*Московский Государственный Университет*
*им. М.В. Ломоносова*
*Факультет Вычислительной Математики и Кибернетики*
*Кафедра Автоматизации Систем Вычислительных Комплексов*

дипломная работа

# *Разработка и реализация многопрофильной системы фильтрации спама на основе методов машинного обучения*

***Автор***

*Петров Александр Владимирович*

*Группа 522*

*кафедра АСВК, ЛВК*

***Научный руководитель***

*Герасёв Александр Витальевич*

*Москва 2011*

**Аннотация**

Данная работа посвящена задаче фильтрации нежелательной электронной почты (спама). В антиспам-системах с открытым исходным кодом традиционно используются методы фильтрации, основанные на байесовской классификации. В некоторых коммерческих системах применяются другие методы классификации (например метод опорных векторов). В данной работе метод опорных векторов успешно применен в рамках открытой системы фильтрации спама.

Многие методы обнаружения спама могут работать в двух различных режимах - персонифицированном и неперсонифицированном. В работе на базе метода опорных векторов предложен третий подход - многопрофильной фильтрации, сочетающий достоинства обоих подходов к классификации электронных сообщений.

# СОДЕРЖАНИЕ

# 1 ВВЕДЕНИЕ

## 1.1 Задача фильтрации спама

Данная работа посвящена фильтрации спама. Под спамом обычно понимают массовую рассылку сообщений. Такими рассылками могут быть рекламные материалы приходящие по обычной почте, SMS, через системы обмена мгновенными сообщениями, по электронной почте.

Электронная почта в настоящее время является одним из основных способов общения в сети Интернет. В протокол доставки электронной почты [1] [2] при его разработке не были включены никакие средства проверки подлинности личности отправителя, что существенно облегчило рассылку сообщений. Кроме того, поскольку копирование электронного сообщения практически бесплатно, при использовании электронной почты проблема спама стоит особо остро. В данной работе под термином **спам** будет подразумеваться нежелательная электронная почта, то есть такая почта, которую пользователь не хотел бы получать даже зная о факте её отправки.

Термином **легитимная почта** мы будем обозначать электронные письма, не являющиеся спамом.

Так как получение нежелательных сообщений отвлекает пользователей и создает ненужную нагрузку на сеть, со спамом необходимо бороться. С момента появления спама было придумано множество методов позволяющих отличить спам от легитимной почты. Перечислим некоторые классы этих методов:

1. Методы, использующие информацию о сетевом соединении. В своей работе они анализируют IP-адреса, доменные имена и другую информацию на сетевом уровне.
2. Методы работающие на уровне протокола передачи электронной почты. Эти методы используют особенности протокола передачи почтовых сообщений (SMTP) и точность его соблюдения клиентом.
3. Анализ заголовков и тела письма. Эти методы используют информацию содержащуюся в самом письме: комбинации заголовков, ключевые слова, соответствие регулярным выражениям. Могут применять-

ся различные статистические методы.

Подробное описание методов было приведено в курсовой работе [13].

В настоящее время системы, рассылающие спам научились достаточно хорошо имитировать работу корректных почтовых сервисов и таким образом обходить фильтры первых двух типов. Поэтому, практически все современные антиспам-системы так или иначе используют статистические фильтры, так как они позволяют наиболее качественно отсеивать нежелательные сообщения. В данной работе остановимся именно на статистических методах.

## 1.2 Статистические методы

Статистические методы можно представить в виде «черного ящика», получающего на вход письмо, и на выходе возвращающего оценку принадлежности этого письма к спаму. Внутри черного ящика выполняется алгоритм, который выдает оценку для письма в зависимости от некоторых статистических данных. Наборов данных может быть несколько, например отдельный набор данных для каждого адресата. Совокупность таких данных для одного адресата будем называть **профилем**.

Профиль строится при помощи **обучающей выборки** – некоторого количества сообщений, для которых известно являются они спамом или легитимной почтой. **Обучением** статистического метода называется процесс построения профиля по обучающей выборке. Поскольку со временем содержание спам-рассылок имеет тенденцию меняться, необходимо поддерживать профиль пользователя в актуальном состоянии, то есть перестраивать его с учетом новых данных. Для некоторых алгоритмов необходимо полностью повторять процесс обучения, однако существуют алгоритмы, в которых можно использовать информацию полученную на ранних этапах добавив к ним новые знания. Процесс добавления в профиль новой информации называется **дообучением**.

Существует два подхода к построению профиля пользователя, необходимого для работы статистических методов:

- В обучающей выборке содержатся письма от всех пользователей. В этом случае при добавлении нового пользователя в систему он сразу

может начинать пользоваться фильтром, не помечая приходящие к нему письма как спам или легитимная почта. Однако возникает другая проблема, связанная с тем, что схожие письма для одних пользователей являются спамом, а для других легитимной почтой. (Например, менеджер вполне может получать письма с коммерческими предложениями, при этом если такие письма получает программист, то они почти наверняка являются спамом.)

- Все письма из обучающей выборки принадлежат одному пользователю. В этом случае алгоритм работает достаточно хорошо (отсеивая практически весь спам, при этом вероятность ложного срабатывания достаточно мала). Однако, для того, чтобы алгоритм начал работать пользователь должен получить и добавить в свою обучающую выборку достаточно большое количество спама и легитимной почтой.

Оба метода обладают недостатками, и, по этому, является актуальной задача разработки подхода, комбинирующего достоинства обоих подходов.

## 1.3 Размещение системы фильтрации спама

Фильтрацию спама можно проводить либо на стороне клиента, либо на стороне сервера. При фильтрации на стороне клиента почтовый клиент скачивает письмо с сервера, после этого классифицирует почту.

Основным достоинством метода является то, что пользователь сам может решить нужно ли ему фильтровать спам, как его фильтровать.

К недостаткам метода можно отнести необходимость настройки фильтрации на каждой машине пользователя, создание дополнительной нагрузки на компьютерную сеть, невозможность использовать неперсонифицированный подход к фильтрации спама.

При фильтрации на стороне сервера письмо классифицируется до передачи его в почтовый клиент пользователя. При этом пользователь может даже не скачивать письмо. Достоинствами метода являются настройка спам-фильтра только в одном месте (на почтовом сервере), низкая нагрузка на сеть,

возможность использовать как персонифицированный, так и неперсонифицированный подход к настройке спама.

В дальнейшем будет рассматриваться только серверная фильтрация спама, так как такой подход обладает рядом достоинств по сравнению с клиентской фильтрацией, а недостатки не являются существенными. Кроме того, как будет показано в разделе 4, разрабатываемый в данной работе метод можно применить только на почтовом сервере.

## 1.4 Методы машинного обучения

Фильтрация спама статистическими методами тесно связана с теорией машинного обучения. Многие термины приведенные выше при описании статистического подхода к фильтрации спама были заимствованны из этой теории.

**Машинное обучение** – это научная дисциплина, изучающая построение алгоритмов, способных обучаться[17]. При помощи машинного обучения решаются такие задачи как восстановление регрессии, кластеризация, классификация, прогнозирование.

Задача фильтрации спама является **задачей классификации**: нужно определить принадлежность письма к одному из классов: спам или легитимная почта.

Формально задача классификации ставится следующим образом: имеется множество **объектов**, разделённых некоторым образом на классы. Для конечного подмножества объектов известно, к каким классам они относятся. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

**Классифицировать** объект – значит указать номер (или наименование) класса, к которому принадлежит объект. Классификация объекта – номер или наименование класса, выдаваемое алгоритмом классификации в результате его применения к данному конкретному объекту.

**Обучением** называется процесс постройки алгоритма классификации по обучающей выборке. **Дообучение** – процесс перестроения алгоритма классификации после добавления в обучающую выборку новых объектов. Для мно-

гих алгоритмов их не приходится перестраивать их с ноля, можно воспользоваться информацией полученной при ранних этапах обучения. Такие алгоритмы называются **дообучаемыми**.

Некоторые из методов машинного обучения подвержены проблеме **переобучения**. Проблема заключается в том что алгоритм может пытаться находить закономерности в обучающей выборке там где их на самом деле нету.

Задачу классификации можно поставить **вероятностно**. В этом случае построенный алгоритм должен выдавать не конкретный класс, а вероятность принадлежности объекта тому или иному классу.

Существует достаточно большое множество методов машинного обучения (см. например [17])

## 1.5 Актуальность разработки многопрофильного средства фильтрации спама

Все существующие на сегодняшний день открытые средства фильтрации спама (имеются ввиду развивающиеся и поддерживаемые системы) используют либо персонифицированный, либо неперсонифицированный подход к фильтрации. Персонифицированный подход при должном обучении может обеспечить более качественную фильтрацию, однако процесс обучения необходимо повторять для каждого нового пользователя почтовой системы, и, кроме того, при появлении нового типа спама каждый из пользователей должен добавить сообщения такого типа в свою обучающую выборку. Поэтому, как было показано выше, является актуальной разработка третьего подхода, сочетающего достоинства обоих подходов.

Такой подход в рамках данной работы мы будем называть **многопрофильным**.

В курсовой работе четвертого курса [13] мною был реализован многопрофильный подход на основе байесовских классификаторов (о фильтрации спама байесовскими методами см. [7] и [8]). В работе А. Розинкина[14] было показано, что применив более сложные методы машинного обучения (в частности метод опорных векторов) можно получить более качественный классификатор.

В данной работе разработан метод, основанный на методе опорных векторов, поддерживающий многопрофильность.

# 2 ЦЕЛИ РАБОТЫ И ПОСТАНОВКА ЗАДАЧИ

Данная работа имеет две основные цели:

- Реализовать в рамках так из свободных антиспам-систем фильтрацию почты, основанную на методе опорных векторов.
- Разработать и реализовать в рамках той же системы многопрофильную фильтрацию, основанную на методе опорных векторов.

Для этого предлагается решить следующие подзадачи:

1. Провести обзор существующих открытых систем фильтрации спама и выбрать средство для расширения.
2. В рамках выбранного средства реализовать фильтрацию спама, основанную на методе опорных векторов.
3. Разработать на базе метода опорных векторов метод, который позволит классифицировать почтовые сообщения по нескольким профилям.
4. Реализовать данный метод в рамках выбранного ранее средства.
5. Провести экспериментальное исследование на открытых тестовых наборах
6. Произвести апробацию разработанного средства на реальных данных.

Разрабатываемое средство должно удовлетворять следующим требованиям:

- должно распространяться под свободной лицензией;
- должно работать демоном на сервере;
- может быть интегрировано в большинство распространенных MTA;
- в случае работы для одного пользователя качество классификации не хуже чем у наивного байесовского классификатора;

- в случае работы для нескольких пользователей в режиме многопрофильной фильтрации качество классификации лучше чем у наивного байесовского классификатора;
- производительность классификации должна быть сравнима с байесовскими методами.

# 3 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

## 3.1 Методы машинного обучения

В разделе 1.4 было приведено неформальное введение в теорию машинного обучения. В данном приведена более формальная постановка задачи машинного обучения и описан один из наиболее используемых методов ее решения - метод опорных векторов, которой, как показано в [14] может быть успешно применен к задаче фильтрации спама.

### 3.1.1 Обучение по прецедентам

Сформулируем основную задачу машинного обучения.

*Пусть есть неизвестная функция* $F : X \to Y$*, переводящая объекты множества* $X$ *в объекты множества* $Y$*, причем для некоторых* $x_1, x_2, ...x_n$ *известны соответствующие им значения* $y_1 = F(x_1), y_2 = F(x_2), ...y_n = F(x_n)$*. Необходимо построить функцию* $F^*(X)$*, наилучшим образом приближающую* $F(X)$*.*

Под фразой **наилучшим образом приближает** подразумевается, что для некоторого функционала качества $\mu(y, y')$, матожидание

$$E\mu(F(x), F^*(x)), x \in X \tag{1}$$

будет минимальным.

В качестве $\mu(y, y')$ можно брать например модуль разности, квадрат разности и т. п.

Объект $x_i$ для которого известно соответствующее значение $y_i$ называется **прецедентом**

**Методом обучения** называется процесс построения $F^*(x)$ по парам $(x_1, y_1), (x_2, y_2), ...(x_n, y_n)$. Множество таких пар называется **обучающей выборкой**

Построенную функцию $F^*(x)$ часто также называют **алгоритмом**, подразумевая, что она должна быть эффективно вычислимой на компьютере.

Задача построения такого алгоритма в общем случае не может быть решена точно, так как неизвестна природа исходной функции $F(x)$. Кроме того минимизация $\mu$ на обучающей выборке не обязательно приводит к тому, что и на новых объектах из $X$ значение $\mu$ также будет мало (см. раздел 3.1.2).

Процесс построения модели можно также рассматривать как выбор конкретного значения функции $F^*(x)$ из семейства $F^*(x, \pi)$. Значение выбранного параметра $\pi$ в таком случае называется **моделью** для алгоритма $F^*(x)$

В целом процесс решения задачи машинного обучения называют **машинным обучением** или **обучением по прецедентам**

Если множество $Y$ конечно, то говорят, что поставленная задача является **задачей классификации** с $m$ классами, где $m$ - мощность множества $Y$.

Задачу классификации можно также поставить вероятностно:

*Пусть есть неизвестная функция* $F : X \to Y$, *переводящая объекты множества* $X$ *в объекты множества* $Y$, *причем множество* $Y$ *конечно. Пусть также для некоторых* $x_1, x_2, ...x_n$ *известны соответствующие им значения* $y_1 = F(x_1), y_2 = F(x_2), ...y_n = F(x_n)$ *Необходимо построить функцию* $F^*(x, y) : (X, Y) \to [0, 1]$, *возвращающую оценку вероятности того, что* $F(x) = y$

Задача фильтрации спама по своей сути является задачей классификации с двумя классами. В качестве объектов в этом случае выступают письма, а множество классов состоит из двух элементов : $\{$*спам, легитимная почта*$\}$

Наиболее распространенным способом описания объектов является **признаковое описание**. Фиксируется совокупность n показателей, измеряемых у всех прецедентов. Если все n показателей числовые, то признаковые описания представляют собой числовые векторы размерности n. Большая часть методов обучения (в частности метод опорных векторов) работают с прецедентами представленными именно в виде числового вектора признаков.

### 3.1.2 Переобучение

Пусть поставлена задача классификации, и при помощи некоторого метода обучения $A$ получено приближение для функции $F$ - функция $F^*$.

Пусть частота ошибок (т. е. вероятность события $F(x) \neq F^*(x)$) на объектах из обучающей выборке равна $a_1$, а на всем множестве $X$ $a_2$. Говорят что алгоритм $A$ обладает **обобщающей способностью**, если $a_2$ не сильно отличается от $a_1$

Нежелательная ситуация, при которой

$$a_1 \ll a_2 \tag{2}$$

называется **переобучением**.

Проблема возникает, когда модель используемая алгоритмом черезмерно сложна и при построении классификатора находятся закономерности между объектами из обучающей выборки в реальности не существующие. В частности проблема часто возникает, когда используется черезмерно большое количество признаков, часть из которых явно никак не влияет на классовую принадлежность объекта.

Практически все методы машинного обучения в той или иной степени подвержены проблеме переобучения, поэтому при настройке метода необходимо подбирать его параметры таким образом, чтобы она проявлялась как можно меньше.

### 3.1.3 Скользящий контроль

Так как вид функции $F(X)$ в общем случае неизвестен, то прямо посчитать значение матожидания 1 не возможным. Для того чтобы хоть как-то оценить качество построенного алгоритма обычно пользуются методом **скользящего контроля**. Метод заключается в следующем: первоначальная обучающая выборка делится на несколько частей. Обучение проводится по очереди на каждой из частей, а значение $\mu$ оценивается по оставшимся частям.

Итоговую оценку $\mu$ считают как

$$\mu' = 1/n \sum_{1}^{n} \mu_i, i \in 1..n, \tag{3}$$

В приведенной формуле $n$ - количество частей, на которое делится выборка, $\mu_i$ - оценка матожидания 1 по всем частям, кроме части с номером $i$, с учетом того, что обучение производилось по части с номером $i$

### 3.1.4 Метод опорных векторов

Метод опорных векторов - это метод обучения, предложенный В. Вапником в работе [18] Метод основан на **принципе максимизации зазора**.

Пусть поставлена задача классификации с двумя классами. В пространстве признаков классифицируемых объектов проводится гиперплоскость таким образом, чтобы объекты обучающей выборки принадлежащие одному классу лежали по одну сторону от этой гиперплоскости, а объекты принадлежащие второму классу по другую.

Понятно, что если гиперплоскость существует, то она не единственна. Среди всех таких гиперплоскостей выбирается такая, которая максимизирует **зазор** – минимальное расстояние от гиперплоскости до ближайшей точки обучающей выборки. Таким образом расстояние от гиперплоскости до ближайшего объекта каждого из классов будет одинаково.

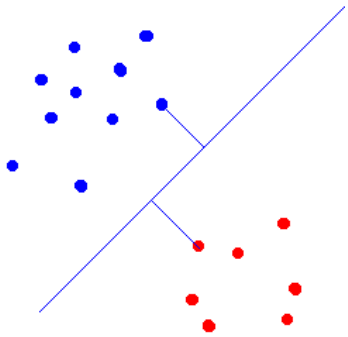


Рисунок 1 – Наилучшая разделяющая прямая в двухмерном пространстве

Если разделяющая гиперплоскость существует выборка называется **линейно разделимой**.

Для обобщения на случай линейно неразделимой выборки используется следующая идея: выборку можно сделать линейно разделимой, если увеличить размерность пространства. Для этого используется так называемая **ядерная функция**

Пусть признаковое пространство имеет размерность $N$, а обучающая выборка состоит из $M$ пар $(x_i, y_i)$. В таком случае обучение методом опорных векторов происходит за время, оцениваемое как

$$O(M^2 * N) \tag{4}$$

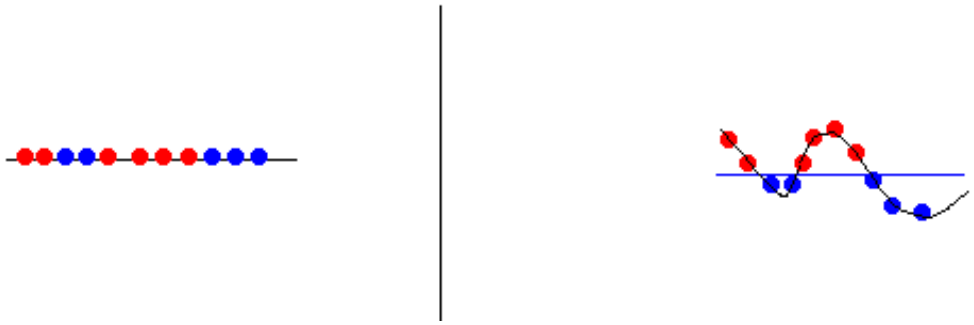


Рисунок 2 – Неразделимая в одномерном пространстве выборка стала разделимой после перевода в двумерное пространство

а время работы построенного классификатора $F^*(X)$ оценивается как

$$O(N) \tag{5}$$

Метод опорных векторов в настоящее время рассматривается как один из наиболее универсальных и хорошо работающий на широком классе задач. Кроме того в работе[14] было показано что этот метод показывает хорошие результаты при применении его к задаче фильтрации спама. Для решения поставленной задачи мы также воспользуемся этим методом.

## 3.2 Обзор существующих открытых систем фильтрации спама

Любая система фильтрации состоит из нескольких компонентов состоит. Часть из этих модулей реализует взаимодействие с почтовым сервером, часть отвечает за выявление заголовков и тела письма, часть реализует логику фильтрации. Так как модуль, реализующий логику фильтрации, является лишь одним из нескольких необходимых модулей, имеет смысл не реализовывать систему фильтрации спама с нуля, а выбрать одну из систем с открытым исходным кодом для доработки.

Для выбора системы для доработки произведем обзор существующих открытых систем фильтрации спама в соответствии со следующими критериями:

- Лицензия. Лицензия должна быть свободной и позволять доработку.

- Дата последнего релиза. Средство должно быть поддерживаемым разработчиками, и чем позднее был выпущен последний релиз, тем лучше.
- Язык программирования. Для простоты модификации средство должно быть написано на одном широко используемых языков программирования, например: C, C++, Python.
- Размещение системы фильтрации спама. Система должна размещаться на стороне сервера для того, чтобы было возможно произвести многопрофильную фильтрацию.
- Метод фильтрации спама - для того чтобы произвести тестирование разработанного метода, необходимо, чтобы система использовала всего один метод фильтрации.

**SpamAssassin**

SpamAssassin - одно из наиболее известных открытых средств фильтрации спама. Этот проект динамично развивается и показывает хорошие результаты производительности и качества фильтрации. SpamAssassin использует в своей работе большое количество методик обнаружения спама.

- Лицензия: свободная (Apache License 2.0);
- Язык программирования: Perl;
- Дата последнего релиза: 2010г.;
- Размещение системы фильтрации: клиент или сервер;
- Метод фильтрации спама: большое количество различных эвристик.

**Spamprobe**

Система Spamprobe написана на языке C++ и использует байесовский подход к фильтрации спама. В настоящее время практически не поддерживается, последняя версия была выпущена в 2007 году.

- Лицензия: свободная (Q Public License);
- Язык программирования: C++;
- Дата последнего релиза: 2007г.;
- Размещение системы фильтрации: сервер;

- Метод фильтрации спама: байесовская классификация.

**Bogofilter**

Система работает на стороне клиента, может интегрироваться в различные почтовые клиенты (такие как kmail и evolution). Использует в своей работе байсовские методы.

- Лицензия: свободная (GPL v.2);
- Язык программирования: C;
- Дата последнего релиза: 2010 г.;
- Размещение системы фильтрации: почтовый клиент;
- Метод фильтрации спама: байесовская классификация.

**bmf**

Очень маленький по размеру (около 4000 строк кода) спам-фильтр. Использует байесовский подход к фильтрации спама. в настоящее время не поддерживается.

- Лицензия: свободная (GPL);
- Язык программирования: C;
- Дата последнего релиза: 2002 г.;
- Размещение системы фильтрации: сервер;
- Метод фильтрации спама: байесовская классификация.

**Quick Spam Filter**

Использует байесовский подход к фильтрации спама. Написан на языке C, в связи с чем работает достаточно быстро. В настоящее время не поддерживается.

- Лицензия: свободная (Artistic License 2.0);
- Язык программирования: C;
- Дата последнего релиза: 2004г.;
- Размещение системы фильтрации: сервер;
- Метод фильтрации спама: байесовская классификация.

**Scmail**

Написан на языке scheme. Для фильтрации использует различные эвристические подходы. В настоящее время не поддерживается.

- Лицензия: свободная (BSD License);
- Язык программирования: Scheme;
- Дата последнего релиза: 2004г.;
- Размещение системы фильтрации: сервер;
- Метод фильтрации спама: различные эвристики.

**Spambayes**

Написан на языке python, поэтому достаточно прост в модификации. Работает на стороне почтового клиента.

- Лицензия: свободная (PSF License);
- Язык программирования: Python;
- Дата последнего релиза: 2011г.;
- Размещение системы фильтрации: почтовый клиент;
- Метод фильтрации спама: байесовская классификация.

**dspam**

Открытая система фильтрации спама. Dspam изначально проектировался для работы в многопользовательском режиме.

- Лицензия: свободная (GPL v.2);
- Язык программирования: C;
- Дата последнего релиза: 2011г.;
- Размещение системы фильтрации: Сервер.
- Метод фильтрации спама: байесовская классификация.

Для фильтрации спама dspam может использовать одну из нескольких разновидностей байесового классификатора. Dspam написан на языке C и работает достаточно эффективно. Dspam имеет большое сообщество разработчиков и активно развивается в настоящий момент. Из недостатков системы dspam

можно отметить использование устаревшей системы сборки (autools) и использование низкоуровневой разработки, что усложняет понимание и модификацию его исходного кода.

| Средство | Место расположения | Дата последнего релиза | Язык программирования | Метод фильтрации |
|---|---|---|---|---|
| Spamassassin | Сервер или клиент | 2010г. | Perl | Совмещает большое количество различных методов |
| Spamprobe | Сервер | 2007г. | C++ | Байес |
| Bogofilter | Почтовый клиент | 2010г. | C | Байес |
| BFM | Сервер | 2002г. | C | Байес |
| QSF | Сервер | 2004г. | C | Байес |
| Scmail | Сервер | 2004г. | Scheme | Различные эвристики |
| Spambayes | Почтовый клиент | 2011г. | Python | Байес |
| Dspam | Сервер | 2011г. | C | Байес |

Рисунок 3 – Сравнение систем фильтрации спама

**Вывод**

По результатам обзора выбор системы для доработки происходил в основном между системами SpamAssassin и dspam, так как только два этих средства работают на серверной стороне и при этом в настоящее время активно поддерживаются сообществом. Dspam в качестве кандидата на доработку обладает двумя преимуществами перед SpamAssassin:

1. Написан на языке C, поэтому работает более эффективно чем SpamAssassin написанный на языке Perl.
2. Фильтрация спама осуществляется всего одним алгоритмом, что упрощает тестирование разработанного метода.

В итоге для доработки был выбран dspam, так как он удовлетворяет нашим требованиям, а именно:

- Ориентирован на работе на стороне сервера
- Распространяется под свободной лицензией
- Ориентирован на многопользовательский режим

# 4 ИССЛЕДОВАНИЕ И ПОСТРОЕНИЕ РЕШЕНИЯ ЗАДАЧИ

Для того, чтобы применить метод опорных векторов к задаче фильтрации спама, необходимо научиться представлять письма в виде, пригодном для применения этого метода - в виде вектора признаков. Кроме того, нам необходимо построить многопрофильную систему, а для этого классифицируемый вектор должен содержать какую-то информацию о пользователе.

## 4.1 Выделение лексем

Под **лексемами** мы будем понимать последовательности символов, на которые мы разбиваем исходный текст. В некотором смысле лексема - это аналог слова. Выделить слова из текста зачастую бывает затруднительно, кроме того зачастую смысловую нагрузку несет лишь часть слова, а иногда напротив лишь последовательность слов. Для разбиения текста на лексемы существует несколько способов:

- **Разбиение до пробельных символов или знаков препинания**. В этом случае лексема получается практически полным аналогом слова. Например текст “Как хорошо, замечательно жить в этом мире!” разобьется на следующие лексемы: “Как” , “хорошо”, “замечательно”, “жить”, “в”, “этом” и “мире”.
- **Выделение n-грамм**. В этом случае из текста выделяются все цепочки содержащие ровно n символов. Например при n=6 фраза “hello, world” породит следующие 6-граммы: “hello,”, “ello, ”, “llo, w”, “lo, wo”, “o, wor”, “, worl” и “ world”.
- **Выделение цепочек**. Использование в качестве лексем цепочек из нескольких слов. При этом в качестве лексем выделяются как сами слова, так и последовательности из нескольких подряд идущих слов. Например “free viagra” породит лексемы “free”, “viagra” и “free viagra”.

- Выделение частичных цепочек. Этот алгоритм рассматривает окна из нескольких слов, и генерирует цепочки состоящие из некоторых (необязательно идущих подряд) слов этого окна. Например, из текста “the quick brown fox jumped” при размере окна 4 будут сгенерированы такие, как “the quick brown *”, “quick * * jumped”, “* * * jumped” и другие.
- Выделение биграмм. Алгоритм похож на использование частичных цепочек, однако генерирует только те цепочки, в которых содержится ровно два слова. Для предыдущего примера алгоритм сгенерирует цепочки “the quick * *” “* * brown jumped”, “the * * fox” и другие.

В dspam можно по желанию использовать любой из алгоритмов, но по умолчанию используется метод выделения цепочек. Разработчики dspam указывают[15] что применительно к задаче фильтрации спама такой метод работает не хуже других. Для решения нашей задачи мы также воспользуемся этим методом.

Так как термин **лексема** по смыслу близок к слову в обычном понимании, то в дальнейшем говоря о частотах слов мы будем подразумевать частоты лексем.

## 4.2 Представление письма в векторном виде

Метод опорных векторов работает с объектами, представленными в векторном виде. Письма имеют текстовую, структуру. Однако, у электронного сообщения возможно выделить большое количество числовых признаков, например:

- Количество слов в письме;
- Дата отправления;
- Количество заголовков;
- Количество вложений.

Все эти признаки в принципе могут быть использованы для построения векторного представления письма, однако они никак не передают самое важное, что отличает спам от легитимной почты - смысл сообщения.


Рисунок 4 – Векторное представление текста

Человек, для того, чтобы понять смысл текста, анализирует совокупность слов в этом тексте. Хорошей характеристикой тематики текста являются частоты входящих в него слов. Таким образом мы можем взять все слова в языке, посчитать количество каждого из них в письме и получить вектор частот.

Проблема заключается в том, что размерность такого вектора чрезмерно высока, причем почти все признаки в этом случае окажутся нулевыми. Русский язык содержит по некоторым оценкам более 500000[19] слов, а с учетом словоформ количество слов может превысить несколько миллион.

Скорость работы метода опорных векторов линейно зависит размерности пространства классифицируемых векторов. Большая размерность вектора приведет к увеличению времени построения модели и времени классификации. Кроме того метод склонен к переобучению при больших размерностях векторов. Поэтому необходимо каким-то образом ограничить количество используемых для классификации слов.

Для решения задачи выбора используемых для классификации была произведена серия экспериментов. Были протестированы следующие способы

отбора:

- Выбор наиболее частых слов
- Выбор случайно отобранных слов
- Выбор слов для которых разница между частотой появления в спаме и в легитимной почте максимальна

Экспериментальным путем было установлено что наилучшие результаты метод опорных векторов проявляет при использовании 1500 самых частых слов. При меньшем количестве информации слишком мало для классификации, при большем - начинают проявляться проблемы переобучения.

## 4.3 Многопрофильность

Нам необходимо построить такой классификатор, который с одной стороны для классификации пользуется данными от всех пользователей, а с другой стороны учитывает специфику адресата письма. Это обозначает, что используемый метод обучения должен использовать некоторую информацию об адресате письма.

В качестве информации, которую можно использовать для классификации удобно использовать некоторый числовой идентификатор, который уникален для каждого пользователя почтовой системы. Если добавить такой идентификатор к векторному представлению письма, то одинаковые письма для разных пользователей будут различаться.

Рассмотрим ситуацию, при которой определенный класс писем (векторные представления которых достаточно близки) один пользователь считает спамом, а другой легитимной почтой. В этом случае добавив в признаковое описание письма идентификатор адресата мы дадим классификатору информативный признак, позволяющий отличить спам от легитимной почты. Если метод обучения должным образом учтет этот признак при построении модели, то в последствии построенный классификатор будет различать приходящие письма по этому признаку и классифицировать их правильно.

Теперь рассмотрим другую ситуацию, при которой определенный класс писем все пользователи почтовой системы считают спамом (или легитимной

почтой). Пусть в обучающей выборке есть некоторое количество писем из этого класса от разных пользователей. В этом случае признаки, соответствующие лексемам, входящим в такие (и только такие) письма будут иметь высокую информативность, и классификатор при построении модели сможет учесть это обстоятельство.

Пусть письмо из этого класса пришло пользователю, который еще не добавлял в свою выборку писем из этого класса. В этом случае, так как признаки, соответствующие лексемам из этого письма имеют большую информативность, такое письмо также правильно определится как спам (или соответственно как легитимная почта).

Таким образом добавление к векторному представлению письма числового идентификатора позволит решить задачу построения многопрофильного спам-фильтра, если метод обучения корректно обработает такой признак.

Уникальный идентификатор адресата в почтовой системе является **номинальным** признаком (т. е. признаком, который может принимать лишь конечное количество различных значений). К сожалению, метод опорных векторов (как и многие другие методы, применяемые для решения задачи классификации) не всегда корректно обрабатывает номинальные признаки.[9] Для решения этой проблемы применяется **бинаризация**.

Пусть есть номинальный признак $a$, способный принимать одно из $n$ значений $a_1, a_2, ...a_n$. При бинаризации такой признак заменяется на $n$ признаков $b_1, b_2, ...b_n$, причем $b_i$ равно 1, если исходный признак равен $a_i$ и 0 в противном случае.

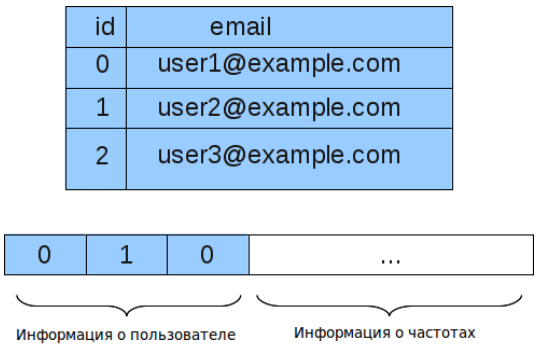



Рисунок 5 – Добавление информации о профиле в вектор письма. Письмо адресовано второму пользователю.

После бинаризации признака, соответствующего идентификатору адресату метод опорных векторов сможет корректно обрабатывать такой признак

и успешно справляться с классификацией сообщений в многопрофильном режиме.

Таким образом, общая схема построения признакового описания письма такова:

- Строится вектор частот.
- Строится вектор, идентифицирующий пользователя.
- Два вектора конкатенируются, получается результирующий вектор.

# 5 РЕАЛИЗАЦИЯ

## 5.1 Библиотека libsvm

Предложенный в разделе 4 метод использует в своей работе метод опорных векторов. Этот метод используется во многих задачах и поэтому существует некоторое количество открытых реализаций метода. Одной из самых известных является библиотека libsvm[16]. Библиотека libsvm содержит реализацию метода опорных векторов. Она содержит интерфейсы для построения моделей по обучающей выборки и предсказания значения целевой функции по существующей модели. Дополнительно эта библиотека содержит интерфейсы для сохранения модели в файл и загрузки ее из файла.

Для создания модели можно пользоваться несколькими ядровыми функциями (линейная, полиномиальная, сигмоид и другие), а также задавать все необходимые для работы метода параметры.

Кроме того можно указывать тип решаемой задачи: классификация, классификация с двумя классами, регрессия и др. Так как libsvm имеет интерфейс для ее использования из языка C (это необходимо для ее использования в dspam), а также распространяется под свободной лицензией, она была использована при реализации разработанного метода.

## 5.2 Архитектура dspam

В основе dspam лежит библиотека libdspam, которая реализует логику фильтрации. Эта библиотека содержит код нескольких алгоритмов фильтрации (одновременно использоваться может только один из них) Кроме того библиотека содержит несколько вспомогательных классов реализующих некоторые структуры данных (списки, деревья и т. п), а также драйвер хранилища данных.

Данные могут храниться либо в одной из поддерживаемых СУБД (это MySQL, Posgres и SQLite), а также непосредственно в файловой системе (используется собственная методика хранения пар ключ-значение).

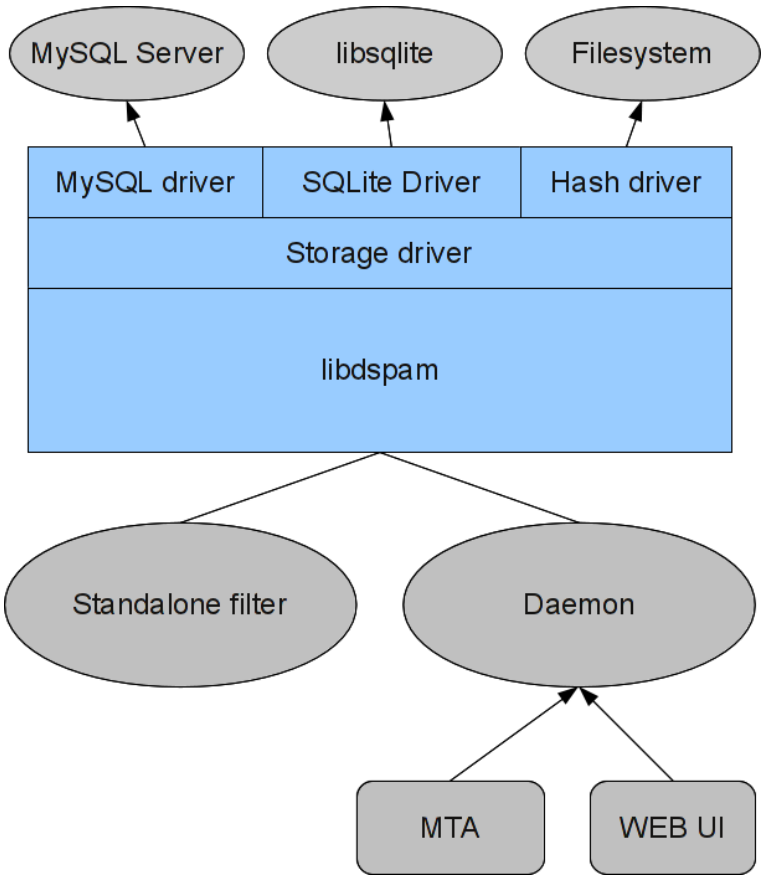



Рисунок 6 – Архитектура dspam

Dspam может работать либо в режиме standalone-фильтра, либо демона. В обоих случаях исполняемый модуль линкуется с библиотекой libdspam.

Работа в standalone режиме подразумевает что бинарный файл запускается по требованию. Тестируемое письмо в этом случае подается исполняемому модулю на стандартный вход, а информацию о произведенном тестировании можно получить со стандартного вывода. Эта информации может быть представлена в виде короткой строчки в которой содержится класс, к которому отнес классификатор письмо и вероятностью или в виде самого письма с установленными заголовками. Этот режим удобно использовать для тестирования системы, а так же в случае кода количество писем достаточно мало и требование к производительности невелики.

Работа в режиме демона подразумевает что исполняемый код постоянно находится в оперативной памяти. Общение с клиентом в этом случае происходит либо через сеть (используется tcp-сокет), либо через UNIX-сокет. Таким образом dspam в этом случае является полноценным серверным приложением. Сервер по запросу от клиента принимает письмо, классифицирует, простав-

ляет необходимые заголовки и возвращает клиенту. Клиентом в таком случае выступает система доставки почты (MTA).

Кроме того сервер может выполнять некоторые сервисные команды: просмотр статистики, очистка хранилища данных и т. п. В этом случае клиентом может быть например web-интерфейс администратора.

## 5.3 Описание реализации

Для обеспечения работы описанного в разделе4 метода были произведены следующие действия:

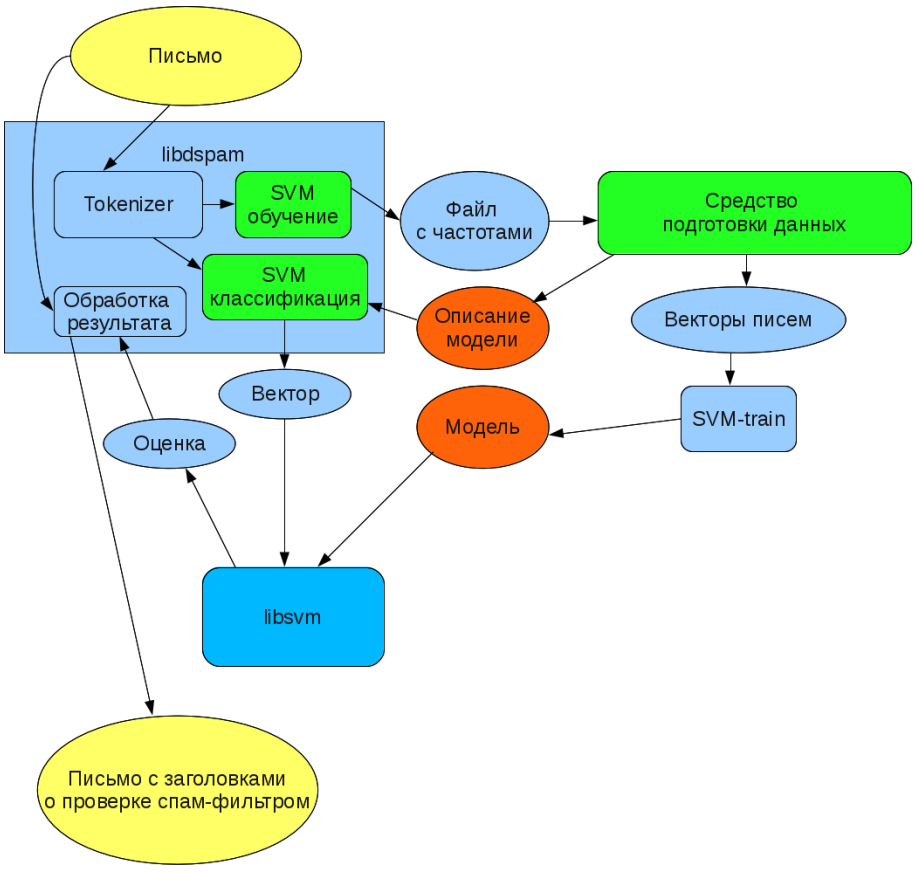
1. В библиотеку dspam добавлен модуль, отвечающий за добавление письма в обучающую выборку метода опорных векторов. Работает с письмами, уже разбитыми на лексемы. Записывает в специальный файл информацию об адресате письма и частотах лексем.
2. Разработана программа на языке python, определяющая частоты каких лексем будут использоваться в качестве признаков методом опорных векторов, подготавливающая данные для инструмента svm-train из комплекта библиотеки libsvm и вызывает этот инструмент, который строит модель для метода опорных векторов. Эта программа также генерирует файл с описанием модели, используемый в дальнейшем при классификации. Программу необходимо вызывать каждый раз при необходимости перегенерации модели (например можно ее вызывать каждые сутки при помощи стандартной службы CRON).
3. В библиотеку dspam добавлена функциональность, позволяющая произвести классификацию письма по модели и ее описанию. По описан модели и самому письму строится векторное представление письма, затем при помощи библиотеки libsvm по модели и вектору получается оценка принадлежности письма одному из классов.

## 5.4 Схема работы

Опишем схему работы разработанного средства в различных случаях. Всего есть три стадии работы системы:

- **Добавление письма в обучающую выборку**. Реализуется при помощи функциональности, содержащейся в libdspam.
- **Построение модели**. Реализуется внешней программой, вызываемой по требованию.
- **Классификация**. Реализуется при помощи функциональности содержащейся в libdspam и libsvm.

Опишем подробно что происходит на каждом из этапов.




Зеленым выделены компоненты, реализованные в рамках работы

Рисунок 7 – Схема работы классификатора. Зеленым выделенные компоненты, реализованные в рамках работы

### 5.4.1 Добавление письма в обучающую выборку

На этом этапе информация о письме сохраняется в специальном файле, который в дальнейшем используется для генерации модели. Шаги, которые происходят на этом этапе:

1. Письмо попадает в систему.
2. Письмо разбивается на лексем.
3. Вычисляются частоты лексем.
4. Информация о классе письма, адресате и частотах лексем записывается в файл частот.

### 5.4.2 Построение модели

На этом этапе генерируется модель для метода опорных векторов, а так же описание этой модели. В

На этапе происходят следующие шаги:

1. Вызывается скрипт построения модели (например, раз в день при помощи стандартной службы CRON).
2. Вычисляются общие частоты лексем для каждой лексемы из файла частот.
3. Выбираются наиболее употребимые лексемы (лексемы с наибольшей частотой).
4. Для каждого из писем строится вектор, содержащий частоты наиболее употребимых лексемы, а также индикаторы принадлежности письма пользователю (все кроме одного индикаторы равны нолю).
5. Построенное множество векторов записывается во временный файл вместе с информацией о классах писем, соответствующих векторам.
6. По построенному временному файлу генерируется модель при помощи инструмента svm-train.

7. Информация о лексемах используемых для построения модели сохраняется во временный файл описания модели.

## 5.5 Классификация

На этом этапе новым письмам приходящим в почтовую систему назначается оценка принадлежности этого письма к спаму или легитимной почте. Шаги, которые происходят на этапе:

1. Письмо попадает в систему.
2. Выясняется адресат письма.
3. Письмо разбивается на лексемы.
4. Вычисляются частоты лексем.
5. Загружается описание модели
6. По описанию модели, адресату и частотах лексем строится вектор
7. Загружается модель
8. По модели и вектору вычисляется вероятность принадлежности письма спаму
9. По пороговому правилу выставляется оценка спам/не спам

# 6 ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ

Для экспериментальной проверки разработанного программного средства были проведены две серии экспериментов. Первая серия проведена для сравнения разработанного программного средства с наивным байесовским классификатором, используемым во многих открытых системах фильтрации спама. Кроме того, было произведено сравнение с байесовским классификатором из системы dspam, использующего методы, описанные в [8] . Серия призвана показать, что разработанное средство работает не хуже байесовских методов.

Вторая серия экспериментов проведена для проверки работы системы в многопрофильном режиме.

## 6.1 Сравнение с наивным байесовским классификатором и байес-подобным классификатором из dspam

### 6.1.1 Набор данных

Тестирование производилось на двух публичных наборах email-сообщений.

1. SpamAssassin public corpus [11]
2. CEAS 2008 Live Spam Challenge Laboratory Corpus [12]

### 6.1.2 Методика тестирования

Тестирование производилось с использованием метода скользящего контроля: выборка разбивалась на пять частей, на четырех из которых проводилось обучение, а на пятой - контроль. Выборка [12] ввиду своей большой вели-

чины была разделена на несколько частей, на каждой из которых тестирование было произведено отдельно.

Результат тестирования представлен в виде графика соответствия величин вероятности ложно-положительного срабатывания и вероятности верного обнаружения спама. Точка на графике обозначает что при определенной границе решающего правила система будет работать именно в таком режиме: фильтровать спам с вероятностью показанной на оси ординат и классифицировать легитимные письма как спам с вероятностью показанной на оси абсцисс. Чем выше проходит график, тем более качественно работает система фильтрации.

### 6.1.3 Результаты тестирования

На графике 8 представлен результаты тестирования на наборе [11]. Видно, что разработанное средство всегда работает лучше чем наивный байесовский классификатор.

Байесовский классификатор из системы dspam показывает примерно такие же результаты, что и разработанный метод.

Тестирование на наборе [12] дало аналогичный же результат.

## 6.2 Тестирование работы в многопрофильном режиме

### 6.2.1 Набор данных

Тестирование производилось на наборе данных SpamAssassin Public Corpus [11]. Для проведения эксперимента сообщения, представленные в выборке были разделены на 3 класса:

1. Класс, сообщения из которого все письма считаются легитимной почтой.
2. Класс, сообщения из которого все пользователи считают спамом

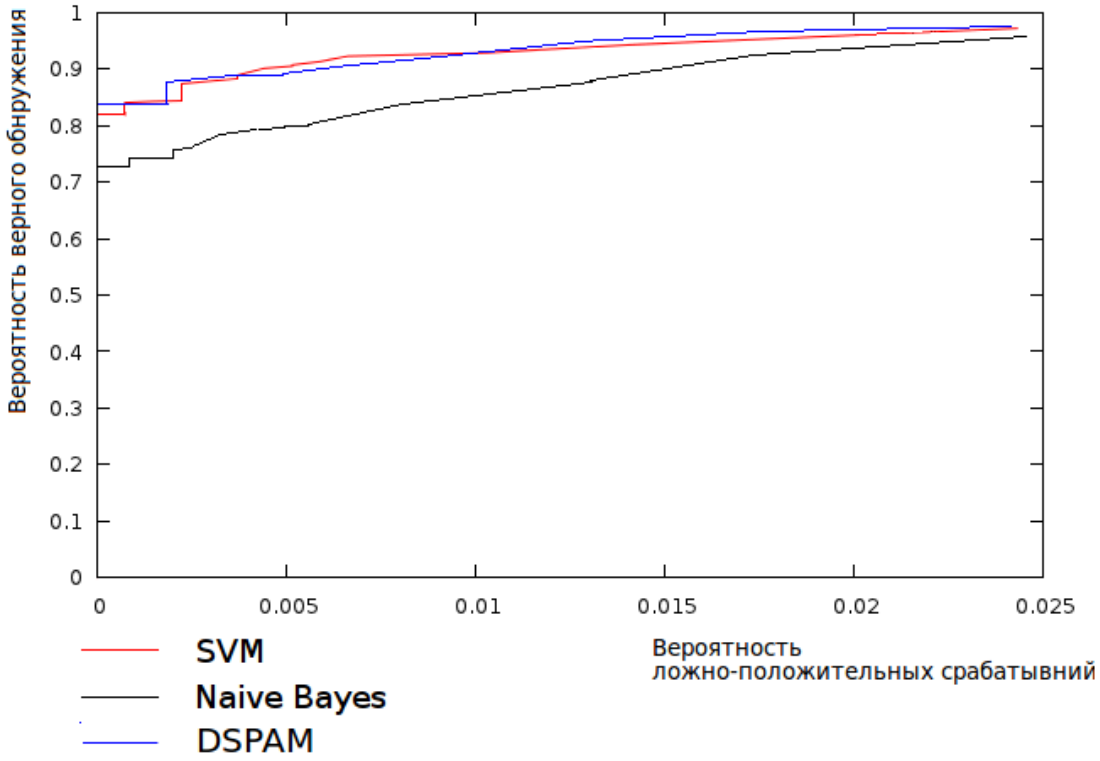



Рисунок 8 – Сравнение качества тестирования реализованного алгоритма, наивного байесовского классификатора на наборе[11]

3. Класс, сообщения из которого считают спамом некоторые из пользователей

В класс 1 были включены все сообщения, отмеченные в наборе [11] как спам. Классы 2 и 3 были получены из писем, отмеченных в наборе [11] как легитимная почта путем кластеризации (метод кластеризации описан в [14])

1. Эксперимент показывающий что многопрофильность работает, и обучение на данных одного пользователя распространяется на другого. Сообщения из всех трех классов были случайным образом разделены на две части (для первого и второго пользователей). Обучение производилось на всех трех наборах для первого пользователя и на первых двух для второго. Далее было произведено тестирование пользователей на сообщениях из третьей группы и посчитаны оценки вероятности принадлежности к спаму для писем. На результирующем графике показаны диапазоны вероятности и количество писем попавших в этот диапазон для каждого из пользователей.
2. Эксперимент показывающий что разное мнение пользователей о груп-

пе писем не приводит к проблемам. Для этого сообщения из всех трех классов были поделены случайным образом на три части (по одной части для трех тестовых пользователей). Для всех трех пользователей было произведено обучение на выборке из первого класса как на спаме, на выборке из второго класса как на легитимной почте. Дополнительно для первого пользователя было произведено обучение на выборке из третьего класса как на легитимной почте, для второго пользователя как на спаме, для третьего обучение на третьем классе не производилось. Тестирование производилось на контрольной выборке писем из третьего класса. На результирующем графике для каждого пользователя показаны диапазоны вероятности и количество писем попавших в этот диапазон.

### 6.2.2 Результаты тестирования

На графиках представлены результаты тестирования работы средства в многопрофильном режиме.

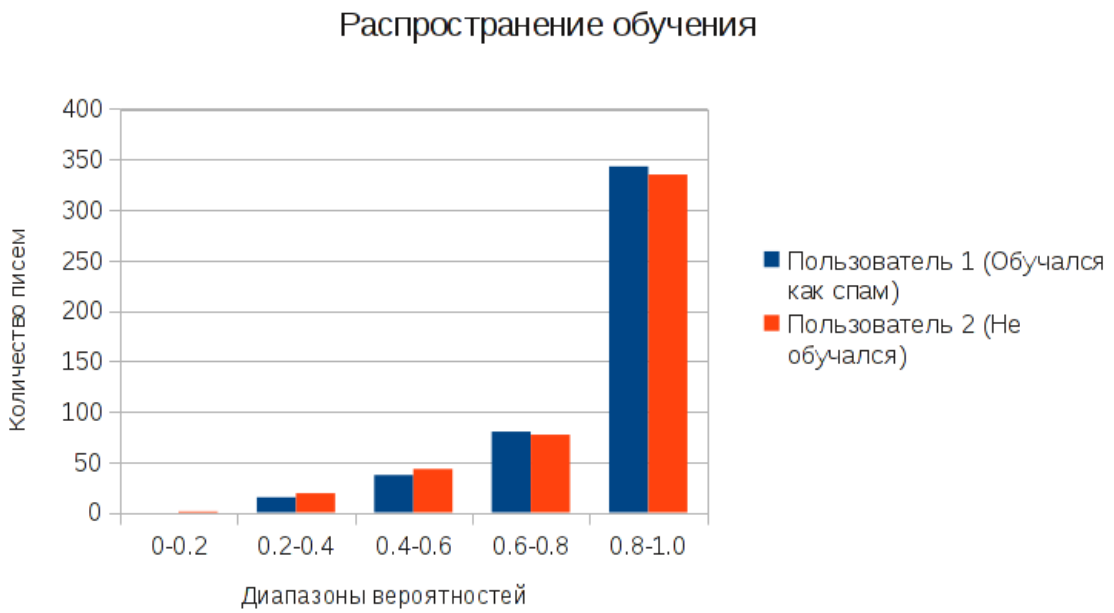



Рисунок 9 – Результаты тестирования пользователя на третьем тестовом наборе. Обучение было произведено на другом пользователе [11]

Из графика 9 видно, что обучение на одном пользователе распространяется на другого пользователя.

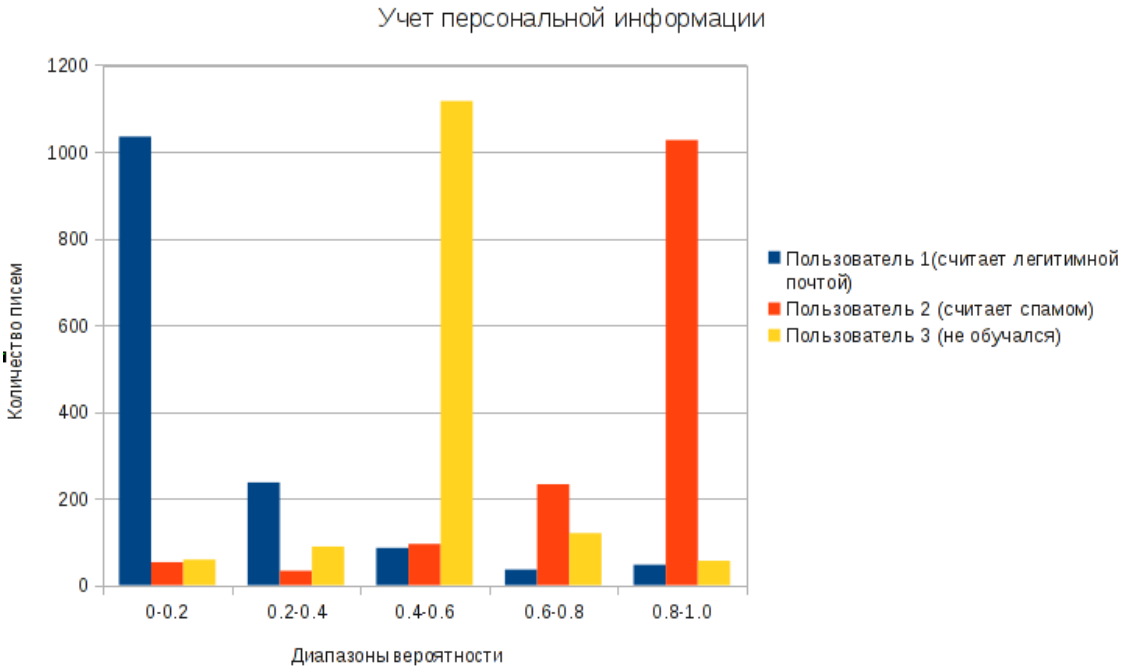



Рисунок 10 – Результаты тестирования на третьем тестовом наборе после обучения на данных из этого набора первого и второго пользователя.

Из графика 10 видно, что если мнения пользователей о каком-то классе писем различны, то система так же будет работать корректно. При этом если мнения пользователей о классе писем различаются, то для пользователя, который еще не добавлял писем из этого класса в обучающую выборку система будет выдавать некоторые промежуточные оценки.

## 6.3 Производительность

Для проверки производительности метода были произведены замеры времени добавления писем в выборку, времени генерации модели и время, которое занимает классификация писем. Кроме того, для сравнения, аналогичные замеры были произведены для стандартного байесовского классификатора, включенного в систему dspam. Стадия построения модели для байесовских алгоритмов отдельно не выделяется, поэтому для dspam такие замеры произведены не были.

Были произведены замеры времени добавления и построения моделей для 20 (10 спам + 10 легитимная почта), 200 (100 спам + 100 легитимная почта), и 2000 (1000 спам + 1000 легитимная почта) сообщений.

Для каждой построенной модели были произведены замеры времени классификации n/2 сообщений (где n - размер выборки).

Результаты замеров времени добавления в писем в модель, а также времени генерации модели представлены на графике 11

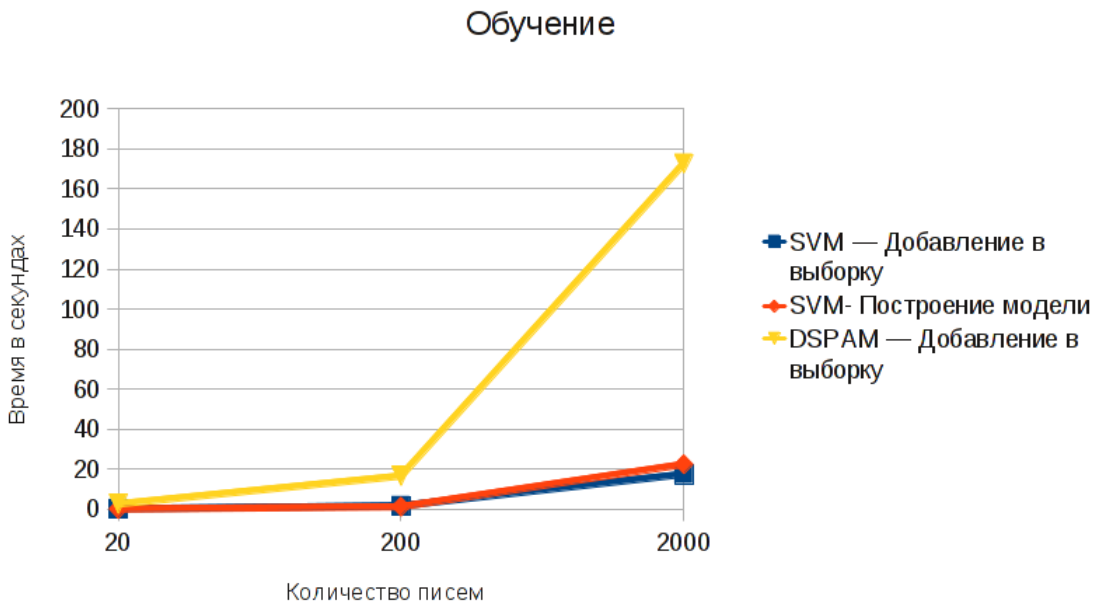



Рисунок 11 – Замеры времени обучения и генерации модели. Разработанное средство работает эффективнее стандартного алгоритма из системы dspam

Результаты замеров времени классификации представлены на графике 12

## 6.4 Апробация на реальном почтовом трафике

Данный эксперимент призван показать, что разработанное средство работает не только на тестовых сообщениях из открытых почтовых наборов, но и на реальных почтовых сообщениях.

Тестирование производилось на почтовых сообщениях одного из активных пользователей почтовой системы ЛВК, полученных им в течение одного


Рисунок 12 – Замеры времени классификации. Стандартный dspam эффективнее разработанного средства

месяца.

Эксперимент производился следующим образом:

- В обучающую выборку пользователя были добавлены письма, полученные пользователем в течение первого дня.
- Построена модель для метода опорных векторов.
- Для каждого из последующих дней производились следующие действия:
  - Произведена классификация всей почты, полученной пользователем за этот день, и вычислены верного обнаружения спама и ложно-положительного срабатывания.
  - Те сообщения, на которых разработанное средство допустило ошибку были добавлены в обучающую выборку.
  - Перегенерирована модель для метода опорных векторов.

В приложении в таблице 1 представлены частоты верного определения спама и вероятности ложно-положительного срабатывания для каждого из дней. Из таблицы видно, что уже после первого дня обучения разработанное средство показало хорошие результаты (вероятность верного обнаружения

близка к 1, вероятность ложного срабатывания к 0). Таким образом, тестирование показало, что разработанное средство может применяться не только на открытых тестовых наборах, но и на реальном почтовом трафике.

# 7 ЗАКЛЮЧЕНИЕ И РЕЗУЛЬТАТЫ

В рамках данной работы были решены следующие задачи:

- Произведен обзор существующих свободных систем фильтрации спама.
- Реализована фильтрация сапама на основе метода опорных векторов в рамках открытого средства.
- На основе метода опорных векторов разработан подход к фильтрации спама, позволяющий классифицировать сообщения по нескольким профилям.
- В рамках выбранного открытого средства фильтрации спама реализован такой подход.
- Произведено экспериментальное исследование, подтвердившее работоспособность метода.
- Произведена апробация метода на реальных данных.

Произведенное экспериментальное исследование на реальном почтовом трафике показало эффективность разработанного средства, поэтому в настоящее время производится подготовка к его промышленному внедрению в почтовую систему ЛВК.

Разработанное реализовано в виде модификации системы dspam. Осмысленным является включение этих модификаций в основное дерево разработки проекта. Для этого необходимо совместно с разработчиками dspam оформить изменений в соответствии с требованиями предъявляемыми к исходному коду проекта dspam.

# 8 ПРИЛОЖЕНИЕ

Таблица 1 – Апробация на реальных данных

| День | Верные срабатывания | ложно-положительные срабатывания |
|---|---|---|
| 2 | 0.997175141243 | 0.0 |
| 3 | 0.991176470588 | 0.00833333333333 |
| 4 | 0.988023952096 | 0.00724637681159 |
| 5 | 0.988826815642 | 0.0 |
| 6 | 0.993846153846 | 0.0382165605096 |
| 7 | 0.968208092486 | 0.0 |
| 8 | 0.990909090909 | 0.0 |
| 9 | 0.984802431611 | 0.0 |
| 10 | 1.0 | 0.00787401574803 |
| 11 | 0.994505494505 | 0.0298507462687 |
| 12 | 0.994750656168 | 0.0 |
| 13 | 0.98275862069 | 0.0 |
| 14 | 0.997084548105 | 0.0220588235294 |
| 15 | 0.990963855422 | 0.00662251655629 |
| 16 | 0.992518703242 | 0.0 |
| 17 | 0.997229916898 | 0.00704225352113 |
| 18 | 0.997159090909 | 0.0 |
| 19 | 0.994350282486 | 0.00757575757576 |
| 20 | 0.997326203209 | 0.0 |
| 21 | 0.994047619048 | 0.0 |
| 22 | 0.997142857143 | 0.00854700854701 |
| 23 | 1.0 | 0.0 |
| 24 | 0.997191011236 | 0.00724637681159 |
| 25 | 0.994202898551 | 0.00826446280992 |
| 26 | 0.994011976048 | 0.008 |
| 27 | 0.997395833333 | 0.0 |
| 28 | 1.0 | 0.0 |
| 29 | 0.997126436782 | 0.0149253731343 |
| 30 | 0.997014925373 | 0.0 |

# СПИСОК ЛИТЕРАТУРЫ


1. J. Klensin, Simple Mail Transfer Protocol, 2001. [HTML] http://www.faqs.org/rfcs/rfc2821.html
2. P. Resnick, Internet Message Format, 2001. [HTML] http://www.faqs.org/rfcs/rfc2822.html
3. Robert McMillan, What will stop spam?, SunWorld 1997. [HTML] http://sunsite.uakom.sk/sunworldonline/swol-12-1997/swol-12-vixie.html
4. Sender Policy Framework. Introduction. [HTML] http://www.openspf.org/Introduction
5. Evan Harris, The Next Step in the Spam Control War: Greylisting, 2004 [HTML] http://projects.puremagic.com/greylisting/whitepaper.html
6. Оффициальная документация проекта RAZOR. [HTML] http://razor.sourceforge.net/
7. P. Graham, A Plan For Spam, 2002 [HTML] http://www.paulgraham.com/spam.html
8. Gray Robinson, A Statistical Approach to the Spam Problem, Linux Journal, 2003 [HTML] http://www.linuxjournal.com/article/6467
9. Юрий Лифшиц, "Лекции для интернета: метод опорных векторов". [PDF] http://yury.name/internet/07ianote.pdf
10. A. Athanasopoulos, A. Dimou, V. Mezaris, I. Kompatsiaris, "GPU Acceleration for Support Vector Machines Proc. 12th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2011), Delft, The Netherlands, April 2011 [PDF] http://mklab.iti.gr/files/wiamis11.pdf
11. Набор сообщений для тестирования SpamAssassin public corpus [HTML] http://spamassassin.apache.org/publiccorpus/
12. Набор сообщений для тестирования CEAS 2008 Live Spam Challenge Laboratory Corpus [HTML] http://plg.uwaterloo.ca/ gvcormac/ceascorpus/
13. А. Петров, Разработка и реализация модифицированной подсистемы байес-анализа в системе SpamAssassin, Курсовая работа, МГУ, 2010

14. А. Розинкин, Система защиты от массовых несанкционированных рассылок электронной почты на основе методов DATA MINING, Диссертация на соискание ученой степени кандидата физико-математических наук, МГУ 2006

15. Stevan Bajić, Tokenizers in dspam, Dspam official documentation [HTML] http://sourceforge.net/apps/mediawiki/dspam/index.php?title=Tokenizers

16. Оффициальная страница libsvm, [HTML] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

17. К. В. Воронцов, Курс лекций по машинному обучению, [PDF] http://bit.ly/voroncov_ml

18. C. Cortes, V. Vapnik, Support-Vector Networks, Machine Learning, 20, 1995 [HTML] http://www.springerlink.com/content/k238jx04hm87j80g/

19. Сергей Тютин, "Сколько слов в русском языке". [HTML] http://linguistics-msu.livejournal.com/10968.html?thread=13784#t13784