***Текст для выступления.***

Всем здравствуйте!

Сейчас(Нам) хотелось бы рассказать о нашем совместном проекте « Блокировка спам сообщений»

1. Введение

Итак, начнем с того, что разработка антиспам-технологий приобрела свою актуальность с момента возникновения электронных почт и проблема получения спам-рассылок остается не менее актуальной сейчас. Наверняка, каждому из вас приходило приветственное письмо от интернет-магазина, где, скажем, перед текстом самого письма вы видите промо-код скидки. Этот ход помогает удержать внимание на письме. Далее идет информация о: продукции, компании, специальных предложениях. И вот такая сбалансированная структура письма помогает постепенно вовлекать новых подписчиков в диалог с магазином.

Это был только 1 пример рассылок спама, а их в этом мире миллионы.

Если в 2000 году средний пользователь получал не более двух спам-писем в день, то в 2004-м году это количество выросло до 8-10. Объем спама в почтовом трафике среднего провайдера или почтовой службы не превышал 20%, на сегодняшний день спам в среднем составляет до 56% входящего на почтовые сервера трафика.

Было совершено множество попыток раз и навсегда решить проблему спама технологическим путем, однако сейчас уже с полной уверенностью можно сказать, что единого решения — некой «серебряной пули» против спама — не существует.

Спам крайне разнообразен и постоянно меняется, поэтому в борьбе с ним абсолютно необходимо использование все новых и новых методов распознавания.

На начальном этапе наша команда столкнулась с проблемой выбора реализации нашей программы. Все мы знаем, что одну и ту же функциональность можно реализовать с помощью совершенно разных технологий: можно написать программу для Windows, сделать web-сервис, мобильное приложение для Android и/или iOS и т.д. Что же нас подтолкнуло сделать выбор в пользу приложения для Chrome?

Во-перых, приложения Chrome без каких-либо дополнительных усилий работают на Windows, Linux и OS X. Конечно, есть множество других способов сделать приложение переносимым, но большинство из них оказываются заметно более затратными.

Во-вторых, С недавних пор появилась возможность [портировать Chrome apps на Android и iOS](http://www.theverge.com/2014/1/28/5355064/chrome-apps-are-coming-to-ios-and-android).

И наконец, на большинстве систем приложения Chrome выглядят для пользователя как обычные программы. Они запускаются из меню «Пуск», открывают обычные окна без браузерных контролов, могут использоваться в качестве программ по умолчанию для открытия файлов, и в остальном ведут себя как полноправные программы.

Итак, для решения нашей проблемы, мы создали расширение в браузере- это компьютерная программа, которая в некотором роде расширяет функциональные возможности браузера

Фундаментом всего расширения является файл manifest.json, в котором описываются все важные свойства и зависимости.

4)

Далее для решения проблемы нам понадобился API (Application Programming Interface) —это описание способов , которыми одна компьютерная программа может взаимодействовать с другой. Если говорить более понятным языком, то API — это готовый код для упрощения жизни программисту.

API определяет функциональность, которую предоставляет программа( модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована.

Наша цель в применение API заключается в том, что сервер сайта должен напрямую обращаться к серверу Google с запросом на создание события с указанными деталями, получать ответ Google, обрабатывать его, и передавать соответствующую информацию в браузер, например, сообщение с запросом на подтверждение пользователю.

Также Google API нужен был для того, чтобы реализовать следующие функции : \*покажи на слайд\* (вход пользователя в свой аккаунт gmail, проверка новых сообщений, смена статуса сообщений(перемещение его из папки входящие в папку спам).)

На этом этапе у нас уже появились некоторое проблемы, а именно : чтобы использовать Api нужно создать аккаунт разработчика и зарегистрировать свое приложение в google cloud platform и тогда уже можно выбрать gmail api и использовать его внутри приложения. ( Но , как оказалось, при регистрации, нужно отдать небольшую сумму, чтобы нас зарегистрировали на платформе, следовательно финансово в наш поет мы уже вложились).

5)

Следующим шагом выполнения нашей работы была реализации отправки и получения запроса, которые должны выполняться неоднократно, независимо от срока жизни любой конкретной веб-страницы или окна браузера. Для этого, как мы выяснили, нужны фоновые сценарии.

Фоновые сценарии загружаются сразу после первого запуска расширения в браузере и будут работать загрузки до тех пор, пока расширение не будет отключено или удалено. Если запросить необходимые разрешения, то можно использовать любой API расширений в сценарии.

Мы подключили фоновый скрипт, используя ключ “background” в «manifest.json»:

В нашей программе background скрипт используется для регулярной проверки появления новых сообщений в почте.

Стоит отметить, что для отправки запросов на сервер мы использовали обьект XMLHttpRequest, который дает возможность с помощью JavaScript делать HTTP-запросы к серверу без перезагрузки страницы.

6)*\*откррываем приложуху\**

Для начала мы решили добавить в наше приложение, наверное, самую легкую возможность фильтрации спама - по черному списку. Чтобы хранить ЧС для каждого пользователя мы использовали syncstorage – оно позволяет сохранять информацию для пользователя вне зависимости от того, с какого устройства он заходит, в отличие от локального, который сохраняет информацию только для одного устройства.

С помощью этого хранилища мы можем показать весь черный список, мы можем удалить весь черный список, добавить пользователя в ЧС и удалить его оттуда. ( \*показываем\*)

7)

Современная спам-рассылка распространяется в сотнях тысяч экземпляров всего за несколько десятков минут, поэтому просто реализации черного списка недостаточно для решений проблемы.. Современная индустрия IT-безопасности предлагает множество решений, и в арсенале антиспамеров есть различные технологии. Однако универсального решения просто не существует. Большинство современных продуктов используют несколько технологий, иначе эффективность продукта будет не высока.

При разработке основного алгоритма фильтрации спама, просмотрев много источников и литературы, мы выявили 2 наиболее продуктивных и глобальных метода борьбы со спамом :

Первый-это метод на основе искусственной нейронной сети :

Предлагаемый метод фильтрации спама основан на использовании

нейронной сети, выступающей в качестве механизма принятия решений, давая на выходе

вероятность того, что сообщение относится к спаму. Искусственная нейронная сеть

обладает способностью обучаться (в том числе, обобщать свои знания, накапливать опыт),

является наиболее приближенной моделью человеческого мозга, как по архитектуре, так и по

принципам работы.

Второй –который мы использовали в своей работе - метод, основанный на применении наивного байесовского классификатора, в основе которого лежит применение теоремы Байеса. Другими словами, НБА предполагает, что наличие какого-либо признака в классе не связано с наличием какого-либо другого признака. Например, фрукт может считаться яблоком, если он красный, круглый и его диаметр составляет порядка 8 сантиметров. Даже если эти признаки зависят друг от друга или от других признаков, в любом случае они вносят независимый вклад в вероятность того, что этот фрукт является яблоком. В связи с таким допущением алгоритм называется «наивным».

При обучении фильтра для каждого встреченного в письмах слова высчитывается и сохраняется его «вес» — оценка вероятности того, что письмо с этим словом — спам. В простейшем случае в качестве оценки используется частота: «появлений в спаме / появлений всего». В более сложных случаях возможна предварительная обработка текста: приведение слов в начальную форму, удаление служебных слов.

8)

Для наиболее точного определения типа сообщения нам потребовалось большая база сообщений для обучения байесовского классификатора.

Из –за работы с большими данными, нам пришлось использовать локальный сервер .

Речь идет о программе, которая имитирует работу реальных серверов хостинг-провайдеров.

Наш Сервер написан на языке Python с помощью фреймворка Flask. Изначально при запуске сервера внутри кода указывается путь каталога в котором находится 963 сообщения из которых 723 не спам, остальные являются спамом. Программа поочередно открывает каждый из этих файлов и записывает все слова из сообщений и сколько раз это слово встретилось из общего количества сообщений. Затем отбираются 4000 наиболее встречаемых слов, а остальные удаляются. Так же не учитываются одинаковые слова, но заканчивающиеся на разное окончание, к примеру “include”, “includes,” “included”. Стоит еще упоминуть что пока программа прогоняет 963 сообщения для своего обучения, она конечно же знает какое из них спам, а какое не спам. Далее создается матрица, где по строкам находятся сообщения а по столбцам те самые 4000 тысячи слов, наиболее встречающихся. В каждой строчке записан так называемый вектор правдоподобия, где на i-ом месте строки указано сколько раз встретилось то или иное слово. так как сообщений 963 то уже не вектор получается а целая матрица. Далее мы создаем модель, на основе которой будет подсчитываться вероятность. (Просто следует подключить библиотеку и написать строчку model1 = MultinomialNB() и обучить классификатор строчкой model1.fit(train\_matrix,train\_labels)). Все, на этом часть обучения закончена. Далее все очень просто на вход поступает сообщение. Его текст раскладывается на слова и уже на основе обучения и с помощью теоремы Байеса определяется является ли письмо спамом или нет. Сервер вернет 0 если не спам, и 1 если спам.

9)

В конце хочется добавить, что , к сожалению, наша программа не совершенна и по результатам тестирования нашего классификатора мы выявили, что 3 сообщения, которые не являются рекламными, по ошибке оказались в спаме, что составляет примерно 0,62%.  
А 7 сообщений, которые по-настоящему являются спамом не были распознаны программой как таковые, что составляет 7,8%.