Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет информационных технологий и программирования

Кафедра «Компьютерные технологии»

О.С. Ларионов

Определение сайтов-зеркал на основе анализа графа ссылок и дублирования контента

Бакалаврская работа

Научный руководитель: А.А. Романенко

Санкт-Петербург

2013

Оглавление

[Введение 3](#_Toc356082502)

[Глава 1. Основные понятия и обзор существующих методов 4](#_Toc356082503)

[1.1. Основные определения 4](#_Toc356082504)

[1.2. Существующие алгоритмы 4](#_Toc356082505)

[Глава 2. Разработка системы 6](#_Toc356082506)

[2.1. Общий концепт 6](#_Toc356082507)

[2.2. Подбор пар 6](#_Toc356082508)

[2.3. Веса признаков 6](#_Toc356082509)

[2.4. Проверка пар 7](#_Toc356082510)

[Глава 3. Реализация и тестирование 8](#_Toc356082511)

[3.1. Алгоритмы подбора пар 8](#_Toc356082512)

[3.2. Проверка совпадения страниц 9](#_Toc356082513)

[3.3. Диаграмма классов системы 10](#_Toc356082514)

[Список используемых источников 11](#_Toc356082515)

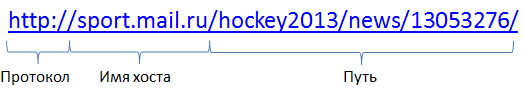
# Введение

В последнее время наблюдается значительный рост интернета. Уже сейчас он хранит в себе огромную коллекцию информации. Например, об этом свидетельствует объем «Машины времени интернета» [1]: более 10 петабайт. Количество сайтов в интернете, по данным аналитических компаний, превысило 620.5 млн. Однако, по данным исследователей, значительная их часть (30-45%) [2] являются дубликатами. Казалось бы, зачем создавать сайты с одним и тем же содержимым? Но у дубликатов есть как положительные, так и отрицательные стороны. С одной стороны, это поможет в случае, если доступ к одному хосту будет невозможен, например, по причине поломки оборудования. С другой стороны, дубликаты представляют проблему для поисковых систем: если интернет на треть заполнен идентичными сайтами, то индекс систем как минимум на треть содержит избыточную информацию.

Таким образом, выявление таких сайтов является актуальной задачей. В настоящей работе разработана и реализована система, позволяющая обнаружить сайты, которые скорее всего являются зеркалами, и проверить это.

# Глава 1. Основные понятия и обзор существующих методов

## Основные определения



**Рис. 1. Схема URL адреса**

Хотя понятие «Сайты-зеркала» доступно на интуитивном уровне, его весьма трудно формализовать. Зеркала бывают как полностью идентичные, отличающиеся только названием хоста, на котором расположен сайт, так и структурно-идентичные, то есть такие, у которых похожи схемы пути к страницам (возможно, с небольшими изменениями), а содержимое страниц по одинаковым путям сильно совпадает. «Сильно совпадает» значит, что контент отличается, например, только в верстке или дизайне. Если бы интернет не был такого размера, то обнаружить идентичные сайты было бы достаточно просто. В таком случае можно было бы для каждого сайта создать документ, в котором сохранить все пути и страницы сайта. После этого несложно среди этих документов найти совпадающие. Однако, интернет слишком велик.

При обнаружении дубликатов мы встречаемся со следующими проблемами:

* Чаще всего доступен только список URL-адресов страниц, полученный веб-пауком, либо созданный прокси-сервером.
* Список может быть неактуален. Часть страниц из него может быть удалена, недоступна, 301 Moved Permanently, и так далее.
* Невозможно скачать и сохранить все страницы в памяти.

Это сильно усложняет задачу.

## 1.2. Существующие алгоритмы

Все существующие сейчас реализации для поиска зеркал являются закрытыми и/или запатентованными. Значительная часть описанных в работах алгоритмов являются вероятностными и действуют по такому принципу:

* Получают на вход список страниц (URL адресов) с различных сайтов.
* Прореживают этот список по какому-либо принципу. Например, оставляют не более пятидесяти сайтов, только сайты с количеством страниц, не меньшим ста и так далее.
* Составляют пары сайтов — кандидатов на зеркала
* Запрашивают об этих сайтах различную информацию:
  + IP-адрес хоста
  + Миниатюру страницы
  + Список ссылок со страницы
  + …
* Производят анализ пар на основе существующей информации

По используемой информации алгоритмы можно разделить на несколько категорий:

* Алгоритмы, основанные на IP адресе. Такие алгоритмы группируют входной список по IP-адресам и считают, что зеркалами являются те сайты, у которых IP-адрес одинаков.
* Алгоритмы, основанные на URL адресе. Такие алгоритмы для оценки сайтов используют различные части адреса (название хоста, схему пути до файла).
* Алгоритмы, основанные на переходах. Этот класс алгоритмов оценивает сайты, основываясь на переходах между страницами сайта, либо на ссылках на внешние ресурсы.
* Алгоритмы, основанные на деревьях сайтов. Основываясь на URL адресах страниц, эти алгоритмы строят деревья сайтов и сравнивают их.

Все эти алгоритмы несовершенны. Например, сейчас хостинг может выкладывать на одном и том же IP адресе много сайтов, благодаря технологии NAT. Алгоритмы, основанные на переходах, могут обмануть новостные сайты, так как часто они ссылаются на один источник или друг на друга (например, для повышения собственного веса в системе *PageRank*). Таким образом, процент верных ответов каждого из этих классов по отдельности невелик. Однако, эти алгоритмы достаточно просты в реализации и удобны в использовании. Поэтому на поверхности лежит идея группировки результатов этих алгоритмов. В работе [3] был проведен анализ алгоритмов из первых трех классов. В результате получилось, что лучшие результаты работы получаются тогда, когда используются алгоритмы из первого и третьего классов в совокупности. В этой работе произведена попытка улучшить результаты путем использования нескольких алгоритмов из разных классов в качестве критериев точности, а так же получение степени влияния различных алгоритмов на правильность итоговой оценки.

# Глава 2. Разработка системы

## 2.1. Постановка задачи

Таким образом, задачу поиска зеркал можно разбить на две: выделить из входного списка URL адресов пары сайтов, которые с большой вероятностью являются дубликатами, и проверить для каждой пары, действительно ли это сайты-зеркала.

## 2.2. Подбор пар

Основным объектом системы подбора пар является классификатор. Он, по входному списку страниц и на основе известных ему алгоритмов сравнения сайтов, а так же на основе их влияния, умеет генерировать список пар сайтов, которые, вероятно, являются зеркалами. В этом ему помогает набор компараторов. Компаратор это сущность, которая умеет создавать из большого списка адресов список пар по определенному признаку:

* По IP адресу сервера, на котором находится сайт
* По URL адресам страниц
* По связности сайтов
* …

Для каждой созданной пары компаратор сообщает вероятность, что эти сайты являются зеркалами. Также у каждого компаратора есть своя степень влияния на итоговый список. Она берется из предположения о том, что у каждого компаратора своя ценность. Например, информация о том, что два сайта находятся на одном и том же сервере может быть менее важна, чем информация о сходстве путей.

Основываясь на всех списках от компараторов, а также весах компараторов, классификатор создает итоговый список взвешенных пар сайтов. Для этого он производит следующую операцию:

1. Выбирает произвольную пару сайтов из списков.
2. Находит ее во всех списках.
3. Вычисляет суммарный рейтинг пары на основе вероятностей в каждом из списков и весах признаков.
4. Составляет итоговый список взвешенных пар.

## 2.3. Веса признаков

Для хорошей работы алгоритма необходимо подобрать подходящие веса всем признакам. Хорошие компараторы, которые практически не будут ошибаться в составлении списка, можно построить на основе известной информации о сайтах. Их влияние на итоговый список (вектор весов) также можно подобрать, составить на основе статистической информации или использовать алгоритмы машинного обучения, например, метод опорных векторов.

## 2.4. Проверка пар

После того, как классификатор обработал данные компараторов и составил итоговый список пар для проверки, этот список передается системе проверки. Ее задача подтвердить, либо опровергнуть, что пара сайтов это действительно зеркала. Для каждой пары сайтов система выберет несколько страниц с этих сайтов и проверит их на сходство. Для такого выбора система может использовать любую стратегию, например, тривиальную: выбирать произвольную страницу на одном из сайтов, брать ее и добавлять к имени хоста другого сайта. Проверив таком образом несколько страниц, система анализирует результаты проверки и говорит для этой пары итоговый результат.

При проверке каждой новой пары, система может пользоваться уже известными результатами. Так, например, можно заметить, что отношение «является зеркалом» является транзитивным. То есть: если сайт является зеркалом сайта , а сайт является зеркалом , то сайт — зеркало . Таким образом, можно накапливать уже известные данные, и использовать свойство транзитивности, как один из критериев финальной оценки.

# Глава 3. Реализация и тестирование

## 3.1. Алгоритмы подбора пар

В системе реализованы следующие алгоритмы подбора пар:

1. Алгоритм IP-адресов. Этот алгоритм работает следующим образом. От всех входных ссылок оставляется только название хоста, остальной путь убирается. После этого для каждого хоста с помощью DNS определяется соответствующий ему IP-адрес. После этого для каждого из полученных IP-адресов составляется список соответствующих ему хостов. После этого алгоритм составляет пары кандидатов из адресов, находящихся на одном IP-адресе. Чтобы уменьшить вероятность ложных срабатываний, считается, что если количество сайтов на одном IP-адресе больше порогового значения, то этот IP-адрес игнорируется, и все хосты отбрасываются.
2. Алгоритм URL-адресов. Для определения кандидатов на основе URL-адресов используется модификация алгоритма шинглов [4]. Анализ происходит следующим образом. Каждый URL-адрес из списка разделяется на части по символам “/” и “.”. Например, адрес

[www.google.com/calendar/events/Saint-Petersburg-2012/](http://www.google.com/calendar/events/Saint-Petersburg-2012/)

разбивается следующим образом:

www google com calendar events Saint-Petersburg-2012

После этого все числа заменяются на символ “\*” для удобства анализа, так как остается только значащая часть.

www google com calendar events Saint-Petersburg-\*

Далее несколько подряд идущих частей группируется в терм, и к нему добавляется номер первой части. Например, если группировать в терм по 2 части, то из указанных выше частей получатся следующие термы:

* www\_google\_1
* google\_com\_2
* com\_calendar\_3
* calendar\_events\_4
* events\_Saint-Petersburg-\*\_5

Позиции помогут обнаружить адреса, сходные по структуре, а не только по частям адреса. Алгоритм преобразует все URL-адреса в списке, и для каждого терма сохраняет всех хосты, у которых встречался этот терм. Теперь, для того, чтобы построить пары, алгоритм анализирует все полученные термы и высчитывает оценку похожести хостов . Для каждого терма, который встречается на нескольких сайтах, вычисляется его вес в оценке по следующей формуле:

,

где — количество хостов, на которых встретился этот терм. Этот вес добавляется в оценку. После обработки всех термов общая оценка нормализуется по формуле:

где и — количества страниц на host1 и host2 соответственно. Нормализация делается, так как у сайтов с большим числом адресов получается много термов, и, соответственно, оценка сходства получается очень большой. Знаменатель дроби помогает сделать ее пропорциональной размеру сайта.

Далее, упорядочив по этой оценке, алгоритм выдает список пар, у которых эта оценка больше пороговой, чтобы убрать случайные помехи.

## 3.2. Проверка совпадения страниц

После того, как общий список кандидатов получен, необходимо протестировать каждую пару. Для этого будем выбирать адрес страницы одного из сайтов и пытаться создать адрес аналогичной страницы другого сайта. В простейшей стратегии можно выбирать произвольную страницу первого сайта, брать ее путь и конкатенацией соединять с хостом второго сайта.

Получив таким образом несколько пар страниц, необходимо проверить, существуют ли вообще эти страницы, и насколько сходно их содержимое. Как уже было сказано, для проверки двух страниц на совпадение можно использовать несколько стратегий.

1. Можно сравнивать на точное совпадение. Для этого, например, можно вычислять хэш-функцию. Однако, эта проверка слишком строга, так как в реальных ситуациях сайты часто могут отличаться только версткой или картинками, оставаясь сходными по содержанию.
2. Другая идея состоит в том, чтобы брать только текст страницы и оценивать их сходство. В данной работе для такой проверки страниц используется алгоритм шинглов. Он работает следующим образом. Несколько подряд идущих слов в тексте объединяются в кортежи, называемые чешуйками (shingles). При этом выборка происходит внахлест. Например, для текста «а роза упала на лапу азора» можно получить три 4-шингла (4 — это количество слов в кортеже):

* (а, роза, упала, на)
* (роза, упала, на, лапу)
* (упала, на, лапу, азора)

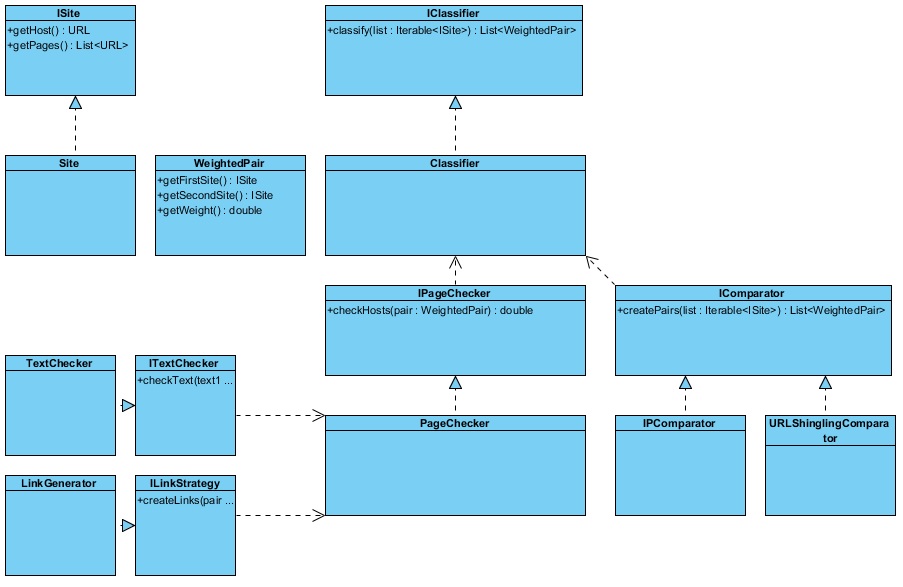
Для каждой страницы page1 и page2 строятся множества шинглов и , которые встретились в тексте. На основе этих множеств вычисляется оценка сходства:

Нетрудно заметить, что эта оценка лежит в интервале , при этом 1 соответствует идентичности страниц, а 0 полному различию.

Проверив таким образом для пары сайтов предлагаемые пары страниц на сходство, можно сделать итоговое предположение о том, действительно ли сайты являются зеркалами. Если более 50 процентов страниц оказывались похожими более чем на 0.5, то сайты признаются зеркалами.

## 3.3. Диаграмма классов системы

На этой диаграмме можно увидеть взаимодействие компонентов системы. Для проверки список с URL-адресами передается в Classifier, который делегирует составление списков классам, реализующим интерфейс IComparator. После этого он соединяет полученные списки пар в один и передает каждую его пару на обработку классу PageChecker. Получив итоговые результаты, Classifier выдает итоговый список пар, которые с наибольшей вероятностью являются зеркалами.



**Рис. 3. Диаграмма классов системы**

Как можно видеть из диаграммы классов, в большинстве случаев в программе используются интерфейсы, а не конкретные реализации классов. Благодаря этому систему достаточно просто расширять, добавлять новые компараторы, изобретать свои схемы подбора ссылок и проверки текста, даже переделывать принцип построения итогового списка. Также в такой системе несложно покрывать unit-тестами каждый компонент программы в отдельности и использовать фреймворки для внедрения зависимостей (dependency injection) для облегчения совместного взаимодействия компонентов. При реализации большинство компонентов программы было покрыто unit-тестами при помощи *JUnit* [6], а для внедрения зависимостей использовался *Spring Framework* [7].

# Список используемых источников

1. *Wayback Machine: Internet Archive*.[http://web.archive.org/‎](http://web.archive.org)
2. *N. Shivakumar and H. Garcia-Molina.* Finding near replicas of documents on the web. **Proceedings of Workshop on Web Databases (WebDB’98)**, March 1998.
3. *K. Bharat, A. Broder, J. Dean, M. Henzinger.* A Comparison of Techniques to Find Mirrored Hosts on the WWW. **Fourth ACM Conference on Digital Libraries**, 1999.
4. *A. Broder, S. Glassman, M. Manasse, G. Zweig.* Syntactic Clustering of the Web. <http://www.std.org/~msm/common/clustering.html>
5. *K. Bharat, A.Broder.* Mirror, mirror on the Web: a study of host pairs with replicated content.
6. *JUnit —* A programmer-oriented testing framework for Java. <http://junit.org/>
7. *Spring Framework* — open source application framework and inversion of control container. <http://www.springsource.org/>