Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра суперкмпьютеров и квантовой информатики

Мацук Егор Александрович

# Разработка сервиса для автоматического поиска изменений web-страниц

Отчет о научной работе

**Научный руководитель:**
к.ф-м.н., в.н.с.
С.А.Жуматий

Москва, 2018

# Оглавление

# Введение

## Цель работы:

Разработка и реализация средств для автоматического поиска текста на произвольной web-странице и контроля изменений этого текста.

## Проблема:

Произвольность используемых разметок для отображения текста на странице ведет к отсутствию единого стандарта, согласно которому можно было бы однозначно определить местоположение текста в HTML-коде.

## Постановка задачи:

Исследование существующих методов поиска текста на web-странице, разработка эвристических методов поиска текста, их тестирование; разработка средства для нахождения разницы между двумя символьными последовательностями, а также визуализации изменений; разработка интерфейса пользователя для конфигурирования программы; разработка системы оповещения пользователей об обновлениях web-страниц.

## План работы:

1) Исследование существующих методов
2) Выявление и формулировка признаков нахождения текста в данной части страницы (в данном теге)
3) Создание алгоритма, осуществляющего поиск текста на основании выявленных признаков
4) Создание алгоритма, осуществляющего поиск различий между двумя текстами
5) Тестирование
6) Разработка средств визуализации и интерфейса пользователя
7) Разработка системы оповещения

# 1. Исследование существующих методов

Идея создания сервиса для отслеживания новостей не нова, как и идея поиска текста на web-страницах. На самом деле, алгоритмы поиска текста встречаются часто – почти в каждом современном браузере присутствует «режим чтения», в котором браузер отображает лишь основной контент на странице. Алгоритмы, осуществляющие такое отображение, носят название readability-алгоритмов.

Исходный код или хотя бы принцип работы таких алгоритмов найти очень трудно, и даже по этим немногочисленным примерам можно сделать вывод, что эти алгоритмы сугубо эвристические и просто перебирают огромное количество вариантов расположения текста. Сделать точный метод невозможно в силу особенностей языка разметки HTML, а для любого эвристического алгоритма всегда можно подобрать такую разметку, в которой алгоритм неправильно выделит основной контент, что часто наблюдается в алгоритмах даже таких больших компаний, как Apple и Google. Но, например, браузер Safari для некоторых страниц просто не предоставляет режим чтения, ввиду невозможности точного определения местоположения текста.

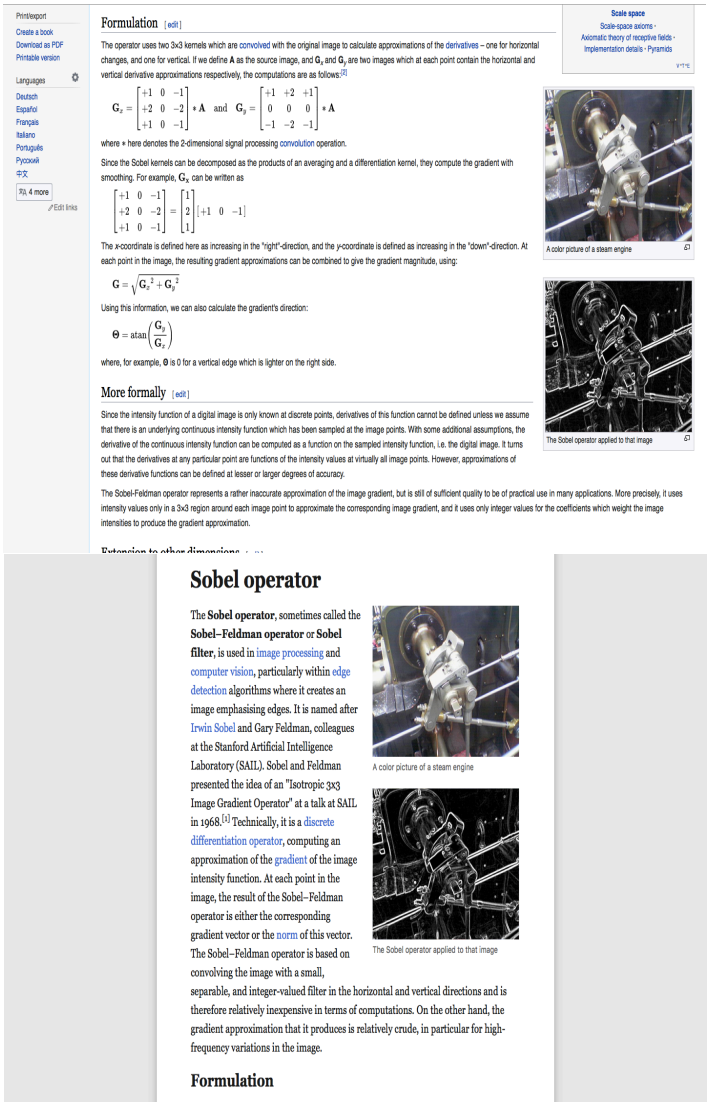


Рисунок 1. Страница на википедии и ее же вариант в режиме чтения в браузере Safari

В контексте программ для отслеживания новостей наблюдаются различные подходы. Например, сервис visualping[1] предлагает пользователю выделить интересующую часть на скриншоте необходимой web-страницы, после чего сервис будет регулярно перезагружать страницу и сравнивать области на двух скриншотах. Такой метод может

[1] https://visualping.io - сервис для отслеживания новостей
[2] http://feed43.com/ - сервис для отслеживания новостей на произвольных

быть полезным в некоторых случаях, однако встает ряд проблем, например, преобразование графической информации в текстовую, изменение размеров страницы по мере обновлений и др.

Сервис feed43.com[2] преобразует код страниц в RSS каналы, после чего их можно отслеживать с помощью новостного клиента. Серьезное ограничение сервиса заключается в необходимости вручную указывать шаблон поиска – тег, в котором предположительно находится текст. Сервис будет преобразовывать исключительно содержимое указанного тега. Для пользователя это выливается в необходимость вручную анализировать HTML-код интересующих страниц, чтобы найти нужный тег, а если требуется проверять множество страниц, то этот процесс может сильно затянуться.

Существует интересный российский сервис websvodka[3], отслеживающий новости и подсвечивающий изменения в коде страницы без необходимости что либо указывать непосредственно. Однако, сервис является платным и более детально изучить принцип его работы не представляется возможным.

В 2003 году в Microsoft Research Asia было опубликовано исследование, в ходе которого был разработан и протестирован алгоритм VIPS (Vision-based Page Segmentation algorithm) для разделения страницы на «информационные блоки» - независимые логические части web-страницы. Из блоков затем можно извлекать различные параметры и давать их на вход регрессионной модели для оценки условной важности блока. Такой метод, очевидно, склонен к переобучению модели опять же в силу особенностей HTML-разметки, однако, возможно, при обучении на большом количестве разных страниц (десятки-сотни тысяч) алгоритм может дать хороший результат. Очевидно, разметка такого большого количества страниц является очень трудозатратной.

[2] http://feed43.com/ - сервис для отслеживания новостей на произвольных страницах
[3] http://www.websvodka.ru/ - сервис для отслеживания новостей

# 2. Формулирование признаков присутствия текста

Требовалось разработать алгоритм, способный осуществлять поиск основного контента на произвольной web-странице, при этом минимизировав участие пользователя в процессе поиска. Для этого было решено определить особенности расположения контента на реальных страницах и, исходя из полученных требований, разработать алгоритм.

Собственно, почему нельзя сравнивать исходный код HTML-страниц целиком? Дело в том, что на любой современной странице присутствуют скрипты, постоянно изменяющие код страницы – одна и та же страница в разные моменты времени почти гарантировано имеет отличия в коде. Это могут быть динамически изменяемые аттрибуты тегов, различные метрики об использовании ресурсов сайта, информация, загружаемая с сервера в реальном времени и т.д.

Были выявлены следующие особенности расположения текстов в HTML-коде:

1) Текст – это последовательность буквенных символов и знаков препинания, как правило большой длины
2) Искомый текст, как правило, либо целиком содержится внутри одного тега, либо разбит на абзацы с помощью тегов `<p>`, `<br>` или `<hr>`.
3) Иногда текст подразделяется на логические блоки или фреймы (см. Рисунок 2). В этом случае части текста расположены в разных тегах, но всегда объединены тегом-предком на один или несколько уровней выше (возможно, тексты расположены на разной глубине для некоторых видов страниц).
4) Тексты статей иногда заключаются в тег `<article>`, при этом в анализируемой выборке не было найдено страниц, где этот тег использовался бы не по назначению.
5) Тексты, как правило, имеют заглавие, отображаемое при помощи тегов `<h1>`, `<h2>`, …, `<h6>`
6) На некоторых страницах, например, блогах, часто располгаются комментарии пользователей, аннотации к другим статьям и другой контент “второго плана”. Такой контент зачастую состоит из нескольких небольших буквенных последовательностей находящихся в одинаковых равноправных тегах с одинаковыми атрибутами, объединенных родительским тегом.
7) Современная страница, как правило, состоит из блока с основным контентом и второстепенных навигационных блоков. Такие навигационные блоки целиком

состоят из ссылок. Таким образом, можно отбрасывать из рассмотрения части страницы, текст в которых полностью расположен в тегах `<a>` с атрибутом `href`.

Эти особенности и легли в основу эвристик, используемых в разработанном алгоритме.

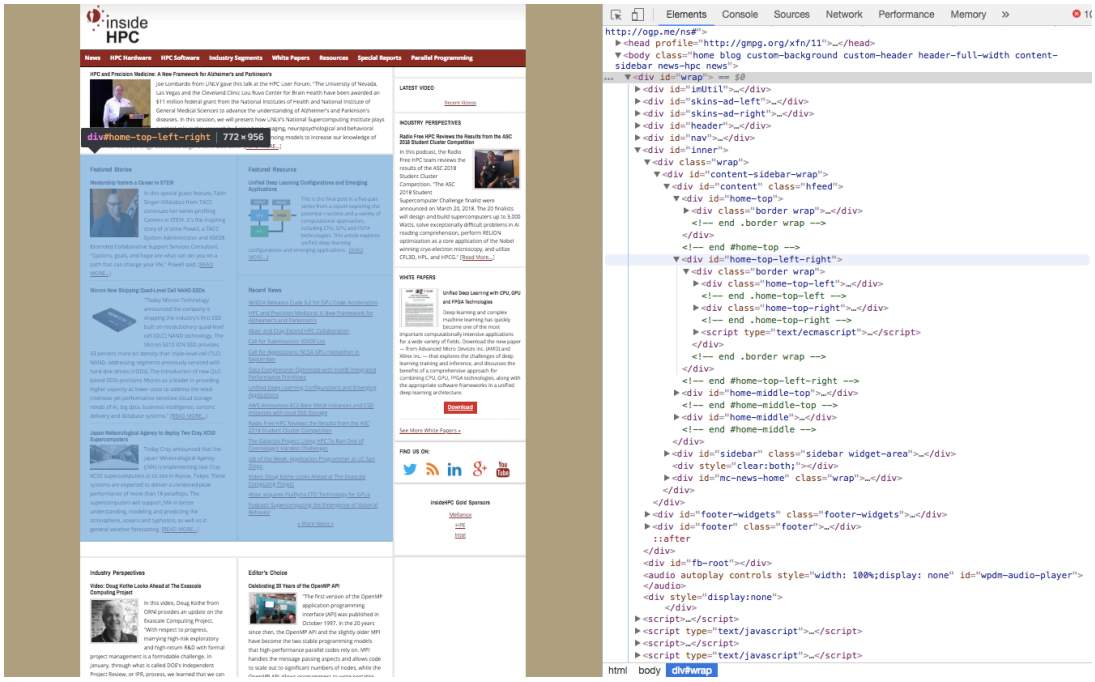


Рисунок 2. Пример страницы с древовидной структурой блоков с ветвями разной глубины: выделенный блок расположен на одном уровне с верхним блоком, но сам подразделяется на два блока, т.е. текст верхнего блока будет не на той же глубине, что и тексты в выделенном блоке

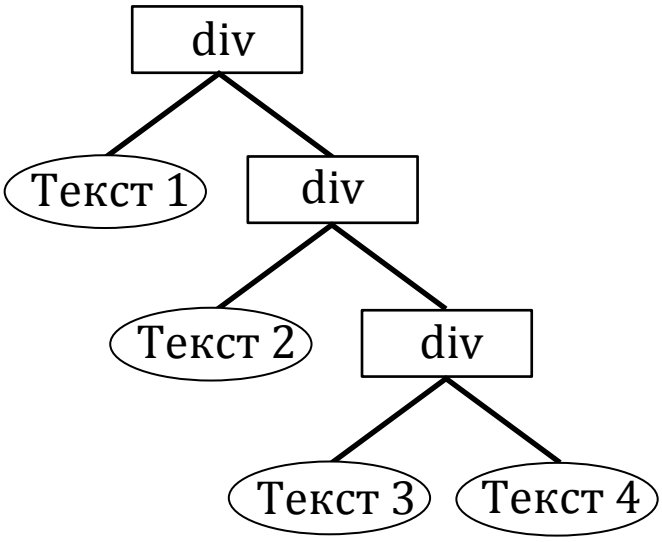


Рисунок 3. Схематичное изображение дерева к предыдущему рисунку

# 3. Практическая часть

## 3.1. Устройство основной программы

Для отслеживания новостей была разработана соответствующая программа. Алгоритм читает файлы конфигурации, содержащие информацию о пользователях и проверяемых сайтах, после чего последовательно проверяет каждый предложенный сайт на предмет обновлений и осуществляет почтовую рассылку результатов.

Разработанная программа обладает следующими возможностями:

- Поиск основного текста на странице и его подсветка
- Сохранение истории рассматриваемых сайтов
- Сравнение текущей версии страницы с прыдущей, хранящейся в истории; сохранение модифицированной страницы с подсвеченными изменениями
- Рассылка результатов по указанным e-mail адресам

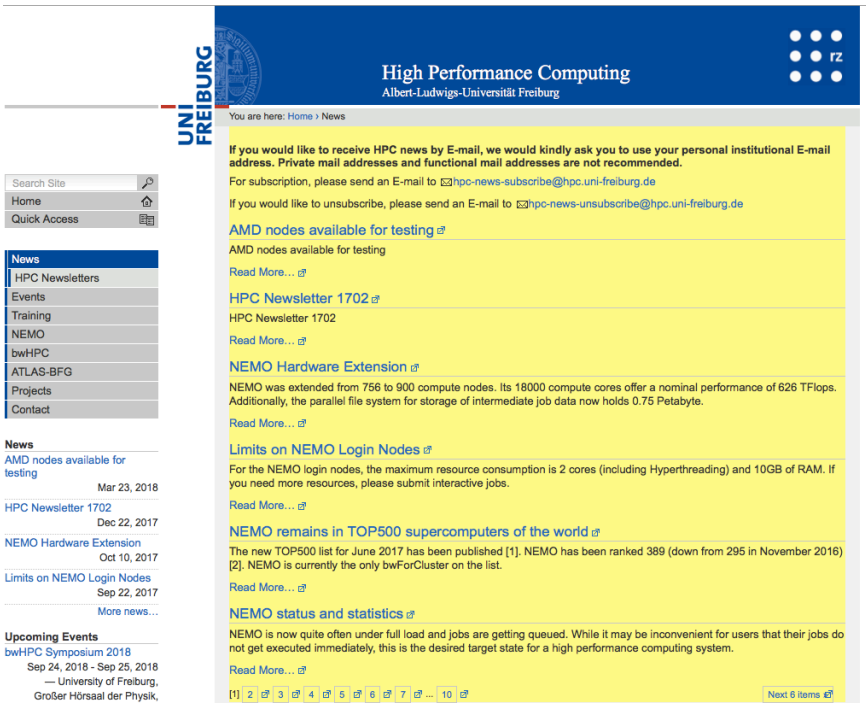


Рисунок 4.Пример автоматичсеки выделенного текста на странице (подсвечен желтым) http://www.hpc.uni-freiburg.de/news

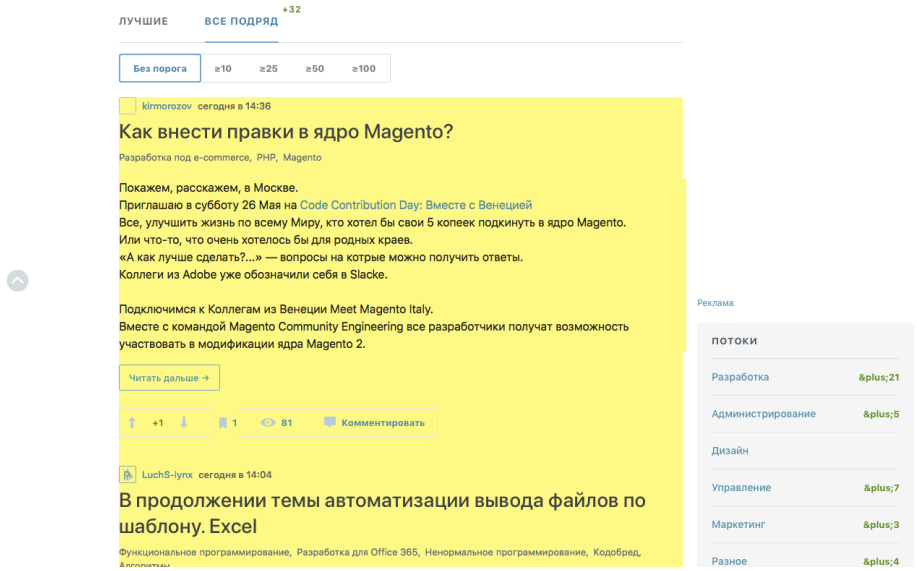


Рисунок 5. https://habr.com


Рисунок 6. https://insidehpc.com

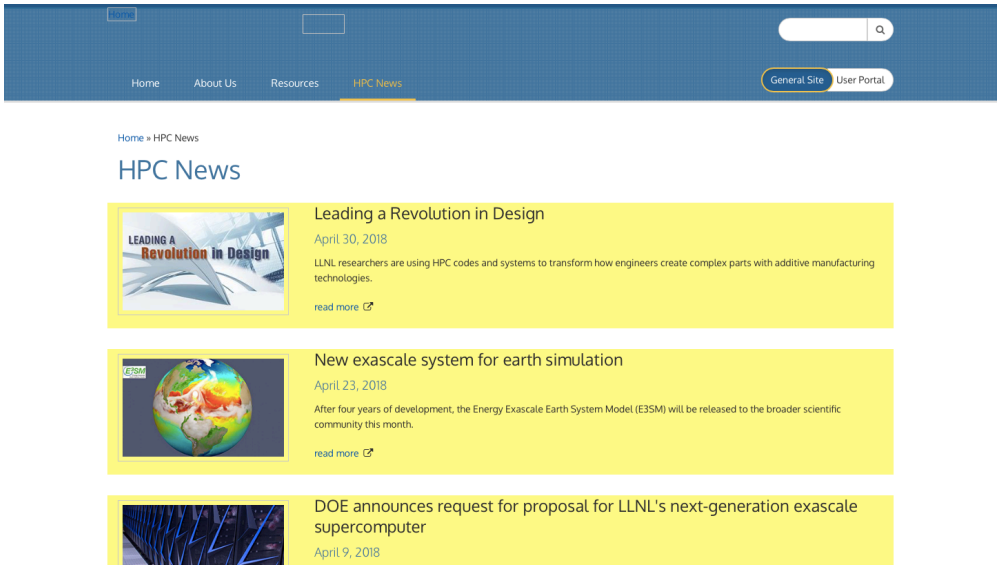


Рисунок 7. https://hpc.llnl.gov/news

Для поиска текста используются эвристики, позволяющие также предположить к какому классу относится найденный текст:

- Статья («article») – на странице содержится один крупный текст без меньших текстов на других планах;
- Статья с комментариями («article with comments») – на странице содержится один крупный текст, а также несколько равноправных, меньших по размеру на других планах;
- Новостные сводки («briefs») – на странице содержится несколько примерно одниаковых по размеру текстов, среди которых нельзя выделить один доминирующий;
- Отутствие текст («no text») – на странице нет символьной последовательности, удовлетворяющей порогу наличия текста
- Неопределенный класс («unsolved») – на странице присутствуют несколько неравноправных текстов.

Выше была также упомянута проблема возможного нахождения нескольких частей искомого текста на разных уровнях дерева тегов. Для этого при анализе дерева вводится небольшая возможная погрешность между глубинами найденных символьных последовательностей: если при проходе по дереву от текстов в сторону корня не все родительские теги являются одним и тем же тегом, то этим тегам «дается попытка» подняться на уровень выше и попасть в общий родительский тег. Если этого не происходит, то теги отбрасываются из рассмотрения. Однако, при отбрасывании слишком большого количества тегов, определяемого пороговым значением, тексту будет присвоен тип unsolved.

Помимо эвристического поиска текста, программа также позволяет жестко задать искомый тег, то есть его название и аттрибуты. В этом случае никакого поиска происходить не будет – программа будет сравнивать именно содержимое этого тега.

Программа поддерживает загрузку страниц как на латинице, так и на кириллице. Если в HTML-коде страницы явно не указана используемая кодировка, то программа пытается использовать одну из указанных. Во избежание ошибки 403 на некоторых сайтах, в HTTP запросе явно указывается заголовок `User-Agent: Mozilla 5.0`.

Итак, при первой загрузке страницы, ее код просто сохраняется в историю. При повторной загрузке с целью проверки обновлений, алгоритм также загружает последнюю

страницу из истории. Затем, из обоих страниц извлекаются тексты и разбиваются на лексемы, представляющие собой либо слова вне тегов, либо теги целиком (от “<” до “>”). Для этого программа моделирует простой конечный автомат. Далее, массивы слов подаются на вход функции поиска разницы между текстами. Из двух текстов и информации об их различиях при помощи другого автомата генерируется модифицированный текст, в который вставлены теги форматирования для подсветки различий.

Рисунок 8. Пример работы алгоритма поиска обновлений (одна из аннотаций была перенесена в другой блок) https://insidehpc.com

Далее, после нахождения обновлений (при их наличии) на всех страницах из списка, осуществляется email-рассылка краткой информации пользователям, содержащей количество добавленных и удаленных элементов на интересующих страницах, а также ссылки на сами страницы. В случае отсутствия изменений на каких-то (или всех) страницах, письмо все равно отправляется, но напротив имен страниц явно указывается, что изменения отсутстсвуют.

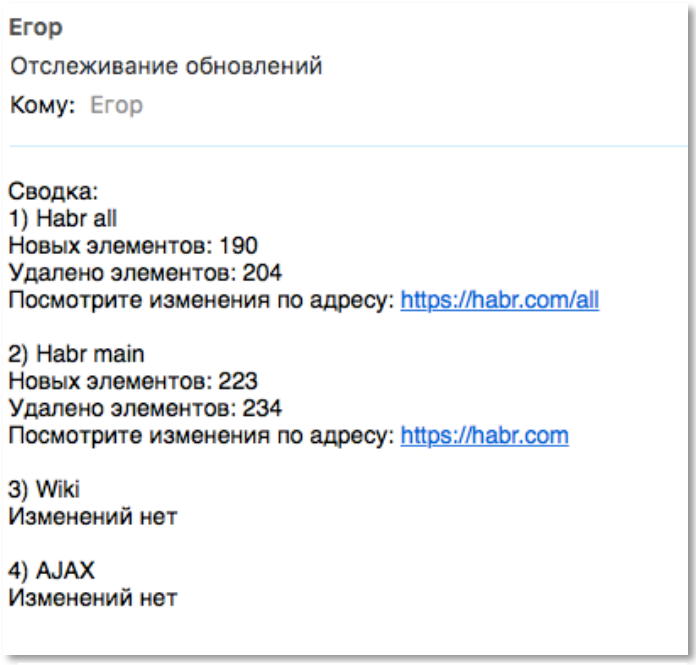


Рисунок 9. Пример письма из рассылки

## 3.2. Пользовательский интерфейс

Для конфигурирования программы требовалось разработать пользовательский интерфейс, так как модификация файла с настройками оказалась неудобной из-за необходимости ручного ввода большого количества URL'ов, e-mail-адресов и другой информации. Поэтому было решено создать веб-интерфейс, открываемый через браузер и позволяющий сконфигурировать программу.

Интерфейс является отдельным приложением, которое может работать независимо от основной программы.

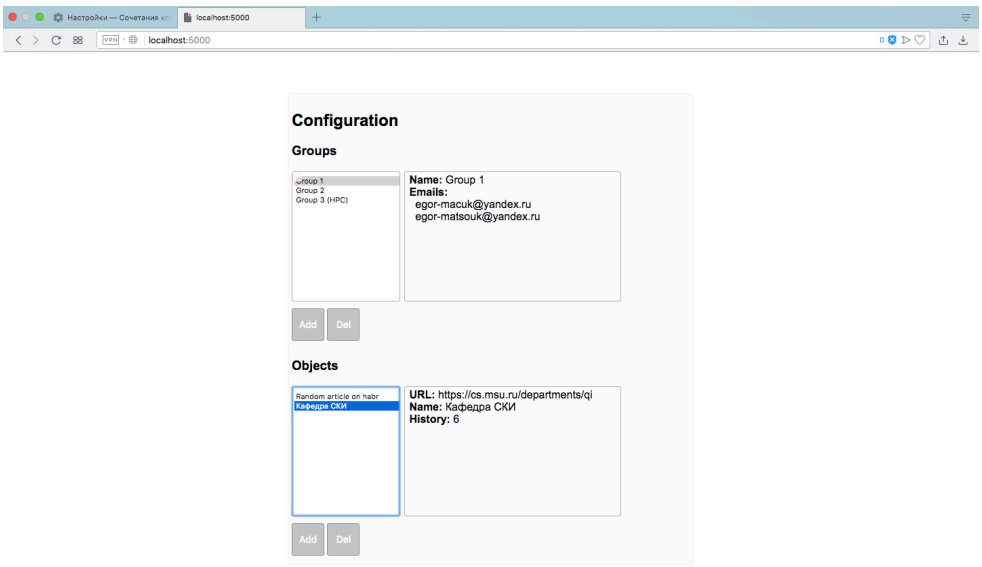


Рисунок 10. Вид web-интерфейса

За время работы над программой было решено создать следующую пользовательскую логику: все пользователи принадлежат некоторым группам, и в каждой группе есть свой список веб-страниц, информация о которых должна рассылаться пользователям группы. Поэтому, интерфейс разрабывался согласно такой логике.

Итак, интерфейс позволяет:

- Создавать группы, обладающие именем и списком email-адресов
- Для каждой группы добавлять структуры-страницы, состоящие из URL, имени и длины истории, т.е. количсетва страниц, которые должны храниться на диске
- Удалять группы
- Удалять страницы

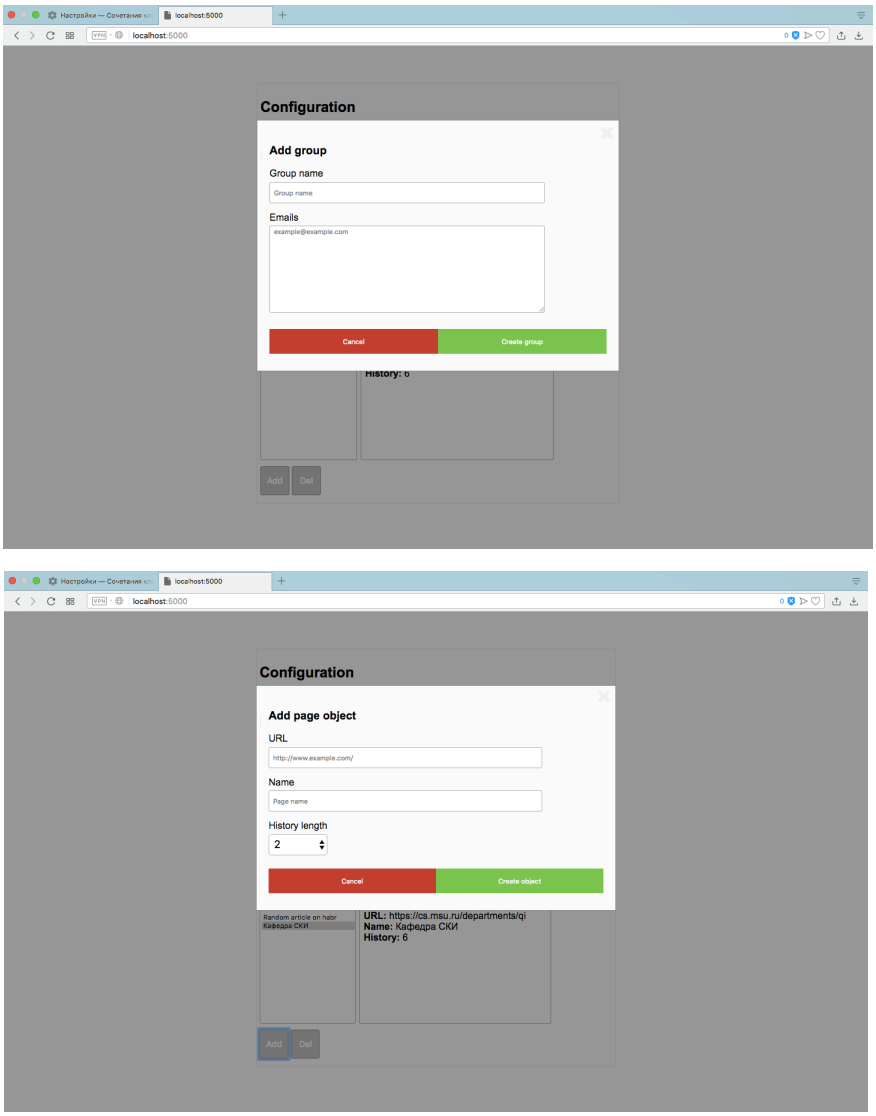


Рисунок 11. Вид окон для создания группы и страницы соответственно

Каждое нажатие, будь то выбор группы или создание страницы, отправляет на сервер запрос, после чего сервер отвечает и JavaScript-код на стороне клиента динамически изменяет страницу. Таким образом реализуется AJAX-модель интерфейса.

При создании группы или страницы, на сервере генерируется маленький файл с введеными данными. Эти файлы затем могут быть прочитаны сервером при взаимодействии с интерфейсом и отправлены клиенту. При нажатии кнопки «Del», файл выбранной группы или страницы удаляется.

Сервер может быть запущен как на одной машине, так и внутри локальной сети. В этом случае создавать и удалять объекты сможет любой учатсник сети.После конфигурирования, сервер может продолжать работу или может быть закрыт – на работу программы это никак не повлияет.

Основная программа же при запуске загружает содержимое созданных файлов и использует их для инициализации внутренних объектов.

## 3.3 Существующие проблемы

Как было сказано ранее, для любой эвристики всегда можно подобрать такую страницу, на которой текст будет выделяться неверно. Тем не менее, разработанный алгоритм корректно выделяет область с основным контентом для всех протестированных страниц.

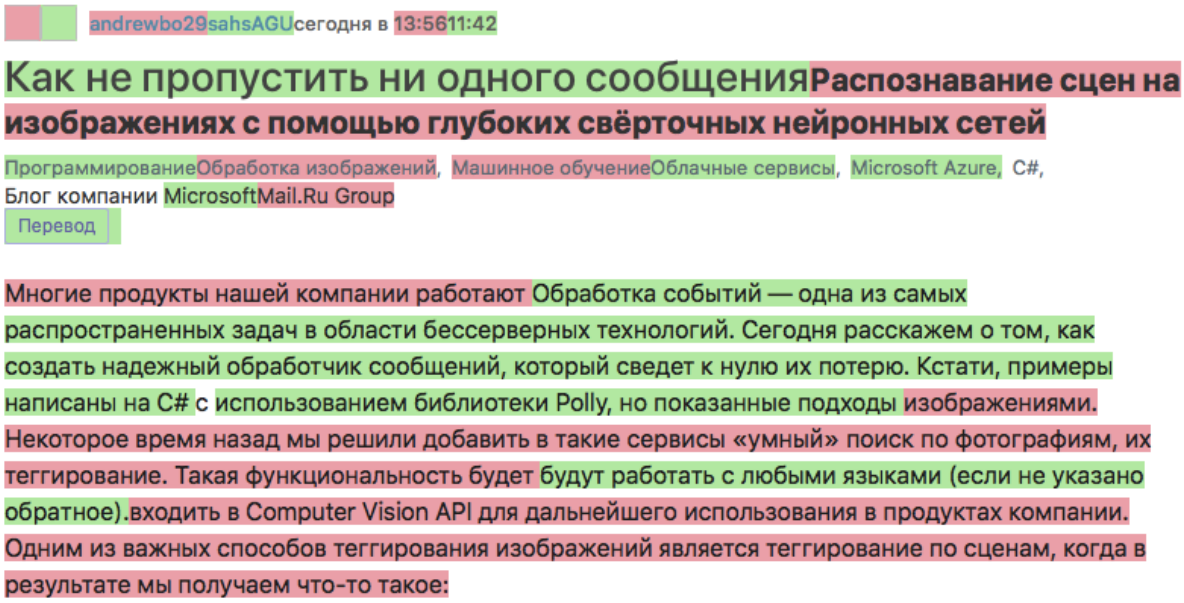


Рисунок 12. Здесь видно, что из-за нескольких общих слов алгоритм расставляет изменения не в удобочитаемом порядке, https://habr.com

Другой фундаментальной проблемой является поиск разницы между текстами. Дело в том, что для сильно отличающихся текстов трудно определить разницу между ситуацией, когда текст был просто сильно изменен по сравнению с предыдущим, и ситуацией, когда текст был заменен поностью другим, но перескающиейся слова (например союзы и предлоги) остались (см. Рисунок 12). Для более точного определения этих двух ситуаций следует проводить более сложный анализ выделенных текстов, например, учитывать части речи слов, что уже является совсем другой, также нетривиальной задачей. Разработанная же программа ищет разницу подобно тому, как это делает утилита diff в Unix.

# Заключение

В ходе работы были проанализированы существующие методы умного отслеживания обновлений на веб-страницах. Была определена архитектура разрабатываемой программы.

Далее вручную был проанализирован исходный код нескольких десятков страниц, в результате были получены некоторые признаки, по которым можно найти текст на странице. По этим признакам был составлен соответствующий эвристический алгоритм.

По итогу работы имеется готовая к использованию программа, позволяющая задать список интересующих веб-страниц и регулярно получать уведомления об их изменениях. Наличие предобработки кода в лице эвристического поиска блока с оснвным текстом позволяет отслеживать изменения именно смысловой части страницы, не затрагивая посторонний мусор, например, рекламные блоки и блоки навигации.

# Список используемой литературы


1. Райан Митчелл, Скрапинг веб-сайтов с помощью Python, М.: ДМК Пресс, 2016, 279 с.
2. Страница с исследованием касательно алгоритма VIPS, https://www.microsoft.com/en-us/research/publication/vips-a-vision-based-page-segmentation-algorithm/?from=http%3A%2F%2Fresearch.microsoft.com%2Fapps%2Fpubs%2Fdefault.aspx%3Fid%3D70027
3. Сервис для отслеживания обновлений на сайтах, https://visualping.io
4. Сервис для отслеживания обновлений на сайтах, http://feed43.com/
5. Сервис для отслеживания обновлений на сайтах, http://www.websvodka.ru/