Оглавление

Введение

Реклама – это часть маркетинговых коммуникаций, в рамках которой производится распространение не персонализированной информации, с целью привлечения внимания к объекту рекламирования.

В связи с развитием средств телекоммуникаций процесс распространения рекламы охватил телевидение, радио и глобальную сеть интернет. Благодаря снижению цен на отправку текстовых сообщений за последние несколько лет увеличилось внимание компаний к смс-рассылке, наблюдается резкий рост доли спам сообщений среди всех получаемых абонентами смс. Зачастую люди не подписываются на такую рассылку, но тем не менее получают ее.

Распространение рекламы без подписки абонента неправомерно. Операторы связи имеют возможность не пропускать спам-рассылки, разрешаются те рассылки, на получение которых абонент давал согласие. Для этого создается механизм обратной связи – при получении нежелательной информации у абонента будет возможность оповестить оператора об этом и заблокировать номер отправителя. Тем не менее такие методы не обеспечивают полную фильтрацию рекламной рассылки, а для отказа от ее получения, требуется время и дополнительные трудозатраты.

На рынке программных продуктов существуют решения по обнаружению спам сообщений, преимущественная реализация для e-mail писем. Следует отметить, что основные разработки ведутся для анализа английского языка, поэтому распознавание спама на русском языке является актуальным. В большинстве случаев ограничиваются использованием «черных списков» и не прибегают к методам классификации на основе анализа текста смс сообщений.

Целью квалификационной бакалаврской работы является разработка методов фильтрации коротких сообщений, несущих рекламный характер.

Разрабатываемый продукт должен быть адаптирован к русскому языку, а также реализовывать обработку сообщения посредством анализа его текста.

1. Аналитический раздел

1.1 Постановка задачи

Система фильтрации спам-сообщений должна отвечать следующим требованиям:

- определять является ли анализируемый кроткий текст спамом;
- минимизировать ошибки классификации коротких сообщений;
- выполнять анализ текста за гарантированное время ответа.

Для решения поставленной задачи необходимо:

- проанализировать существующие аналоги;
- сформировать базу спам-сообщений;
- на основе анализа базы спам-сообщений разработать систему признаков спама;
- проанализировать и выбрать методы анализа коротких сообщений для выявления спама;
- на основе выбранных методов разработать алгоритмы выявления спама;
- разработать структуру системы;
- провести тестирование системы на широком классе спам-сообщений.

1.2 Анализ существующих решений

Фильтрация спам в e-mail рассылке

Для фильтрации спама в почтовых ящиках существует множество решений, поскольку задача распознавания e-mail спама является актуальной в настоящее время. Некоторые почтовые службы используют свои внутренние программные продукты, нацеленные на борьбу со спамом. Например, известная почтовая служба компании Mail.ru [1] собирает различные метрики спам-сообщений, в реализации внутренних алгоритмов распознавания спама используются статистические методы и деревья решений, кроме того дополнительно используется анти-спам компании KasperskyLab [2].

Существуют и плагины для почтовых клиентов, например плагин AGAVA Spampotexx [3] для почтового клиента The Bat!. Данный плагин использует фильтрацию на основе «черных» и «белых» списков писем с дополнительными ограничениями и проверками, а также основывается на модифицированном байесовском классификаторе: использует нелинейную функцию весовых коэффициентов. AGAVA Spampotexx не рассматривает стоп-слова, обладает своим словарем стоп-слов. Плагин избегает переобучения (когда для обучения представляются примеры одного спам-класса, что ведет к увеличению весов слов текста) предварительно классифицируя сообщение. В случае удачной классификации, обучения на этом сообщении не проводится.

К преимуществам перечисленных реализаций можно отнести:

- использование большого объема обучающей выборки, что позволяет повысить точность классификации;
- усложненные алгоритмы известных классификаторов, усложняющее составление спама таким образом, чтобы он прошел проверку

К недостаткам реализаций можно отнести:

- накладные расходы на ресурсы
- платность приложений
- использование спам-фильтров только в определенных существующих решениях
- в случае пробной или бесплатной версии необходимо просматривать рекламу
- ориентация в основном на анализ больших текстов
- опора на анализ полей e-mail сообщения, а также html-инъекций в теле электронного письма

Таким образом, спам-фильтры email сообщений не адаптированы для анализа коротких текстовых сообщений.

Фильтрация спама в приложениях под ОС Android

Фильтр SMS, Clear Inbox - бесплатные приложения под ОС Android, которые позволяют фильтровать спам только путем блокирования номеров телефонов, созданием фильтров (по вхождению слов или наборов специальных символов). Данные приложения обладают самым высоким рейтингом среди приложений своего класса задач, но некоторые пользователи отмечают плохую работу приложений.

SMS SpamBlocker также является бесплатным приложением под ОС Android, согласно описанию характеристик приложения на GooglePlay данный спам-фильтр обладает встроенным модулем распознавания спама. Не указывается какие методы распознавания используются в SMS Spam Blocker. Согласно отзывам русскоязычных пользователей во многих случаях не блокирует сообщения, в большинстве своем основывается на «белом» списке.

В качестве преимуществ перечисленных приложений можно отметить бесплатное использование и гибкие настройки «черных» и «белых» списков. К недостаткам Фильтр SMS и Clear Inbox относится отсутствие анализа спама по тексту сообщения, а SMS SpamBlocker плохое распознавание спама.

1.3 Анализ и выбор алгоритмов подготовки текста для автоматической обработки

1.3.1 Разбиение текста на смысловые элементы

Разбиение текста на смысловые элементы (токенизация) — процесс разбиения текста на слова, фразы, символы или другие значимые элементы, называемые токенами. Разбиение текста на слова в самом простом подходе осуществляется по пробельным символам и пунктуационным знакам. Но в некоторых случаях такое удаление влечет последующую некорректную обработку. Например, разделение дефисом составного имени компании приведет к образованию двух слов и потере смыслового элемента — названия компании или разбиение по пробельному символу фразеологизма приводит к потере смысловой нагрузки токена.

В результате токенизации уменьшается размерность исходного текса, и происходит утрата некоторых его характеристик:

- связь слов в предложении
- эмоциональность (которую частично можно определить по знакам препинания)
- смысла (обеспечиваемого специальными символами в зависимости от контекста)

Как правило после токенизации игнорируются регистры слов, что тоже сужает отображение исходного текста в пространство токенов.

Одно и тоже слово может появляться в текстах в своих словоформах, но все словоформы имеют близкую смысловую нагрузку и поэтому разумно приводить слова к нормальным (начальным) формам, кроме того нормализованные слова легче поддаются последующей машинной обработке и классификации.

Следует отметить проблему написания слов с грамматическими ошибками – даже при правильном разбиении текста на слова грамматические ошибки не позволят правильно осуществить последующую нормализацию. Вероятность возникновения опечатки при наборе смс значительно выше, чем в художественном тексте, прошедшем редакционно-издательскую обработку, или рекламе. Абонент может допускать ошибки случайно или же специально искажать слова в целях экономии средств при отправке смс.

1.3.2 Выявление неизменной части слова

Выявление неизменной части слова связано с отсечением от слова изменяемых частей. Такой процесс принято называть стеммингом (от англ. Stemming).

Стемминг основывается на правилах отсечения для каждого конкретного языка. Для некоторых языков он не применим. Как правило, применение стемминга по формообразующим морфемам дает большую точность чем по словообразующим морфемам. Русский язык относится к группе флективных синтетических языков, т.е. языков, в которых доминирует словообразование с использованием аффиксов (аффикс – это суффикс, которому может соответствовать несколько значений), поэтому можно применить алгоритмы стемминга для нормализации текста.

Алгоритм Мартина Портера

Одним из самых простых алгоритмов стеммига является алгоритм Мартина Портера. На каждом шаге отсекается формо или словообразующий

суффикс и оставшаяся часть проверяется на соответствие правилам. Если полученное слово удовлетворяет правилам, происходит переход на следующий шаг. Если
нет – алгоритм выбирает другой суффикс для отсечения. Сначала отсекаются наиболее длинные суффиксы, далее более простые.

Преимуществом данного подхода является быстрота и легкость реализации, алгоритм не требует использования баз данных норм слов. В качестве недостатка можно отметить не высокую точность правильного образования нормальных форм. Например, алгоритм может отсечь исходя из правил больше, чем нужно, он также не справится со случаем выпадающей и беглой гласных в слове.

Алгоритм Андрея Коваленко

Другой алгоритм стемминга был предложен в программной реализации stemka Андреем Коваленко. Он использует вероятностную модель: происходит разбиение слов из обучающего текста на пары «последние две буквы основы» + «суффикс» и, если такая пара уже присутствует в модели, увеличивается ее вес, иначе она добавляется в модель. После чего полученный массив данных ранжируется по убыванию веса и модели, вероятность реализации которых составляет менее 1/10000, отсекаются. В конце этапа сбора статистических данных выявляется набор потенциальных окончаний с условиями на предшествующие символы, который инвертируется для удобства сканирования словоформ "справа налево" и представляется в виде таблицы переходов конечного автомата.

Алгоритм возвращает набор из нескольких стемм, удовлетворяющих таблицам.

В отличие от подхода Портера только что рассмотренный алгоритм обладает большей точностью в задаче стемминга, возвращает варианты стемм, но требует предварительного сбора данных, их обработки для формирования весов.

Алгоритм Ильи Сегалович

Данный алгоритм был предложен Ильёй Сегаловичем и реализован в программе Mystem. В алгоритме строится лес инвертированных префиксных деревьев суффиксов и инвертированного префиксного дерева для основ, для этого используется словарь с перечислением всех грамматических форм слова.

На первом шаге при помощи дерева суффиксов во входном слове определяются возможные границы между стеммой и суффиксом, после чего для каждой потенциальной стеммы (начиная с самой длинной) бинарным поиском по дереву основ проверяется ее наличие в словаре либо нахождение наиболее близких к ней основ. В качестве меры близости рассматривается длина общего «хвоста». Если слово словарное – алгоритм заканчивает работу, иначе – переходит к следующему разбиению. Если вариант основы не совпадает ни с одной из «ближайших» словарных основ, то это означает, что анализируемое слово с данным вариантом основы в словаре отсутствует. Тогда по имеющейся основе, суффиксу и модели «ближайшей» словарной основы генерируется гипотетическая модель изменения данного слова. Гипотеза запоминается, а если она уже была построена ранее – увеличивает свой вес. Если слово так и не было найдено в словаре – длина требуемого общего окончания основ уменьшается на единицу, идет просмотр дерева на предмет новых гипотез. Когда длина общего «хвоста» достигает двух, поиск останавливается и идет ранжирование гипотез по продуктивности: если вес гипотезы в пять и более раз меньше самого большого веса – такая гипотеза отсеивается.

Результатом работы стеммера является получившийся набор гипотез для несуществующего или одна гипотеза для словарного слова. Для каждого варианта начальной формы реализация алгоритма Mystem предлагает всю грамматическую информацию, в случае не существующего в словаре слова

информация синтезируется, и частоту его использования в IPM (instances per million).

Преимуществом данного алгоритма является хорошая точность нормализации слов, а также есть этап синтеза стемм слов.

1.3.3 Нормализация

Нормализация (леммитизация)— процесс приведения словоформы к лемме — её нормальной (словарной) форме. Нормализация может применяться к результатам алгоритмов стемминга путем образования нормальной формы слова с использованием словаря суффиксов и окончаний для основ.

Другой алгоритм нормализации — это поиск по словарю словоформ и соответствующих им нормальных форм слов. Недостаток метода в том, что необходимо обладать таким словарем, а также нужно обеспечить добавление новых словоформ и норм слов.

Согласно закону Хипса, который гласит, что число различных слов в тексте растет примерно как корень квадратный из количества всех слов текста, скорость появления новых слов определяет необходимое требование постоянного пополнения словаря.

$$v(n) = \alpha n^{\beta}\ , \qquad (1.1)$$

где v - это объем словаря уникальных слов, составленный из текста, который состоит из n уникальных слов, α и β - определенные эмпирически параметры. Для европейских языков α принимает значение от 10 до 100, а β от 0.4 до 0.6.

В рамках исследуемой темы размеры текстов малы, поэтому общий прирост новых слов будет не большой. Но с течением времени количество не словарных слов будет увеличиваться, поэтому рекомендуется обеспечить возможность пополнения словаря.

1.3.4 Анализ возможности потери информации при автоматической обработке текста

Процесс автоматической обработки текста на русском языке влечет ошибки, связанные с омонимичностью (омонимы - слова, совпадающие одновременно по звучанию и по написанию, но различные по значению), омоформами (омоформы - это разные слова, совпадающие в отдельных грамматических формах. Например, опасно совпадение нормальных форм слов), омографами (омографы - слова, совпадающие по написанию, но различные по звучанию и значению — различные ударения). Последние ошибки устранить крайне сложно, так как при восприятии текста человек сам определяет нужное ударение слова исходя из смысловой нагрузки сообщения.

При нормализации происходит потеря информации о части речи, числе, форме слова, эмоциональной окраске. Для сохранения утрачиваемой информации применяют теги к каждому слову, но при этом скорость обработки текста снижается.

В качестве средств по улучшению качества дальнейшей обработки слов на этапе нормализации в работе предлагается использовать разбиение множества слов на группы по признаку синонимии, а каждой такой группе назначить «представителя» - слово в начальной форме. Данное преобразование уменьшит мощность анализируемого множества нормальных форм слов, что в свою очередь увеличит скорость обработки текста алгоритмами классификации.

1.3.5 Выбор метода нормализации слов текста

В качестве метода нормализации текста был выбран метод, основанный на использовании тезауруса. Согласно исследованиям [] размер активного словаря среднестатистического гражданина России равен 1,5 тыс слов, а пассивного 6 тыс слов.

Данный подход зависит от мощности словаря словоформ. Существует несколько вариантов словарей [перечислить]. Среди них словарь Зализняка выгодно отличается тем, что [], .Словарь Зализняка содержит 1696633 форм слова и 155090 уникальных нормальных форм, что вполне приемлимо в рамках решаемой задачи.

1.4 Анализ методов классификация текста и выбор метода выявления спама

1.4.1 Классификаторы на основе правил

В данном подходе разрабатываются правила, по которым определяется является ли текстовое сообщение спамом или нет. Недостатки подхода:

- можно определить набор правил и составить спам сообщение так, чтобы их обойти;
- правила не покрывают случаи, когда абонент специально использовал те или иные символы, и сообщение классифицируется как спам;
- не учитывается, что признаки спама меняются в зависимости от аудитории, которой осуществляется рассылка — ключевые слова в одном случае могут расцениваться как спам, в другом случае спамом являться не будут.

В качестве преимуществ можно выделить простоту реализации и скорость работы.

1.4.2 Наивный байесовский классификатор

Данный классификатор опирается на теорему Байеса, описывающую соотношение между условными вероятностями.

$$P(A \mid B) = P(B \mid A) \times P(A) / P(B) \quad (1.2)$$

Определяется два класса, к которым может принадлежать текст — спам и не спам. Выделяются так же характеристики (признаки) спама.

Тогда вероятность того, что при наличии в тексте признака i сообщение относится к классу j:

$$P(\text{Класс j}| \text{Признак i}) = P(\text{Признак i}| \text{Класс j}) \times P(\text{Класс j}) / P(\text{Признак i}) \quad (1.3)$$

Вероятность того, что сообщение относится к классу j:

P(Класс j | Признак1, .. Признак N) = P(Класс j) × P(Признак1,..Признак N| Класс j) / P(Признак1, .. Признак N) (1.4)

P(Признак1,..Признак N| Класс j) = P(Признак1| Класс j) × P(Признак2,..Признак N| Класс j, Признак1) = .. = P(Признак1| Класс j) × P(Признак2,..Признак N| Класс j, Признак1) × .. × P(Признак N| Класс j, Признак1,..ПризнакN-1) (1.5)

В Байесовском классификаторе делается упрощающее априорное предположение о том, что признаки независимы между собой, поэтому:

P(Признак1,..Признак N| Класс j) = P(Класс j) × P(Признак1| Класс j) × .. × P(Признак N| Класс j) (1.6)

Далее составляют отношение правдоподобия:

$$P(\text{Спам} \mid \text{Текст}) / P(\text{Не спам} \mid \text{текст}) \quad (1.7)$$

или логарифмического правдоподобия:

$$\ln (P(\text{Спам} \mid \text{Текст}) / P(\text{Не спам} \mid \text{текст})) \quad (1.8)$$

и сравнивают его с некоторым пороговым значением.

Минусы байесовского классификатора:

- проблема анализа редких слов;
- если слово никогда не встречалось во время фазы обучения — числитель и знаменатель равны в формуле 1.2. Простое решение состоит в том, чтобы игнорировать такие редкие слова т.е. можно выделить ядро слов и увеличить обучающую выборку. Кроме того, для улучшения точности классификации можно использовать словосочетания вместо изолированных слов естественных языков. Этот метод более чувствителен к контексту и устраняет байесовский шум лучше за счет большей базы данных;
- априорное предположение о том, что слова не взаимосвязаны;
- базируется на предположении, что одни слова встречаются в спам-сообщениях чаще чем в других. Существует метод «Байесова отравления», позволяющий добавить много лишнего текста, иногда тщательно подобранного, чтобы «обмануть» фильтр.

Плюсы байесовского классификатора:

- прост в реализации;
- удобен (позволяет обходиться без «черных списков» и подобных искусственных приемов);

- эффективен (после обучения на достаточно большой выборке отсекает до 95—97 % спама, и в случае любых ошибок его можно дообучать).

В исследовании [5] исходный массив данных представлен текстами спам смс на английском языке. Далее каждое сообщение проходит через процедуру токенизации, разбиение по пробельным символам. Токенами считаются слова, символ «$», учитываются такие характеристики как количество чисел в тексте и общее число символов в сообщении, каждый токен рассматривается как отдельная характеристика.

В качестве обучающего множество берется 70 % всех спам писем, остальные 30 % используются как тестовые данные. Согласно результатам исследования 1,5 % - ложно негативное срабатывание, 93 % всех спам сообщений обнаружено и 0,74 % ложно положительное срабатывание. Делается вывод о том, что добавление новых характеристик спама уменьшит количество ошибок. Добавляются такие характеристики как:

- длина сообщения <= 40, <= 60, <=80, <=120, <=160
- наборы символов «://» и «@» также считаются характеристиками сообщения на предмет наличия адреса электронной почты и адреса веб-сайта в Интернете.

1.4.3 Метод Фишера

В отличие от Байесовского классификатора в методе Фишера вычисление ведется по следующей формуле:

$$P(\text{Категория j}|\ \text{Признак i}) = \text{Ndocj}(\text{Признак i}) / \text{Ndoc}, \quad (1.9)$$

где

Ndocj(Признак i) - количество документов с данным признаком в этой категории,

Ndoc - общее количество документов с данным признаком

В формуле 1.9 делается предположение, что для всех категорий представлено одинаковое число документов.

Полную вероятность можно получить простым перемножением, но по методу Фишера необходимо перемножить P(Категория j| Признак i) по всем признакам при зафиксированной категории, а после взять натуральный логарифм и умножить результат на -2.

$$P(\text{Категория } j|\ \text{Признак } 1,..\text{Признак } N) = -2 \times \ln(P(\text{Категория } j|\ \text{Признак } 1) \times .. \times P(\text{Категория } j|\ \text{Признак } N)) \quad (1.10)$$

Фишер доказал, что если бы вероятности были независимы и случайны, то результат этого вычисления подчинялся бы распределению хи-квадрат. Тогда образец, не принадлежащий некоторой категории, будет содержать слова с различными вероятностями признаков для данной категории (их распределение было бы случайным), а в образце, принадлежащем категории, будет много признаков с высокими вероятностями.

Таким образом при подачи результата вычисления по формуле 1.10 на вход обратной функции хи-квадрат, получается вероятность того, что случайный набор вероятностей вернет такое большое число.

1.4.4 Метод опорных векторов

Метод опорных векторов так же известный как SVM — support vector machine - является бинарным линейным классификатором.

Математическая постановка задачи следующая: имеется вектор x характеристик объекта из пространства всех характеристик объектов X, также имеется множество ответов Y. Необходимо построить аппроксимирующую целевую зависимость a: $X \rightarrow Y$ на всем пространстве X.

Предполагается так же, что имеется некоторое обучающее множество — набор пар (x, y) — характеристик объекта пространства X и соответствующих ответов из пространства Y.

В качестве вектора x в задачи фильтрации спама используется вектор выделенных характеристик текста, заданный в пространстве Rn.

Множество ответов Y состоит из двух классов — спам и не спам.

Зависимость a изначально имеет вид:

a(x) = sign (sum<wj x j> − w0) = sign (<w x> − w0) , (1.11)

где x = (x1 , . . . , xn) — признаковое описание объекта x;

вектор w = (w1 , . . . , wn) ∈ Rn

и скалярный порог w0 ∈ R являются параметрами алгоритма.

Уравнение (w, x) = w0 описывает гиперплоскость, разделяющую классы в пространстве Rn.

Случай линейной разделимости

Если выборка x линейно разделима, то разделяющая гиперплоскость не единственная. Необходимо, чтобы разделяющая гиперплоскость максимально отстояла от ближайшей к ней точек обоих классов.

Точки из обучающей выборки не могут лежать внутри этой полосы. Границами полосы служат две параллельные гиперплоскости с направляющим вектором w. Точки, ближайшие к разделяющей гиперплоскости, лежат в точности на границах полосы. При этом сама разделяющая гиперплоскость проходит ровно по середине полосы. Для того, чтобы разделяющая гиперплоскость находилась максимально удаленно от точек выборки ширина полосы должна быть максимальна.

В случае разделимой выборки построение оптимальной разделяющей гиперплоскости сводится к минимизации квадратичной формы при ограничениях-неравенствах относительно n + 1 переменных w, w0. Решение этой системы сводится к решению двойственной задачи поиска седловой

функции Лагранжа. В результате вектор весов w является линейной комбинацией векторов обучающей выборки, причем только тех, для которых соответствующая компонента λ двойственных переменных не равна нулю. Опорный вектор — это объект обучающей выборки xi, для которого (w, xi) - w0 = yi, а λi > 0. В результате решения задачи квадратичного программирования находится вектор w и w0.

Таким образом, алгоритм классификации может быть записан в следующем виде:

a(x) = sign (sum (λi * yi (xi , x) − w0) (1.12)

Случай линейно неразделимой выборки

Для обобщения алгоритма SVM на случай линейно неразделимой выборки вводится разрешение классификатору допускать ошибки, при условии минимизации ошибок. Вводится также штраф за приближение объекта к границе классов, после задача сводится к поиску седловой точки функции Лагранжа, что приводит к образованию деления объектов на следующие типы:

1) λi = 0, ni = C, qi = 0, mi > 1 (ni — переменные двойственные к ошибкам qi, C - управляющий параметр метода, позволяющий находить компромисс между максимизацией разделяющей полосы и и минимизации суммарной ошибки).

Объект xi классифицируется правильно и находится далеко от разделяющей полосы. Такие объекты называются периферийными.

2) 0 < λi < C, 0 < ni < C, qi = 0, mi = 1

Объект xi классифицируется правильно и лежит в точности на границе разделяющей полосы. Такие объекты называются опорными.

3) λi = C, ni = 0, qi > 0, mi < 1

Объект xi либо лежит внутри разделяющей полосы, но классифицируется правильно, либо попадает на границу классов, либо относится к чужому классу. Такие объекты называются нарушителями.

Как правило, гарантировать условие линейной разделимости оказывается весьма сложно, а решение задачи в условиях линейной неразделимости более трудоемко. Поэтому используют приведение исходного пространства с большей размерностью, что увеличивает вероятность того, что в этом пространстве выборка окажется линейно разделимой.

Спрямляющие пространства

Пространство H называется спрямляющим, если оно получено путем некоторого преобразования из исходного пространства с меньшей размерностью. В новом пространстве H используется метод классификации на линейно разделимой выборке с той лишь разницей, что скалярные произведение векторов переопределяются на скалярное произведение в пространстве H т. е. требование к пространству H — наличие скалярного произведения.

Скалярное произведение в свою очередь формально заменяют ядром. Функция $K : X \times X \to R$ называется ядром (kernel function), если она представима в виде

$$K(x, x') = (\psi(x), \psi(x')) \qquad (1.13)$$

при некотором отображении ψ: X -> H, где H — пространство со скалярным произведением. Это позволяет не строить само пространство H, а подбирать непосредственно само ядро.

Существует несколько «стандартных» ядер, которые приводят к уже известным алгоритмам: полиномиальным разделяющим поверхностям, двухслойным нейронным сетям, потенциальным функциям и другим. Но до сих пор не найдено эффективного общего подхода к их подбору в конкретных задачах.

Преимущества SVM

- Принцип оптимальной разделяющей гиперплоскости приводит к максимизации ширины разделяющей полосы между классами, следовательно, к более уверенной классификации. Градиентные нейросетевые методы выбирают положение разделяющей гиперплоскости произвольным образом, «как придётся».
- дает хорошие результаты в задачах классификации [4];
- показывает хорошие результаты при активном обучении. Активное обучение — это процесс выборки наиболее подходящего примера класса и добавление его в обучающее множество для дообучения.

Недостатки SVM

- неустойчив по отношению к шуму в исходных данных.
- не разработаны общие методы построения спрямляющих пространств или ядер, наиболее подходящих для конкретной задачи.
- в общем случае, когда линейная разделимость не гарантируется, приходится подбирать управляющий параметр алгоритма C.

В работе [5] исследуется точность классификации SVM в зависимости от выбранного ядра. Типы ядер:

- Полиномиальное
  - Однородное $k(x,y) = (x \cdot y)^d$ (1.14)
  - Неоднородное $k(x,\ y) = (x \cdot y + \mathrm{C})^d$ (1.15)
- Радиальное базисное

$$k(x,y) = \exp\left(-\gamma\left|\left|x-y\right|\right|^2\right) \text{ при } \gamma > 0 \quad (1.16)$$

- Сигмоид

$$k\ (x,y) = \tanh(kx \cdot y + C) \quad (1.17)$$

Результаты в таблице 1.1.

Таблица 1.1 Результаты работы SVM с различными ядрами

Тип классификации	Ошибка	Пойманный спам (в %)	Ложно позитивное срабатывание (в %)
Линейное разделение	1.18	93.8	0.47
Ядро: Полином 2ой степени	2.03	85.7	0.27
Ядро: Полином 3ей степени	1.64	89.7	0.4
Ядро: Полином 4ой степени	1.7	90.5	0.6
Ядро: Радиальная функция	2.61	81.4	0.32
Ядро: Сигмоид	13.4	0	0

Согласно таблице наилучшие результаты достигаются при линейном ядре. В случае полиноминального ядра уменьшается число ошибок при увеличении степени полинома до двух, но при дальнейшем увеличении уменьшения ошибок не происходит. Так же рассматривается и сигмоид в качестве функции ядра. Наилучшие результаты SVM достигает при предположении линейной разделимости исходного множества.

1.4.5 k-ближайших соседей (k-NN — k-nearest neighbor)

Метод k-ближайших соседей используется для создания автоматических классификаторов. Основная идея метода — это находить в массиве текстов с выделенными признаками самые похожие на анализируемый текст и на основе знаний об категориальной принадлежности уже имеющихся текстов классифицировать неизвестное сообщение.

При классификации неизвестного документа находится заранее заданное число k текстов из обучающей выборки, которые в пространстве признаков расположены к ближе всего. Принадлежность текстов к распознаваемым классам считается известной. Параметр k обычно выбирают от 1 до 100. Близость классифицируемого документа и документа, принадлежащего категории, определяется как косинус угла между их векторами признаков. Чем значение ближе к 1, тем документы больше друг на друга похожи.

Решение об отнесении документа к тому или иному классу принимается на основе анализа информации о принадлежности k его ближайших соседей.

Главной особенностью, выделяющей метод k-NN среди остальных, является отсутствие у этого алгоритма стадии обучения.

Основным преимуществом такого подхода является возможность обновлять обучающую выборку без переобучения классификатора. Классический алгоритм предлагает сравнивать анализируемый документ со всеми документами из обучающей выборки и поэтому главный недостаток метода k-ближайших соседей заключается в длительности времени работы рубрикатора на этапе классификации.

В таблице 1.2 приведены результаты анализа k-NN, проведенные в исследовании [5].

Таблица 1.2 Результат работы классификации методом k-NN

Количество соседей	Ошибка (в %)	Пойманный спам (в %)	Ложно позитивное срабатывание (в %)
2	2.78	81.3	0.46
10	2.53	82.6	0.4
20	2.98	78.8	0.35
50	3.4	74.8	0.24
100	4.14	68.4	0.16

1.4.6 Дерево принятия решений

Дерево принятия решений используется в области статистики и анализа данных для прогнозных моделей, в том числе используются и в задачах классификаций. Дерево состоит из «листей» и «веток». На ребрах («ветках») дерева решения записаны атрибуты, от которых зависит целевая функция, в «листьях» записаны значения целевой функции, а в остальных узлах— атрибуты, по которым различаются случаи. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение. Цель состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной на основе нескольких переменных на входе.

1.4.7 Случайный лес (Random forest) и AdaBoost

Случайный лес (Random forest)— алгоритм машинного обучения, предложенный Лео Брейманом и Адель Картер , заключающийся в использовании комитета (ансамбля) решающих деревьев.

AdaBoost (сокращение от Adaptive Boosting)— алгоритм усиления классификаторов, путем объединения их в комитет. AdaBoost является адаптивным: каждый следующий комитет классификаторов строится по объектам, неверно классифицированным предыдущими комитетами.

AdaBoost вызывает слабые классификаторы в цикле. После каждого вызова обновляется распределение весов, которые отвечают важности каждого из объектов обучающего множества для классификации. На каждой итерации веса каждого неверно классифицированного объекта возрастают, таким образом новый комитет классификаторов «фокусирует своё внимание» на этих объектах.

1.4.8 Выбор алгоритмов классификации текста

В таблице 1.3 представлены результаты работы алгоритмов по нескольким параметрам: процентное соотношение ошибок, допустимых алгоритмами, процент пойманного спама и процент ложно позитивного срабатывания.

Таблица 1.3 Сравнение алгоритмов классификаций

Метод	Ошибка (в %)	Пойманный спам (в %)	Ложно позитивное срабатывание (%)
Байес	1,12	94,5	0,51
SVM (линейное ядро)	1,18	93,8	0,47
K-ближайших соседей (10)	2,53	82,6	0,4
Случайный лес	2,16	90,62	0,73
AdaBoost с деревьями решений	1,41	92,2	0,51

Наиболее эффективными алгоритмами в задаче классификации спам — сообщений является наивный Байесовский классификатор, также следует использовать другой статистический алгоритм - алгоритм Фишера. Для реализации данных алгоритмов необходимо выделить признаки спам-сообщений, а также создать тренировочное множество обычных текстовых сообщений и спам-сообщений.

1.5 Формирование базы данных коротких спам-сообщений

Для работы алгоритмов фильтрации спама необходимо осуществить сбор данных двух типов: спам-сообщений и обычных текстовых сообщения абонентов. В случае рекламной рассылки информация не несет личный характер, а обращена к аудитории и поэтому поиск смс-рассылок является более легкой задачей, чем поиск данных для класса не спам сообщений. В случае обычных текстовых сообщений нельзя обратиться к базе сотового оператора, потому что текстовые сообщения носят информацию личного характера и не будут предоставлены в общий доступ.

В качестве источника спам сообщений используются данные с сайта РосСпама [6], на котором абоненты сами регистрируют примеры рекламных смс-рассылок, полученных ими на сотовые телефоны. Другим источником является сайт организации «Арамба» [7], предоставляющей маркетинговые услуги для компаний. «Арамба» предлагает также собственный программный продукт для массового оповещения клиентов компаний. На сайте приведены примеры смс-рассылок различных компаний, сгруппированных по сферам деятельности.

1.5.1 Формирование базы спам-сообщений

Сбор подобной информации является рутинной работой, требующей временных затрат, поэтому создана специальная система по сбору информации. Данная система преобразует форматы данных РосСпама и «Арамбы» в формат текстового файла, каждая строка которого является

отдельным сообщением. На вход системе подается два файла: база текстовых сообщений с сайта РосСпама и с сайта «Арамбы», система конверитрует форматы и сохраняет результаты в едином текстовом файле.

Мощность собранной базы спам-сообщений — 4936 сообщений.

1.5.2 Формирование базы обычных сообщений

Для сбора данных, не являющихся рекламной рассылкой, были скачаны истории диалогов личной переписки одного пользователя из социальной сети «В контакте».

Поскольку загруженная таким образом информация имеет свой формат, было необходимо разработать программу, преобразующую формат диалогов из социальной сети «В контакте» в выходной формат - каждая строка выходного файла является отдельным сообщением.

В целях ликвидации доминирующих сообщений пользователя, диалоги которого использовались для формирования базы, его сообщения игнорировались.

Из текстовых сообщений диалогов попали в базу обычных сообщений лишь те тексты, количество символов в которых принадлежит диапазону от 30 до 140.

Количество проанализированных диалогов 23. Размер базы обычных текстовых сообщений 11045.

1.6 Анализ признаков, позволяющих выделить спам-сообщения

Как правило, спамом являются сообщения, содержащую призыв к действию. Спам-сообщения имеют структуру: некоторый заголовок, в виде обращения или в виде объявления темы сообщения, затем следует побуждение к действию и контактная информация.

Можно разбить текст спам-сообщения на элементы, представленные на рис. 1.2.

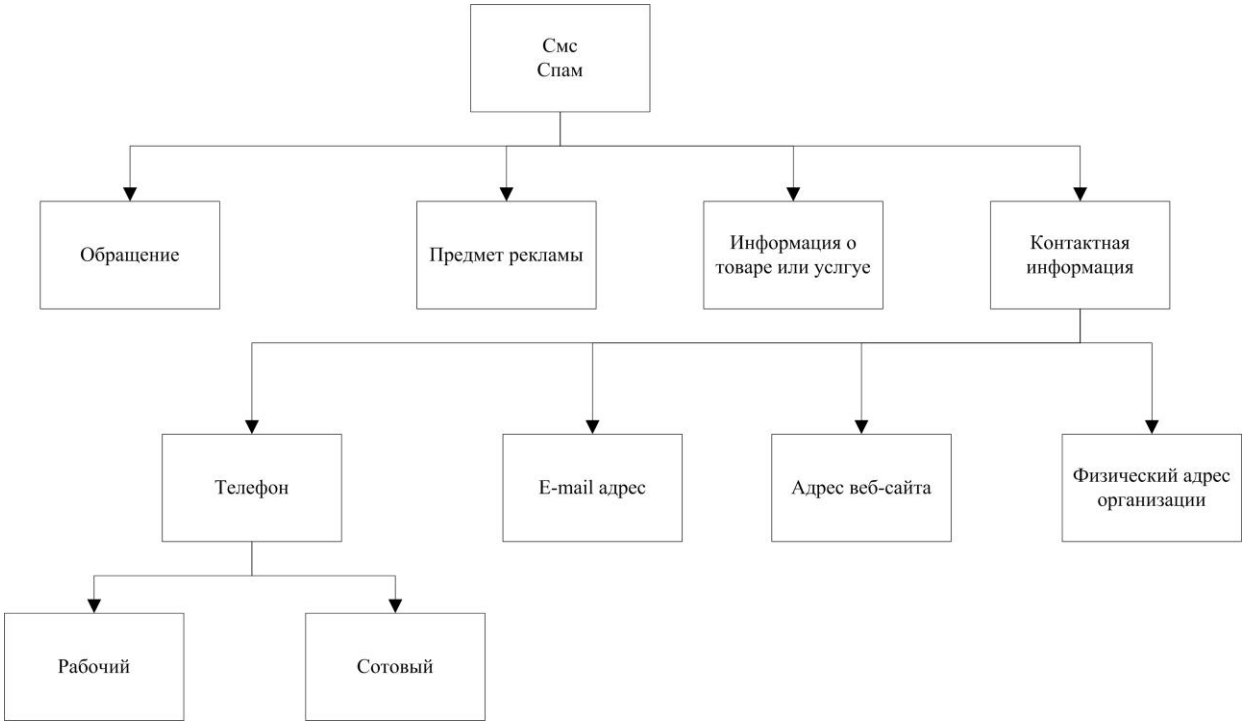


Рисунок 1.2 Структура смс спам сообщения

В спам-сообщениях присутствует, как правило, контактная информация в виде телефона (сотового или рабочего), веб адрес сайта, физический адрес. Реже встречается указание физического адреса и графика работы. Числа в большинстве случаев представляют размер скидки (с последующим знаком «%»), дату или стоимость товара/услуги, также используются для обозначения графика работы или номера дома.

В связи с тем, что размер одного сообщения ограничен для русского языка 70 символами, компании стараются умещать всю необходимую информацию в одну или две смс. Осуществляется это посредством использования сокращений.

Составители спам сообщений для обхода фильтров помимо сокращения слов используют прием замены русских букв в слове на схожие по написанию английские буквы или символы.

1.7 Анализ и выбор признаков спама

В известных работах [5] в качестве признаков спам-сообщений предлагаются: слова, количество символов, числа, знак $.

Проведенный анализ собранной базы спам–сообщений показал, что выделенные признаки не являются достаточными. В результате анализа предлагается использовать следующие дополнительные признаки спам смс:

1) количество слов в сообщении
2) количество сокращений
3) количество слов, полностью написанных в верхнем регистре
4) количество слов, начинающихся с заглавных букв
5) количество слов, состоящих только из английских букв
6) количество слов, состоящих как из русских, так и из английских символов
7) наличие специального символа %
8) наличие номеров телефонов
9) наличие адреса электронной почты
10) наличие адреса веб-сайта

1.8 Разработка взаимодействия с системой

Отображение взаимодействия пользователей системы и отношения между основными модулями системы представлено с помощью диаграммы прецедентов на рис. 1.3.

Рисунок 1.3 Диаграмма прецедентов для библиотеки фильтрации спам сообщений

Выводы

В результате проведенного анализа предметной области были определены:

- выделяемые признаки спам-сообщений при разбиении текста на смысловые элементы;
- алгоритм нормализации слов с использованием словаря словоформ Зализняка;
- алгоритмы классификации текста: алгоритм наивного байесовского классификатора и алгоритм Фишера;
- проблемы алгоритмов и предложены пути их решения в рамках данной дипломной работы.

2. Конструкторский раздел

2.1. Структура программного обеспечения фильтрации спама

В качестве методов классификации текстов была выбрана комбинация следующих методов:

- наивного байесовского классификатора
- классификатора Фишера

Для реализации этих методов на предварительном этапе обработки текста необходимо провести так называемую токенизацию текста и нормализацию слов с использованием выбранного тезауруса. Первый уровень формализации можно представить следующим образом (рис. 2.1). Второй уровень формализации можно представить следующим образом (рис. 2.2).

Рисунок 2.Первый уровень формализации фильтрации спам сообщений

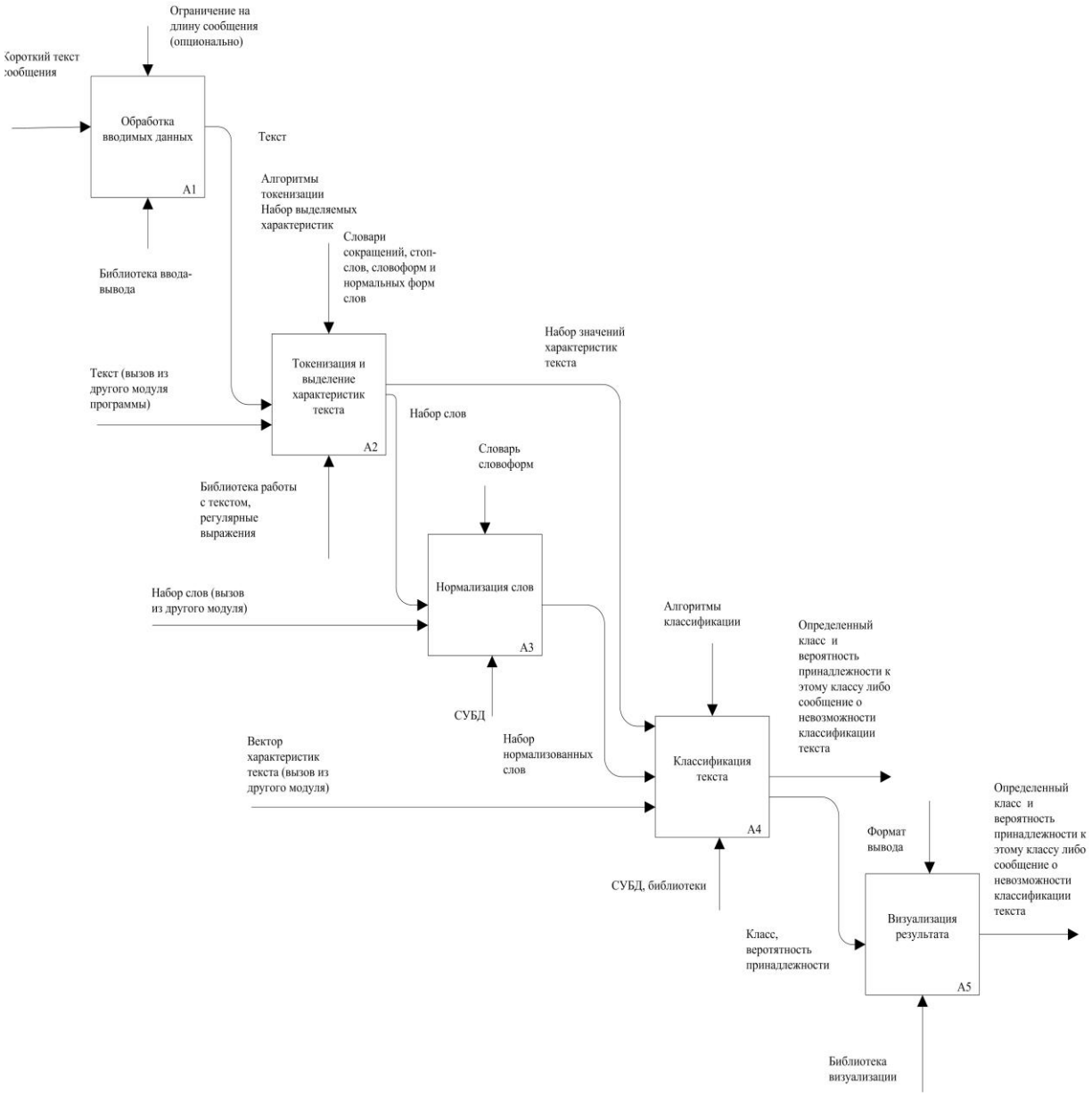


Рисунок 2.2 Второй уровень формализации фильтрации спам сообщений

2.2 Токенизация

2.2.1. Определение токенов

В качестве токенов будем рассматривать:

- слова
- телефонные номера
- числа
- e-mail адрес
- адрес веб-сайта

2.2.2. Разработка алгоритма выделение токенов из текста

Выделение токенов из текста производится с помощью следующих разработанных регулярных выражений:

- телефонный номер

  MobTel = "(((\\+7)|8|7)\\s*((\\(\\s*\\d{3}\\s*\\))|([\\s-]?\\d){3})([\\s-]?\\d){7})"

  HomeTel = "(\\d([\\s-]?\\d){2}-?\\d([\\s-]?\\d){2,3})"

  TelRegexp = "[^\\d](?<telephone> MobTel | HomeTel)(?:[^\\d]|$)"
- число

  (?<number>[\\d]+)
- email адрес

  "(?<email>[a-zA-Z1-9]+[a-zA-Z1-9-\\._]*@[a-z1-9]+(?:\\.[a-z1-9]+)+)"
- адрес веб сайта

  Eng = "([a-z1-9]+(\\.[a-z]+)+)";

  Rus = "([a-яё1-9]+(\\.[a-яё][a-яё1-9-]*)+)";

  WebSiteRegexp = "(((https?://)|(www\\.))?(" + Eng + "|" + Rus + ")(/[^\\s]+)*)";

2.2.3. Выделение и обработка сокращений

Сокращения бывают четырех видов:

- образованные путем усечения слова до нескольких начальных букв
- образованные путем усечения слова посредством удаления символов из середины слова и разделения начало и конца слова дефисом
- сокращения, написанные через «/»
- при отсечении начальной части сокращаемого слова

Поскольку сокращение «кодирует» слово или словосочетание, то разумно определять в тексте сокращения и не исключать их из дальнейшего рассмотрения, а заменять их на полные слова. Для этого необходимо составить словарь сокращений и осуществлять замену по нему.

Анализ сокращений в текстах базы данных спам сообщений осуществляется с использованием следующих разработанных правил определения сокращений:

- для сокращений, образованных усечением до начальных букв "([а-яА-ЯёЁ]{1,4})";
- сокращения с дефисом

  "([а-яА-ЯёЁ]{2,4}-[а-яА-ЯёЁ]{2})";

- сокращения с разделителем «/»

  "([а-яА-Я]/[а-яА-Я])";

  CommonAbbr = SimpleAbbr + "|" + HyphAbbr + "|" + SlashAbbr;

  Итоговое регулярное выражение - AbbrRegexp = "([\\s[^a-zA-Za-яА-ЯёЁ]]|\\G)(?<word>" + CommonAbbr + ")([^a-zA-Za-яА-ЯёЁ]|\\z)";

Длина сокращений может варьироваться. Определено, что оптимальный диапазон длины сокращений — от 1 до 4. При увеличении до 5 символов резко возрастает процент попадания обычных слов в класс сокращений, что увеличивает время анализа, но не улучшает его качество.

Установлено, что если сокращение вида w1-w2,то оптимальная длина сокращения w1 от 2 до 3, w2 2. Иначе появляются составные слова (которые можно разделить на два слова w1 и w2 сохранив их смысл). Возникает следующая проблема: сб-вс или пн-пт т.е. дни недели будут восприняты как одно сокращение и такая важная информация как график работы будет проигнорирована. Тогда если w1 и w2 принадлежат подмножеству: {пн, пон, вт, ср, чт, пт, пят, сб, суб, вс}, то w1-w2 разбиваются на слова w1 w2, а после заменяются полными словами.

Таким образом предлагается все слова, подходящие регулярному выражению "([а-яА-ЯёЁ]{2,4}-[а-яА-ЯёЁ]{2})" рассматривать как сокращения, которые сложнее всего расшифровать и проанализировать, поэтому они игнорируются и не участвуют в дальнейшем анализе.

2.2.4. Выделение и обработка стоп-слов

Стоп -слово - это слово, не несущее какой-либо самостоятельной смысловой нагрузки. К стоп-словам относятся:

- предлоги
- союзы
- частицы
- междометия
- местоимения

Стоп-слова определяются по следующему разработанному регулярному выражению:

(?:(?:\\G)|(?:[\\s,\\-!?]+))(?<auxiliary_word>[а-яА-ЯёЁ]{1,4})[\\s?\\-!,]+

После определения токенов по регулярному выражению необходимо осуществить поиск стоп-слова в словаре и, если оно найдено, удалить стоп-слово из дальнейшего рассмотрения.

2.2.5. Поиск английских символов, заменяющих русские буквы и замена их на соответствующие русские буквы

После токенизации слово, состоящее из русских и английских букв, не будет корректно обработано, поэтому его необходимо либо привести к русской раскладке либо проигнорировать его.

После анализа базы данных спам сообщений были выявлены списки замен, по которым следует осуществлять замену английских букв русскими.

Такую операцию стоит производить над словами, являющимися комбинацией символов из разных алфавитов. В случае обработки текстового сообщения в английской раскладке такое сообщение системой будет проигнорировано, поскольку резко увеличиваются затраты ресурсов на обработку текста: нужно определить, набрано сообщение транслитом или же представляет собой текст на английском языке. В обоих случаях приведение сообщения к русской раскладке с сохранением смысловой нагрузки является трудоемкой задачей и в рамках данного проекта не решается.

2.2.6. Алгоритм токенизации

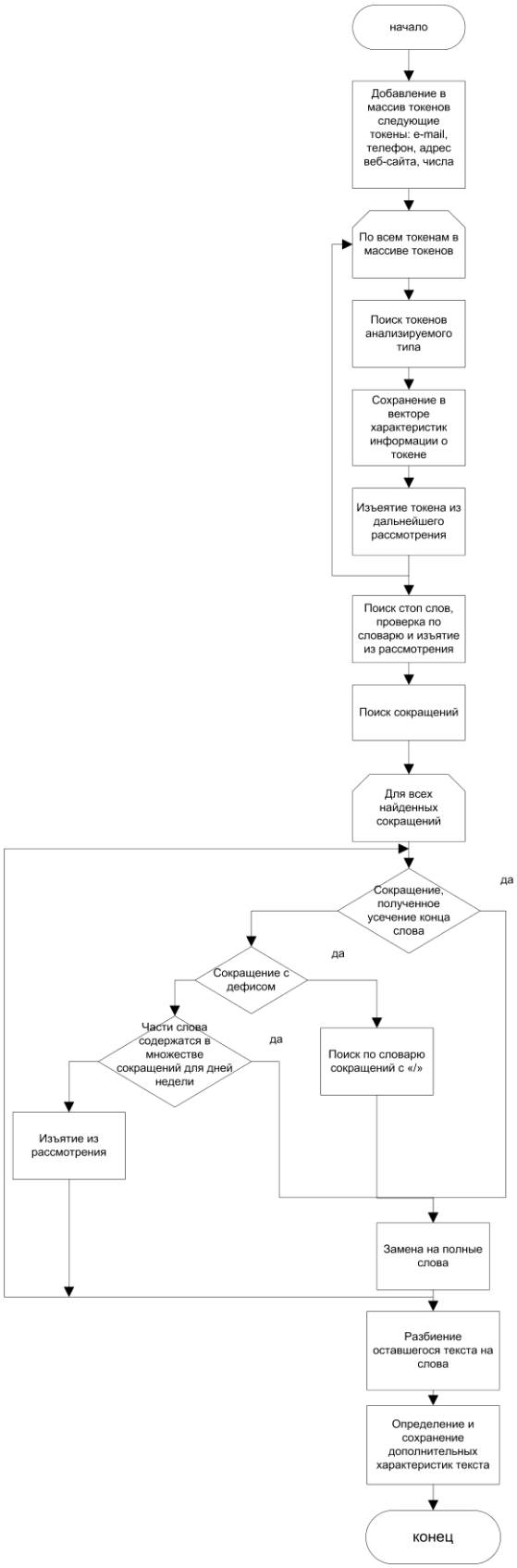


Рисунок 3.3 Алгоритм токенизации

Алгоритм токенизации представлен на рисунке 3.3.

Алгоритм выделения характеристик текста определяет следующие признаки:

- количество слов в верхнем регистре
- количество символов в верхнем регистре
- количество слов в тексте
- количество только русских слов
- количество только английских слов
- количество смешанных слов (русские, английские буквы и/или символы)
- количество чисел
- Поиск смешанных слов и замена английских букв русскими
- Количество слов не известных словарю

По окончании выявления признаков происходит итоговая склейка токенов-слов.

2.2.7. Разработка алгоритма нормализация

После этапа токенизации каждое слово из множества токенов слов ищется в словаре словоформ и, если оно найдено, сохраняется его нормальная форма, если нет, то слово не рассматривается дальше.

2.3. Классификаторы

2.3.1. Применение наивного Байесовского классификатора

Входные данные

Набор слов, статистика встречаемости слов

Алгоритм

Определим множество классов: S — спам, NS — не спам. Каждое слово в тексте сообщения будем рассматривать в качестве признака. Тогда множество всех признаков – выделенные признаки текста на этапе токенизации и множество всех найденных слов (после токенизации и нормализации остаются только русские слова), обозначим его W. Обозначим i-ый признак этого множества как Wi.

Тогда согласно теореме Байеса для одного признака:

$$P(S \mid Wi) = P(Wi \mid S) * P(S) / P(Wi), \quad (2.1)$$

где

P(S) — вероятность попадания случайного текста в категорию спам.

P(Wi) — вероятность встречи слова Wi.

P(Wi | S) — вероятность того, что слово встречается в спам сообщении

P (S) примем за 0.5 поскольку не будем делать априорного предположения о вероятности попадания наугад взятого сообщения в класс спам.

$$P(Wi \mid S) = N(Wi)s / N(W)s, \quad (2.2)$$

где

N(Wi)s — сколько раз встретилось слово Wi в спам текстах,

N(W)s — общее количество слов в спам текстах.

Вероятность P(Wi) не стоит вычислять поскольку она одинаковая для всех классов, а для определения класса сообщения необходимо лишь сравнит вероятности принадлежности к классам.

Определяемая вероятность для одного признака имеет следующий вид:

для класса спама: $P(Wi \mid S) = N(Wi)s * P(S) / N(W)s$ (2.3)

для класса не спама: $P(Wi \mid S) = N(Wi)ns * P(NS) / N(W)ns$ (2.4)

В случае, если признак Wi еще не встречался в спам тексте, согласно формуле 2.1 вероятность P(S | Wi) = 0, что неверно. В данном случае целесообразно применить сглаживание Лапласа:

$$P(Wi \mid S) = (N(Wi)s + 1) * P(S) / (N(W)s + M), \quad (2.5)$$

где M — мощность словаря

Комбинация всех признаков производится по формуле 2.6, полученной из формулы 1.6 путем логарифмирования обоих частей равенства.

$$P(S \mid W1..WN) = \log(P(S)) + \sum_{i=1}^{N} P(Wi \mid S) \quad (2.6)$$

Поскольку результат выбора основывается на статистике встречаемости слов в сообщениях различных классов, он зависит от первоначальной выборки и может давать ложно положительные ответы т.е. классифицировать обычное сообщение как спам, а так же и ложно отрицательное срабатывания - отнесение спам сообщения к классу обычного текста. Критично ложно положительное срабатывание потому что из-за него пользователю не дойдет обычное сообщение.

Для снижения вероятности появления ложно положительного срабатывания введем некий порог m: текст попадает в класс спама, если вероятность P(S|W1...WN) больше в m раз чем P(NS | W1..WN).

Если P(NS | W1..WN) > P(S|W1...WN), то сообщение рассматривается как обычное. Иначе классификатор не может отнести сообщение ни к одному классу.

В рамках данной работы не происходит дообучения классификатора, но это можно исправить при последующем развитии путем введения дополнительного словаря. Предлагается следующий алгоритм: редкие слова,

которых нет в основном словаре, ищутся в дополнительном словаре. Все слова дополнительного словаря, встречаемость которых больше или равна заранее определенному (эмпирически) значению, добавляются в основной словарь — происходит дообучение классификатора.

Выходные данные

Решение о принадлежности текста к одной из категорий: спам, не спам, не определено.

2.3.2. Применение классификатора Фишера

Входные данные

Набор слов, статистика встречаемости слов

Алгоритм

Для реализации данного подхода необходимо собрать следующие статистические данные:

- Ndocj(Признак i) - количество документов с данным признаком в этой категории,
- Ndoc - общее количество документов с данным признаком

По методу Фишера необхоимо подать на вход обратной функции хи-квадрат значение

$$-2 \times \ln(P(\text{Категория j}|\text{ Признак 1}) \times .. \times P(\text{Категория j}|\text{ Признак N})), \quad (2.7)$$

где

$$P(\text{Категория j}|\text{ Признак i}) = Ndocj(\text{Признак i}) / Ndoc.$$

Для принятия решения о принадлежности текста классу необходимо сравнить полученные вероятности принадлежности сообщения к каждому классу. Если абсолютное значение разности вероятностей меньше некоторой точности (определяется эмпирически), то необходимо сообщить о не возможности отнесения сообщения к одному из классов. В другом случае

вероятность принадлежности для какого класса больше, тот класс и стоит рассматривать как результат классификации в данном алгоритме.

Дообучение происходит таким же образом как в случае с Байесовским классификатором.

Выходные данные

Решение о принадлежности текста к одной из категорий: спам, не спам, неопределено.

2.3.3 Объединение результатов классификаторов

Каждому классификатору присваивается определенный вес (на основе анализа точности работы), после происходит суммирование вероятностей и весов, а результат нормируется и сравнивается с пороговыми значениями (как в случае классификации в методе наивного Байесовского классификатора).

2.4 Структуры данных

Для реализации работы классификаторов необходимо хранить словарь словоформ и нормальных форм слова, а также статистические данные о встречаемости слов в текстах разных классов сообщений.

В качестве структуры данных можно использовать сбалансированные деревья поиска, хэши или базу данных.

Тип данных словаря стоп-слов

На этапе токенизации используется представление словаря стоп-слов в виде сбалансированного дерева.

Тип данных словаря сокращений

На этапе токенизации используется представление словаря сокращений в виде хэш, поскольку одному сокращению могут соответствовать разные варианты полных слов.

2.4.1 База данных

В случае словарей сокращений и стоп-слов можно использовать деревья поиска и хэш таблицы, поскольку размеры словарей небольшие и они могут быть загружены в память на время выполнения программы. Что касается словарей словоформ, то они большие и хранить их предпочтительней в базе данных. Кроме того, в базе данных можно хранить статистическую информацию о встречаемости слов в разных классах сообщений. Очевидным плюсом использования базы данных является организация связанности и целостности данных.

Инфологическое проектирование

Инфологическая модель – это ориентированная на человека и не зависимая от типа СУБД модель предметной области, определяющая совокупности информационных объектов, их атрибутов и отношений между объектами, динамику изменений предметной области, а также характер информационных потребностей пользователей. Инфологическая модель предметной области может быть описана моделью "сущность—связь", в основе которой лежит деление реального мира на отдельные различимые сущности, находящиеся в определенных связях друг с другом, причем обе категории — сущность и связь полагаются первичными, неопределенными понятиями.

В рамках задачи дипломного проекта в качестве сущностей рассматриваются:

- нормальная форма слова;
- словоформа;
- количество слов, встречающихся в классах спам и не спам сообщений;
- количество документов, относящихся к выделенным классам, в которых встречаются слова.

Инфологическая модель базы данных представлена с помощью ER-диаграммы на рисунке 2.3. Таблицу признаков слов в последующем можно модифицировать на хранения не только количественных параметров, но и самих токенов для сбора статистики.

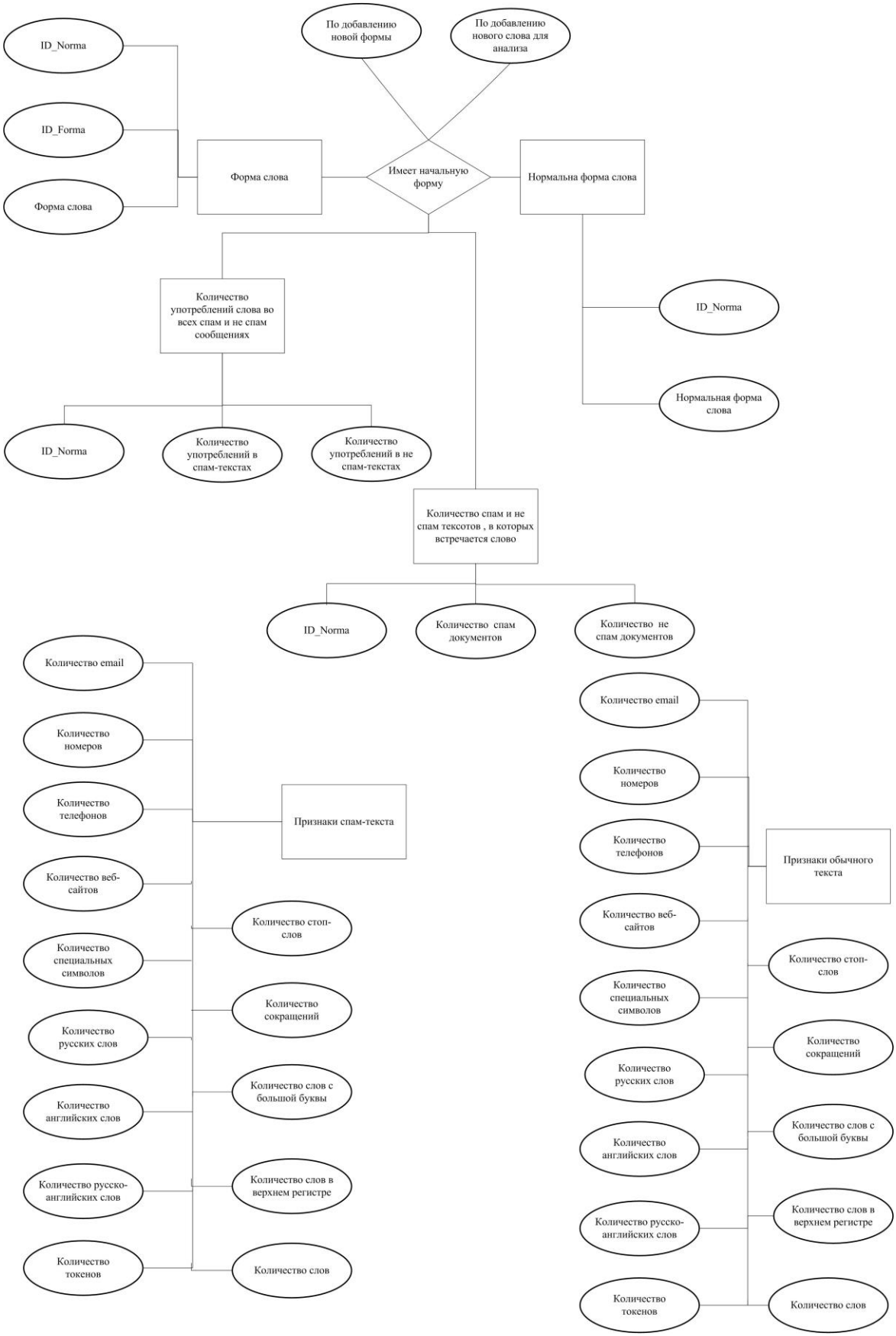


Рисунок 2.3 ER-диаграмма базы данных

Даталогическое проектирование

Даталогическая модель без признаков текста представлена на рисунке 2.4 и в таблицах 2.1 – 2.4.

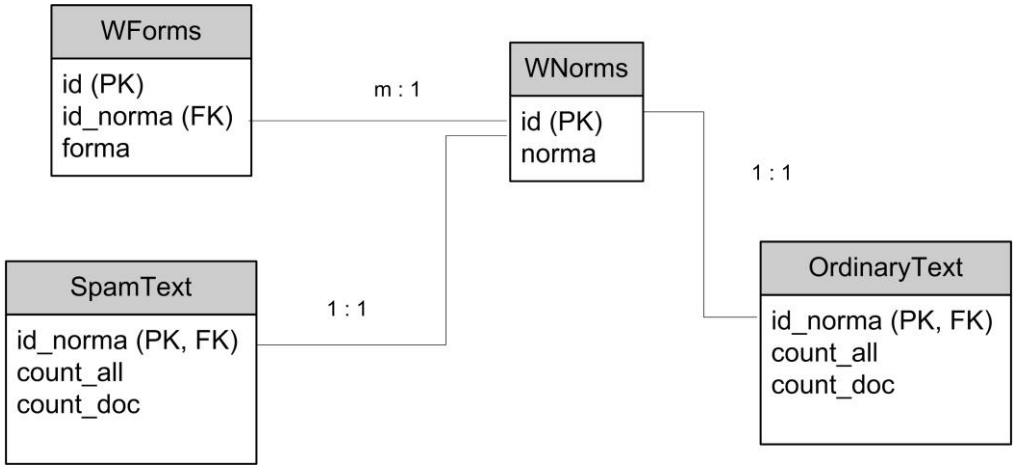


Рисунок 2.4 Схема базы данных

Таблица 2.1 Таблица норм слов WNorms

Имя поля	Тип	Свойства
id	integer	Primary key
norma	text	not null

Таблица 2.2 Таблица словоформ WForms

Имя поля	Тип	Свойства
Id	integer	Primary key
id_norma	integer	Foreigh key references WForms(id)
Forma	text	not null

Таблица 2.3 Таблица встречаемости слов в спам текстах

Имя поля	Тип	Свойства
id_norma	Integer	Foreign key references WNorms(id)
Count_all	Integer	-
Count_docs	Integer	-

Таблица 2.4 Таблица встречаемости слов в обычных текстах

Имя поля	Тип	Свойства
id_norma	Integer	Foreign key references WNorms(id)
Count_all	Integer	-
Count_docs	Integer	-

3. Технологический раздел

3.1. Выбор средств программной реализации

3.1.1 Выбор языка программирования

Для реализации программной части дипломного проекта был выбран язык Java (версии 7).

Причины выбора языка программирования:

- объектно-ориентированный язык, позволяет использовать подходы ООП и разрабатывать более гибкий и удобный в поддержке код
- приложения на Java могут работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры и могут запускаться на любом устройстве, на котором есть виртуальная машина
- на Java разрабатываются приложения под ОС Android, что позволяет подключать разрабатываемые проекты в качестве библиотек к приложениям
- обладает автоматическим управлением памятью
- имеется широкий набор коллекций классов, реализующих различные структуры данных
- обладает средствами создания многопоточных приложений
- унификация доступа к базам данных (на уровне отдельных SQL-запросов — на основе JDBC)

3.1.2 Выбор среды разработки

В качестве среды разработки была выбрана Eclipse, потому что:

- обладает инструментами разработки на Java
- позволяет разрабатывать кроссплатформенные приложения
- обладает средствами генерации документации
- отображает список текущих задач

- позволяет осуществлять автоматическое обновление/установку кода
- имеет текстовый редактор с подсветкой синтаксиса, систему дополнения при наборе
- использует потокобезопасный отладчик
- обладает возможностями подключения проекта к системе контроля версий (GIT, CVS)
- позволяет подключать модули

3.1.3 Выбор базы данных

В качестве базы данных была выбрана Sqlite 3, потому что:

- простота использования
- возможность использовать язык Java
- экономичная в плане ресурсов архитектура (приложение — клиент — backend (файлы))
- кроссплатформенность
- полностью свободная лицензия
- использование в ОС Android
- безопасность, БД хранится в одном файле, права доступа к которому можно контролировать стандартными средствами ОС

3.2 Основные моменты реализации

3.2.1 Организация файлов исходного кода

Весь проект располагается в одной директории. Поскольку проект реализован на языке Java, он состоит из пакетов. Пакет Java — это механизм, позволяющий организовать Java-классы в пространстве имен. Каждый пакет располагается в своей поддиректории. Папка пакет состоит из папок «bin» и «src». В папке «bin» содержатся фалы расширения *.class — файл содержащий байт код, который может исполняться виртуальной машиной Java. В папке «src» содержится исходный код классов.

Так же имеется полная сборка библиотеки в формате jar-файла, который может непосредственно подключить к другому проекту.

В директории «db» находится файл базы данных на sqlite, там же находятся файлы конфигурации базы и инициализирующие скрипты для создания новой базы.

3.2.2. Основные классы

Разработанная библиотека состоит из двух основных модулей: предобработка текста и классификация. Приложение разработано на объектно-ориентированном языке с применением принципов ООП.

Для реализации задач сбора информации, анализа, обработки текста, классификации, связи с базой данных, тестирования и интерфейса было разработано более 50 классов. Далее будут приведены основные классы.

1) Token родительский класс, реализующий метод выделения токенов из строки символов, а также хранит основную информацию о токене и некоторые статистические данные.
2) Auxiliary – класс по выявлению стоп-слов, использует словарь в виде сбалансированного дерева
3) Abbreviation – класс по выявлению сокращений, использует словарь, представленный хэш таблицей
4) SqlAgent – класс, осуществляющий соединение с базой и реализующий последующее взаимодействие с базой
5) TextParser – класс, который из входящего сообщения выделяет токены, сохраняет нормализованные слова и сохраняет признаки текста в класс-контейнер признаков
6) TextFeatures – класс- контейнер признаков, также умеет находить некоторые признаки текста. Взаимодействует с классом LangDetector, от которого для каждого языка наследуются свои классы.

7) Baies – класс, реализующий наивный байесовский классификатор
8) Fisher – класс, реализующий метод Фишера в задачи классификации
9) SpamDetector – класс, который на вход принимает текстовое сообщение, а на выходе предоставляет ответ о принадлежности входных данных классу спам сообщений или обычного текста.

Диаграммы основных классов представлены на рисунках 4.1 и 4.2.

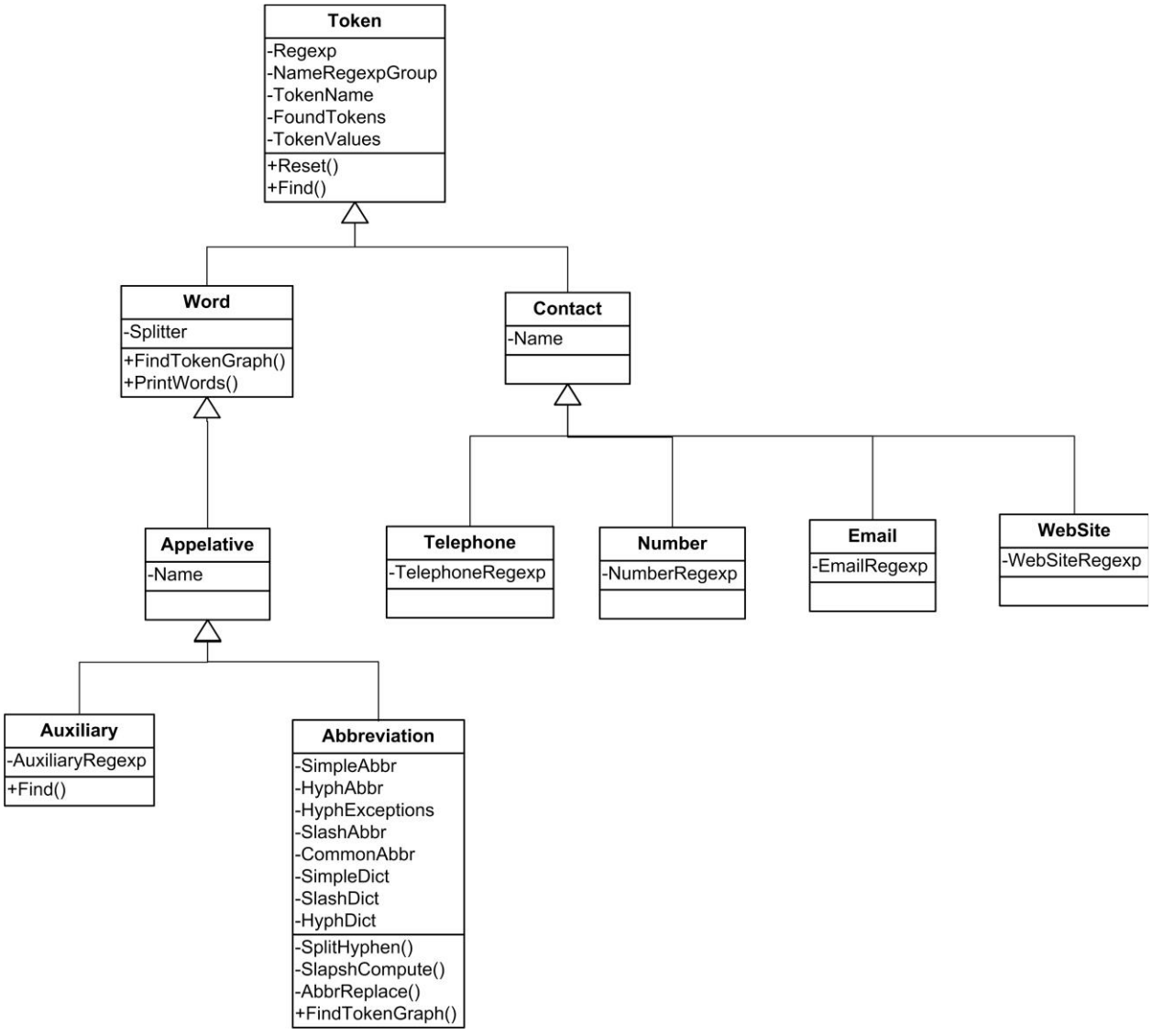


Рисунок 4.1 UML диаграмма классов

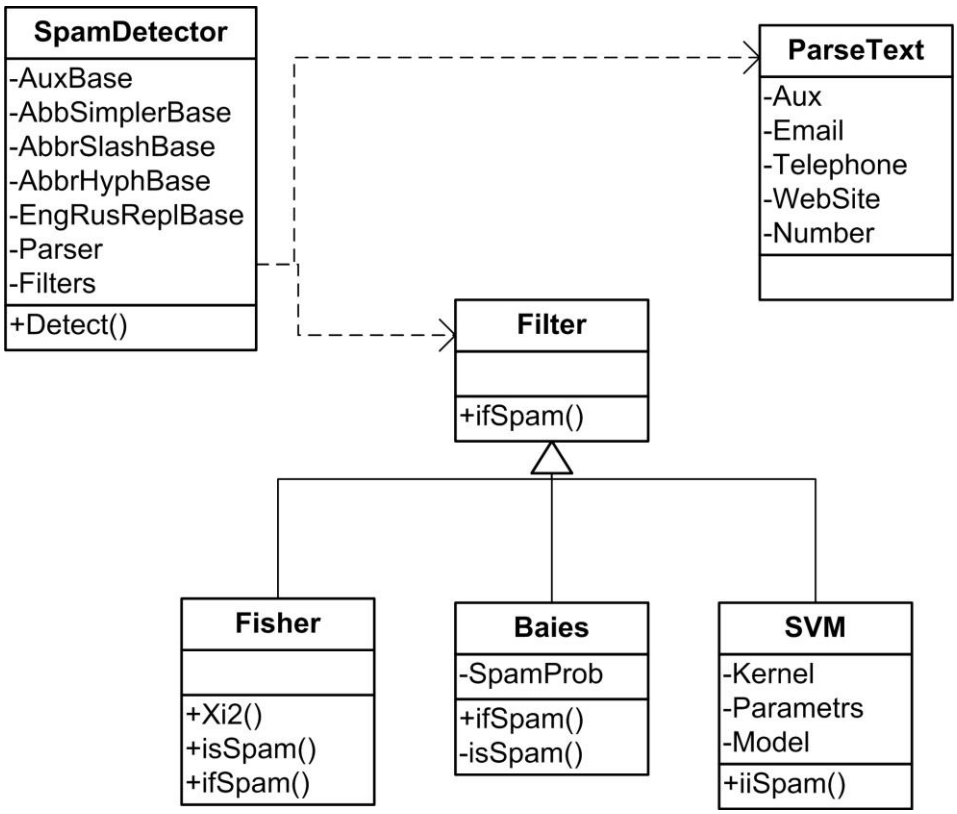


Рисунок 4.2 UML диаграмма классов

3.2.3 Создание базы данных

База данных создавалась посредством запуска скрипта, который осуществляет конфигурацию среды sqlite и создает таблицы.

После с помощью команды «.import <file>» заранее подготовленные данные массово вставляются в таблицы.

3.2.4. Методика разработки

В качестве методики разработки используется разработка через тестирование - это техника разработки программного продукта, которая основывается на повторении очень коротких циклов разработки. Сначала пишется тест на вводимое изменение, после пишется код, который реализует поставленные задачи и позволит пройти тест. Конечным этапом является изменения (рефакторинга) нового кода к соответствующим стандартам.

Цикл разработки через тестирование:

- Добавление теста
- Запуск всех тестов, если есть среди тестов те, которые проходят успешно, значит они не тестирую новые задачи и должны быть удалены или изменены.
- Написание кода
- Запуск всех тестов, убедиться, что все тесты проходят
- Рефакторинг (при достижении требуемой функциональности) - изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы, устранить дублирование кода, облегчить внесение изменений в ближайшем будущем.
- Повторение цикла

Преимущества данного подхода:

- разработанное приложение более пригодно для тестирования, потому что заставляет разработчика обдумать то, как приложение будет тестироваться
- способствует тому, что тестами будет покрыта вся функциональность
- способствует декомпозиции системы
- модули программного продукта становятся менее связными
- повышение качество кода продукта
- тесты позволяют проводить рефакторинг без риска испортить код
- тесты могут использоваться в качестве документации
- программисты, пишущие больше тестов, склоны быть более продуктивными согласно [9]
- реже используется отладчик

Стоит отметить, что на начальном этапе значительную часть ресурсов придется отвести на написание тестов, также есть задачи, которые невозможно решить только при помощи тестов (безопасность данных, взаимодействие между процессами).

Разработку через тестирование следует применять, когда необходимо прохождение функциональных тестов,

3.3 Тестирование библиотеки

При тестировании разработанного программного решения были использованы два подхода: модульное и функциональное.

При модульном тестировании для каждого логически законченного набора классов был реализован класс-тестировщик, который вызывал методы тестируемых классов с заранее подготовленными данными и формировал вывод промежуточных данных во время работы тестируемых классов либо на стандартный поток вывода, либо в лог-файл.

Пример файла с результатами тестирования пакета токенизации текста представлен на рисунке 4.1.

Complex Test

Analysing text:
Назовите Вашу цену - Распродам автомобили с пробегом до 31 октября! Гарантия на а/м, Диагностика бесплатно. РОЛЬФ Центр: +74957858289

Removed Email
Назовите Вашу цену - Распродам автомобили с пробегом до 31 октября! Гарантия на а/м, Диагностика бесплатно. РОЛЬФ Центр: +74957858289
Removed Telephone
Назовите Вашу цену - Распродам автомобили с пробегом до 31 октября! Гарантия на а/м, Диагностика бесплатно. РОЛЬФ Центр:
Removed WebSite
Назовите Вашу цену - Распродам автомобили с пробегом до 31 октября! Гарантия на а/м, Диагностика бесплатно. РОЛЬФ Центр:
Removed Number
Назовите Вашу цену - Распродам автомобили с пробегом до октября! Гарантия на а/м, Диагностика бесплатно. РОЛЬФ Центр:
Removed Auxiliary Word
Назовите Вашу цену - Распродам автомобили пробегом октября! Гарантия а/м, Диагностика бесплатно. РОЛЬФ Центр:
Removed/changed abbreviations
Назовите Вашу цену - Распродам автомобили пробегом октября! Гарантия автомобиль, Диагностика бесплатно. РОЛЬФ Центр:
Назовите Вашу цену - Распрода машина автомобили пробегом октября! Гарантия автомобиль, Диагностика бесплатно. РОЛЬФ Центр:

After Splitting
Text variant
Назовите|Вашу|цену|Распродам|автомобили|пробегом|октября|Гарантия|автомобиль|Диагностика|бесплатно|РОЛЬФ|Центр|

Text variant 0: (only rus words)
Назовите
Вашу
цену
Распрода
пробегом
октября
Гарантия
автомобиль
Диагностика
бесплатно
РОЛЬФ
Центр

End test

Рисунок 4.1 Результат тестирования модуля разбиения текста на смысловые элементы

При функциональном тестировании был составлен универсальный класс, который позволяет тестировать разные методы классификации. В качестве входных параметров он принимает файл, содержащий тестируемое множество спам сообщений, а также файл, содержащий тестируемое множество обычных текстов сообщений. В качестве параметров тестирования рассматриваются: рабочая директория, пути к файлам словарей и базе данных, префикс для образования имени файла результата тестирования от имени исходного файла, тестируемый фильтр спама.

Формат выходных данных:

1) <результат прохождения теста> - OK/ER/UN — (правильно определился класс, ошибка, не удалось определить)
2) <класс сообщения> - 1- спам, 0 — не удалось определить, -1 обычный текст
3) <выходное значение вероятности/логарифма вероятности спама>
4) <выходное значение вероятности/логарифма вероятности обычного текста>

В конце теста приводится:

- countErr — суммарное количество ошибок;
- countUnknown — число текстов, которых не удалось классийицировать;
- coutCorrect — число текстов, которые корректно были классифицированы.

Результат работы функционального тестирования представлен на рис. 4.2

```
UN      0 -44.825454850959986 -55.86311148473866
OK        1 -41.11981030474349 -62.587029298849025
OK        1 -206.59136064720664 -237.55445781776817
UN        0 -75.66900442949701 -85.43334662575334
OK        1 -71.34313054010346 -89.92284866664687
OK        1 -83.57331071007981 -122.67555965556825
UN        0 -48.49112934907698 -61.31653217268257
OK        1 -86.07997042893068 -125.93625367500974
OK        1 -135.11303331407373 -156.51425765798874
OK        1 -107.86303960969576 -123.7406606145956
OK        1 -67.4061142808149 -86.18810454518876
OK        1 -136.16241529086653 -158.67323623614854
UN        0 -100.5344198510453 -108.34556519001971
OK        1 -65.33040741488428 -86.39384727499066
OK        1 -70.87186793195741 -90.8260182055871
UN        0 -63.4833955337364 -74.49511850766976
OK        1 -90.59227349207313 -112.95855029651874
OK        1 -78.44418523940051 -99.0475878505326
OK        1 -129.52480150576935 -157.16411509892538
OK        1 -159.64034111887213 -195.71834521688695
OK        1 -73.44875759620649 -98.88622121012078
OK        1 -93.88076221405694 -133.1711991331314
OK        1 -123.42353946593471 -149.2323222210584
OK        1 -164.28106912549404 -184.6581810365317
OK        1 -106.59700773085882 -128.8607665534549
OK        1 -106.42739317123458 -135.50883464558132
OK        1 -122.81971529328051 -163.62618813748267
OK        1 -112.21847811565827 -134.21184673351328
OK        1 -69.30526204761651 -93.34992248212383
OK        1 -193.70848096825145 -248.51214284723764

countErr = 0        countUnknown = 5        countCorrect = 25
```

Рисунок 4.2 Пример выходных данных при функциональном тестировании классификаторов

3.4. Сборка и запуск ПО

Использование разработанного ПО возможно в двух режимах:

- с использованием пользовательского интерфейса
- подключение библиотеки к разрабатываемому проекту

Пользовательский интерфейс

достаточно запустить исполняемую сборку библиотеки smap_detector.jar из командной строки:

java -jar spam_detector.jar «текст сообщения»

Подключение библиотеки к проекту

Для подключения библиотеки к проекту следует в зависимости от среды разработки подключить к проекту spam_detector.jar

Так же можно вносить свои изменения в библиотеку, запуск проектами с изменениями осуществляется в обычном режиме запуска программы на исполнение/отладку.

3.5 Входные и выходные данные

Входные данные

Входными данными является текстовое сообщение. Не накладывается жесткое ограничение на длину сообщения, но в рамках данного дипломного проекта реализована классификация для смс текстов т.е. коротких тестов. В случае подачи большого текста точность классификации не гарантируется.

Выходные данные

Выходными данными является строка. Формат строки:

<имя класса> [<вероятностная характеристика принадлежности классу спама> < вероятностная характеристика принадлежности классу обычных сообщений>]

<имя класса>

- спам — если сообщение принадлежит классу рекламных смс-рассылок
- обычный текст — если сообщение определено как обычный текст
- не известно — если не удалось отнести сообщение ни к одному классу

вероятностные характеристики для классов спама и обычного текста опционные и предназначены для дальнейшего использования при подключении библиотеки к проекту.

3.6. Требования к системе

Для использования библиотеки фильтрации спама система пользователя должна удовлетворять следующим требованиям:

- Не менее 512 Мб оперативной памяти.
- Процессор с частотой не менее 1.5 ГГц.
- Установленный пакет Java версии не менее 7

4. Экспериментальный раздел

В экспериментальном разделе исследовались разработанные реализации алгоритмов выделения значимых элементов текста и распознавания спам-сообщений.

4.1 Анализ базы

При разбиении текста 92% выделенных слов в спам-сообщениях было найдено в словаре словоформ Зализняка. 9% всех слов в спам-сообщениях состояли только из английских символов, 1,9% слов содержали как русские, так и английские буквы.

При разбиении текста 83% выделенных слов в обычных текста было найдено в словаре словоформ Зализняка. 1,1% всех слов в обычных сообщениях содержали только английские символы, 0,84% слов содержали как русские, так и английские буквы.

Количество номеров телефонов в спам-сообщениях к общему количеству токенов — 2,4%, в обычных сообщениях — 0,01%.

Количество электронных адресов сайтов в спам-сообщениях к общему количеству токенов — 1,7%, в обычных сообщениях — 0,06%.

Количество чисел в спам-сообщениях к общему количеству токенов — 13,5%, в обычных сообщениях — 1,4%.

Количество e-mail адресов в спам-сообщениях к общему количеству токенов — 0,006%, в обычных сообщениях — 0,01%.

Количество стоп-слов в спам-сообщениях к общему количеству токенов — 19,5%, в обычных сообщениях — 34,6%.

Количество сокращений в спам-сообщениях к общему количеству токенов — 3,7%, в обычных сообщениях — 0,18%.

Количество слов в спам-сообщениях к общему количеству токенов — 60,5%, в обычных сообщениях — 63,7%.

4.2 Исследование работы модифицированного наивного байесовского классификатора

Фильтру спам-сообщений на основе модифицированного байесовского классификатора подавалась на вход:

1) база спам-сообщений из тренировочной выборке (из 150 сообщений)

- верно классифицировалось — 97,4%
- не удалось классифицировать — 2,6%
- ошибок — 0%

2) база обычных сообщений из тренировочной выборки (из 150 сообщений)

- верно классифицировалось — 90,7%
- не удалось классифицировать — 9,3%
- ошибок — 0%

3) база спам-сообщений, не содержащихся в тренировочной выборке (30 сообщений)

- верно классифицировалось — 83,3%
- не удалось классифицировать — 16,6%
- ошибок — 0%

4) база обычных сообщений, не содержащихся в тренировочной выборке (150 сообщений)

- верно классифицировалось — 88,6%
- не удалось классифицировать — 10,6%
- ошибок — 0,6%

В результате по пунктам 1 и 2:

- мощность тестируемой базы — 300 сообщений
- не удалось определить 0,6%
- ошибок — 0%
- верно классифицировалось (точность классификации) — 99,4%
- процент найденного спама (полнота классификации) — 97,4%

В результате по пунктам 3 и 4:

- мощность тестируемой базы - 180 сообщений, 150 из которых обычный текст, 30 — спам-сообщений.
- Процент правильного определения класса сообщения(точность классификации) — 87,9%
- Процент найденного спама (полнота классификации) — 83,3 %
- не удалось определить — 11,6%
- ошибка - 0,5%

Анализ результатов исследования

Тестирование классификатора на сообщениях из тренировочной выборки показало, что если на сообщении обучался классификатор, то он правильно его классифицирует в 99,4 процентах случаев, при этом в исследовании ошибок при классификации не найдено.

В случае тестирования на ранее не обученной выборке, процент правильной классификации данных велик, но согласно исследованиям он может достигать значений больше 94[5]. При этом стоит отметить, что процент ошибок очень маленький — 0,5%, в то время как в исследованиях [5] он равен 1,2%. Процент ошибок удалось снизить благодаря введению дополнительного класса, в который попадают сообщения, когда классификатор «затрудняется» в выборе. Предпочтительней рассмотреть спам-сообщение как обычный текст, чем заблокировать сообщение и оно не дойдет до пользователя.

Для улучшения точности классификации следует провести дополнительное исследование по выявлению значения параметра, который используется при выборе класса.

4.3 Исследование работы классификатора по методу Фишера

1) база спам сообщений, на которой обучалась система

- верно классифицировалось — 65%
- не удалось классифицировать — 35%
- ошибок — 0%

2) база обычных сообщений, на которых обучалась система

- верно классифицировалось - 84%
- не удалось классифицировать – 10,6%
- ошибок – 5,4%

3) база спам сообщений, не содержащихся в тренировочной выборке (30 сообщений)

- верно классифицировалось — 63,3%
- не удалось классифицировать — 33,3%
- ошибок — 3,3%

4) база спам сообщений, не содержащихся в тренировочной выборке (30 сообщений)

- верно классифицировалось — 66,7 %
- не удалось классифицировать — 30%
- ошибок — 3,3%

Анализ результатов исследования

Исследование показало, что метод Фишера распознает спам хуже, чем модифицированный наивный байесовский классификатор. Это связано с тем, что обучающие множества для класса спам сообщений и класса обычных сообщений должны быть равномощны т.е. содержать одинаковое количество документов. Поскольку мощность базе данных спам-сообщений меньше в 2,23 раза мощности базы обычных сообщений, пришлось сократить обучающее множество для обычных сообщений в процессе обучения. С увеличением количества обучающей выборки можно ожидать улучшение работы классификатора.

5. Организационно-экономическая часть

5.1 Организация и планирование процесса разработки

Этап организации и планирования разработки программного продукта в рамках использования традиционного метода планирования представляет собой разбиение процесса на следующие работы:

1. Формирование состава выполняемых работ и группировка их по стадиям разработки.
2. Расчет трудоемкости выполнения работ.
3. Установление профессионального состава и расчет количества исполнителей.
4. Определение затрат времени на выполнение каждого этапа работ.
5. Построение календарного графика выполнения разработки.
6. Контроль выполнения календарного графика.
7. Формирование состава выполняемых работ и группировка их по стадиям разработки

5.2 Основные этапы разработки программного продукта

Разработку программного продукта разбивают на следующие стадии:

1. Техническое задание

Постановка задач и установление состава набора прикладных программ, структуры информационной базы. Выбор языков программирования. Предварительный выбор методов выполнения работы. Разработка календарного плана выполнения работ.

2. Эскизный проект

Предварительная разработка структуры входных и выходных данных. Разработка общего описания алгоритмов реализации решения задач.

Разработка пояснительной записки. Консультации разработчиков постановки задач. Согласование и утверждение эскизного проекта.

3. Технический проект

Разработка алгоритмов решения задач. Разработка пояснительной записки. Согласование и утверждение технического проекта. Разработка структуры программы. Разработка программной документации и передача ее для включения в технический проект. Уточнение структуры, анализ и определение формы представления входных и выходных данных. Выбор конфигурации технических средств.

4. Рабочий проект

Комплексная отладка задач и сдача в опытную эксплуатацию. Разработка проектной документации. Программирование и отладка программ. Описание контрольного примера. Разработка программной документации. Разработка, согласование программы и методики испытаний. Предварительное проведение всех видов испытаний.

5. Внедрение

Подготовка и передача программной документации для сопровождения с оформлением соответствующего акта. Передача программной продукции в фонд алгоритмов и программ. Проверка алгоритмов и программ решения задач, корректировка документации после опытной эксплуатации программного продукта.

5.3 Расчет трудоемкости проекта

Трудоемкость разработки программной продукции зависит от ряда факторов, основными из которых являются:

1. Степень новизны разрабатываемого программного обеспечения. По степени новизны разрабатываемые продукт относится к группе новизны «В» - поскольку есть существующие аналоги
2. Сложность моделей и алгоритмов его функционирования. По степени сложности алгоритма функционирования разрабатываемая программная продукция относится к первой группе сложности - программная продукция, реализующая оптимизационные и моделирующие алгоритмы.
3. Объем используемой информации, вид её представления и способ обработки.
5. Уровень используемого алгоритмического языка программирования.

Трудоемкость разработки программной продукции $\tau_{ПП}$ может быть определена как сумма величин трудоемкости выполнения отдельных этапов разработки ПП из выражения

$$\tau_{ПП} = \tau_{ТЗ} + \tau_{ЭП} + \tau_{ТП} + \tau_{РП} + \tau_{В} \quad (5.1)$$

$\tau_{ТЗ}$ – трудоемкость разработки технического задания на создание ПП;

$\tau_{ЭП}$ – трудоемкость разработки эскизного проекта;

$\tau_{ТП}$ – трудоемкость разработки технического проекта ПП;

$\tau_{РП}$ – трудо- емкость разработки рабочего проекта ПП;

$\tau_{В}$ – трудоемкость внедрения разработанного ПП.

Трудоемкость разработки технического задания рассчитывается по формуле

$$\tau_{ТЗ} = T_{ЗРЗ} + T_{ЗРП} \quad (5.2)$$

$T_{ЗРЗ}$ – затраты времени разработчика постановки задач на разработку ТЗ, чел.-дни;

$T_{ЗРП}$ – затраты времени разработчика программного обеспечения на разработку ТЗ, чел.-дни.

Значения величин ТЗРЗ и $Т_{ЗРП}$ рассчитываются по формулам

$$T_{ЗРЗ} = t_З K_{ЗРЗ} \quad (5.3)$$

$$T_{ЗРП} = t_З K_{ЗРП} \quad (5.4)$$

$t_З$ – норма времени на разработку ТЗ на ПП в зависимости от функционального назначения и степени новизны разрабатываемого ПП, чел.-дни.

Для задач, относящихся по степени новизны к группе «В», $t_З$ = 47 чел-дни т.к. функциональное значение ПП носит расчетный характер

$К_{ЗРЗ}$ – коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задач на стадии технического задания

$К_{ЗРЗ}$ = 0,65 т.к. разработка ТЗ производится вместе с разработчиком ПП

$К_{ЗРП}$ – коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком ПП на стадии технического задания

$К_{ЗРП}$ = 0,35 т.к. разработка ТЗ производится вместе с разработчиком ПП

Приходим к следующему расчету:

$$\tau_{ТЗ} = 47 * 0,65 + 47 * 0,35 = 47 \text{ [чел.-дни]}$$

Трудоемкость разработки эскизного проекта ПП $\tau_{ЭП}$ рассчитывают по формуле

$$\tau_{ЭП} = T_{ЭРЗ} + T_{ЭРП} \quad (5.5)$$

$Т_{ЭРЗ}$ – затраты времени разработчика постановки задач на разработку эскизного проекта, чел.-дни;

$Т_{ЭРП}$ – затраты времени разработчика ПП на разработку эскизного проекта, чел.-дни.

Значения величин $Т_{ЭРЗ}$ и $Т_{ЭРП}$ рассчитываются по формулам

$$T_{ЭРЗ} = t_Э K_{ЭРЗ} \quad (5.6)$$

$$T_{ЭРП} = t_Э K_{ЭРП} \quad (5.7)$$

где $t_Э$ – норма времени на разработку эскизного проекта на ПП в зависимости от функционального назначения и степени новизны разрабатываемого ПП, чел.-дни.

Для задач, относящихся по степени новизны к группе «В» и функциональное значение ПП носит расчетный характер , $t_Э$ = 45 [чел.-дни]

$К_{ЭРЗ}$ – коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задач на стадии эскизного проекта. $К_{ЭРЗ}$ = 0,6.

$К_{ЭРП}$ – коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком ПП на стадии эскизного проекта. $К_{ЭРП}$ = 0,4.

Таким образом, для рассматриваемого случая:

$$\tau_{ЭП} = 67* 0,6 + 67 * 0,4 = 67 \text{ [чел.-дни]}$$

Трудоемкость разработки технического проекта $\tau_{ТП}$ зависит от функционального назначения ПП, количества разновидностей форм входной и выходной информации и определяется как сумма времени, затраченного разработчиком постановки задач и разработчиком ПП:

$$\tau_{ТП} = (t_{ТРЗ} + t_{ТРП}) \cdot K_В K_Р \quad (5.8)$$

$t_{ТРЗ}$, $t_{ТРП}$ – норма времени, затрачиваемого на разработку технического проекта разработчиком постановки задач и разработчиком ПП соответственно, чел.-дни.

Для рассматриваемого случая $t_{ТРЗ}$ = 17 [чел.-дни], $t_{ТРП}$ = 8 [чел.-дни].

$К_В$ — коэффициент учета вида используемой информации;

$К_Р$ — коэффициент учета режима обработки информации (при разработке технического проекта КР = 1,36)

Значение коэффициента $К_В$ определяют из выражения:

$$К_В = (К_П\, n_П + К_{НС}\, n_{НС} + К_Б\, n_Б) / (n_П + n_{НС} + n_Б) \quad (5.9)$$

$К_П$, $К_{НС}$, $К_Б$ – значения коэффициентов учета вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно.

Для рассматриваемого случая $К_П = 1$; $К_{НС} = 0,72$; $К_Б = 2,08$.

$n_П$, $n_{НС}$, $n_Б$ – количество наборов данных переменной, нормативно-справочной информации и баз данных соответственно ($n_П = 1$, $n_{НС} = 10$, $n_Б = 2$).

Таким образом, для рассматриваемого случая по формулам 5.9 и 5.8 соответственно:

$$К_В = (1 * 1 + 0,72 * 10 + 2 * 2,08) / (1 + 10 + 2) = 0,95$$

$$\tau_{ТП} = (17 + 8) * 1,36 * 0,95 = 32 \text{ [чел.-дни]}$$

Трудоемкость разработки рабочего проекта $\tau_{РП}$ зависит от функционального назначения ПП, от количества разновидностей форм входной и выходной информации, от сложности алгоритма функционирования, а также от сложности контроля информации, от степени использования готовых программных модулей и уровня алгоритмического языка программирования. Трудоемкость $\tau_{РП}$ определяется по формуле

$$\tau_{РП} = (t_{РРЗ} + t_{РРП}) \cdot К_К\, К_Р\, К_Я\, К_З\, К_{ИА} \quad (5.10)$$

$t_{РРЗ}$, $t_{РРП}$ – нормы времени, затрачиваемые на разработку рабочего проекта на алгоритмическом языке высокого уровня разработчиком постановки задач и разработчиком ПП соответственно, чел.-дни.

$К_К$ – коэффициент учета сложности контроля информации.

$К_Р$ – коэффициент учета режима обработки информаци.

$К_Я$ – коэффициент учета уровня алгоритмического языка программирования.

$К_З$ – коэффициент учета степени использования готовых программных модулей.

$К_{ИА}$ – коэффициент учета вида используемой информации и сложности алгоритма ПП.

Значение коэффициента КИА определяют из выражения

$$К_{ИА} = (К'_П n_П + К'_{НС} n_{НС} + К'_Б n_Б)/(n_П + n_{НС} + n_Б) \quad (5.11)$$

где $К'_П$, $К'_{НС}$, $К'_Б$ – значения коэффициентов учета сложности алгоритма ПП и вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно.

Таким образом, для рассматриваемого случая:

$$К_К = 1,07;\ К_Р = 1,44;\ К_Я = 1,00;\ К_З = 0,6$$

$$t_{РРЗ} = 8\ [чел.-дни],\ t_{РРП} = 54\ [чел.-дни]$$

$$К'_П = 1,1;\ К'_{НС} = 0,65;\ К'_Б = 0,4$$

$$К_{ИА} = (1,1 * 1 + 0,65\ *10 + 0,4 * 2)\ /\ (1 + 10 + 2) = 0,64$$

$$\tau_{РП} = (8 + 54) * 1,07 * 1,44 * 1,0 * 0,6 * 0,64 = 37\ [чел.-дни]$$

Поскольку при разработке ПП стадии «Технический проект» и «Рабочий проект» объединяются в стадию «Технорабочий проект», то трудоемкость ее выполнения τТРП определяется по формуле

$$\tau_{ТРП} = 0,85\tau_{ТП} + \tau_{РП} \quad (5.12)$$

То есть для рассматриваемого случая по формуле 5.12:

$$\tau_{ТРП} = 0,85 * 32 + 37 \approx 64\ [чел.-дни]$$

Трудоемкость выполнения стадии «Внедрение» рассчитывается по формуле

$$\tau_B = (t_{ВРЗ} + t_{ВРП}) \cdot K_K \, K_P \, K_З \quad (5.13)$$

$t_{ВРЗ}$, $t_{ВРП}$ – норма времени, затрачиваемого разработчиком постановки задач и разработчиком ПП соответственно на выполнение процедур внедрения ПП, чел.-дни

Таким образом, для рассматриваемого случая:

$K_K = 1,07; K_P = 1,26; K_З = 0,6$

$$t_{ВРЗ} = 25 \text{ [чел.-дни]}; \; t_{ВРП} = 73 \text{ [чел.-дни]}$$

$$\tau_B = (25 + 73) * 1,07 * 1,26 * 0,6 = 80 \text{ [чел.-дни]}$$

Подставив полученные данные в формулу 5.1, получим:

$$\tau_{ПП} = \tau_{ТЗ} + \tau_{ЭП} + \tau_{ТРП} + \tau_B = 47 + 67 + 64 + 80 = 258 \text{ [чел.-дни]}$$

Полученные результаты можно отобразить в таблице 5.2.

Таблица 5.2 — Трудоемкости по стадиям разработки проекта

Этап разработки	Трудоемкость этапа
ТЗ	47
ЭП	67
ТРП	64
Внедрение	80
Всего	258

5.4 Определение численности исполнителей

Средняя численность исполнителей при реализации проекта разработки и внедрения ПО определяется соотношением:

$$N = \frac{Q_p}{F} \quad (5.14)$$

где Q_P — это затраты труда на выполнение проекта (разработка и внедрение ПО),

а F — фонд рабочего времени.

Величина фонда рабочего времени определяется соотношением:

$$F = T * F_M \quad (5.15)$$

где T – время выполнения проекта в месяцах, равное 4 месяцам,

F_M – фонд времени в текущем месяце, который рассчитывается из учета общества числа дней в году, числа выходных и праздничных дней:

$$F_м = (t_р * (D_К – D_В – D_П)) / 12 \quad (5.16)$$

где – $t_р$ — это продолжительность рабочего дня, $D_К$ – число дней в году, $D_В$ – число выходных дней в году, $D_П$ – число праздничных дней в году.

Подставив в формулы 5.14, 5.15 и 5.16 величины, соответствующие этому проекту и 2014 году, получим:

$$F_M = (8 * (365 - 118)) / 12 = 165$$

$$F = 4 * 165 = 660$$

$$N = (8 * 258) / 660 = 3$$

Таким образом, число исполнителей проекта – 3.

5.5 Календарный план-график

Планирование и контроль хода выполнения разработки проводится по календарному графику выполнения работ. Планирование процесса разработки представлено в таблице 5.3.

Таблица 5.3 — Планирование процесса разработки

Этап разработки	Трудоемкость этапа	Работник	Распределение трудоемкости
ТЗ	47	1	31
		2	8
		3	8
ЭП	67	1	39
		2	9
		3	9
ТРП	64	1	22
		2	21
		3	21
Внедрение	80	1	26
		2	27
		3	27

Далее в таблице 5.4 представлен календарный ленточный план выполнения проекта.

Таблица 5.4 - календарный ленточный план выполнения проекта

Этапы													
ТЗ	31												
	8												
	8												
ЭП					39								
					9								
					9								
ТРП									22				
									21				
									21				
Внедрение											26		

Таким образом, при распараллеливании работы можно добиться сокращения срока разработки и внедрения программного продукта с 258 до 119 дней.

5.6 Расчет затрат на разработку программного продукта

Затраты на выполнение проекта состоят из затрат на:

- заработную плату исполнителям;
- закупку или аренду оборудования;
- организацию рабочих мест;
- накладные расходы.

Затраты на заработную плату и отчисления на социальное страхование в пенсионный фонд и фонд обязательного медицинского страхования составляют 30%.

Для всех исполнителей предполагается оклад в размере 40000 рублей в месяц.

В таблице 5.5 приведены затраты на заработную плату исполнителям и отчисления на социальные нужды.

Таблица 5.5 — Затраты на зарплату исполнителям и отчисления на социальные нужды

Месяц	Исполнитель	Рабочие дни	Зарплата (руб.)	Отчисления (руб.)
Февраль	1	22	40000	12000
	2	8	14545	4363
	3	8	14545	4363
Март	1	22	40000	12000
	2	9	16363	4909
	3	9	16363	4909
Апрель	1	22	40000	12000
	2	-		
	3	-		
Май	1	22	40000	12000
	2	18	36000	10800
	3	18	36000	10800
Июнь	1	22	40000	12000
	2	21	38181	11454
	3	21	38181	11454
Июль	1	8	14545	4363
	2	9	16363	4909
	3	9	16363	4909
		Всего	457 449	

Расходы на материалы, необходимые для разработки программной продукции, указаны в таблице 5.6.

Таблица 5.6 — Материальные затраты

№	Наименование материала	Единица измерения	Кол-во	Цена за единицу (руб.)	Сумма (руб.)
1	Бумага А4	Пачка 500 л.	1	200	200
2	Картридж струйного принтера	Шт.	1	1000	1000
3	Канцелярские товары	Упаковка	1	200	200
				Всего:	1400

В работе над проектом используется специальное оборудование – персональные электронно-вычислительные машины (ПЭВМ) в количестве 3 шт. Стоимость одной ПЭВМ составляет 16000 рублей. Месячная норма амортизации ПЭВМ К = 4% (срок службы 25 месяцев). Также необходима компьютерная мебель (столы — 5лет с КА = 1,67%, стулья — 2 года с КА = 4,17%) и компьютерные аксессуары со сроком службы 2 года и с КА = 4,17%. Затраты на амортизацию представлены в таблице 5.7.

Таблица 5.7 — Затраты на амортизацию

Наименование оборудования	Цена (руб.)	Кол-во (шт.)	$К_А$ (%)	Время использования (мес.)	Сумма отчислений (руб.)
ПЭВМ	16000	3	4	6	11520
Стол	3000	3	1,67	6	901
Стул	1000	3	4,17	6	300
Мышь	500	3	4,17	6	150
				Всего:	12871

Общая величина затрат: 457499 + 1400 + 12871 = 471770 (руб.)

5.7 Расчет экономической эффективности

Основными показателями экономической эффективности является чистый дисконтированный доход (ЧДД) и срок окупаемости вложенных средств.

Чистый дисконтированный доход определяется по формуле:

$$ЧДД = \sum_{t=0}^{T} (R_t - З_t) \cdot \frac{1}{(1+E)^t} \quad (5.17)$$

где T – горизонт расчета по месяцам;

t – период расчета;

R_t – результат, достигнутый на t шаге (стоимость);

$З_t$ – затраты;

E – приемлемая для инвестора норма прибыли на вложенный капитал.

Коэффициент E установим равным ставке рефинансирования ЦБ РФ – 8,5% годовых. В результате анализа рынка программной продукции, аналогичной разрабатываемой, планируется продажа 300 единиц ПП каждые два месяца. Планируемая цена ПП составляет 1000 рублей. Предполагаемые накладные расходы, связанные с реализацией составят 500 рублей в месяц.

Таблица 5.8 – Чистый дисконтированный доход

Месяц	Текущие затраты (руб)	Затраты с начала года (руб)	Текущий доход (руб)	Общий доход (руб)	ЧДД в месяц (руб)
1	78628,33333	78628,33333	0	0	-77919,26799
2	78628,33333	157256,6667	0	0	-77216,59696
3	78628,33333	235885	0	0	-76520,26257
4	78628,33333	314513,3333	0	0	-75830,20768
5	78628,33333	393141,6667	0	0	-75146,37566
6	78628,33333	471770	0	0	-74468,7104
7	500	472270	150000	150000	140314,2404
8	500	472770	150000	300000	139048,8954
9	500	473270	150000	450000	137794,9613
10	500	473770	150000	600000	136552,335
11	500	474270	150000	750000	135320,9147
				ЧДД	96609,01083

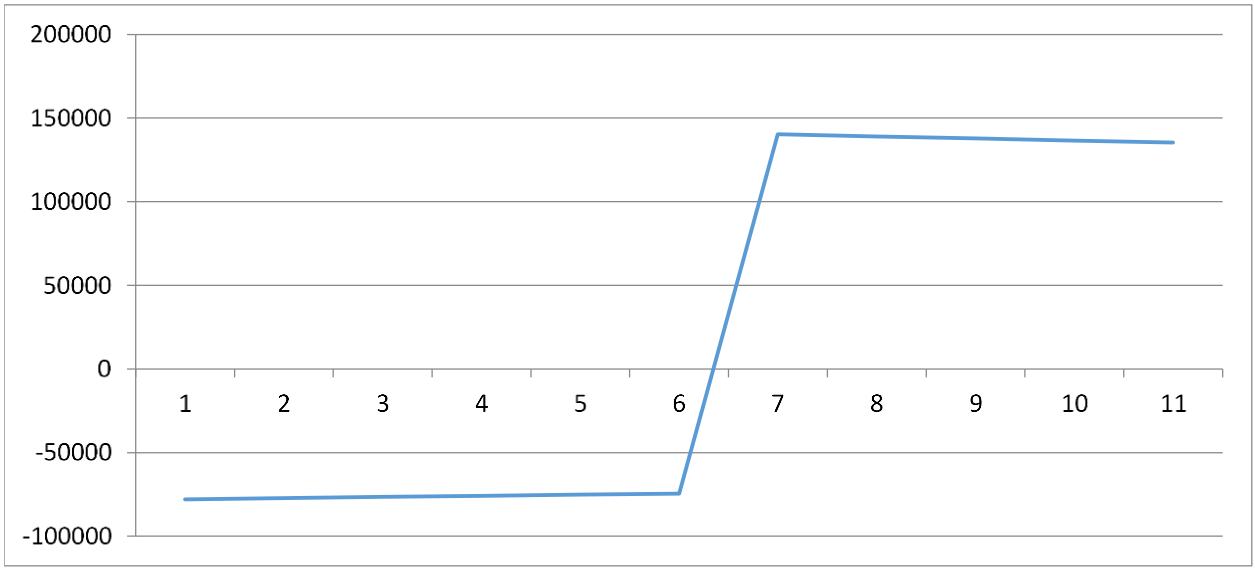


Рисунок 5.1 Чистый дисконтированный хоход

В результате срок окупаемости продукции составляет 11 месяцев, с последующим доходом 150 000 руб в месяц.

Выводы

Принимая во внимание экономический анализ в предшествующих пунктах, разработка и реализация ПП оказывается рентабельной. Разрабатываемый программный продукт имеет аналоги, но менее точные по результатам обработки спам сообщений, а также они мало адаптированы для работы с русским языком и самой спецификой смс рассылок, в результате сложно оценить спрос рынка и затраты на реализацию продукции. В связи с этим приведенные выше расчеты является оценочными и оптимистичными.

6. Безопасность жизнедеятельности и охрана труда

6.1 Анализ опасных и вредных факторов при разработке программного обеспечения и мероприятия по их устранению

Процесс создания программного продукта не возможен без использования персональных электронно-вычислительных машин (ПЭВМ). Эксплуатация ПЭВМ сопряжена с рядом опасных факторов, воздействию которых подвергаются программисты. К таким факторам относят электро-магнитное излучение, отраженный свет и блики, вибрация, шум.

В данном разделе рассматриваются основные виды опасных и вредных факторов, а также допустимые нормы излучений, требования к освещенности, помещению в целом, уровням шума на рабочем месте, которые регламентируются СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации рабочего места».

6.1.1 Требования к помещению для работы с ПЭВМ

Работа программистов должна проходить в помещении, которое не наносит вреда здоровью и создает комфортные условия труда. Следует придерживаться общих рекомендаций при выборе помещения и материалов для его внутренней отделки:

- Окна в помещениях, где эксплуатируется вычислительная техника, преимущественно должны быть ориентированы на север и северо-восток. Оконные проемы должны быть оборудованы

регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

- Для внутренней отделки интерьера помещений, где расположены ПЭВМ, должны использоваться диффузно отражающие материалы с коэффициентом отражения для потолка - 0,7 - 0,8; для стен - 0,5 - 0,6; для пола - 0,3 - 0,5.

- Помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации.

- Не следует размещать рабочие места с ПЭВМ вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ.

6.1.2 Общие требования к организации рабочих мест пользователей ПЭВМ

Разработчики программного продукта в первую очередь подвержены воздействию излучения от экрана мониторов, поэтому экран видеомонитора должен находиться от глаз разработчика на расстоянии 600 - 700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Для комфортной организации рабочих мест расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора), должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м. СанПин 2.2.2/2.4.1340-03 рекомендует

изолировать рабочие места программистов друг от друга перегородками высотой 1,5 - 2,0 м.

Для снижения утомляемости сотрудников рабочий кресло должно быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья.

6.1.3 Требования к микроклимату

Программист большую часть рабочего времени проводит за ПЭВМ. Процесс написания программного продукта требует повышенной концентрации, внимания, умственных усилий, что сопровождается нервно-эмоциональным напряжением. Данный вид работ относится к категории 1а. Такие помещения согласно СанПиН должны обеспечиваться оптимальными параметрами микроклимата. Оптимальные параметры микроклимата указаны в таблице 6.1.

К вредным факторам при работе с ЭВМ относят также и запыленность

Таблица 6.1

	Температура воздуха, ^{o}C	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	22-24	40-60	0,1
Теплый	23-25	40-60	0,1

помещения.

Этот фактор усугубляется влиянием на частицы пыли электростатических полей персональных компьютеров. В качестве норм по уменьшению запыленности, поддержания температуры и важности следует использовать систему кондиционирования. Необходимо производить влажную уборку и проветривать помещение.

6.1.4Требования к уровню шума и вибрации

В отделе разработки программ при выполнении основных или вспомогательных работ с использованием ПЭВМ уровень шума на рабочем месте не должен превышать 50 дБ, а уровни вибрации на рабочих местах не должны превышать предельно допустимых значений, указанных в таблице 6.2.

Таблица 6.2. Допустимые нормы вибрации на рабочих местах с ВДТ и ПЭВМ

Среднегеометрические частоты октавных полос, Гц	Допустимые значения по виброскорости	
	м/с	дБ
2	4,5x10	79
4	2,2x10	73
8	1,1x10	67
16	1,1x10	67
31,5	1,1x10	67
63	1,1x10	67
Корректированные значения и их уровни в дБ	2,0x10	72

К внутренним источникам шума относятся вентиляторы, принтеры и другие периферийные устройства ЭВМ.

Мощные источники шума такие как сервера должны располагаться в отдельных помещениях с использованием средств звукоизоляции (звукопоглощающих материалов для облицовки стен и потолка помещения) и толстых перегородок (стен).

К внешним источникам шума можно отнести шум с улицы и соседних помещений. Для снижения шума улицы следует использовать более толстые шумопоглощающие стеклопакеты, а проветривание помещения осуществлять

посредством системы кондиционирования. Для уменьшения шума соседних комнат следует использовать облицовку стен звукопоглощающими материалами.

6.1.5 Требования к освещенности

Наиболее важным условием эффективной работы программистов и пользователей является соблюдение оптимальных параметров системы освещения в рабочих помещениях.

Естественное освещение осуществляется через светопроемы, ориентированные в основном на север и северо-восток (для исключения попадания прямых солнечных лучей на экраны компьютеров) и обеспечивает коэффициент естественной освещенности (КЕО) не ниже 1,5%.

В качестве искусственного освещения проектом предусмотрено использование системы общего равномерного освещения. В соответствии с СанПиН 2.2.2/2.4.1340-03 освещенность на поверхности рабочего стола находится в пределах 300-500 лк. Разрешается использование светильников местного освещения для работы с документами (при этом светильники не должны создавать блики на поверхности экрана).

Правильное расположение рабочих мест относительно источников освещения, отсутствие зеркальных поверхностей и использование матовых материалов ограничивает прямую (от источников освещения) и отраженную (от рабочих поверхностей) блескость. При этом яркость светящихся поверхностей не превышает 200 кд/кв.м, яркость бликов на экране ПЭВМ не превышает 40 кд/кв.м, и яркость потолка не превышает 200 кд/кв.м.

В соответствии с СанПиН 2.2.2/2.4.1340-03 проектом предусмотрено использование люминесцентных ламп типа ЛБ в качестве источников света при искусственном освещении. В светильниках местного освещения допускается применение ламп накаливания.

Применение газоразрядных ламп в светильниках общего и местного освещения обеспечивает коэффициент пульсации не более 5%.

6.1.6 Требования к уровням электромагнитных полей на рабочих местах, оборудованных ПЭВМ

Программисты во время использования ПЭВМ подвергаются воздействию такого вредного фактора как электромагнитное излучение. Уровни электромагнитных полей нормируются по СанПиН 2.2.2/2.4.1340-03 и представлены в таблицах ниже:

Таблица 6.3. Временные допустимые уровни ЭМП, создаваемых ПЭВМ

Наименование параметров		ВДУ ЭМП
Напряженность электрического поля	в диапазоне частот 5 Гц - 2 кГц	25 В/м
	в диапазоне частот 2 кГц - 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц - 400 кГц	25 нТл
Электростатический потенциал экрана видеомонитора		500 В

Таблица 6.4. Визуальные параметры ВДТ, контролируемые на рабочих местах

N п/п	Параметры	Допустимые значения
1	Яркость белого поля	Не менее 35 кд/кв. м
2	Неравномерность яркости рабочего поля	Не более +/- 20%
3	Контрастность (для монохромного режима)	Не менее 3:1
4	Временная нестабильность изображения (мелькания)	Не должна фиксироваться
5	Пространственная нестабильность изображения (дрожание)	Не более 2 х 1E(-4L), где L - проектное расстояние наблюдения, мм

6.1.7 Требования к визуальным параметрам ВДТ, контролируемым на рабочих местах

Для организации комфортных условий на рабочих местах программистов следует обращать внимание на визуальные параметры

окружения - удобство считывания информации. Плохая организация рабочего места, обусловленная неудобствами восприятия информации, приводит к ухудшению здоровья сотрудников, быстрой утомляемости, раздражительности.

Требования к визуальным параметрам, их внешнему виду, дизайну, возможности настройки представлены в СанПиН 2.2.2/2.4.1340-03. Визуальные эргономические параметры монитора и пределы их изменений приведены в таблице ниже:

Таблица 6.5. Визуальные эргономические параметры ВДТ и пределы их изменений

Наименование параметров	Пределы значений параметров	
	миним. (не менее)	максим. (не более)
Яркость знака (яркость фона), кд/кв.м (измеренная в темноте)	35	120
Внешняя освещенность экрана, лк	100	250
Угловой размер знака, угл. мин.	16	60

В данном проекте используются мониторы с разрешением экрана от 1920 * 1080, диагональю от 21 дюйма, с матовой поверхностью экрана, полупрофессиональной матрицей TFT IPS, углом обзора не менее 178 градусов, а также возможность настройки положения монитора по горизонтали и вертикали. Набор вышеперечисленных параметров обеспечивает комфортное восприятие как текстовой, так и графической информации.

6.2 Расчет уровня шума в серверной комнаты

В качестве источника шума рассматривается сервер. Схема расположения источников шума представлена на рисунке 6.1.

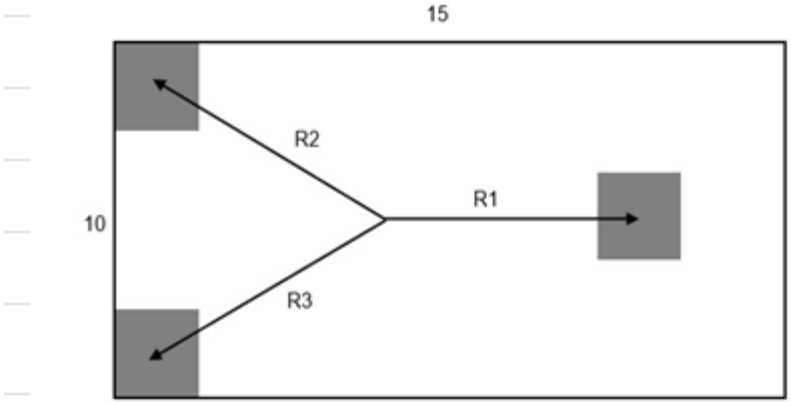


Рисунок 6.1. Расположение источников шума (серверов)

Уровни звуковой мощности каждого из источников шума представлены в таблице 6.6. Уровни звуковой интенсивности источников шума и расчеты даны в таблице 7.

Таблица 6.6. Уровни звуковой мощности источников шума.

№ источника	Расположение источника	Расстояние до источника	Уровни звуковой мощности, Lw								
			63	125	250	500	1000	2000	4000	8000	Площадь S
1	на полу	5	80	84	83	87	84	82	94	96	157
2	на полу	5	81	82	83	84	83	81	80	77	39,25

Таблица 6.7. Уровни звуковой интенсивности источников шума.

Используемые данные для расчётов	Октавные полосы	63	125	250	500	1000	2000	4000	8000
	y	0,8	0,75	0,7	0,8	1	1,4	1,8	2,5
	B	30	28,125	26,25	30	37,5	52,5	67,5	93,75
Номер источника шума	1	71,45205	75,71994	75,00715	78,45205	74,53217	71,16769	82,17094	82,90516
	2	73,00881	74,24533	75,50075	76,00881	74,21049	71,07185	69,28073	65,3343

Характеристики помещения с источниками шума отображены на рисунке рисунке 6.2.

Габариты комнаты	м / м3
Длина	10
Ширина	15
Высота	5
объем	750

Рисунок 6.2. Характеристики помещения с серверами

Предельный спектр уровня шума в помещении с серверами представлен на рисунке 6.3.

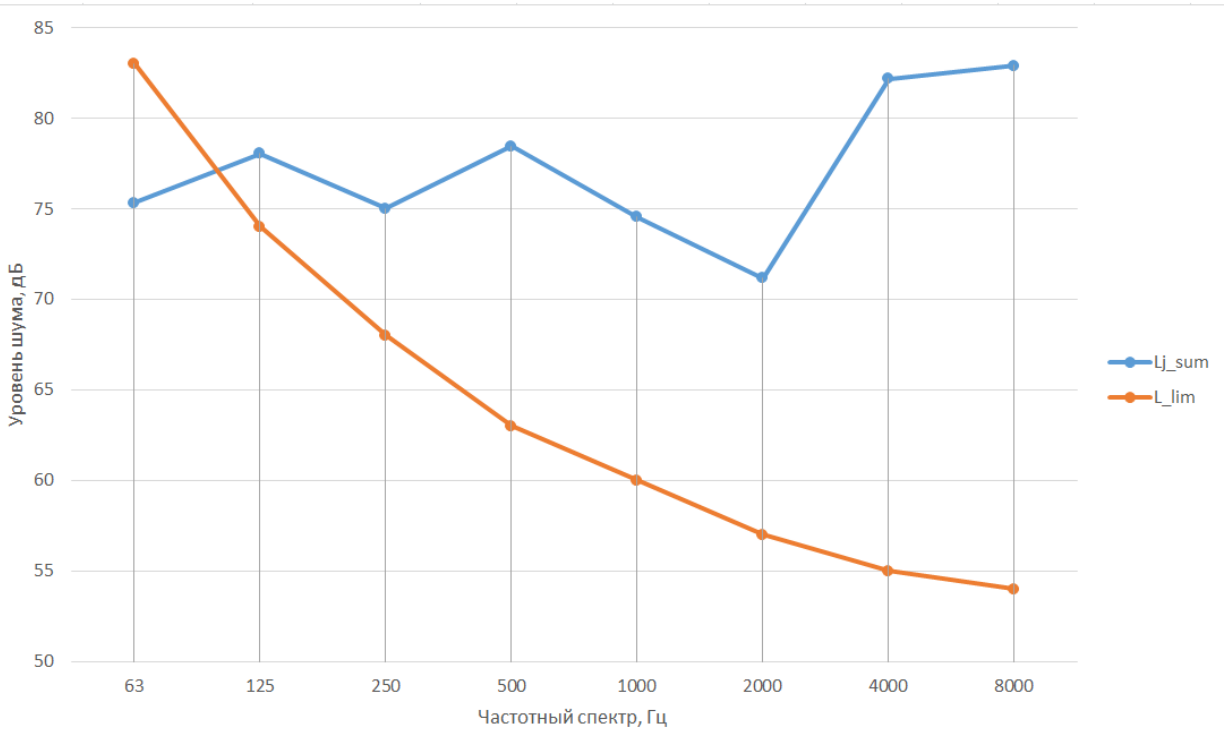


Рисунок 6.3. Предельный спектр суммарной звуковой интенсивности в различных октавных полосах.

Полученные при расчёте уровни шума для заданного помещения и источников превышают предельно допустимые величины уровней шума. Так как увеличение расстояния от источника до расчётной точки слабо влияет на уровень шума, а изменение размеров комнаты не снижает уровень шума до допустимых значений, рекомендуется использовать дополнительные методы борьбы с шумом такие как звукоизоляция. Звукоизоляция помещения осуществляется посредством звукоизолирующих материалов, облицовки стен, кожухов и кабин, заполнением воздушного пространства в двойных легких перегородках звукопоглощающими материалами, повышением воздухонепроницаемости преграды или применением экранов.

Так как уровни шума превышают допустимые уровни шума, то программисты не должны работать в данном помещении, а во время нахождения рядом с серверами следует использовать индивидуальные средства защиты (например, беруши).

6.3 Расчет освещенности офиса

В данном проекте были разработаны план офисного помещения программистов, расстановка мебели, а также расчет освещения производится с помощью программы DIALux 4.12.01.

План помещения представлен на рисунке 6.4. Ведомость объектов помещения представлена в таблице 6.8

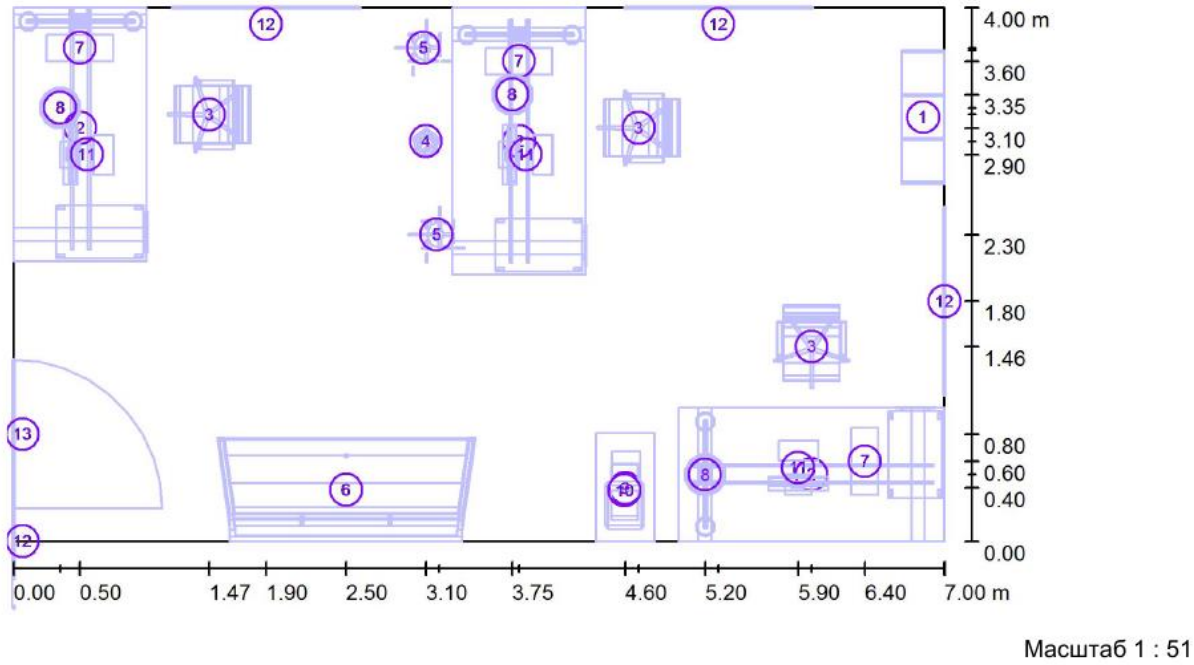


Рисунок 6.4.. План помещения.

.

Таблица 6.8. Ведомость объектов

Номер объекта	Количество	Наименование
1	1	Стеллаж
2	3	Контейнер
3	3	Вращающееся кресло
4	1	растение
5	2	Растение
6	1	Диван
7	3	Компьютер
8	3	Корзина для бумаг
9	1	Малый контейнер
10	1	Принтер
11	3	Экран TFT IPS и клавиатура
12	4	Окно
13	1	Дверь

Схема расположения светильников на рисунке 6.5.

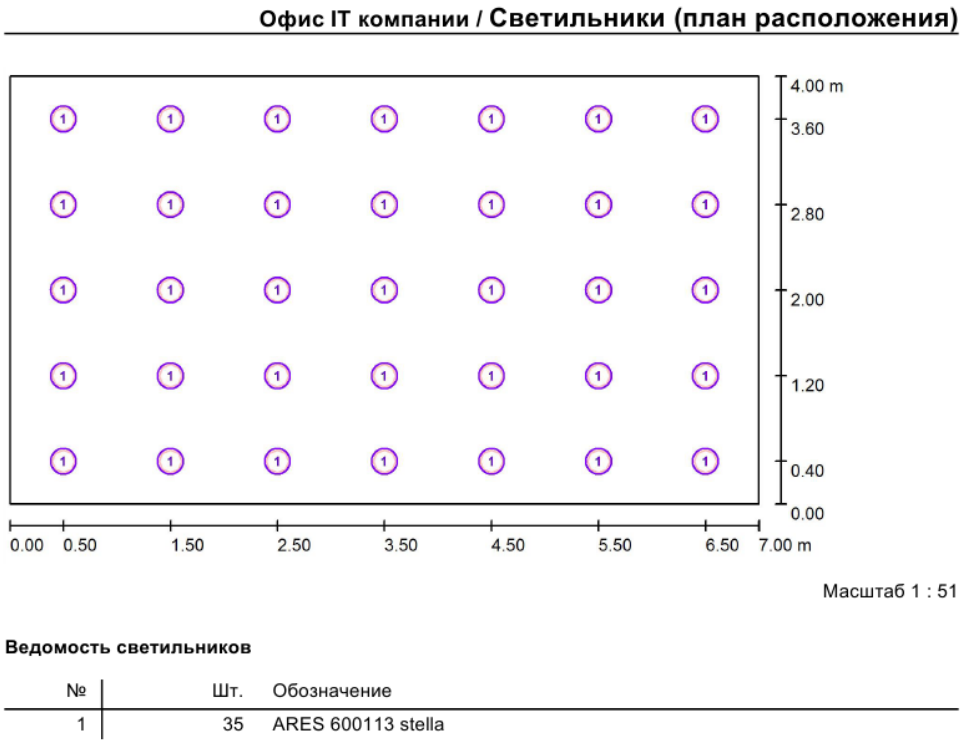
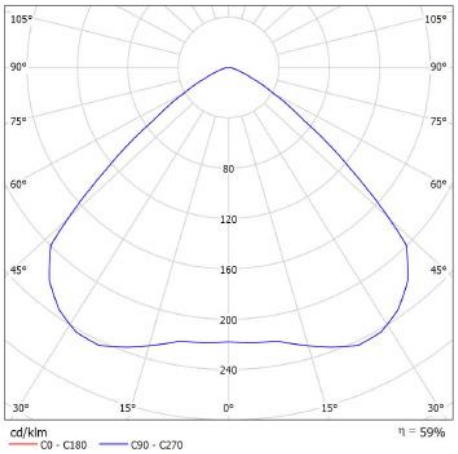


Рисунок 6.5. Схема расположения светильников

Паспорт используемого типа светильника представлен на рисунке 6.6.

ARES 600113 stella / Паспорт светильника

Место выхода света 1:



Классификация светильников по CIE: 100
CIE Flux Code: 62 95 99 100 59

Recessed luminaire for outdoor installation on ceiling. Configuration: phospho-chromatised and polyester powder coated die-cast aluminium structure with aluminium (four colours available) finish frame, moulded silicone gasket and tempered transparent or sandblasted glass.

Место выхода света 1:

Оценка экранирования по UGR

ρ Потолок		70	70	50	50	30	70	70	50	50	30
ρ Стенки		50	30	50	30	30	50	30	50	30	30
ρ Полы		20	20	20	20	20	20	20	20	20	20
Величина помещения X	Y	Направление взгляда поперек к оси ламп					Направление взгляда вдоль к оси ламп				
2H	2H	18.5	19.6	18.7	19.8	20.0	18.5	19.6	18.7	19.8	20.0
	3H	18.5	19.5	18.8	19.7	20.0	18.5	19.5	18.8	19.7	20.0
	4H	18.4	19.4	18.8	19.6	19.9	18.4	19.4	18.8	19.6	19.9
	6H	18.4	19.2	18.7	19.5	19.8	18.4	19.2	18.7	19.5	19.8
	8H	18.4	19.2	18.7	19.5	19.8	18.4	19.2	18.7	19.5	19.8
	12H	18.3	19.1	18.7	19.4	19.7	18.3	19.1	18.7	19.4	19.7
4H	2H	18.5	19.4	18.8	19.7	19.9	18.5	19.4	18.8	19.7	19.9
	3H	18.5	19.3	18.9	19.6	19.9	18.5	19.3	18.9	19.6	19.9
	4H	18.5	19.2	18.9	19.5	19.9	18.5	19.2	18.9	19.5	19.9
	6H	18.5	19.1	18.9	19.4	19.8	18.5	19.1	18.9	19.4	19.8
	8H	18.5	19.0	18.9	19.4	19.8	18.5	19.0	18.9	19.4	19.8
	12H	18.4	18.9	18.9	19.3	19.7	18.4	18.9	18.9	19.3	19.7
8H	4H	18.4	19.0	18.9	19.4	19.8	18.4	19.0	18.9	19.4	19.8
	6H	18.4	18.8	18.9	19.3	19.7	18.4	18.8	18.9	19.3	19.7
	8H	18.4	18.7	18.8	19.2	19.7	18.4	18.7	18.8	19.2	19.7
	12H	18.3	18.7	18.8	19.1	19.6	18.3	18.7	18.8	19.1	19.6
12H	4H	18.4	18.9	18.9	19.3	19.7	18.4	18.9	18.9	19.3	19.7
	6H	18.4	18.7	18.8	19.2	19.7	18.4	18.7	18.8	19.2	19.7
	8H	18.3	18.7	18.8	19.1	19.6	18.3	18.7	18.8	19.1	19.6

Варьирование позиции наблюдателя для расстояний между светильниками S

S = 1.0H	+1.4 / -1.9	+1.4 / -1.9
S = 1.5H	+2.4 / -4.4	+2.4 / -4.4
S = 2.0H	+4.0 / -6.6	+4.0 / -6.6
Стандартная таблица	BK01	BK01
Корректировочное слагаемое	-1.2	-1.2

Скорректированные индексы экранирования, отнесенные к 1500lm Общий световой поток

Рисунок 6.6. Паспорт светильника

План помещения с указанием изолиний освещенности на рисунке 6.7.

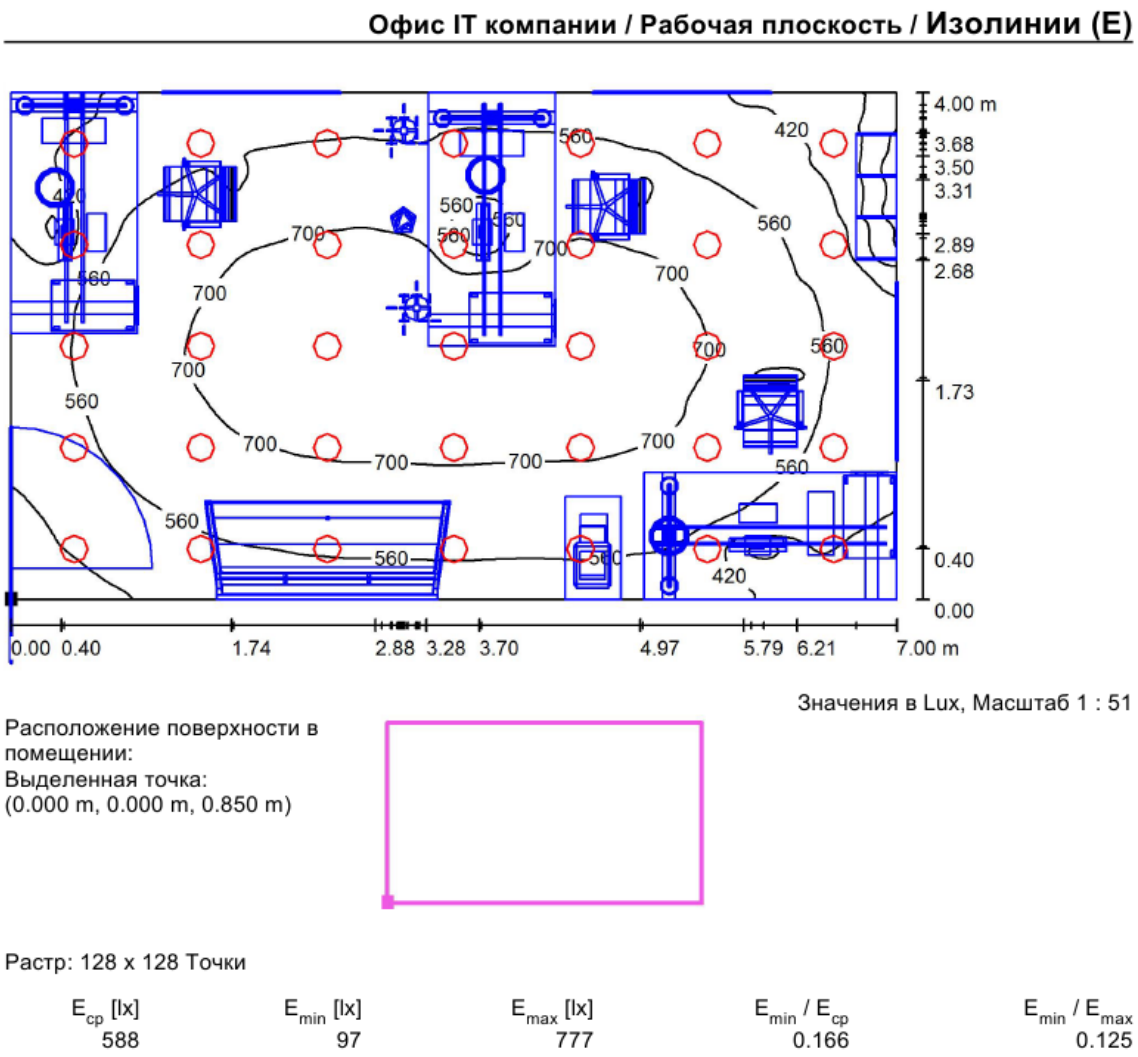


Рисунок 6.7. План помещения с указанием изолиний освещенности

Светотехнические результаты расчетов представлены на рисунке 6.8.

Офис IT компании / Светотехнические результаты

Общий световой поток: 30865 lm
Общая мощность: 3500.0 W
Краевая зона: 0.000 m

Поверхность	Средние освещенности [lx]			Коэффициент отражения [%]	Средние Яркость [cd/m²]
	Напрямую	Опосредовано	Всего		
Рабочая плоскость	465	122	588	/	/
Полы	258	85	343	20	22
Потолок	0.00	148	148	70	33
Стенка 1	142	118	259	50	41
Стенка 2	104	101	205	53	35
Стенка 3	147	117	264	54	45
Стенка 4	135	124	259	50	41

Равномерность на рабочей плоскости
E_{min} / E_{cp}: 0.166 (1:6)
E_{min} / E_{max}: 0.125 (1:8)

Удельная подсоединенная мощность: 125.00 W/m² = 21.27 W/m²/100 lx (Поверхность основания: 28.00 m²)

Рисунок 6.8. Светотехнические результаты расчетов.

Уровни освещенности поверхностей помещения указаны на рисунке 6.9.

Высота помещения: 2.800 m, Монтажная высота: 3.015 m — Значения в Lux, Масштаб 1:

Поверхность	ρ [%]	E_{cp} [lx]	E_{min} [lx]	E_{max} [lx]	E_{min} / E
Рабочая плоскость	/	588	97	777	0.1
Полы	20	343	20	613	0.0
Потолок	70	148	106	194	0.7
Стенки (4)	52	251	29	558	

Рабочая плоскость:
Высота: 0.850 m
Растр: 128 x 128 Точки
Краевая зона: 0.000 m

Ведомость светильников

№	Шт.	Обозначение (Поправочный коэффициент)	Φ (Светильник) [lm]	Φ (Лампы) [lm]	P [
1	35	ARES 600113 stella (1.000)	882	1500	100
			Всего: 30865	Всего: 52500	3500

Удельная подсоединенная мощность: 125.00 W/m² = 21.27 W/m²/100 lx (Поверхность основания: 28.00 m²)

Рисунок 6.9 Уровни освещенности поверхностей помещения

3D визуализация офиса на рисунке 6.10.

Рисунок 6.10. 3D визуализация офиса

Заключение

В ходе проделанной работы был осуществлен анализ предметной области, рассмотрены существующие программные реализации фильтрации спама, рассмотрены алгоритмы автоматической обработки и классификации текста, выявлены дополнительный признаки спама, осуществлено формирование баз данных спам-сообщений и обычных сообщений. Были проанализированы недостатки алгоритма наивного байесовского классификатора, предложены пути увеличения точности и полноты классификации.

В соответствии с техническим заданием на квалификационную работу, разработана структура библиотеки фильтрации спам-сообщений, реализованы алгоритмы выделения смысловых элементов текста, нормализации слов и классификации текста.

В соответствии с выбранной структурой было реализовано программное обеспечение, выполняющие поставленные перед ним задачи фильтрации спам-сообщений. Программное обеспечение позволяет определять класс сообщения, а в случае затруднения при классификации информирует о том, что не удалось определить класс сообщения.

Список литературы

1. Проект Почта@mail.ru. [Электронный ресурс] //Официальный сайт. – Режим доступа: https://e.mail.ru – Загл. с экрана (дата обращения 22.06.2014)
2. Продукт Anti-Spam KasperskyLab [Электронный ресурс] //Официальный сайт. – Режим доступа: http://www.kaspersky.ru/products/business/targeted-solutions/anti-spam – Загл. с экрана (дата обращения 22.06.2014)
3. Анти-спам фильтр AVEGA Spamprotexx [Электронный ресурс] //Официальный сайт. – Режим доступа: http://www.spamprotexx.ru/ Загл. с экрана (дата обращения 22.06.2014)
4. Martin, M., On-Line Support Vector Machines for Function Approximation, in Technical Report(LSI-02-11-R). 2002, Catalunya: Department of Software, Universitat Politecnica de Catalunya
5. Houshmand Shirani-Mehr, SMS Spam Detection using Machine Learning Approach
6. РосСпам [Электронный ресурс] //Официальный сайт. – Режим доступа: https://rosspam.org Загл. с экрана (дата обращения 15.03.2014)
7. Арамба [Электронный ресурс] //Официальный сайт. – Режим доступа: https://aramba.ru Загл. с экрана (дата обращения 17.03.2014)
8. Erdogmus, Hakan; Morisio, Torchiano. On the Effectiveness of Test-first Approach to Programming. Proceedings of the IEEE Transactions on Software Engineering, 31(1). January 2005. (NRC 47445). — «We found that test-first students on average wrote more tests and, in turn, students who wrote more tests tended to be more productive.»
9. The Standford Natural Language Processing Group [Электронный ресурс] //Официальный сайт. – Режим доступа: http://nlp.stanford.edu/IR-book/html/htmledition/index-1.html Загл. с экрана (дата обращения 5.04.2014)

10. Сегаран Т. Программируем коллективный разум.-Пер. с англ. – СПб:Символ-Плюс,2008.-368с.,ил.
11. Полный словарь сокращений, акронимов, аббревиатур и сложносоставных слов русского языка [Электронный ресурс] //Официальный сайт. – Режим доступа: http://www.sokr.ru Загл. с экрана (дата обращения 29.03.2014)
12. Википедия – электронная энциклопедия [Электронный ресурс] //Официальный сайт. – Режим доступа: http://ru.wikipedia.org/wiki/Частица_в_русском_языке Загл. с экрана (дата обращения 29.03.2014)
13. Полезные справочные и занимательные матриалы по русскому языку[Электронный ресурс] //Официальный сайт. – Режим доступа: http://tutrus.com/morpholog/razryadyi-mestoimeniy Загл. с экрана (дата обращения 29.03.2014)

Приложение А. Пользовательский интерфейс.

Начало работы

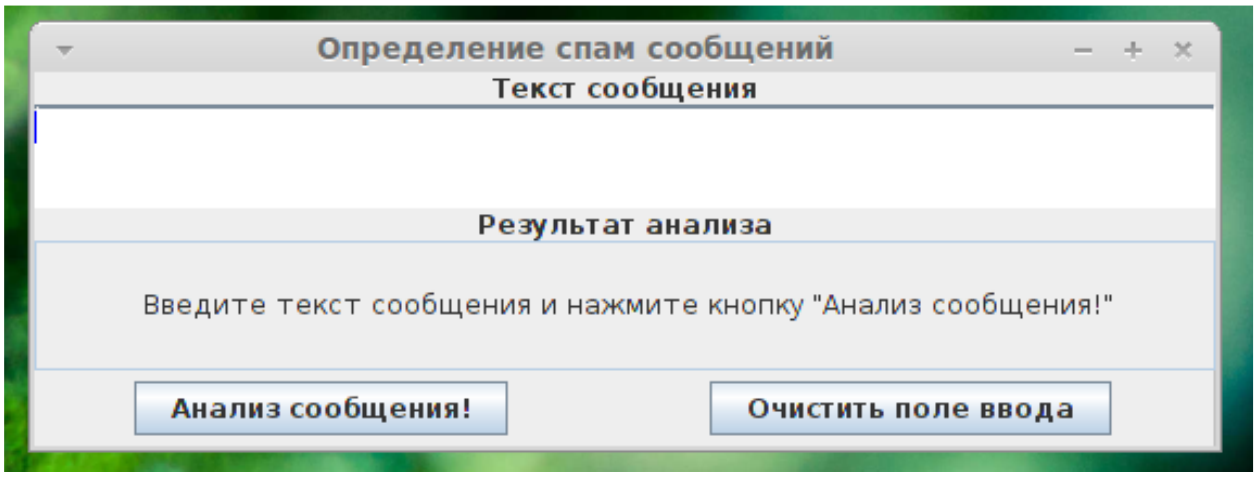


Рисунок А1 Начало работы

Пример распознанного спам-сообщения

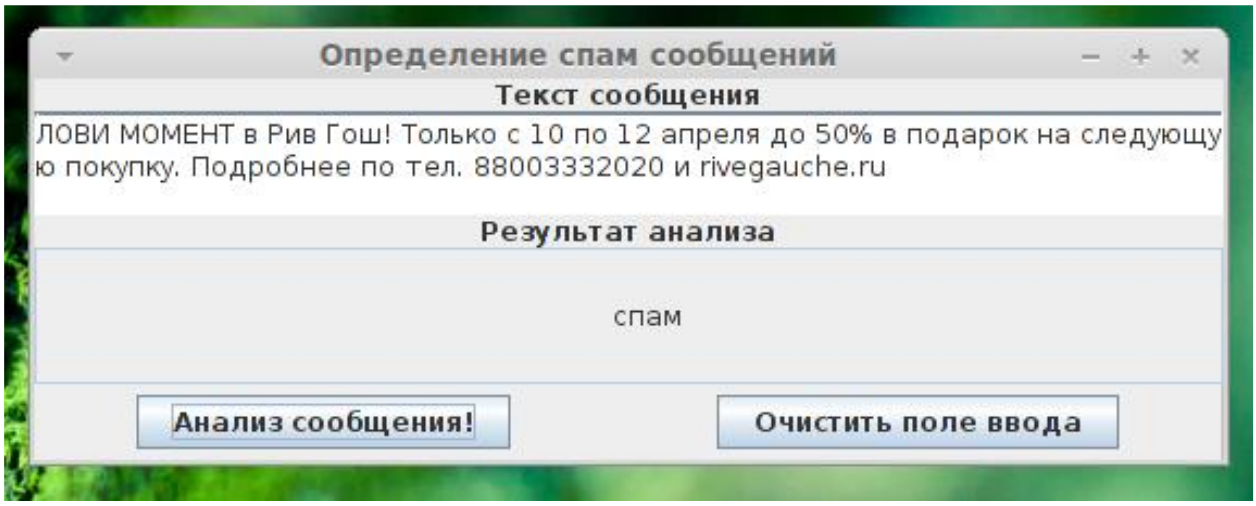


Рисунок А2 Распознанное спам-сообщение

Пример правильной классификации обычного текста

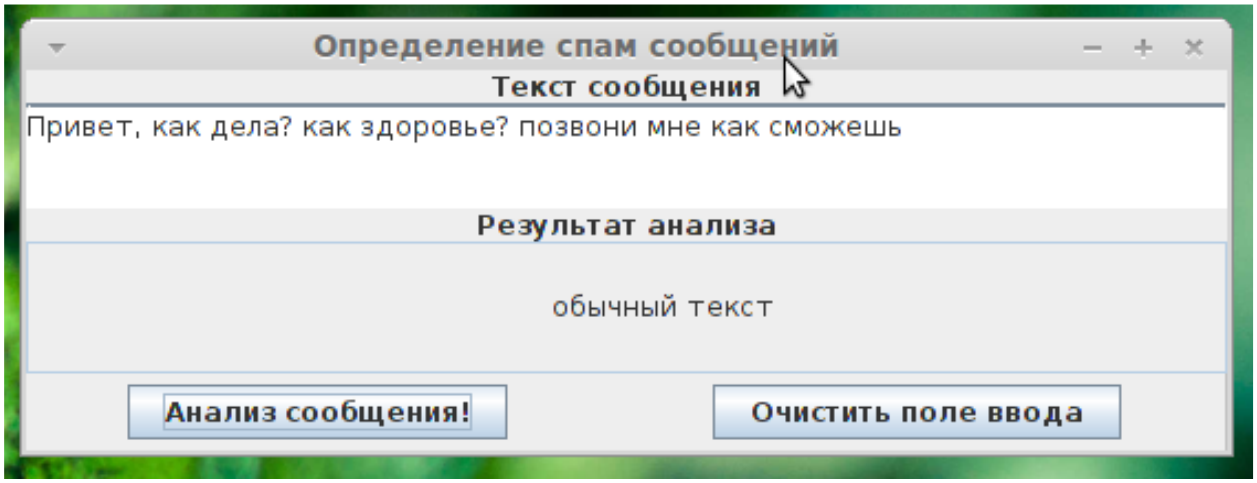


Рисунок А3 Пример правильной классификации обычного текста

Пример не распознанного сообщения

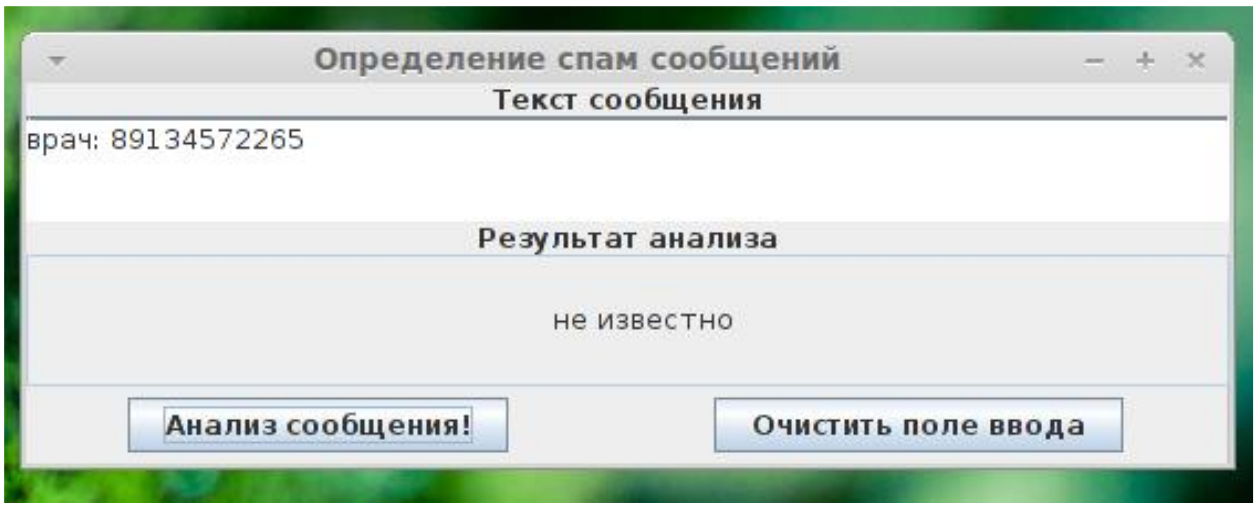


Рисунок А4 Пример не распознанного сообщения