Санкт-Петербургский государственный университет

Факультет прикладной математики — процессов управления

Пискунова Анна Сергеевна

Выпускная квалификационная работа бакалавра

Разработка приложения для автоматического определения спам-сообщений для устройств, работающих на платформе Android.

100400

Прикладная математика и информатика

кафедра «Технологии программирования»

Заведующий кафедрой,
Кандидат технических наук, доцент — Блеканов И. С.

Научный руководитель,
старший преподаватель — Малинана М. А.

Рецензент,
Доктор физ-мат наук,
профессор,
почетный работник Высшей школы РФ — Андрианов С. Н.

Санкт-Петербург

2017

Содержание

Введение

На сегодняшний день человечество не может обойтись без компьютерных и сетевых технологий. Средства связи вторглись в большинство сфер жизнедеятельности общества, начиная с помощи в образовании и заканчивая рекламой и продвижением какой-либо платной продукции. Отсюда возникла проблема получения нежелательных сообщений, иными словами, спама.

Слово «спам» произошло от названия марки консервов «SPAM» производства американской компании Hormel Foods. Во время Второй мировой войны этот продукт использовался в качестве продукта питания американских солдат, но, когда война закончилась, остались большие запасы продукции, и, чтобы избавиться от них, компания стала вести очень активную рекламу. С тех пор слово «спам» прижилось как название рекламной рассылки [1].

Коммерческие организации делают массовую рассылку, в том числе и людям, которые не хотели бы получать подобные сообщения. Иногда они даже представляют опасность, потому что могут содержать компьютерные вирусы, мошеннические ссылки. Кроме того, широковещательные рассылки влекут за собой большие затраты ресурсов сервера, отнимают время пользователя, затрачиваемое на прочтение и сортировку подобных писем.

Выделяют различные виды спама:

- Фишинг — попытка узнать секретные данные, такие как пароли, номера банковских карт и прочее.
- Реклама.
- Антиреклама — информация, направленная на уменьшение интереса пользователя к продукции какой-либо компании, к известной личности или нашумевшему событию.

- «Нигерийские письма» — носящие мошеннический характер, в которых говорится о якобы полученном наследстве и просьбе получателя прислать немного денег для оформления документов. Таким образом нарушитель закона выманивает деньги у обманутого человека.

По способам распространения спам классифицируется:

- Отправляемый на электронную почту — как правило, это спам в виде «нигерийских писем» или рекламы.
- Посылаемый в виде SMS по мобильной сети — обычно реклама или фишинг.
- Отправляемый пользователям социальных сетей.

По статистике доля спама в электронной почте составляет около 60%[1], в SMS-сообщениях - 15%[2]. Такое положение дел является поводом для развития технологий фильтрации сообщений.

Рост пользователей мобильных телефонов привел к резкому увеличению количества нежелательных SMS-сообщений. Несмотря на то, что для электронной почты существует много различных фильтров, борьбе со спамом на мобильных телефонах уделяется не так много внимания. Исходя из этого факта, перед автором данной работы была поставлена задача исследовать проблему SMS-спама, рассмотреть существующие решения и их недостатки и найти новый подход для детектирования нежелательных SMS-сообщений.

[1]По исследованию Лаборатории Касперского в 2016 году: `https://kasperskycontenthub.com/securelist-russia/files/2016/08/Spam-report_Q2-2016_final_RUS.pdf`

[2]Согласно аналитическим исследованиям РАЭК `raec.ru/activity/analytics/`

Цели и задачи

Целью данной работы является написание приложения для Android-устройств, блокирующего SMS-спам.

Основные задачи, необходимые для достижения поставленной цели:

1. Анализ существующих методов фильтрации спама.
2. Реализация приложения для смартфонов на платформе Android, перехватывающего входящие SMS-сообщения.
3. Установка и настройка удаленного сервера, на котором производится основная обработка текстовых сообщений.
4. Анализ некоторых SMS -сообщений, выявление среди них характерных признаков спама.
5. Обучение фильтра и его реализация на удаленном сервере.
6. Достижение полноты и точности классификации >85%.
7. Разработка графического интерфейса клиентской части приложения:
   (a) Реализация возможности отправлять SMS.
   (b) Появление уведомления, звукового и визуального оповещения при получении сообщения, не распознанного как спам. Отсутствие любых оповещений при получении спама.
   (c) Возможность удалять сообщения.
8. Размещение приложения на открытом ресурсе.

Для написания клиентской части приложения была выбрана система Android. На текущий момент это одна из самых популярных и быстроразвивающихся операционных систем. Под управлением Android работают не только телефоны и планшеты, но и телевизоры, наручные часы, нетбуки, в будущем планируется разработка автомобилей. Согласно статистике 65,17% смартфонов, проданных в июле 2016 года, работают под управлени-

ем операционной системы Android, и эти показатели не перестают расти[3].

Серверная часть разрабатываемого в рамках данной работы приложения реализована на Java. Выбранный язык программирования имеет существенные преимущества [2]:

- предоставляет разработчику широкие возможности;
- простота применения;
- независимость от платформы;
- встроенные функции защиты.

[3]По исследованиям statcounter (сервис, анализирующий веб-траффик) : `http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201607-201701`

Глава 1. Существующие подходы для решения задачи фильтрации спама

Для решения проблемы обнаружения спама используются различные фильтры. Они делятся на два типа: ручные и автоматические. Автоматические, в свою очередь, разделяются на кластеризаторы и классификаторы. Ниже приведены описания некоторых из них.

1.1. Ручная фильтрация

В основе неавтоматических фильтров лежат списки доступа. При использовании политики «черного списка», пользователю необходимо настраивать его самостоятельно, выбирая номера телефонов (в случае SMS) или e-mail адреса (в случае электронных писем) для блокировки. Но ручные способы фильтрации нежелательных сообщений имеют низкую эффективность и требуют постоянного обновления списков доступа, дополнительно нагружая пользователя [3, 4].

1.2. Кластеризация

Характерная особенность таких фильтров заключается в том, что их не нужно «обучать» вручную. В статье [5] в качестве одного из методов нахождения спама был использован EM-алгоритм.

1.2.1. Описание EM-алгоритма

Пусть $X = (x_1, ..., x_m)$ — множество объектов, подвергающихся кластеризации. Первоначально имеются априорные вероятности ω_j, плотности p_j

для k кластеров и начальный вектор параметров $\Theta = (\omega_1, ..., \omega_k, \theta_1, ..., \theta_k)$. Плотность распределения по всем кластерам:

$$p(x) = \sum_j^k \omega_j p_j(x),$$

причем $\sum_j^k \omega_j = 1, \omega_j \geqslant 0$.

Задача: оценить веса ω_j в данной сумме.

Идея метода состоит в использовании вектора скрытых переменных G, который вычисляется при известном векторе параметров Θ. Итерационно выполняются следующие действия [6]:

1. E-шаг — считается ожидаемое значение вектора G

$$g_{i,j} = \frac{\omega_j p_j(x_i)}{\sum_{s=1}^k \omega_s p_s(x_i)},$$

где $\sum_j^k g_{i,j} = 1,\ i = 1, ..., m$.

2. M-шаг — максимизация логарифма правдоподобия:

$$L(\Theta) = \ln \prod_i^m p(x_i) = \sum_{i=1}^m \ln \sum_{j=1}^k \omega_j p_j(x_i) \to \max_\Theta.$$

Отсюда следует:

$$\omega_j = \frac{1}{m} \sum_{i=1}^m g_{i,j},\ j = 1, ..., k,$$

$$\theta_j = \arg\max_\theta \sum_{i=1}^m g_{i,j} \ln \psi(x_i, \theta),\ j = 1, ..., k,$$

где $\psi(x, \theta_j) = p_j(x)$.

При использовании данного метода на выборке из 3900 сообщений получены результаты: точность = 85,5%, полнота = 17,04% [5]. Очевидным недостатком является неустойчивость алгоритма: получаемый результат зависит от начальных данных. Этим объясняется низкий процент найденного спама.

1.3. Классификация

По-другому подход называется «обучение с учителем». В роли «учителя» выступает конечная выборка заранее размеченных вручную объектов. На основании этих данных алгоритм восстанавливает функцию зависимости классов объектов от их признаков и в дальнейшем предсказывает принадлежность новых объектов к тому или иному классу.

1.3.1. Логистическая регрессия

Модель, в которой обучающие данные подгоняются к логистической кривой. Такой метод хорошо подходит для бинарной классификации объектов (разбиения всех объектов на 2 класса).

Этапы:

1. Сформировать обучающее множество $(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})$, где $x^{(i)}$ — n-мерный вектор значений признаков i-го объекта, $y^{(i)}$ — размеченный вручную маркер значения класса, к которому i-й объект относится.
2. Подбор весовых коэффициентов признаков $\theta_1, ..., \theta_n$, таких, что функция правдоподобия

$$L(\theta) = \prod_{i}^{n} P\{y = y_i | x = x_i\} \tag{1.1}$$

достигает при них максимума. Это эквивалентно нахождению максимума её логарифма.

Для этого применяется метод градиентного спуска, принцип которого состоит в итерационном вычислении θ по формуле [7]:

$$\theta := \theta + \alpha \sum_{i}^{m} \left(y_i - f(\theta^T x_i)\right) x_i, \quad \alpha > 0, \tag{1.2}$$

до тех пор, пока значение θ на предыдущем шаге не будет отличаться

от следующего на малую величину ε, здесь α — константа, которая подбирается вручную в зависимости от поставленной задачи.

3. Подстановка полученных параметров в логистическую функцию:

$$f(z) = \frac{1}{1+e^z}, z = \theta^{\mathrm{T}} x = \theta_1 x_1 + ... + \theta_n x_n.$$

Здесь $(x_1, ..., x_n)$ бинарный вектор признаков объекта, элементы которого принимают значение 1, если данный признак присутствует у объекта. Значение логистической функции представляют собой вероятность принадлежности элемента некоторому классу. Как правило, в случае бинарной классификации, граничное значение берется равным 0,5 [8] .

1.3.2. Множественная линейная регрессия

Общая форма множественной линейной регрессии имеет вид:

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p + \varepsilon,$$

где y — зависимая переменная, $\beta_0, ..., \beta_p$ — коэффициенты регрессии, а $x_1, ..., x_p$ — независимые переменные модели. Обычно принято считать, что ошибка ε стремится к нормальному распределению с мат. ожиданием $E(\varepsilon) = 0$ и с постоянной дисперсией [9]. Для нахождения коэффициентов используется метод наименьших квадратов [10].

1.3.3. Наивный байесовкий классификатор

Наивный байесовский классификатор основывается на теореме Байеса и формуле полной вероятности. Условная вероятность того, что сообщение A — спам, при том, что в нем содержится слово B, рассчитывается по формуле Байеса [11]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Метод является самообучаемым. Однако, у него есть существенные недостатки [12]:

- обрабатывается только текст сообщения;
- не учитывается номер отправителя;
- игнорируются редко встречающиеся слова.

1.4. Анализ и сравнение методов

На практике были рассмотрены и проанализированы следующие методы автоматической классификации: множественная линейная регрессия, логистическая регрессия и наивный байесовкий классификатор. Для обучения фильтров собрана выборка из 2100 сообщений, 1200 из которых — спам, 900 — не спам. Обучающая выборка была вручную классифицирована. Для сравнения эффективности методов составлена тестовая коллекция из 100 SMS-сообщений, 44 из которых — спам, 56 — не спам. Все сообщения на русском языке. Оценка качества классификаторов проводилась с использованием стандартных метрик:

- Полнота (*Recall*) — это доля найденных классификатором сообщений, являющихся спамом, относительно всех спам-сообщений в тестовой выборке.
- Точность (*Precision*) — это доля сообщений действительно являющихся спамом относительно всех сообщений которые программа отнесла к спаму.
- *F–mera* — гармоническое среднее между точностью и полнотой.

TP — количество сообщений, классифицируемых как спам и фильтром, и человеком. FP — количество сообщений, которые фильтр ошибочно отнес к спаму. FN — количество спам-сообщений, которые фильтр не отнес к нужному классу. TN — количество сообщений, классифицируемых как не

спам и фильтром, и человеком [13].

$$Pr = \frac{TP}{TP + FP},$$

$$Rec = \frac{TP}{TP + FN},$$

$$F_{mera} = \frac{2PrRec}{Pr + Rec}.$$

		Экспертная оценка	
		спам	не спам
Оценка системы	спам	TP	FP
	не спам	FN	TN

Рис. 1. Значения, используемые для оценки качества классификатора.

В таблице 1 приведены результаты тестирования методов:

Классификатор	Pr	Rec	F-mera
Логистическая регрессия	0,87	0,93	0,899
Множественная линейная регрессия	0,547	0,932	0,73
Наивный байесовский классификатор	0,837	0,818	0,827

Таблица 1. Оценки качества классификаторов.

Исходя из полученных метрик качества можно сделать вывод о том, что для поставленной в работе задачи наиболее точные результаты достигаются при применении логистической регрессии.

Глава 2. Практическая часть

2.1. Подготовка данных

Перед программной реализацией фильтра была проведена подготовительная работа, а именно: выбор наилучшего алгоритма для нормализации строк и подбор признаков, характеризующих SMS-сообщение.

2.1.1. Стемминг и лемматизация

В русском языке практически у каждого слова могут быть различные парадигмы: времена для глаголов, падежи для имен существительных и тому подобное. Отсюда возникает проблема обработки однокоренных слов, например, «акция» и «акцию» будут восприняты фильтром как два разных терма. Чтобы избежать подобных ситуаций, нужно приводить все слова к некой канонической форме. В данной работе было исследовано два пути решения этой задачи: стемминг и лемматизация.

Стемминг — процедура нахождения основы слова и отсечения его оставшейся части [14]. Автором использовалась Java библиотека Lucene Snowball, реализованная компанией Apache. Для работы со строками создается экземпляр класса RussianStemmer.

Алгоритм стемминга [15]:

1. Поиск и удаление окончаний, свойственных деепричастиям«в, вши, вшись, ыв, ывши»). Если такие не найдены, производится поиск возвратного постфикса («ся, сь») и его удаление, затем таким же образом ищутся и удаляются окончания, свойственные причастиям, прилагательным или существительным.
2. Если слово заканчивается на букву «и», удалить её.

3. Удаление словообразовательного суффикса(«ост, ость»), находящегося в абсолютном конце слова $R2$.

   $R1$ — часть слова после первой согласной, следующей за гласной.

   $R2$ — аналогично $R1$, но рассматривается только в области $R1$.

   В слове «абонент» $R1$=«бонент», $R2$=«нент».
4. Если окончание слова относится к превосходной степени («ейш, ейше»), оно отсекается.
5. Замена двойного «н» на одинарное.
6. Удаление «ь», если он стоит в конце слова.

Лемматизация — приведение слова к нормальной форме (лемме) [14]. В данной работе был рассмотрен инструмент TreeTagger, который взаимодействует с основным программным продуктом с помощью библиотеки TT4J. Он был разработан Хелмутом Шмидом в Институте вычислительной лингвистики Штутгартского университета. TreeTagger применим к множеству языков, в том числе и к русскому. Перед подключением библиотеки к проекту установлен пакет Tagger, к нему подключены необходимые скрипты и файл параметров языка. Лемматизация слов осуществляется с помощью объекта класса TreeTaggerWrapper, одним из основных методов работы с которым является процедура process().

Алгоритм	Количество символов (слов)				
	318 (48)	218 (30)	116 (17)	37 (5)	7 (1)
	Время выполнения в секундах				
TreeTagger	323	246	120,9	38,9	19,5
Snowball	0,094	0,073	0,062	0,042	0,041

Таблица 2. Сравнение времени обработки текста лемматизатором и стеммером.

В таблице 2 представлено сравнение времени обработки сообщений лем-

матизатором TreeTagger и стеммером Snowball. По строкам расположены типы обработчика, а по столбцам — время работы в зависимости от количества символов в строке. Дополнительно в скобках указано количество слов в анализируемых строках. Ввиду низкой скорости работы программы TreeTagger, было решено использовать параллелизацию. Для этого проделаны следующие действия:

1. С помощью фабрики класса Executors создается пул с четырмя потоками. ExecutorService service=Executors.newFixedThreadPool(4).
2. Во вложенном классе Task, реализующем интерфейс Callable, в переопределенном методе call() вызывается метод treeTagger(String word). Он принимает на вход слово и возвращает его лемму.
3. В главном потоке создается коллекция ArrayList<Future<String>> results для хранения обработанных строк.
4. В цикле, проходящим по всем словам, содержащимся в сообщении, объект service вызывает метод submit с интстанированием класса Task, полученный результат добавляется в коллекцию и при завершении цикла выполняется процедура shutdown() и awaitTermination().

Алгоритм	Количество слов (символов)				
	318 (48)	218 (30)	116 (17)	37 (5)	7 (1)
	Время выполнения в секундах				
TreeTagger	323	246	120,9	38,9	19,5
TreeTagger с параллелизацией	168,5	148	56,6	13,6	4,1

Таблица 3. Сравнение времени обработки текста лемматизатором с параллелизацией и без.

В таблице 3 показаны результаты, полученные с помощью распарал-

леливания. По строкам таблицы размещаются способы лемматизации, по столбцам — время работы в зависимости от количества символов в строке. Дополнительно в скобках указано количество слов в анализируемых строках. В среднем время при параллельном выполнении уменьшается в 2 раза. Это обусловлено тем, что строки подвергаются лемматизации не по одной. Одновременно обрабатываются несколько строк, каждая в отдельном потоке. Несмотря на увеличение скорости обработки текста лемматизатором при использовании параллельных процессов, время работы стемминга, в зависимости от длины строки, меньше на 2, а иногда и на 3 порядка. Это послужило причиной использования в проекте стеммера Snowball.

2.1.2. Описание признаков

На основе экспертного анализа SMS-сообщений были сформулированы следующие признаки, необходимые для представления сообщения в виде бинарного вектора:

1. Наличие слов, написанных заглавными буквами (обычно такой прием используется в рекламе для привлечения внимания клиента).
2. Сообщение отправлено с короткого номера.
3. Наличие в тексте тройного восклицательного, вопросительного знака или других специфических символов.
4. Присутствие веб-адреса (в большинстве случаев в спам-сообщениях рекламодатели оставляют ссылку на их сайт).
5. Ложное оповещение о пополнении баланса, отправленное не оператором связи абонента.
6. Присутствие ключевых слов, характерных для того или иного класса. Список наиболее часто встречающихся слов содержится в словарях:
   (a) Спамовые слова («выигрыш», «бонус», «кредит» и т.п.)

(b) Неспамовые слова (разговорная лексика и личные местоимения).

7. Наличие номера телефона в тексте SMS-сообщения.
8. Сообщение отправлено с длинного номера, начинающегося с «+7».
9. Смайлы.
10. Полученное SMS — от оператора (как правило, такие сообщения содержат важную для абонента информацию: состояние баланса, сведения о доступных тарифах, предупреждение о роуминге и т.д.).

Для каждого сформулированного свойства методом максимального правдоподобия (1.1) был вычислен весовой коэффициент, значение которого послужило одним из оснований для принятия решения о необходимости использования признака. Также учитывались такие факторы, как относительная частота встречаемости и «зеркальность» свойств. Относительная частота встречаемости вычислялась по формуле:

$$\nu = \frac{C}{N},$$

где C — частота встречаемости признака, N — размер выборки.

Были исключены признаки под номерами: 5 (в связи с редкостью проявления), 2 (так как является взаимоисключающим с 8), 9 (как имеющий вес, практически не влияющий на значение характеристической функции). Оставшиеся признаки и соответствующие им весовые коэффициенты и частоты приведены в таблице 4, где положительные веса имеют свойства, характерные для спама, отрицательные — для не спама.

№ признака	1	3	4	6(a)	6(b)	7	8	10
Весовой коэффициент	0,2969	1,3500	1,0939	3,2225	-2,6735	1,6269	-7,1014	-5,2812
Относительная частота	0,379	0,041	0,244	0,627	0,240	0,202	0,240	0,072

Таблица 4. Весовые коэффициенты и частоты признаков.

2.2. Программная реализация

2.2.1. Реализация логистической регрессии

Обучение классификатора проводилось с помощью выборки из 2100 сообщений, 1200 из которых — спам, 900 — не спам. Спамовые сообщения взяты частично с сайта[4] с примерами массовых рекламных рассылок различных компаний, остальные, включая неспамовые, — из баз данных телефонов реальных людей. Для каждого объекта выборки получен восьмимерный бинарный вектор значений его признаков и размеченный вручную маркер класса, к которому SMS-сообщение относится. Чтобы получить зависимость класса от признаков, была использована логистическая регрессия, определение которой давалось в разделе 1.3.1, а для её реализации пакет прикладных программ MatLab. Данная система очень удобна для программирования алгоритмов, работающих с матрицами, быстро справляется с большими массивами данных и имеет удобный графический интерфейс. В итерационной формуле градиентного спуска (1.2), автором использовались значения: $\alpha = 0,0005$, начальное значение $\theta_0 = (1;1;1;1;1;1;1;1)$, $\varepsilon = 0,005$.

[4] `http://www.aramba.ru/sms-examples`

2.2.2. Реализация клиентской части приложения

Клиентская часть приложения состоит из нескольких классов, каждый из которых выполняет свою функцию. Они делятся на два типа: классы, работающие фоном, и Activities, которые нужны для визуального представления приложения.

Фоновые классы:

MessageReceiver — создан для перехвата входящих SMS-сообщений, обмена информацией с сервером и оповещения клиента. Наследуется от класса BroadcastReceiver — приемника широковещательных сообщений, который включает в себя метод-обработчик onReceive(). В теле метода извлекается сведения о SMS-сообщении (номер отправителя, текст сообщения и название оператора симкарты) и отправляется на сервер. Чтобы отправить данные на сервер, создается отдельный поток и в нем с помощью объекта класса Socket с параметрами: порт и ip адрес удаленного компьютера, происходит соединение с сервером. Затем, используя метод readBoolean(), вызываемого объектом класса DataInputStream, клиент получает результат: true в случае спама, иначе — false. SMS со всеми его характеристиками (спам/не спам, прочитанное/новое, входящее/исходящее) сохраняется в базу данных, а клиенту в случае не спамового сообщения приходит оповещение о его получении, уведомление создано с помощью классов Notification и NotificationManager. Оно появляется в виде всплывающего окна с изображением сообщения, звукового сигнала и мигающего светодиода.

DbOpenHelper — требуется для создания базы данных и таблиц, а также их обновления, наследуется от класса SQLiteOpenHelper.

ItemAdapter — собственная реализация адаптера для отображения списка сообщений. Все элементы списка — объекты класса Item с полями number (номер телефона), message (текст сообщения), filter (спамовое ли сообще-

ние), image(входящее сообщение или исходящее). Макет каждого сообщения прописан в файле list_view.xml, стили которого меняются в зависимости от вида сообщения. Сообщения имеют пометку в виде изображения (входящие — стрелка вниз, исходящие — стрелка вверх).

Классы, организующие визуализацию экранов, устроены похожим образом: все они наследуются от класса AppCompatActivity и к каждому привязан макет в формате xml, который подключается с помощью метода setContentView() при создании Activity, то есть в методе onCreate() [2].

MainActivity — точка входа приложения, то есть этот класс отвечает за отображение экрана при запуске приложения. Создан для вывода существующих в базе данных сообщений. Макет activity_main.xml, содержание которого отображается на экране, включает в себя элемент ListView, каждый пункт которого содержит в себе информацию о сообщении: входящее/исходящее, прочитанное/новое, спам/не спам, тело сообщения и номер отправителя. Также на странице реализованы следующие виды меню:

- контекстное меню с пунктами «написать сообщение», «удалить все сообщения»;
- всплывающее меню с действием «удалить сообщение», «ответить», которое открывается после «длинного нажатия» по одному из элементов списка.

Send — нужен для реализации отправки SMS-сообщений на другие номера. Основную задачу выполняют методы sendMultipartTextMessage() класса SmSManager и registerReceiver() класса Context. При успешной отправке SMS информация об отправленном сообщении помещается в базу данных. Макет «activity_send.xml», загружаемый при открытии данного экрана приложения, содержит в себе поле для ввода номера, поле для набора сообщения, кнопки «Отправить» и «Домой» и иконку, при клике на которую осуществляется переход на поиск номеров и контактов из телефонной

книги.

SecretMessageActivity — необходим для отображения текста конкретного сообщения, подключает макет secret.xml, содержащий 2 TextView: с номером и текстом; и 2 кнопки: «Домой», при нажатии на которую пользователь попадает на главную страницу, и «Отправить SMS», перенаправляющую на форму отправки сообщения.

ActivitySearch — класс для отображения списка контактов, хранящихся в телефоне и поиска среди них нужного, подключает макет activity_search.xml, содержащий поля для ввода имени и список listview, в который динамически выводятся содержащие введенную строку контакты. С помощью объекта класса ContentResolver выполняются запросы к базе данных контактов: с универсальными идентификаторами ресурса (далее URI) ContactsContract.Contacts.CONTENT_URI и ContactsContract.CommonDataKinds.Phone.CONTENT_URI. Здесь первый URI считывает все имена, второй — все номера телефонов. Динамический поиск по контактам осуществляется путем реализации интерфейса TextWatcher. Он включает в себя методы beforeTextChanged(), onTextChanged() и afterTextChanged(). При клике на элемент списка клиент перенаправляется обратно на Send, где в поле ввода номера вставлен номер, соответствующий выбранному из списка имени.

Архитектура приложения устроена так, что со всех Activity можно перейти на любое другое Activity. Каждое из них должно быть прописано в файле Manifest.xml, который находится в корне проекта. Также в данный файл записываются разрешения, необходимые приложению для правильной работы (тег <uses-permission>), приоритет обработчика сообщений.

2.2.3. Реализация серверной части приложения

Для работы сервера, расположенного на удаленной машине, созданы 4 класса: Server, MyMessage, KeyWords и Snow.

Server — главный класс, в котором определен метод main(). Он выполняет задачу соединения с клиентом, получения его данных, их обработки и отправления результата клиенту. Подключение осуществляется с помощью объекта класса ServerSocket и метода accept(). Методом getInputStream() извлекается входной поток сокета, из которого считываются данные, отправленные клиентом. Затем осуществляется обработка данных и подсчет значения характеристической функции.

Отправка результата клиенту осуществляется с помощью методов getOutputStream() и writeBoolean() классов OutputStream и DataOutputStream соответственно. После отправки ответа клиенту, сервер возвращается к поиску новых подключений.

MyMessage — класс, объект которого представляет SMS-сообщение. Полями являются признаки сообщения — переменные типа boolean, значения которых определяется методами класса.

1. upperCaseWords(String message) — проверяет текст сообщения на наличие слов, написанных заглавными буквами, при этом слово должно состоять более, чем из одной буквы, чтобы исключить союзы и предлоги, находящиеся в начале предложения. Помимо этого, если все сообщение написано заглавными буквами, считается, что признак отсутствует.
2. punctuationMarks(String message) — исследует, есть ли в сообщении характерные для спама знаки препинания: тройной восклицательный, вопросительный знак, символы «%», «$», «*».
3. webAddress(String message) — выявляет в сообщении присутствие ссылки на интернет-ресурс.

4. keyWords(String message) — находит в тексте SMS ключевые слова, свойственные спаму.
5. hasNumber(String message) — проверяет, содержится ли в тексте сообщения номер телефона.
6. longNumber(String number) — анализирует номер отправителя SMS, если сообщение пришло от длинного номера, начинающегося с «+7», фиксирует наличие признака.
7. isFromOperator(String number,String operator) — проверяет, является ли отправитель оператором связи абонента.

Snow — класс, импортирующий библиотеку Snowball, при помощи которой можно осуществлять стемминг слов.

KeyWords — класс, представляющий файл с теми или иными ключевыми словами в виде списка строк ArrayList<String>, прошедших процедуру стемминга.

2.3. Настройка удаленного сервера

В качестве сервера используется удаленная машина с операционной системой Ubuntu 16.04.1 LTS x86_64, на которой установлен пакет SSH свободной реализации OpenSSH версии SSH-2. Протокол SSH обеспечивает безопасный канал по незащищенной сети, подключая разработчика к удаленному серверу. Данный протокол шифрует весь трафик, чтобы избежать прослушивания, захвата соединения и других атак. Кроме того, OpenSSH предоставляет большой набор возможностей безопасного туннелирования, методов проверки подлинности и усовершенствованные параметры конфигурации [16]. Подключение локального компьютера к удаленному выполняется через терминал путем выполнения команды:

ssh username@ip_address -p password.

Здесь username — логин пользователя, которому предоставлены права подключения, password — его пароль, ip_address — ip адрес удаленного компьютера.

Такой способ взаимодействия удобен тем, что сервер не привязан к конкретной машине, есть возможность управлять им удаленно. Для того, чтобы SSH сессия не прерывалась и сервер работал без сбоев, необходимо внести изменения в стандартные настройки соединения, а именно добавить «ServerAliveInterval 60» в файл /.ssh/config. В связи с тем, что взаимодействие с удаленной системой происходит через терминал, компиляция кода и сборка проекта с помощью среды разработки невозможна, поэтому данные процедуры также производились посредством командной строки.

Этапы переноса серверной части программы на удаленную машину:

1. Создание каталога проекта со всеми необходимыми подкаталогами и размещение в них файлов исходного кода и библиотек.
2. Компиляция всего проекта целиком:

   javac -Xlint:unchecked -sourcepath src -d bin -classpath /path/library.jar /path/class.java

   Здесь /path/library.jar — путь к библиотеке, используемой в проекте, /path/class.java — путь к главному классу проекта, содержащему метод main.
3. Создание файла манифеста и подключение библиотек к самому проекту:

   echo class-path: lib/org.annolab.tt4j-1.0.15.jar >manifest.mf
4. Формирование jar-архива:

   jar -cmef manifest.mf package_name.MainClass Archive.jar -C bin .

   Здесь package_name — название пакета, MainClass — имя главного класса, Archive — название jar-архива, с помощью которого в дальнейшем

запускается проект.

5. Запуск проекта:

java -jar /path/Archive.jar,

где path — путь к архиву в файловой системе.

Глава 3. Результаты работы

3.1. Архитектура приложения

Разработанное приложение для автоматического определения спама среди SMS-сообщений имеет архитектуру, представленную на Рисунке 2

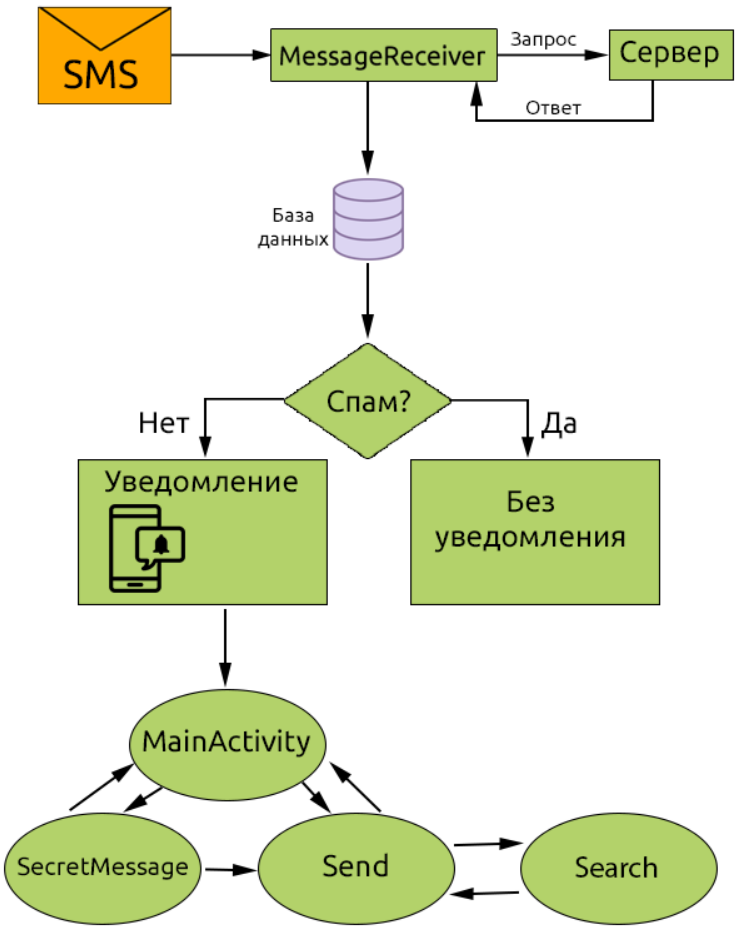

Рис. 2. Архитектура приложения.

1. Класс MessageReceiver перехватывает входящее SMS-сообщение, отправляет информацию о нем на сервер, на котором происходит обработка сообщения и отправка ответа клиенту.
2. После получения результата с сервера все данные о сообщении заносятся в базу данных.

3. Если фильтр не отнес сообщение к спаму, появляется уведомление и осуществляется переход на MainActivity. Иначе, ничего не происходит.
4. С MainActivity клиент может перейти на другие Activity (SecretMessage, Send) и обратно путем нажатия соответствующей кнопки или выбора пункта меню. Также есть возможность напрямую перейти с SecretMessage на Send, с Send на Activity_Search и обратно.

3.2. Оценка качества классификации

В таблице 5 приведены данные для оценки качества фильтра, используемого в приложении, определение которых давалось в разделе 1.4.

Данные	TP	FP	FN	TN
Значение	33	2	2	13

Таблица 5. Данные для оценки качества фильтра, используемого в приложении.

Для окончательной проверки работы классификатора собрана тестовая коллекция, состоящая из 50 SMS-сообщений. В качестве асессоров выступали 5 человек. 35 сообщений они отнесли к спаму, 15 — к не спаму. После этого коллекция отправлена на обработку приложению. Результаты автоматической классификации, а именно вычисленные значения метрик, приведены в таблице 6.

Метрика	Pr	Rec	F-mera
Значение	0,94	0,94	0,94

Таблица 6. Метрики оценки качества фильтра, используемого в приложении.

Для сравнения работы приложения с другими фильтрами была установлена программа «Фильтр СМС Спама» от разработчика Singulapps[5].

[5] https://play.google.com/store/apps/details?id=singulapps.smsfilter.pro&hl=en

Тестирование было произведено на той же коллекции из 50 SMS-сообщений. Его результаты приведены в таблицах 7 и 8. Видно, что фильтр слишком грубый, так как среди 15 неспамовых сообщений 10 классифицировал, как спам.

Данные	TP	FP	FN	TN
Значение	35	10	0	5

Таблица 7. Данные для оценки качества фильтра, используемого в стороннем приложении.

Метрика	Pr	Rec	F-mera
Значение	0,77	1	0,87

Таблица 8. Метрики оценки качества фильтра, используемого в стороннем приложении.

Для достижения хорошего качества классификации SMS-сообщений одинаково важны высокие показания точности и полноты. При стремлении полноты к 1, точность уменьшается. Этот же закон действует в обратном направлении. В ходе проверки работы разработанного в рамках данной задачи фильтра были получены оптимальные и достаточно хорошие значения метрик качества. Значение метрики *F-mera* для разработанного фильтра больше, чем для выбранного существующего в свободном доступе.

3.3. Демонстрация работы приложения

При запуске приложения открывается главное Activity, на котором выводятся все сообщения из базы данных. На рисунке 3 показан скриншот экрана приложения в момент получения нового сообщения.

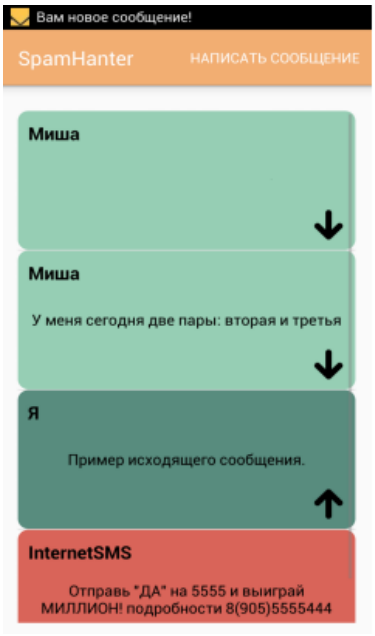

Рис. 3. Главный экран.

На рисунке 4 скриншот экрана MainActivity, сделанный во время долгого нажатия на элемент списка.

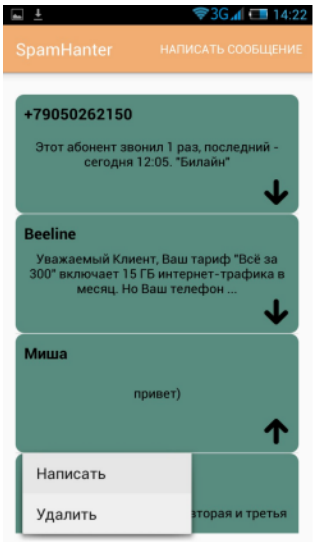

Рис. 4. Демонстация контекстного меню.

На рисунке 5 показан экран «MainActivity» до и после выбора пункта меню «Удалить все сообщения».

На рисунке 6 показаны скриншоты экрана Activity_Search. Производится динамический поиск контактов, содержащихся в телефонной книге,

по именам.

Рис. 5. Результат удаления всех сообщений.

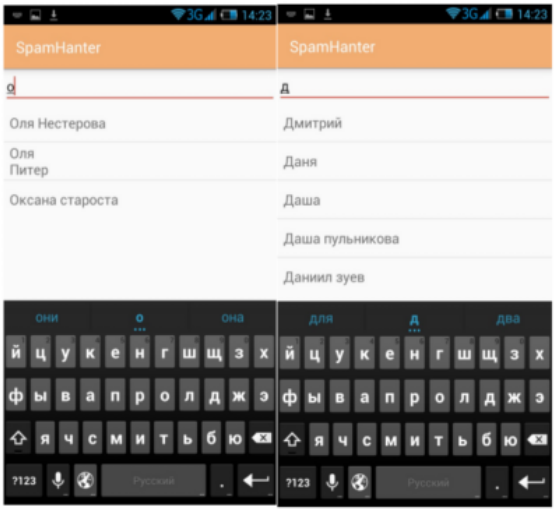

Рис. 6. Поиск контактов по имени.

На рисунке 7 продемонстрирован экран Send, посредством которого производится отправка сообщения.

На рисунке 8 представлен экран SecretMessageActivity, переход на который осуществляется либо при выборе сообщения из списка, отображенного в MainActivity, либо при нажатии на уведомление.

Рис. 7. Экран отправки сообщения.

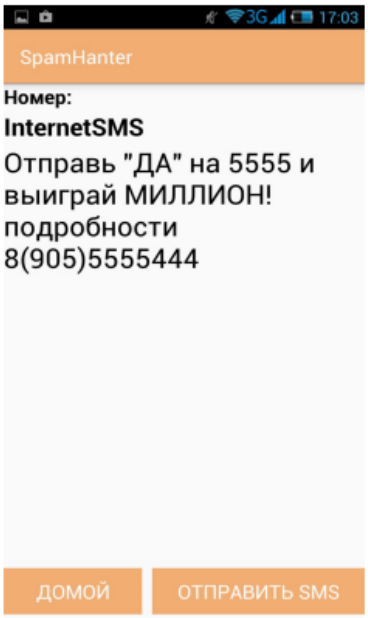

Рис. 8. Экран чтения сообщения.

Выводы

Поставленную задачу можно решить автоматически, так как выбранный метод показывает достаточно точный результат. Подобранные признаки, определяющие векторную модель SMS-сообщения, хорошо подходят для автоматической классификации. Приложение работает стабильно, удаленный сервер функционирует без сбоев. В момент установки на смартфон объем занятой приложением памяти составляет 2.9 Мб. При одновременном запуске приложения на нескольких устройствах сервер имеет возможность перехватывать все сообщения, приходящие на эти устройства.

Заключение

В ходе работы реализовано клиент-серверное для смартфонов, работающих на платформе Андроид. Приложение обладает удобным графическим интерфейсом, с помощью которого пользователь с легкостью может читать, писать и удалять SMS-сообщения. Реализовано клиент-серверное взаимодействие между приложением на смартфоне и удаленной вычислительной машиной, выполняющей всю работу по анализу сообщений. Проведен анализ существующих методов борьбы со спамом и их сравнение с разработанным приложением. В ходе проверки фильтра получены оптимальные и достаточно большие значения метрик качества: точность (Pr), полнота (Rec), $F\text{-}mera$.

Список литературы

1. Борьба со спамом: история и методы [Электронный ресурс] // URL: https://mipt.ru/dmcp/student/diff_articles/no_spam.php (дата обращения: 12.02.17).
2. Брюс Эккель. Философия Java. 4 изд. СПб.: Питер , 2016. 1168 с.
3. Мироненко А. Н. Автоматическая фильтрация спама на базе сети формальных нейронов // Вестник омского университета. 2011. № 2. С. 178–182.
4. Коробейников А. Г., Блинов С. Ю., Лейман А. В., Маркина Г. Л., Кутузов И. М. Алгоритм определения спамности документов на основе фейеровских отображений // Научно-технический вестник информационных технологий, механики и оптики. 2012. № 6. С. 123–127.
5. Tiago A. Almeida, José María G. Hidalgo, Akebo Yamakami Contributions to the Study of SMS Spam Filtering: New Collection and Results // ACM New York. 2011. № 5. С. 259–262.
6. stanford.edu [Электронный ресурс] // URL:http://cs229.stanford.edu/notes/cs229-notes8.pdf (дата обращения: 19.12.16).
7. Supervised learning [Электронный ресурс] // URL:https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf (дата обращения:05.03.2017).
8. Фальк В. Н., Бочаров И. А., Шаграев А. Г. Трансдуктивное обучение логистической регрессии в задаче классификации текстов // Программные продукты и системы. 2014. № 2 (106). С. 115–118.
9. Xin Yan, XiaoGang Su Linera regression analysis. Singapore: World Scientific, 2009. 328 с.
10. Калиткин Н. Н. Численные методы. 2 изд. СПб: БХВ-Петербург, 2011. 592 с.

11. Гмурман В. Е. Теория вероятностей и математическая статистика : учебное пособие для вузов. 11 изд. М.: Высшая школа, 2005. 479 с.
12. Машечкин И. В., Петровский М. И., Розинкин А. Н. Система предотвращения массовых рассылок на основе алгоритмов машинного обучения // Программные продукты и системы. 2005. № 3. С. 10–15.
13. Журавлева Л. В., Стригулин К. А. Анализ тональности отзывов пользователей в мета-области фильмов // Молодой ученый. 2016. № 12. С. 157–161.
14. Яцко В. А. Алгоритмы и программы автоматической обработки текста // Вестник Иркутского государственного лингвистического университета. 2012. № 1. С. 150–161.
15. Russian stemming algorithm [Электронный ресурс] // URL:http://snowball.tartarus.org/algorithms/russian/stemmer.html (дата обращения: 19.11.16).
16. Dale Liu Next Generation SSH2 Implementation. 1 изд. Rockland: Syngress, 2008. 336 с.