой класс c из множества C , при котором вероятность для данного объекта была бы максимальна. Математически это можно записать в следующем виде:

$$c = \arg \max_C P(C|O). \quad (1.2)$$

Для вычисления $P(C|O)$ можно воспользоваться теоремой Байеса и перейти к косвенным вероятностям:

$$P(C|O) = \frac{P(O|C)P(C)}{P(O)} \quad (1.3)$$

Так как в задачах классификации объект O представляется вектором признаков $(o_1 o_1 \dots o_n)$, то предыдущая формула примет вид:

$$P(C|o_1o_2...o_n) = \frac{P(o_1o_2...o_n|C)P(C)}{P(o_1o_2...o_n)} \quad (1.4)$$

С учетом "наивного" допущения о том, что переменные o_i зависят только от класса и не зависят друг от друга, можно переписать числитель в следующем виде:

$$P(o_1o_2...o_n|C)P(C) = P(C) \prod_i P(o_i|C) \quad (1.5)$$

В итоге (1.2) примет вид:

$$c = \arg \max_{c \in C} P(c|o_1o_2...o_n) = \arg \max_{c \in C} P(c) \prod_i P(o_i|C) \quad (1.6)$$

Соответственно, под тренировкой классификатора понимается вычисление значений $P(c)$ и $P(o_i|c)$.

К плюсам данного алгоритма можно отнести возможность его применения для решения задач многоклассовой классификации, высокую скорость обучения классификатора, а также тот факт, что при выполнении допущения о независимости алгоритм НБА может превосходить другие алгоритмы, при этом требуя меньший объем обучающей выборки. Однако, если в тестовом наборе присутствует некоторое значение категориального признака, которое не встречалось в обучающей выборке, тогда модель присвоит нулевую вероятность этому значению, что негативно скажется на результатах прогноза. Также стоит заметить, что в реальности наборы полностью независимых признаков встречаются крайне редко, поэтому допущение о независимости признаков будет выполняться редко при решении реальных задач.

Алгоритм k -ближайших соседей Алгоритм k -ближайших соседей использует гипотезу компактности векторного пространства, которая заключается в том, что объекты одного класса образуют в пространстве признаков компактную область, причем области разных классов не

пересекаются. Тогда можно ожидать, что тестовый объект будет иметь такую же метку класса, как и окружающие его объекты из обучающего множества. Данный алгоритм относит объект к преобладающему классу среди k его соседей. При $k=1$ алгоритм относит объект к классу самого ближайшего ему объекта.

Суть работы алгоритма можно рассмотреть на примере, представленном на рисунке 1.5.

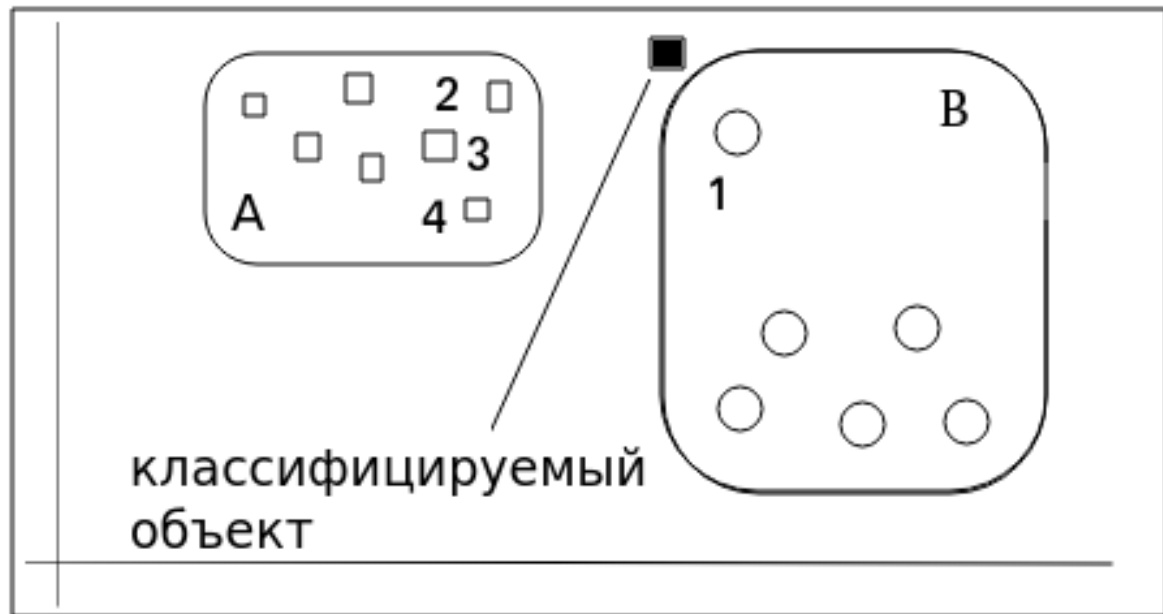


Рисунок 1.5 — Геометрическая интерпретация алгоритма k -ближайших соседей

Пусть имеется множество объектов, принадлежащих двум классам: А и В. При $k = 1$, классифицируемый объект будет отнесен к классу В, так как ближе всего классифицируемый объект располагается к объекту 1, принадлежащему классу В. Однако, при $k=4$, классифицируемый объект будет отнесен к классу А, так как среди четырех объектов, минимально удаленных от классифицируемого объекта, три объекта будут принадлежать классу А(объекты 2,3,4).

Перед тем, как определять k наименее удаленных объектов, необходимо определить функцию дистанции. Классическим вариантом функции дистанции является дистанция в евклидовом пространстве [18]:

$$d(p,q) = \sqrt{\sum_{k=1}^n (p_k - q_k)^2}, \quad (1.7)$$

где $p = (p_1, \dots, p_n)$ и $q = (q_1, \dots, q_n)$ - точки в евклидовом пространстве размерности n .

При использовании данного алгоритма основной задачей, которую необходимо решить, является выбор значения k . Если выбрать значение k слишком малым, то есть опасность, что единственным ближайшим объектом окажется «выброс», т.е. объект с неправильно определённым классом, и он даст неверное решение. Казалось бы, увеличивая значение параметра k , можно понизить вероятность случайного попадания на такие «выбросы» в качестве ближайших соседей исследуемого объекта. Но здесь возникает другая опасность. Чтобы понять в чём она заключается, рассмотрим случай, когда k равно общему числу объектов N . Понятно, что тогда в этом случае объект будет отнесен к классу, который имеет наибольшее количество объектов, и расстояние до исследуемого объекта не будет играть вообще никакой роли. На практике чаще всего полагают $k = \sqrt{N}$.

К плюсам данного алгоритма можно отнести то, что он подходит для решения задачи многоклассовой классификации. Однако применение данного алгоритма осложняется необходимостью подбора значения k , а также низкой скоростью классификации, по сравнению с НБА или алгоритмом опорных векторов.

Алгоритм опорных векторов

Данный алгоритм ищет в векторном пространстве объектов разделяющую гиперплоскость между двумя классами, максимально удаленную от всех точек обучающего множества (рисунок 1.6).

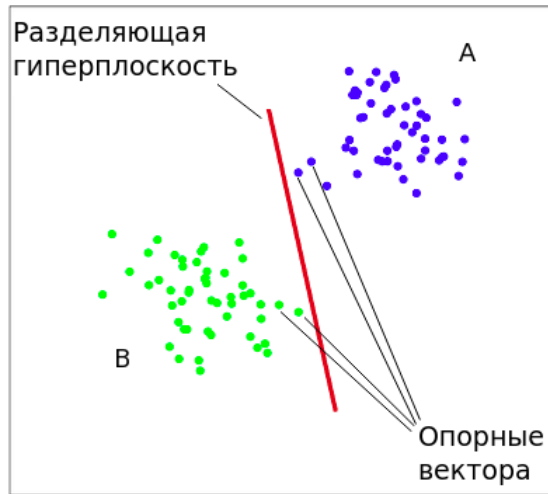


Рисунок 1.6 — Геометрическая интерпретация алгоритма опорных векторов

Расстояние между найденной гиперплоскостью и ближайшей точкой данных называется зазором классификации. В алгоритме опорных векторов разделяющая гиперплоскость полностью определяется небольшим подмножеством объектов. Элементы данного подмножества называются опорными векторами.

Пусть имеется обучающая выборка: $(x_1, y_1), \dots, (x_m, y_m)$, $x_i \in \mathbf{R}^n$, $y_i \in \{-1, 1\}$. Алгоритм опорных векторов строит классифицирующую функцию:

$$f(\vec{x}) = \text{sign}(\vec{w}^T \vec{x} + b), \quad (1.8)$$

где

- \vec{w}^T - вектор нормали к разделяющей поверхности;
- \vec{x} - вектор-представление классифицируемого объекта;
- b - параметр сдвига.

Значение функции классификатора -1 соответствует одному классу, а +1 - другому,

Далее необходимо выбрать такие w и b , которые максимизируют расстояние до каждого класса. Можно подсчитать, что данное расстояние равно $\frac{1}{\|\vec{w}\|}$ (рисунок 1.7).

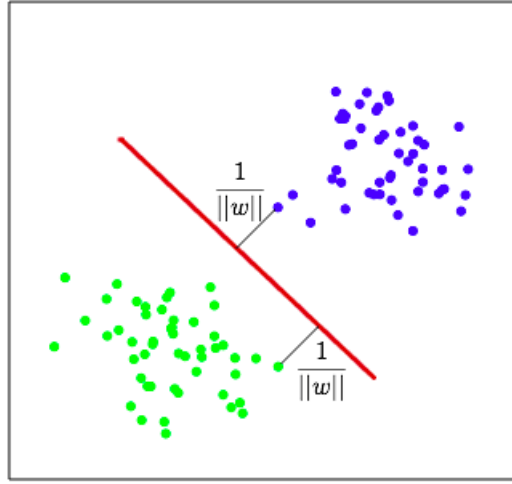


Рисунок 1.7 — Расстояние от каждого класса до гиперплоскости

Проблема нахождения максимума $\frac{1}{\|w\|}$ эквивалентна проблеме нахождения минимума $\|w\|^2$. Данное условие можно записать в виде задачи оптимизации:

$$\begin{cases} \arg \min_{w,b} \|w\|^2, \\ y_i(\vec{w}^T \vec{x} + b) \geq 1, i = 1, \dots, m. \end{cases} \quad (1.9)$$

которая является стандартной задачей квадратичного программирования и решается с помощью множителей Лагранжа [19].

На практике редко встречаются случаи, когда исследуемое множество линейно разделимо. Пример линейной неразделимости показан на рисунке 1.8.

В этом случае поступают следующим образом: все элементы обучающей выборки вкладываются в пространство X более высокой размерности с помощью специального отображения: $\varphi : \mathbf{R}^n \rightarrow X$. При этом отображение φ выбирается таким образом, чтобы в новом пространстве X выборка была линейно разделима.

В таком случае классифицирующая функция примет следующий вид:

$$f(\vec{x}) = \text{sign}(\vec{w}^T \varphi(\vec{x}) + b) \quad (1.10)$$

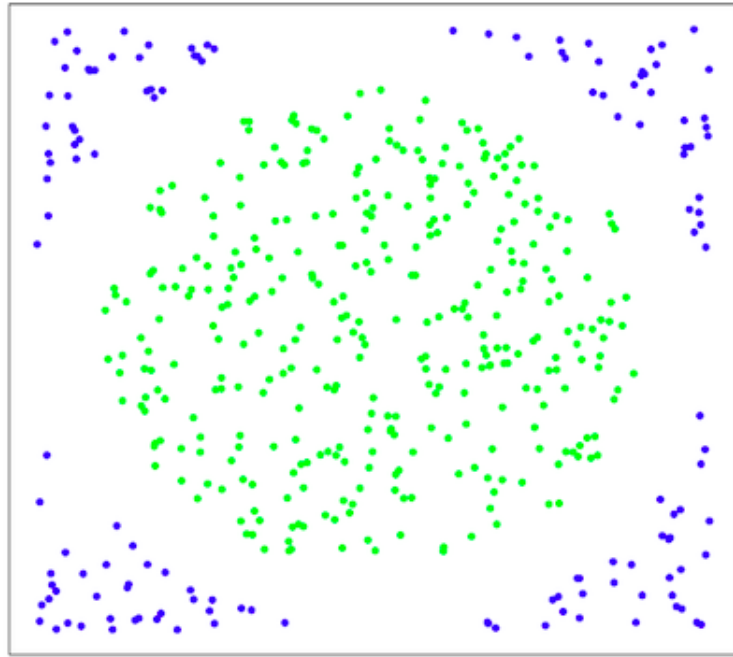


Рисунок 1.8 — Линейно неразделимое множество

Выражение $k(x, x') = (\varphi(x), \varphi(x'))$ называется ядром классификатора. С математической точки зрения ядром может служить любая положительно определенная симметричная функция двух переменных [20]. Выбор ядра влияет на точность работы классификатора.

К плюсам данного метода можно отнести хорошие показатели работы алгоритма в случаях линейной разделимости. Однако, при не выполнении условия линейной разделимости, необходимо прибегать к подбору параметров ядра для получения наилучших результатов, что требует большое количество времени, так как при каждом новом подборе параметра выполняется переобучение классификатора. Также следует обратить внимание, что в классическом варианте данный алгоритм подходит только для решения задач бинарной классификации.

1.5.3 Параметры для оценки качества работы классификатора

В простейшем случае метрикой для оценки качества работы классификатора может быть доля объектов, по которым классификатор принял правильное решение:

$$Accuracy = \frac{P}{N}, \quad (1.11)$$

где P - количество объектов, по которым классификатор принял правильное решение, N - размер тестовой выборки.

Тем не менее, данная метрика имеет одну особенность, которую необходимо учитывать. Она присваивает всем объектам одинаковый вес, что может быть не корректно в случае, если распределение объектов в обучающей выборке сильно смещено в сторону какого-то одного или нескольких классов. В этом случае у классификатора есть больше информации по этим классам и соответственно в рамках этих классов он будет принимать более адекватные решения. Для решения данной проблемы можно воспользоваться специально подобранной сбалансированной обучающей выборкой, однако рекомендуется для оценки качества работы классификатора использовать другие метрики.

Одними из таких метрик являются **точность**(precision) и **полнота**(recall). Точность системы в пределах класса – это доля объектов, действительно принадлежащих данному классу относительно всех объектов, которые система отнесла к этому классу. Полнота системы – это доля найденных классификатором объектов, принадлежащих классу относительно всех объектов этого класса в тестовой выборке. Для расчета значения данных параметров можно воспользоваться таблицей контингентности(рисунок 1.9), которая составляется отдельно для каждого класса.

Категория i		Экспертная оценка	
		Положительная	Отрицательная
Оценка системы	Положительная	TP	FP
	Отрицательная	FN	TN

Рисунок 1.9 — Таблица контингентности

В данной таблице содержится информация о том, сколько раз система приняла верное и сколько раз неверное решение при классификации объектов заданного класса. А именно:

- TP - истинно-положительные;
- FP - ложно-положительные;
- TN - истинно-отрицательные;
- FN - ложно-отрицательные.

Тогда, точность и полнота определяются следующим образом:

$$Precision = \frac{TP}{TP + FP} \quad (1.12)$$

$$Recall = \frac{TP}{TP + FN} \quad (1.13)$$

Очевидно, что чем выше точность и полнота, тем более качественной является работа системы. Однако, максимальная точность и полнота не достижимы одновременно, поэтому приходится искать некий баланс. Для этого разумным является ввести какую-то одну метрику, которая объединяла бы в себе информацию о полноте и точности разрабатываемого классификатора. Одной из таких метрик является **F-мера**, которая представляет собой гармоническое среднее [5] между точностью и полнотой. Она стремится к нулю, если точность или полнота стремится к нулю. F-мера рассчитывается по следующей формуле:

$$F = 2 \frac{Precision * Recall}{Precision + Recall} \quad (1.14)$$

В вычислении F-меры для задачи классификации есть два основных подхода:

- Суммарный: результаты по всем классам сводятся в единую таблицу, по которой затем вычисляется F-мера;

— Средний: для каждого класса F-мера рассчитывается отдельно, затем на основании полученных значений вычисляется среднее арифметическое значение F-меры для всех классов.

1.6 Выводы

В аналитическом разделе была определена цель и решаемые задачи, проведен анализ предметной области. В результате проведенного анализа был сделан вывод о том, что данная задача относится к области задач компьютерной лингвистики, а именно, к классу задач классификации. В связи с этим были также рассмотрены существующие методы и подходы, используемые для решения задач классификации.

В связи с тем, что подпись в электронном письме содержится непосредственно в его теле, то будет разумным ввести ограничение к разрабатываемому ПО, которое заключается в том, что на вход программы будет подаваться текст письма и информация из его заголовков.

Так как для работы разрабатываемого классификатора необходимо провести предварительную обработку входных данных, то в случае данной задачи наиболее подходящим будет подход, который основан на выделении множества признаков, которые будут характеризовать строки письма с точки зрения принадлежности выделенным классам ("подпись"/"не подпись"), что позволит также использовать англоязычные письма для обучения классификатора. Другие подходы, в отличие от описанного, требуют большого корпуса размеченных данных, а также большого количества именно русскоязычных писем для составления словарей. Также следует заметить, что подпись в электронных письмах часто содержит имена собственные, сокращения, а также другую информацию, например, номера телефонов, url-адреса. Данный факт также осложняет применение подхода, использующего словари.

На основании проведенного анализа существующих алгоритмов классификации, можно сделать вывод о том, что все они подходят для решения задач бинарной классификации. Однако, сложно выделить один наиболее подходящий для реализации поставленной задачи, так как качество работы данных алгоритмов классификации зависит от различных параметров, таких как размер обучающей выборки, независимость выделенных параметров, соотношение количества объектов того или иного класса в обучающей выборке. В связи с этим предлагается проверить работу всех описанных алгоритмов в рамках данной работы, а затем

оценить результаты с помощью параметров, описанных в конце аналитического раздела. На основании полученных результатов можно будет сделать вывод о том, какой из алгоритмов является наиболее подходящим для решения данной задачи.

2 Конструкторский раздел

2.1 Выполняемые задачи

Основная задача данной работы - удаление подписей из текстов электронных писем.

На рисунке 2.1 показана диаграмма IDEF0, которая отображает основные этапы удаления подписей из текстов электронных писем.

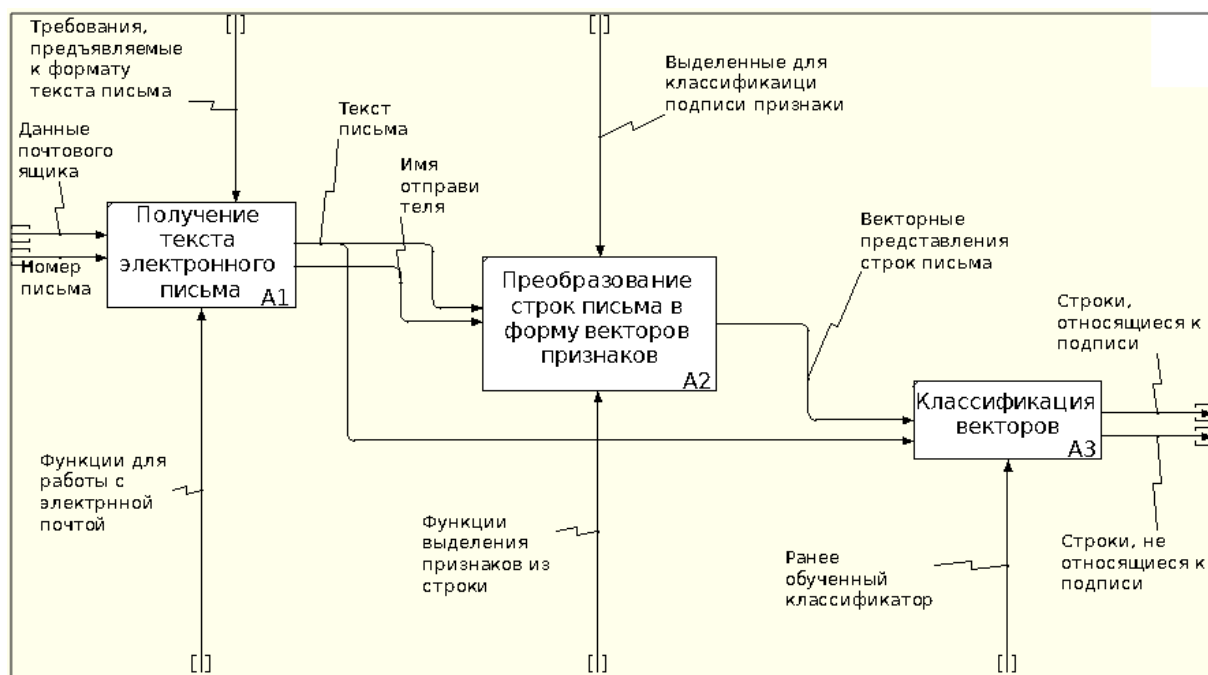


Рисунок 2.1 — IDEF0-диаграмма проекта

Как видно из рисунка, данную задачу можно разделить на три основных этапа:

- получение текста электронного письма;
- преобразование строк письма в форму векторов признаков;
- классификация векторов, полученных на предыдущем этапе.

2.2 Получение текста электронного письма

На данном этапе решается задача получения текста электронного письма из почтового ящика пользователя. Соответственно, входными данными будут являться данные электронного почтового ящика (адрес, пароль), а также номер того письма, текст которого необходимо получить. В результате, помимо текста письма, будет получено еще и значение заголовка "From" оно будет использовано в следующем этапе для приведения строк письма к формату вектора признаков.

Так как в настоящее время, в теле письма может передаваться практически любая по типу информация, то введем ограничение, согласно которому тексту письма будет соответствовать только информация, имеющая значение "text/plain" заголовка "Content-type". Выделение текста из элементов типа "text/html" выходит за рамки разрабатываемого метода и может быть рассмотрено как расширение функционала разрабатываемого программного обеспечения.

На рисунке 2.2 представлена схема работы алгоритма получения текста электронного письма.

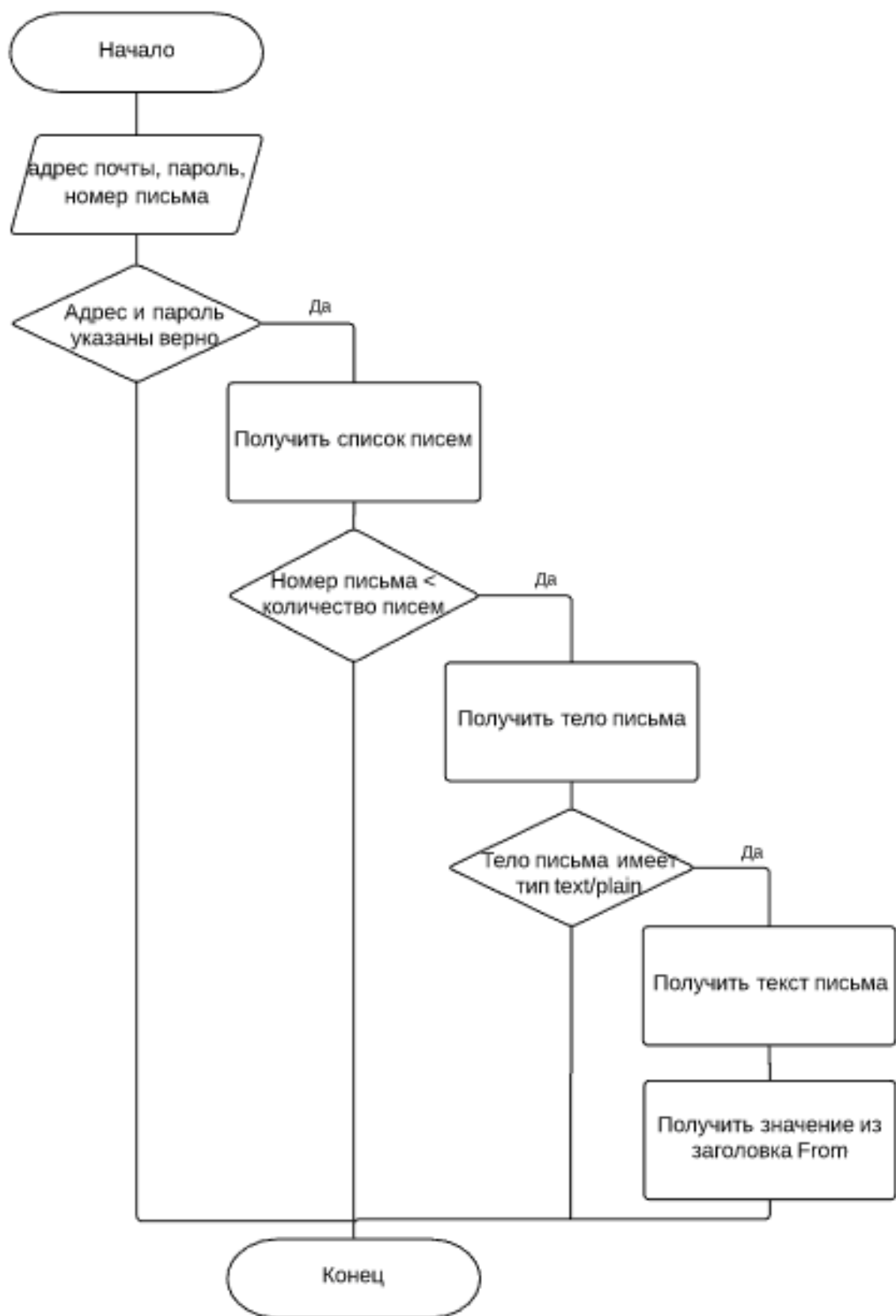


Рисунок 2.2 — Схема алгоритма получения электронного письма

2.3 Преобразование строк письма в форму векторов признаков

Для решения задачи классификации необходимо представить классифицируемые объекты, в данном случае строки текста электронного письма, в форме числовых векторов. В аналитическом разделе были описаны основные подходы, применяемые для приведения классифицируемых объектов к необходимому виду. В рамках текущей работы выбор был сделан в пользу метода, основанного на представлении строки в виде вектора признаков. С целью определения строк, относящихся к подписи, были выделены, следующие признаки:

- строка содержит адрес электронной почты;
- строка содержит URL адрес;
- строка представлена в виде разделителя типа — —";
- строка содержит номер телефона;
- строка содержит имя человека;
- строка содержит часто употребляемые слова (Спасибо, с уважением и т.п.);
- процент слов, начинающихся с заглавной буквы, в строке не менее 50;
- процент цифр среди общего количества всех символов в строке не менее 30;
- длина строки не более 50 символов;
- строка содержит имя отправителя, полученное из заголовка "From" электронного письма;

В результате выделения описанных выше признаков, строка текста электронного письма будет представлена в виде вектора $v = (v_1, v_2, \dots, v_n)$, где n - количество выделенных признаков. Элементы вектора v могут принимать два значения: 0 и 1, где 0 - данный признак отсутствует в строке, 1 - присутствует.

Например, для строки <С уважением, Мельников Дмитрий>, полученной из письма, отправленного от имени <Мельников Д.

citrus_m@mail.ru>, соответствующий вектор будет выглядеть следующим образом: (0, 0, 0, 0, 1, 1, 1, 0, 1, 1).

Стоит заметить, что первые 6 шесть описанных признаков могут быть выделены из строки с использованием регулярных выражений. Данные регулярные выражения представлены в приложении А.

На рисунке 2.3 представлена схема алгоритма преобразования строк письма в вектора признаков.

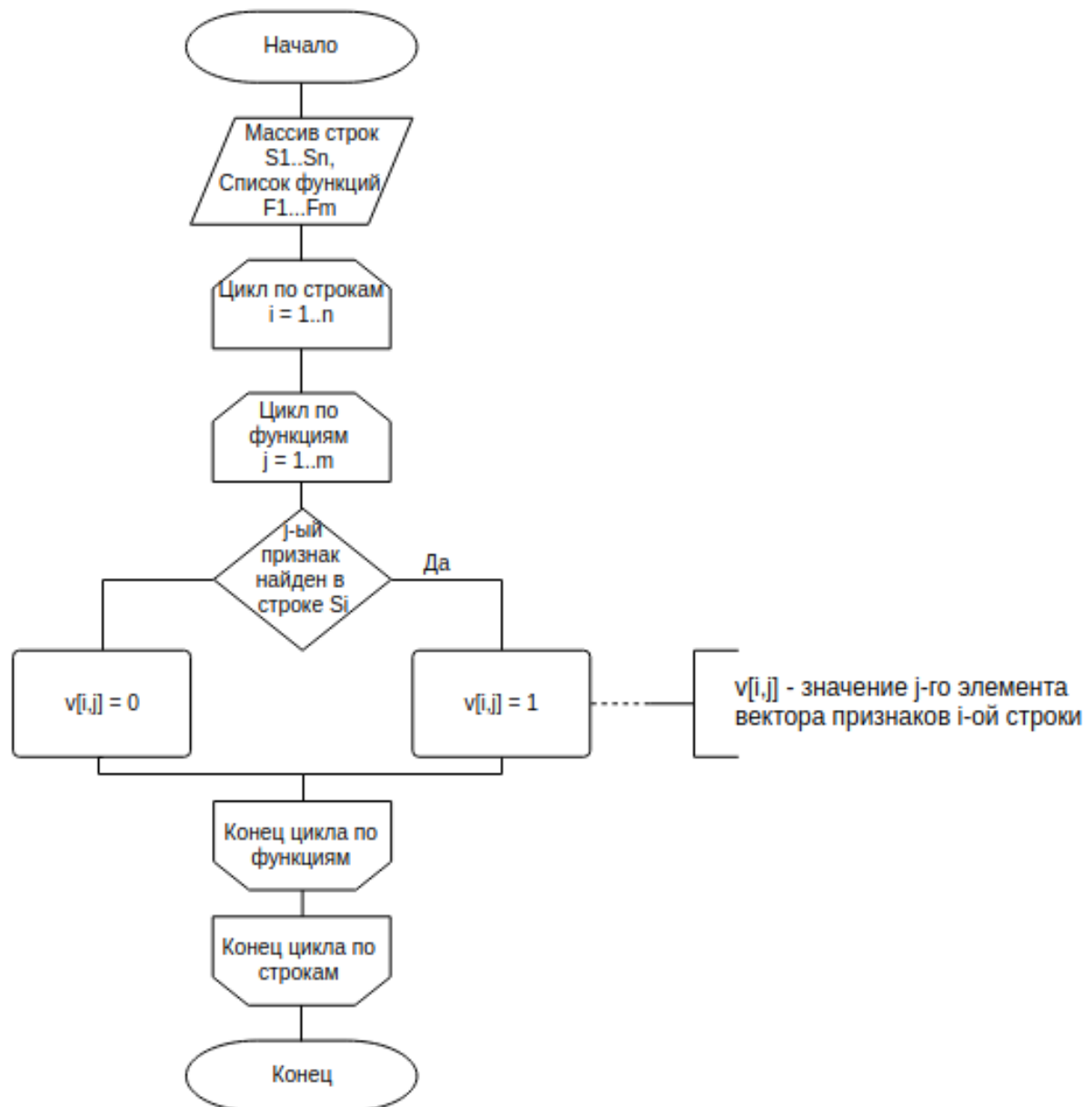


Рисунок 2.3 — Схема алгоритма преобразования строк письма в вектора признаков

2.4 Классификация векторов признаков

Последним этапом выделения подписи является классификация числовых векторов, описывающих строки письма. Для решения данной задачи необходимо иметь заранее обученный классификатор. Поэтому следует описать набор действий, необходимых для получения обученной модели классификатора.

2.4.1 Описание обучающей выборки данных

Так как для решения задач классификации используются алгоритмы обучения с учителем, то возникает необходимость разметки обучающей выборки с целью указания метки класса, к которому принадлежит классифицируемый объект.

В данной работе исходные данные представляют из себя множество из пар файлов, в которых один файл содержит текст электронного письма, а другой - значение, полученное из заголовка From, которое используется для выделения последнего из ранее описанных признаков. Разметке подлежит файл, содержащий текст письма. Пример размеченного файла представлен на рисунке 2.4.

Уважаемый(-ая) Дмитрий Мельников!

Ваш платеж банковской картой в пользу предприятия е-коммерции VELOBIKE PAY совершен успешно. Пожалуйста, сохраните указанную ниже информацию о платеже.

Номер платежа: 129644
Сумма операции: 150.00 RUB
Покупатель: DMITRY MELNIKOV
Номер карты: *****
Код авторизации: 251147|
Дата заказа: 2018-05-05 10:43:10
Комментарий: 1

ВНИМАНИЕ!

По всем вопросам, связанным с выполнением оплаченного Вами заказа, пожалуйста, обращайтесь в предприятие е-коммерции VELOBIKE PAY

#sig#Услуга обработки и безопасности платежей предоставлена процессинговой компанией

#sig#Uniteller (www.uniteller.ru).

#sig#Спасибо, что воспользовались нашими услугами! Отличного Вам настроения!

#sig#С уважением,

#sig#Команда Uniteller

Рисунок 2.4 — Пример разметки текста электронного письма

Префикс #sig# означает, что строка относится к классу "подпись".

2.4.2 Обучение классификатора

На рисунке 2.5 представлена IDEF0 диаграмма, описывающая основные этапы получения обученной модели классификатора.

Как видно из рисунка, данную задачу можно разделить на 4 основных этапа:

- Разметка текстов электронных писем;
- Приведение строк электронных писем к виду векторов признаков;
- Добавления данных для обучения в БД;
- Обучение классификатора;

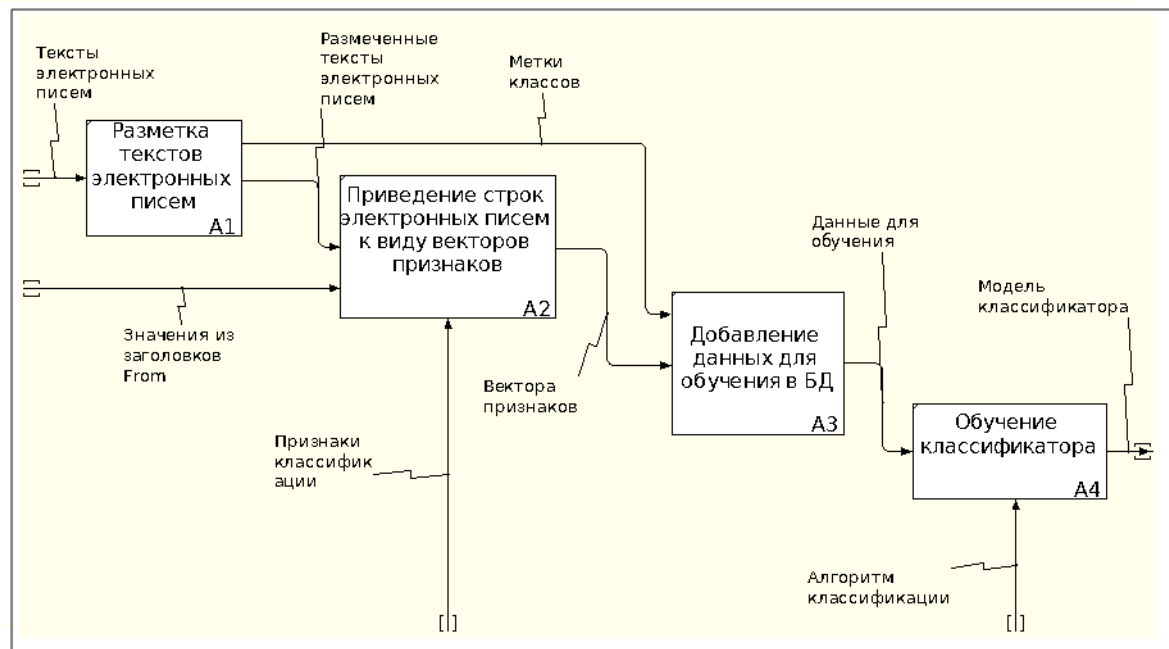


Рисунок 2.5 — IDEF-диаграмма получения обученной модели классификатора

Стоит обратить внимание на то, что третий этап не является обязательным, однако, использование базы данных для хранения обучающей выборки позволяет удобно расширять выборку за счет добавления новых писем, а также позволяет быстрее производить обучение классификатора.

2.5 Структура программы

Общая структура разработанного программного комплекса представлена на рисунке 2.6

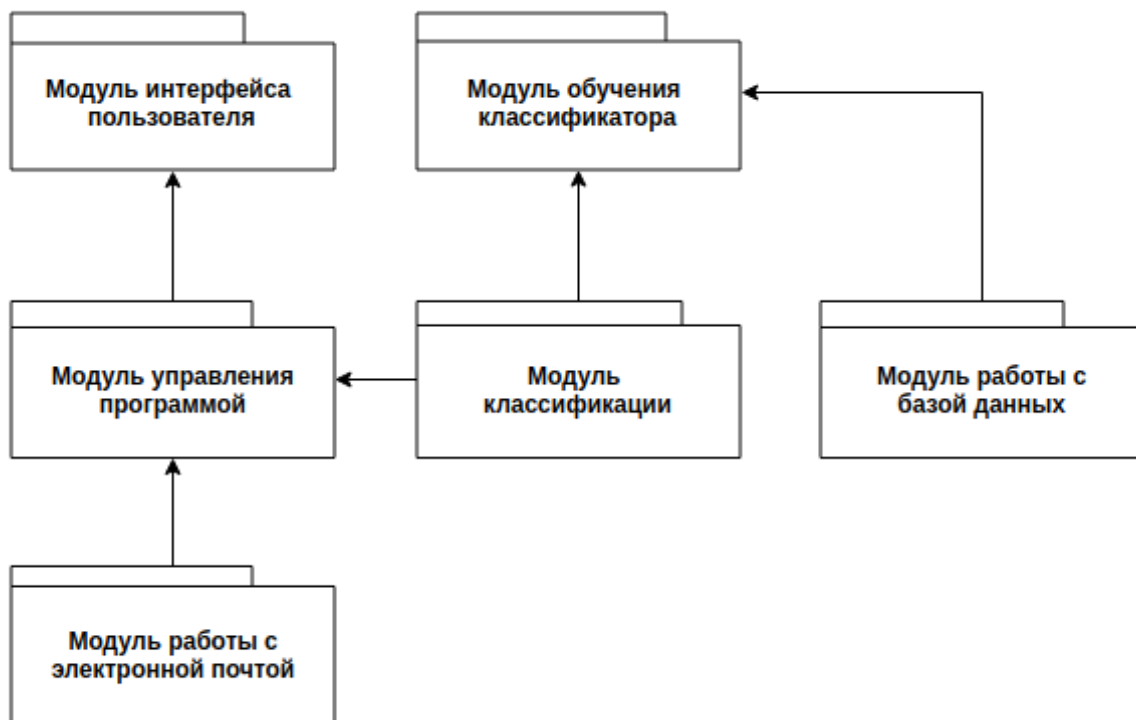


Рисунок 2.6 — Структура программного комплекса

Разработанный программный комплекс можно разделить на две части: пользовательское приложение с графическим интерфейсом и приложение для получения обученной модели классификатора. Основой пользовательского приложения является модуль управления программой, который содержит в себе основную логику работы приложения и осуществляет взаимодействие других компонентов приложения: модулей классификации, работы с электронной почтой и интерфейса пользователя.

Функционал, используемый обеими частями программного комплекса, представлен только в модуле классификации, при этом для работы пользовательского приложения не требуется наличия модуля работы с базой данных.

2.6 Выводы

В конструкторском разделе описывается предлагаемый метод удаления подписей из текстов электронных писем. Дано подробное описание, построены схемы выбранных алгоритмов. При этом выделены основные этапы работы метода с указанием необходимых исходных данных для его работы и полученных результатов на каждом этапе.

3 Технологический раздел

Для реализации программного продукта необходимо выбрать язык программирования, среду разработки, а также библиотеки, которые позволят сделать приложение функциональным и гибким по отношению к возможным будущим изменениям, модификации, расширению функционала. В данном разделе приводится краткое описание выбранных средств, а также требования для функционирования программного обеспечения и описание его архитектуры.

3.1 Выбор средств и инструментов разработки

3.1.1 Выбор языка программирования

В качестве языка программирования для реализации программного обеспечения был выбран язык Python. Это высокоуровневый язык программирования общего назначения, поддерживающий различные парадигмы программирования, а также портированный почти на все известные платформы. Синтаксис ядра Python минималистичен, при этом стандартная библиотека включает большой объем полезных функций. Также Python имеет богатую документацию с большим количеством примеров. Описанные особенности положительно влияют на скорость разработки программного обеспечения.

В данной работе выбор языка Python обусловлен следующими причинами:

- богатый выбор различных библиотек, в том числе и библиотек, используемых для решения задач машинного обучения;
- наличие подробной документации с примерами;
- наличие опыта разработки приложений на данном языке;
- большое количество проектов с открытым исходным кодом в области задач машинного обучения.

3.1.2 Выбор среды разработки

В качестве среды разработки был выбран текстовый редактор Visual Studio Code. Данный выбор обусловлен следующими причинами:

- Данный текстовый редактор поддерживается большинством платформ;
- Является более "легковесным" по сравнению с полноценными интегрированными средами разработки, такими как PyCharm, Eclipse, Visual Studio;
- Имеется возможность установки дополнительных плагинов для разработки на определенном языке, в том числе и для разработки на языке Python;
- Интеграция с системой контроля версий Git.

3.1.3 Выбор базы данных

Так как в данной работе применяются алгоритмы машинного обучения, то встает вопрос хранения данных, используемых для обучения классификатора. В качестве базы данных была выбрана реляционная база данных SQLite. Выбор именно этой базы данных обусловлен рядом следующих причин:

- кросс-платформенность;
- отсутствие необходимости настройки сервера СУБД, так как все данные и параметры настроек базы данных хранятся в одном единственном файле;
- высокая производительность;
- небольшой объем занимаемой памяти;
- возможность использования многих языков программирования, в том числе и Python.

3.1.4 Система контроля версий

В процессе разработки программного обеспечения использовалась система контроля версий Git. Система контроля версий позволяет вно-

сдать в проект атомарные изменения, направленные на решения каких-либо задач. В случае обнаружения ошибок или изменения требований, внесенные изменения можно отменить. Кроме того, с помощью системы контроля версий решается вопрос резервного копирования.

Выбор системы контроля версий Git обусловлен следующими ее особенностями:

- Данная система контроля версий является децентрализованной, что позволяет иметь несколько независимых резервных копий проекта;
- Поддерживается хостингом репозитория GitHub;
- Предоставляет широкие возможности для управления изменениями проекта и просмотра истории изменений.

3.2 Описание используемых библиотек

Как было сказано ранее, существует множество библиотек, написанных для языка python, которые позволяют решать множество самых разнообразных задач. В связи с этим следует привести описание основных библиотек, используемых при разработке данного программного обеспечения.

3.2.1 Библиотека для работы с базами данных

В качестве библиотеки для работы с базой данных была выбрана библиотека `peewee`. Данная библиотека является достаточно легкой и легко устанавливается с помощью пакетного менеджера `pip`. Также она реализована с поддержкой технологии ORM (англ. Object-Relational Mapping), что обеспечивает работу с данными в терминах классов языка программирования, а не в терминах таблиц данных базы SQLite.

3.2.2 Библиотека для графического интерфейса

Для написания части приложения, отвечающего за взаимодействие с пользователем, была выбрана кросс-платформенная библиотека `tkinter`. Данная библиотека позволяет создавать приложения с использованием различных графических компонентов. Также следует

заметить, что данная библиотека содержится в стандартной библиотеке языка Python, что безусловно является преимуществом, так как не требуется дополнительная её установка. Еще одним преимуществом данной библиотеки является хорошая документация с большим количеством примеров.

3.2.3 Библиотека для машинного обучения

В связи с ростом популярности технологий машинного обучения для различных языков программирования было разработано множество библиотек, тем или иным образом связанных с темой машинного обучения. К наиболее популярным библиотекам, поддерживаемыми языком Python, можно отнести Scikit-Learn, Theano, TensorFlow, Keras.

В данной работе для решения задачи классификации была выбрана библиотека Scikit-Learn. Данная библиотека предоставляет следующие возможности для решения задач классификации:

- Готовая реализация множества алгоритмов классификации: алгоритм опорных векторов, алгоритм k-ближайших соседей, алгоритм наивной байесовской классификации и др.;
- Удобный доступ к необходимым компонентам;
- Перекрестная проверка (Cross Validation): для оценки эффективности работы модели на независимых данных;
- Оптимизация параметров алгоритма (Parameter Tuning): для получения максимально эффективной отдачи от модели;
- Встроенные алгоритмы нормализации данных, для более эффективного обучения модели.

3.3 Описание классов

При разработке ПО использовался преимущественно объектно-ориентированный подход. Отношения основных классов, используемых в программе, представлены на диаграмме классов(рисунок 3.1).

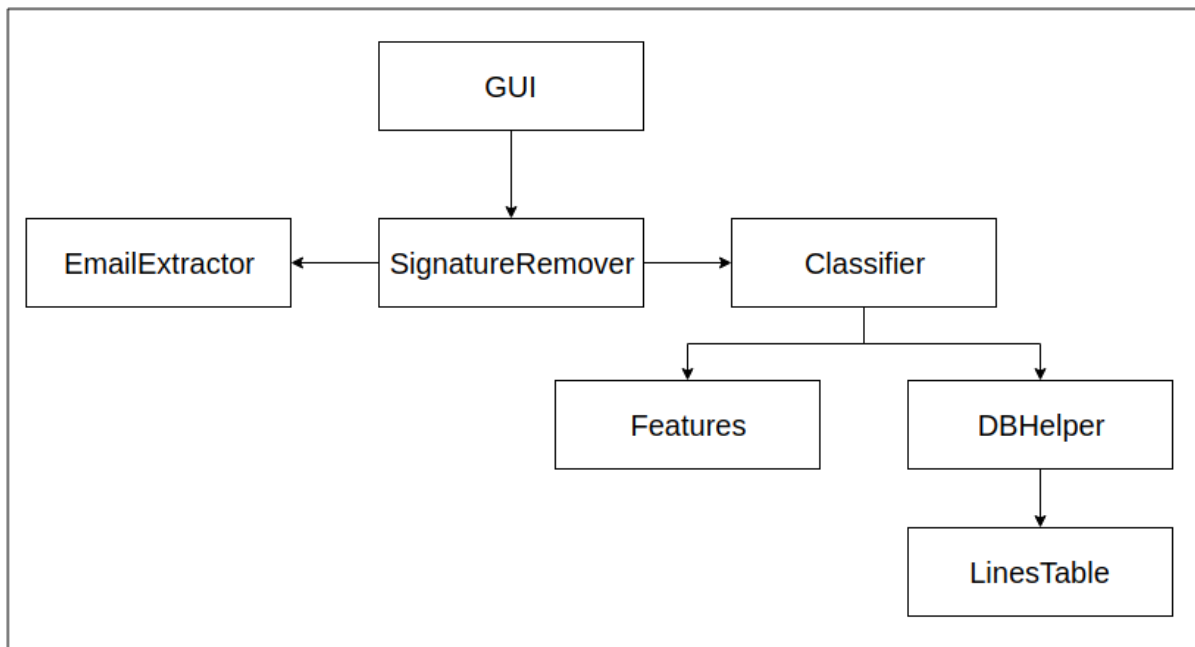


Рисунок 3.1 — Диаграмма классов

Класс **LinesTable** описывает таблицу в базе данных, которая содержит данные для обучения классификатора. Данный класс является наследником класса **Model** библиотеки **reewee**.

Класс **DBHelper** предоставляет набор методов для работы с базой данных: создание таблицы, загрузка и выгрузка данных для обучения.

Класс **Classifier** описывает методы для обучения классификатора, его сохранения в файл, а также ряд методов для классификации строк электронного письма на основе ранее обученного классификатора.

Класс **Features** описывает функционал для предварительной обработки исходных данных с целью приведения их к виду вектора признаков.

Класс **EmailExtractor** отвечает за получение текста электронного письма из почтового ящика пользователя.

Класс **GUI** является контроллером, который отвечает за пользовательский ввод и отображение информации, а так же инициализирует работу алгоритма удаления подписи.

Класс **SignatureRemover** отвечает за решение задачи удаления подписи из текста электронного письма на основе данных, полученных от пользователя, используя методы классов **EmailExtractor** и **Classifier**. Полученный результат передается классу **GUI** с целью выдачи результата пользователю.

3.4 Требования к системе пользователя

Разработанное программное обеспечение состоит из двух приложений: консольное приложение для загрузки исходных данных в базу данных и обучения классификатора, графическое приложение для демонстрации работы метода удаления подписи из текстов электронных писем. Стоит заметить, что для конечного пользователя интерес представляет только графическое приложение, которое использует файл с обученным классификатором, полученный в результате работы консольного приложения, используемого непосредственно разработчиком.

Разработанное графическое приложение способно запускаться на широком круге систем, их технические характеристики не имеют принципиального значения. К основным требованиям, необходимым для корректной работы приложения, относятся:

- Предустановленный интерпретатор Python 3.6 или более поздний;
- Файлы библиотек `scikit-learn`, `numpy`, `scipy`;
- Подключение к сети Интернет;
- Наличие у пользователя почтового ящика `gmail`.

Стоит заметить, что большинство библиотек, используемых приложением (`email`, `imaplib`, `tkinter`) не требуют дополнительной установки, так как являются частью стандартной библиотеки, устанавливаемой вместе с интерпретатором Python.

3.5 Тестирование ПО

Так как результат удаления подписи из текста электронного письма зависит от работы классификатора, то необходимо протестировать корректность функций, влияющих на его обучение. К таким функциям прежде всего следует отнести функции, представляющие строки письма в виде вектора признаков. Некорректное выделение признаков отрицательно скажется на обучении классификатора, что в дальнейшем может привести к снижению качества классификации строк письма. В связи с этим было проведено модульное тестирование функций, отвечающих за выделение признаков.

Для тестирования и оценки параметров обученной модели классификатора имеющаяся выборка была разделена на две части, одна из которых использовалась непосредственно для обучения, а другая - для тестирования полученной модели и ее оценки с использованием параметров, описанных в пункте 1.5.3.

3.6 Описание пользовательского приложения

Разработанное приложение представляет из себя окно, содержащее поля для ввода пользовательских данных (адрес почты, пароль, номер письма), поля для отображения исходного текста письма и текста письма после удаления подписи, а также кнопки "Помощь" и "Удалить подпись". Интерфейс пользовательского приложения представлен на рисунке 3.2.

Приложение предоставляет пользователю две функции: удаление подписи и вывод информации о приложении. Для того, чтобы получить информацию о приложении, пользователь должен нажать на кнопку "Помощь". Для удаления подписи из письма пользователю следует ввести адрес электронной почты, пароль и порядковый номер письма в папке "входящие" почтового ящика, а затем нажать кнопку "Удалить подпись". В случае успешной загрузки электронного письма в соответствующих полях отобразятся исходный текст письма и текст без строк, которые были классифицированы как строки, относящиеся к подписи.

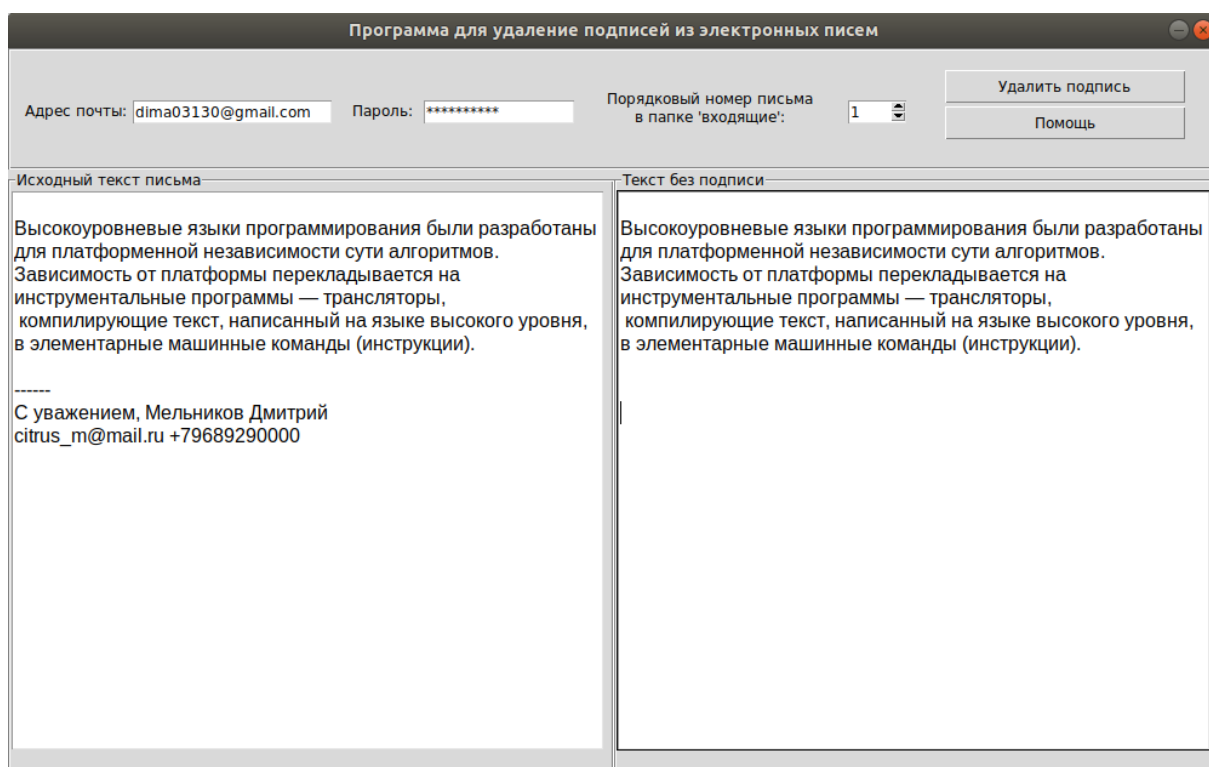


Рисунок 3.2 — Интерфейс пользовательского приложения

Также пользователь получить уведомление о том, сколько строк было удалено из текста письма.

В ходе работы программы возможно следующие ошибки и исключительные ситуации, возникающие в результате некорректных действий пользователя или по причинам, не зависящим от действий пользователя:

- Введены некорректные данные учетной записи gmail;
- Указан порядковый номер письма, превышающий общее количество писем;
- Нестабильное интернет соединение;
- Тело электронного письма представлено не в виде текста.

Все из описанных ситуаций обрабатываются программой и выдаются в виде сообщений пользователю (рисунок 3.3).

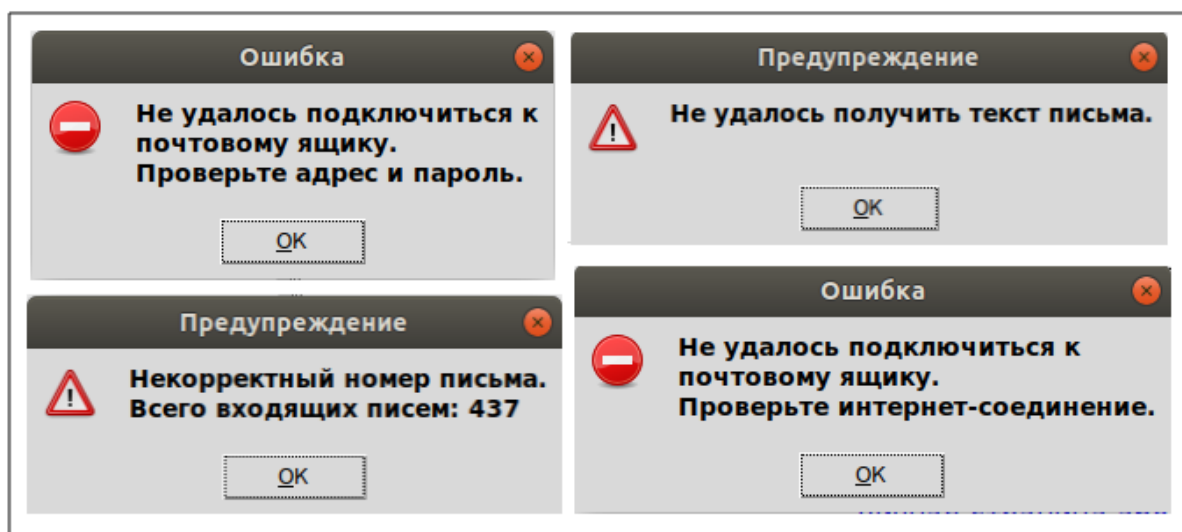


Рисунок 3.3 — Ошибки и исключительные ситуации

3.7 Выводы

В технологическом разделе проведен анализ существующих средств разработки программного обеспечения и описан выбор средств для решения поставленной задачи. Также обоснован выбор языка программирования, среды разработки и системы контроля версий в соответствии с задачами, которые стоят перед программным комплексом, и приведено краткое описание используемых библиотек. Описаны основные классы программы, а также интерфейс разработанного приложения.

4 Экспериментальный раздел

В рамках дипломного проекта был проведен ряд экспериментов, направленных на исследование описанных алгоритмов классификации с целью определения наиболее подходящего для решения поставленной задачи.

В ходе работы были рассмотрены следующие алгоритмы классификации: алгоритм опорных векторов, алгоритм k -ближайших соседей, алгоритм "наивной" Байесовской классификации. Для сравнения эффективности работы алгоритмов были использованы метрики, описанные в аналитическом разделе.

Так как для решения поставленной задачи используются алгоритмы классификации, реализованные в библиотеке Scikit-Learn, то для сравнения описанных алгоритмов между собой необходимо сначала сравнить эффективность работы алгоритмов в отдельности при различных конфигурациях параметров, предоставляемых используемой библиотекой.

4.1 Алгоритм опорных векторов

Согласно документации Scikit-Learn, конструктор класса SVC, описывающий модель классификатора с использованием алгоритма опорных векторов, принимает два основных параметра, характеризующих модель: C и $kernel$. Параметр C минимизирует величину ошибки обучения, т.е. повышение данного параметра приводит к построению более сложной, но в то же время, более корректной гиперплоскости, однако при слишком высоких значениях данного параметра возможно переобучение сети. Параметр $kernel$ определяет тип функции ядра, которая используется в случае линейной неразделимости точек в пространстве признаков. Может принимать следующие значения: 'linear', 'rbf', 'poly', 'sigmoid'.

Библиотека Scikit-Learn предоставляет метод GridSearchCV, позволяющий провести обучение при различных конфигурациях параметров модели и оценить эффективность работы с использованием метода пе-

рекрестной проверки. Метод перекрестной проверки основан на разделении обучающей выборки на k подвыборок, каждая из которых $k - 1$ раз участвует в обучении и один раз используется в качестве тестовой выборки. Таким образом модель обучается k раз на разных подмножествах обучающей выборки, и в результате для оценки используется среднее из полученных значений.

Для оценки эффективности работы алгоритма опорных векторов было проведено обучение классификатора на всех возможных комбинациях параметров $C = [0.5, 1, 1.5, \dots, 10]$ и $kernel = ['linear', 'rbf', 'poly', 'sigmoid']$.

На рисунке 4.1 представлены средние значения метрики recall(полноты) для каждой из комбинаций параметров.

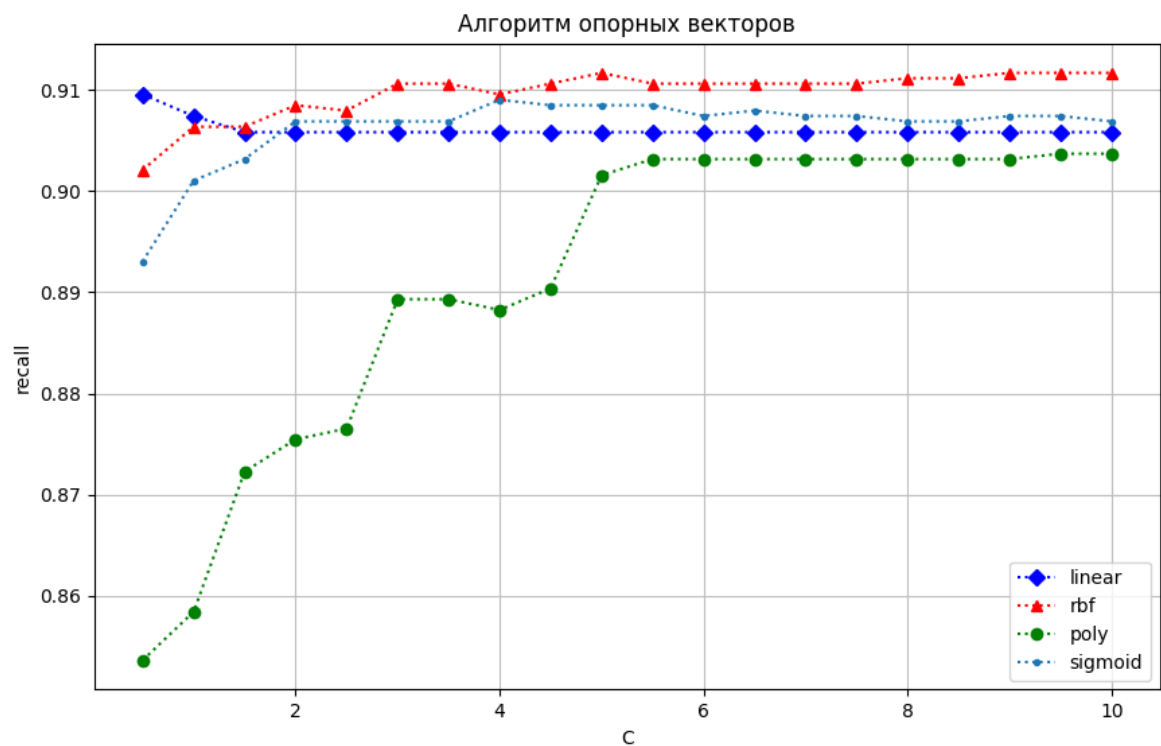


Рисунок 4.1 — Значения метрики recall при различных конфигурациях параметров алгоритма опорных векторов

На основании полученных результатов можно сделать вывод о том, что наилучшие результаты классификации достигаются при использовании следующих параметров: $C = 5$, $kernel = 'rbf'$.

4.2 Алгоритм k -ближайших соседей

Согласно документации Scikit-Learn, конструктор класса `KNeighborsClassifier`, описывающий модель классификатора с использованием алгоритма k -ближайших соседей, принимает параметр `n_neighbors`, характеризующий модель. Данный параметр описывает количество "соседей", на основании которого принимается решение о принадлежности объекта тому или иному классу.

В данном случае можно также применить метод `GridSearchCV`, чтобы определить оптимальное значение параметра `n_neighbors`, при котором классификатор будет давать наилучший результат.

На рисунке 4.2 представлен график зависимости метрики `recall`(полнота) классификатора в зависимости от значения параметра `n_neighbors`.

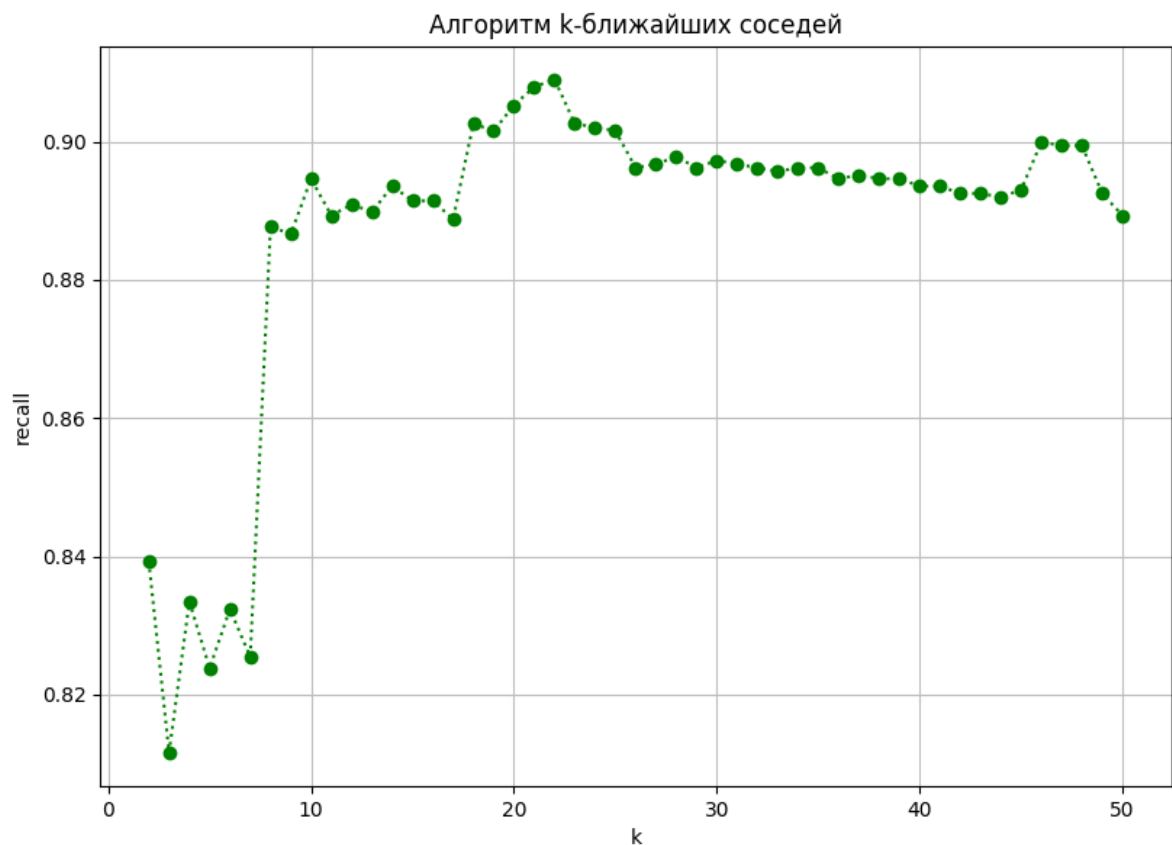


Рисунок 4.2 — Значения метрики `recall` в зависимости от значения параметра `n_neighbors`

На основании полученных результатов можно сделать вывод о том, что наилучшие результаты классификации достигаются при $n_neighbors = 22$.

4.3 Сравнение алгоритмов классификации

Ранее были проанализированы параметры алгоритмов k -ближайших соседей и опорных векторов, при которых классификатор выдает наилучший результат. Следует заметить, что при использовании алгоритма "наивной" байесовской классификации не используются какие-либо параметры, влияющие на работу классификации, соответственно нет необходимости отдельно исследовать работу данного алгоритма.

Ниже представлены результаты сравнения описанных алгоритмов классификации с использованием метрик: точность, полнота и f -мера. Все значения получены с использованием метода перекрестной проверки с разбиением исходной выборки на 5 частей. При обучении классификаторов использовались значения параметров, полученные в предыдущих разделах, за исключением "наивной" байесовской классификации, так как в данном случае такие параметры отсутствуют. Значения для точности (precision), полноты (recall) и f -меры представлены на рисунках 4.3, 4.4, 4.5 соответственно.

На основании полученных результатов, можно сделать вывод, что значения, полученные для алгоритмов опорных векторов и k -ближайших соседей, отличаются не значительно и данные являются наиболее подходящими для решения поставленной задачи по сравнению с алгоритмом "наивной" байесовской классификации.

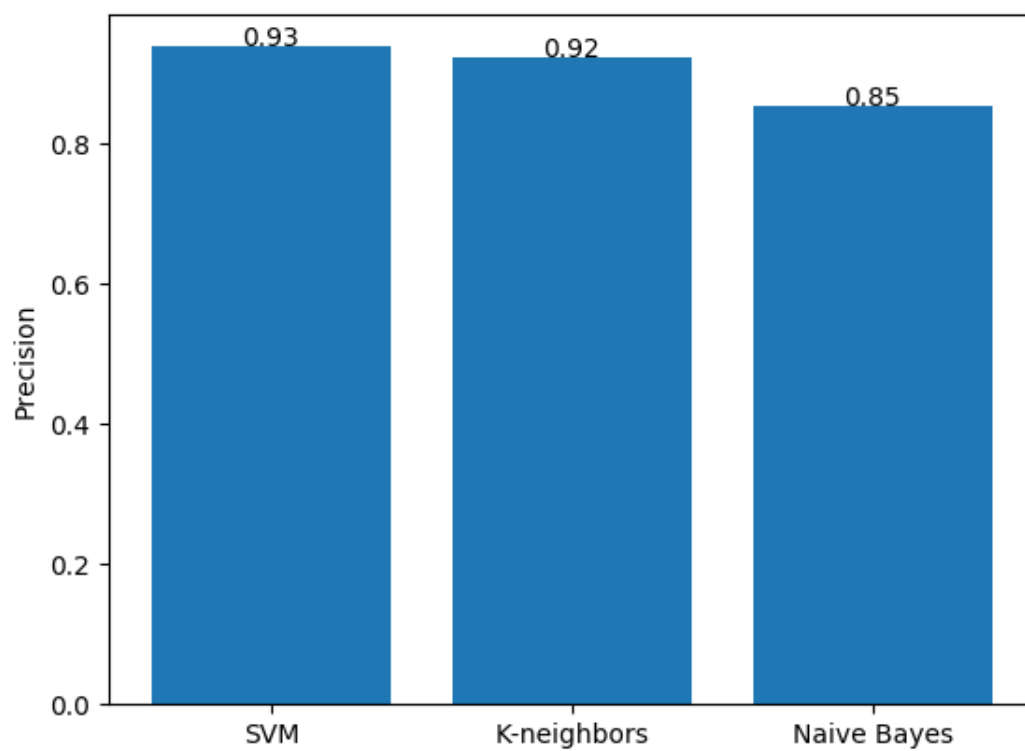


Рисунок 4.3 — Значения точности

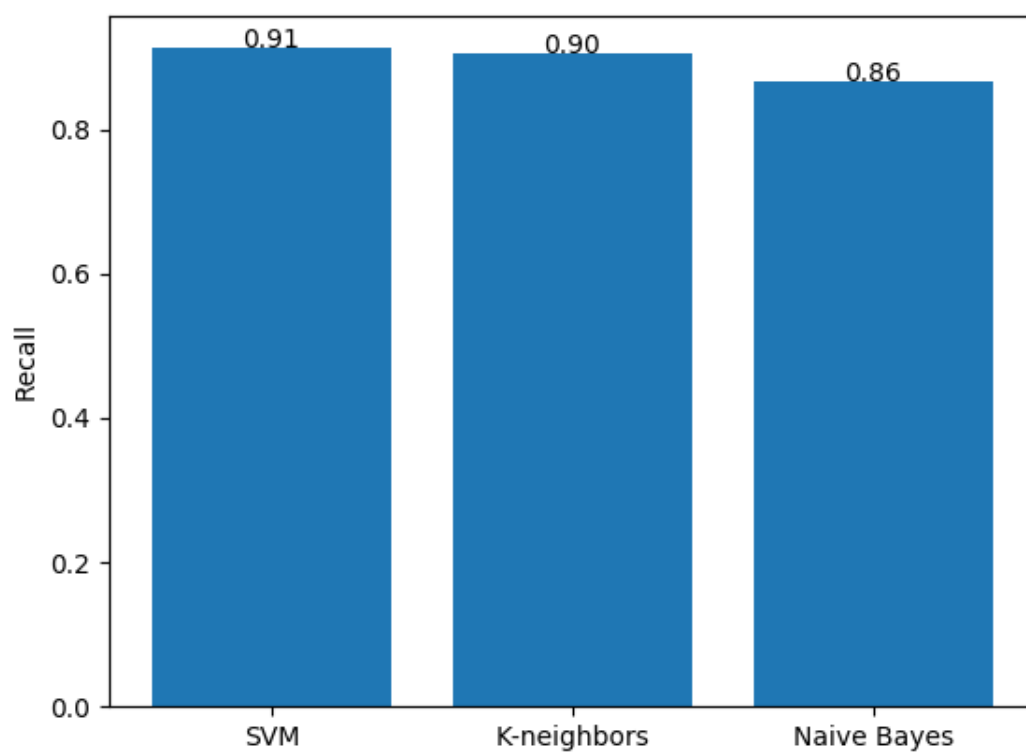


Рисунок 4.4 — Значения полноты

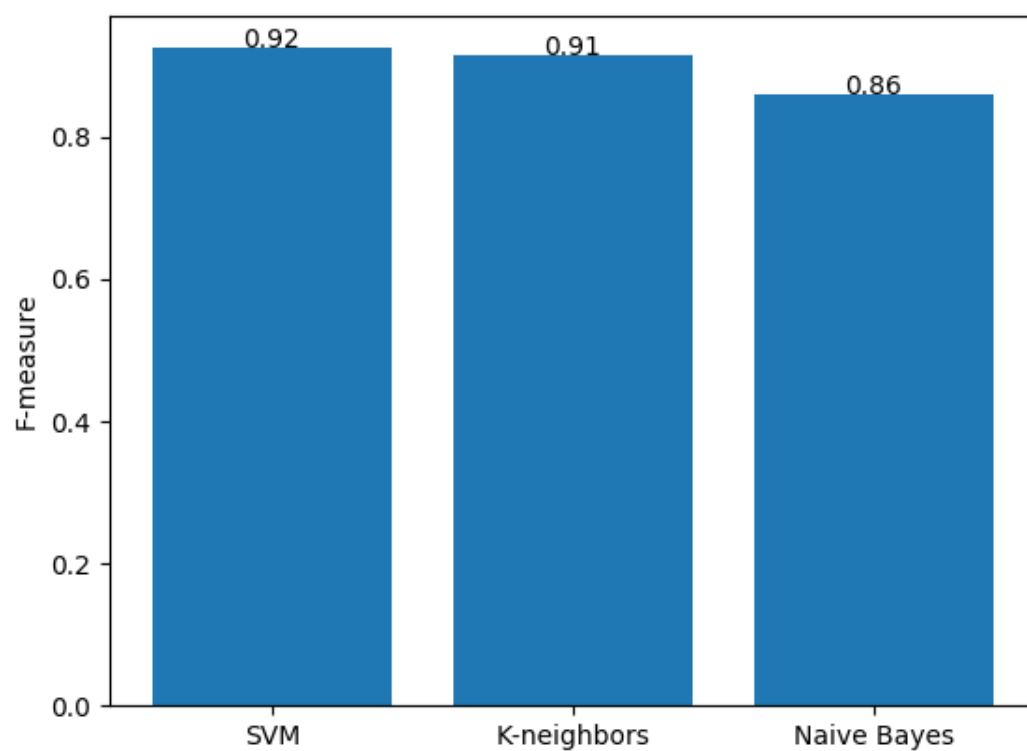


Рисунок 4.5 — Значения f-меры

4.4 Выводы

В рамках проведенных исследований каждый из описанных алгоритмов классификации был проверен при различных конфигурациях параметров, предоставляемых используемой библиотекой, в целях выявления наилучшей конфигурации параметров модели.

Затем с использованием описанных ранее метрик было проведено сравнений алгоритмов классификации между собой. Полученные результаты свидетельствуют о том, что наиболее подходящими алгоритмами для решения данной задачи являются алгоритмы опорных векторов и k -ближайших соседей.

Заключение

В рамках данной работы был разработан программный комплекс, позволяющий удалять подпись из текстов электронных писем. В ходе работы:

- проведен анализ предметной области;
- проведен анализ существующих подходов и методов решения задач, связанных с обработкой текста на естественном языке;
- разработан собственный метод на основании проанализированных;
- разработан программный комплекс, реализующий выбранный метод;
- проведен анализ эффективности работы программного продукта.

В качестве дальнейшего развития данного проекта можно выделить следующие направления:

- увеличение размера обучающей выборки и выделение новых признаков классификации для повышения качества работы классификатора;
- применение других алгоритмов классификации;
- извлечение текстов из электронных писем, имеющих тело в формате "text/html";
- извлечение различной информации об отправителе из строк, принадлежащих к подписи;

Список использованных источников

1. RFC 5322. Internet Message Format. P. Resnick. October 2008.
2. Национальная библиотека им. Н. Э. Баумана Bauman National Library [Электронный ресурс]: SMTP (Simple Mail Transfer Protocol). URL: [https://ru.bmstu.wiki/SMTP_\(Simple_Mail_Transfer_Protocol\)](https://ru.bmstu.wiki/SMTP_(Simple_Mail_Transfer_Protocol)) (дата обращения: 04.04.2018).
3. RFC 2076. Common Internet Message Headers. J. Palme. February 1997.
4. Компьютерная лингвистика [Электронный ресурс] : Материал из Википедии — свободной энциклопедии. Дата обновления: 21.12.2017. URL: <https://ru.wikipedia.org/?oldid=89782735> (дата обращения: 05.04.2018).
5. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика : учеб. пособие / Е.И. Большакова [и др.] — М.: МИЭМ, 2011. — 272 с.
6. Somers, H. Machine Translation: Latest Developments. In: The Oxford Handbook of Computational Linguistics. Mitkov R. (ed.). Oxford University Press, 2003, p. 512-528.
7. Васильев В. Г., Кривенко М. П. Методы автоматизированной обработки текстов. — М.: ИПИ РАН, 2008. - 304 с.
8. Барсегян А.А. и др. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP — 2-е изд. — СПб.: БХВ-Петербург, 2008.
9. Harabagiu, S., Moldovan D. Question Answering. In: The Oxford Handbook of Computational Linguistics. Mitkov R. (ed.). Oxford University Press, 2003, p. 560-582.
10. Grishman R. Information extraction. In: The Oxford Handbook of Computational Linguistics. Mitkov R. (ed.). Oxford University Press, 2003, p. 545-559.
11. Головкин В.А. Нейронные сети: обучение, организация и применение. Кн. 4: Учебное пособие для вузов / Общая ред. А.И.Галушкина. М.: ИПРЖР, 2001. - 256 с.

12. Борисов Е.С. Классификатор текстов на естественном языке [Электронный ресурс]. URL: <http://mechanoid.kiev.ua/neural-net-classifier-text.html> (дата обращения: 15.04.2018).
13. Ландэ Д.В. Поиск знаний в INTERNET (Пер. с англ.) - М.: Изд. дом Вильямс, 2005. - 272 с.
14. Word2vec [Электронный ресурс] : Материал из Википедии — свободной энциклопедии. Дата обновления: 11.02.2018. URL: <https://ru.wikipedia.org/?oldid=90859511> (дата обращения: 21.04.2018).
15. Катасёв А.С., Катасёва Д.В., Кирпичников А.П. Нейросетевая технология классификации электронных почтовых сообщений. / А.С.Катасёв // Вестник технологического университета. - 2015. - Т.18, №5 - С. 180-183.
16. Портал искусственного интеллекта [Электронный ресурс] : Способы нормализации переменных. Дата обновления: 26.03.2015. URL: <http://neuronus.com/theory/931-sposoby-normalizatsii-peremennyykh.html> (дата обращения: 23.04.2018).
17. Печинкин А.В., Тескин О.И., Цветкова Г.М. Теория вероятностей. 3-е изд., испр. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. — 456 с
18. Евклидова метрика [Электронный ресурс] : Материал из Википедии — свободной энциклопедии. Дата обновления: 23.02.2016. URL: <https://ru.wikipedia.org/?oldid=76661972> (дата обращения: 27.04.2018).
19. Метод множителей Лагранжа [Электронный ресурс] : Материал из Википедии — свободной энциклопедии. Дата обновления: 19.03.2018. URL: <https://ru.wikipedia.org/?oldid=91600361> (дата обращения: 27.04.2018).

20. Воронцов К.В. Лекции по методу опорных векторов. [Электронный ресурс]. URL: <http://http://www.ccas.ru/voron/download/SVM.pdf> (дата обращения: 28.04.2018).

Приложение А Регулярные выражения, используемые для выделения признаков

```
# Рег. выражение для поиска email адреса
[a-zA-Z0-9_.$%!\'=/^?~{|}+~]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-~]+

# Рег. выражение для поиска URL адреса
(https?:VV)?(da-z\.-)+\.[a-z\.-]{2,6}(\w \.-)*V?

# Рег. выражение для поиска строки-разделителя
^[ \t]*[-]{1,30}[ \t]*$

# Рег. выражение для поиска строки из спец.символов
^[ \t]*(\*|#|[\+]|[\^]|[-]|[\~]|[\&]|[\$]|_|[\!]|[\V]|[\%]|[\:]|[\=]){10,}[ \t]*$

# Рег. выражения для поиска телефонного номера
(((\+?\d){1}\s?\(? \d){3,4}\)?\s?)|(\d)\d-\d{3}\d-)|(\d){2,3}\s-?\d{2}\s-?\d{2})|
(\( ? \d{2,3} ?\)?[\.- ]{,1})\d{2,3}[\.- ]{,1}\d{2,4}

# Рег. выражение для поиска часто употребляемых слов
(T|t)hank.*,(B|b)est|(R|r)egards| ^sent[ ]{1}from[ ]{1}my[\s,!w]*$|BR|(S|s)incerely|
(C|c)orporation|Group|(O|o)тправлено\sc*|(C|c)\s(Y|y)важением|
(Y|y)дачного|(X|x)орошего|(B|b)ам|доброго|(C|c)пасибо
```

Рисунок А.1 — Используемые регулярные выражения