



ТЕХНОСФЕРА

Фокусировка web-спайдера. Сад камней.

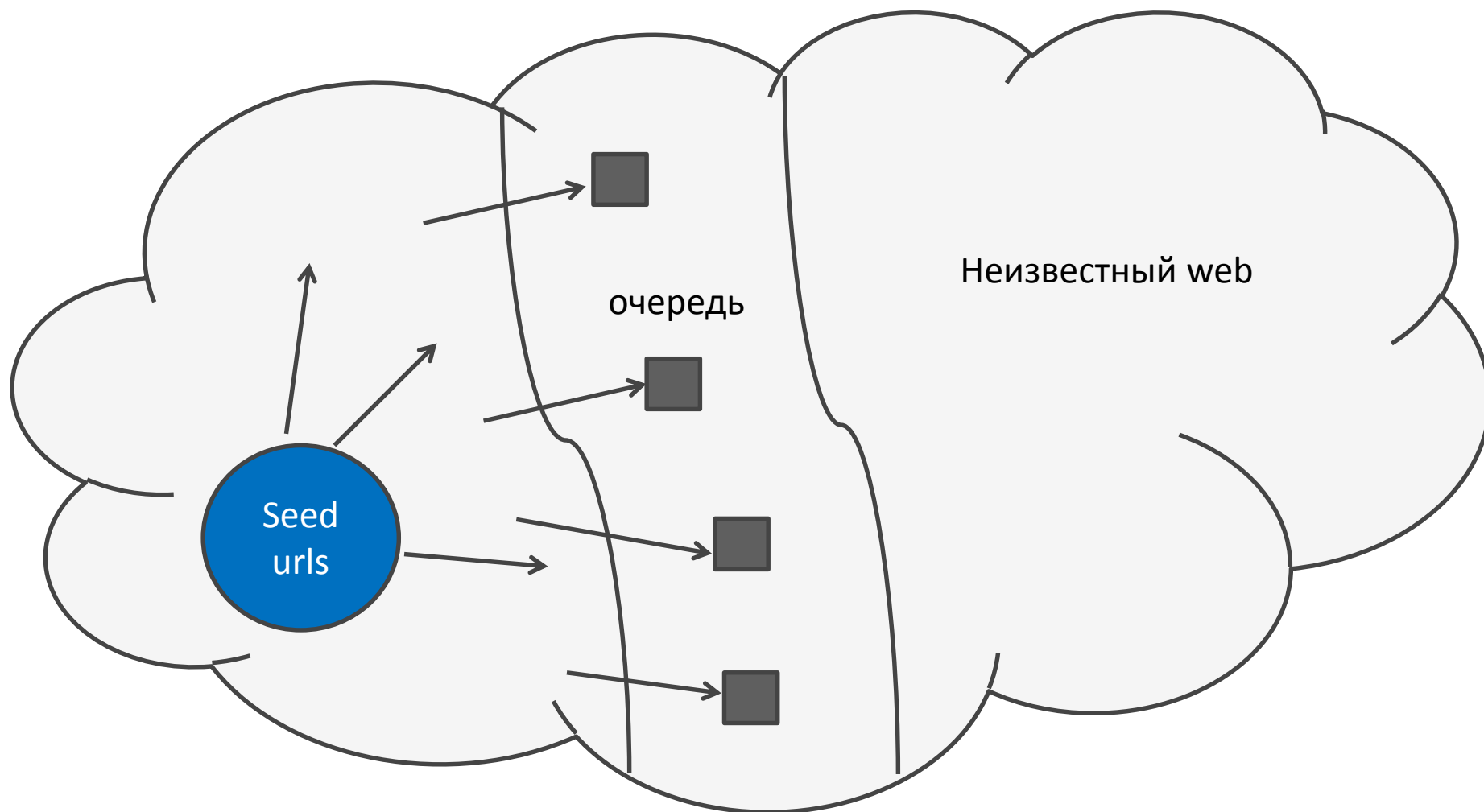
Сергукова Юлия,
программист отдела инфраструктуры проекта
Поиск@Mail.Ru

План лекции:

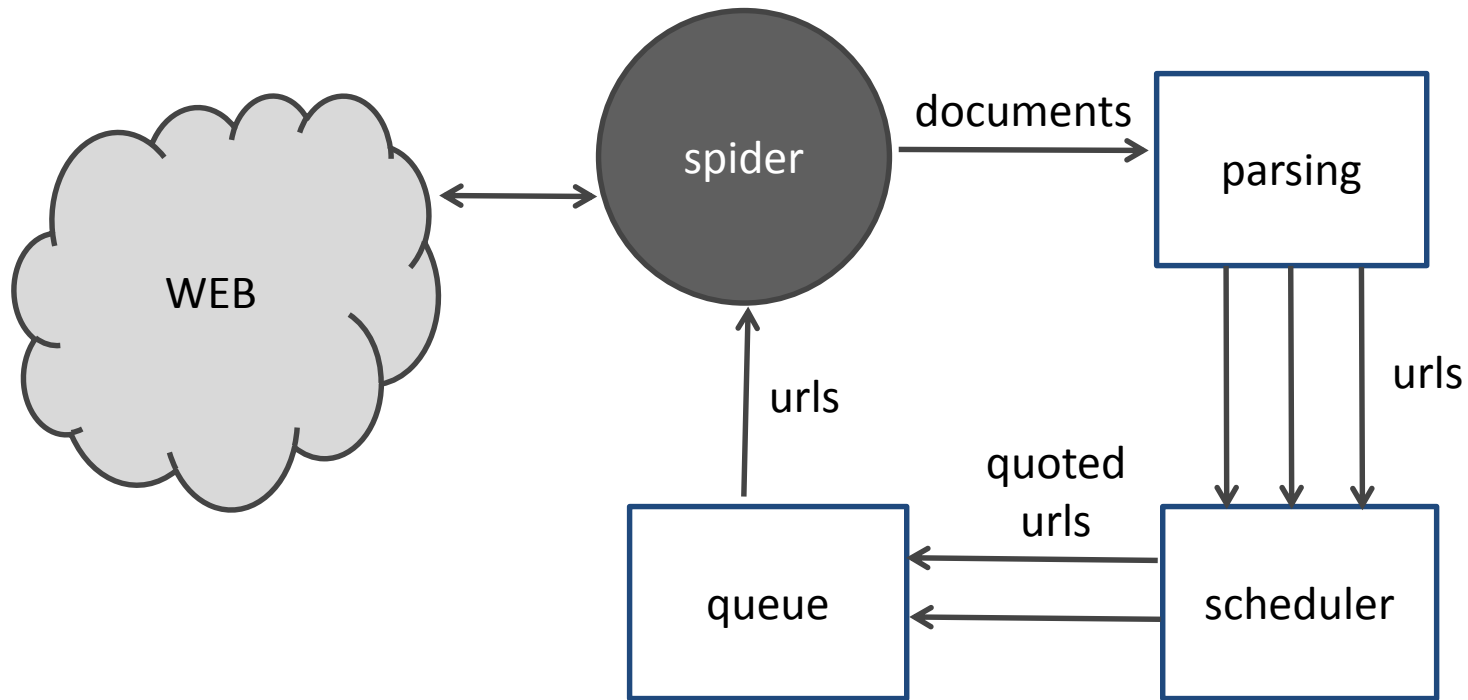
1. Постановка проблемы
2. Алгоритмы обхода сайтов
3. Алгоритмы фокусировки
4. «Сад камней»
5. Квотирование
6. ДЗ (!!!)

Вспомним прошлое занятие

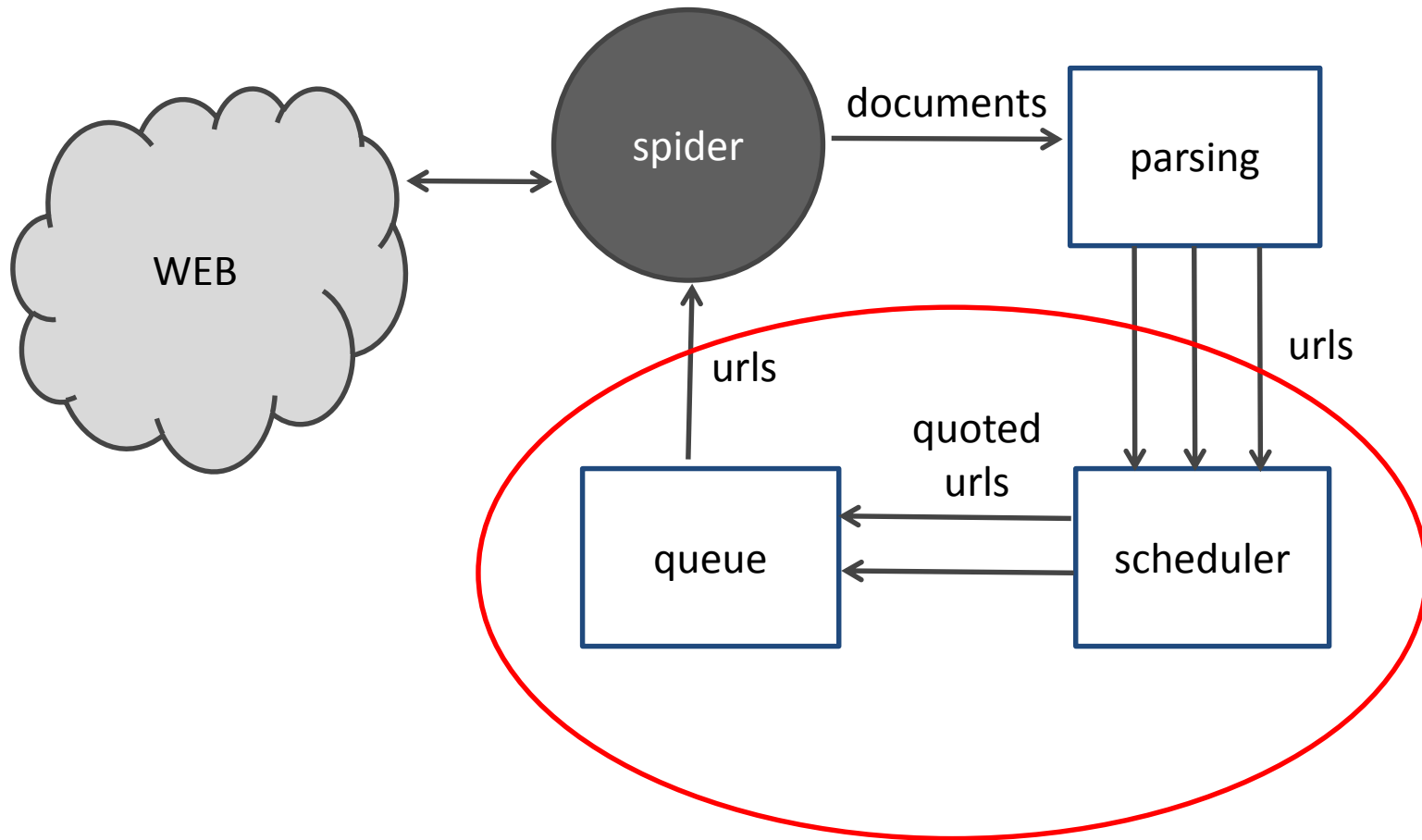
Выкачка



Spider & utils



Spider & utils



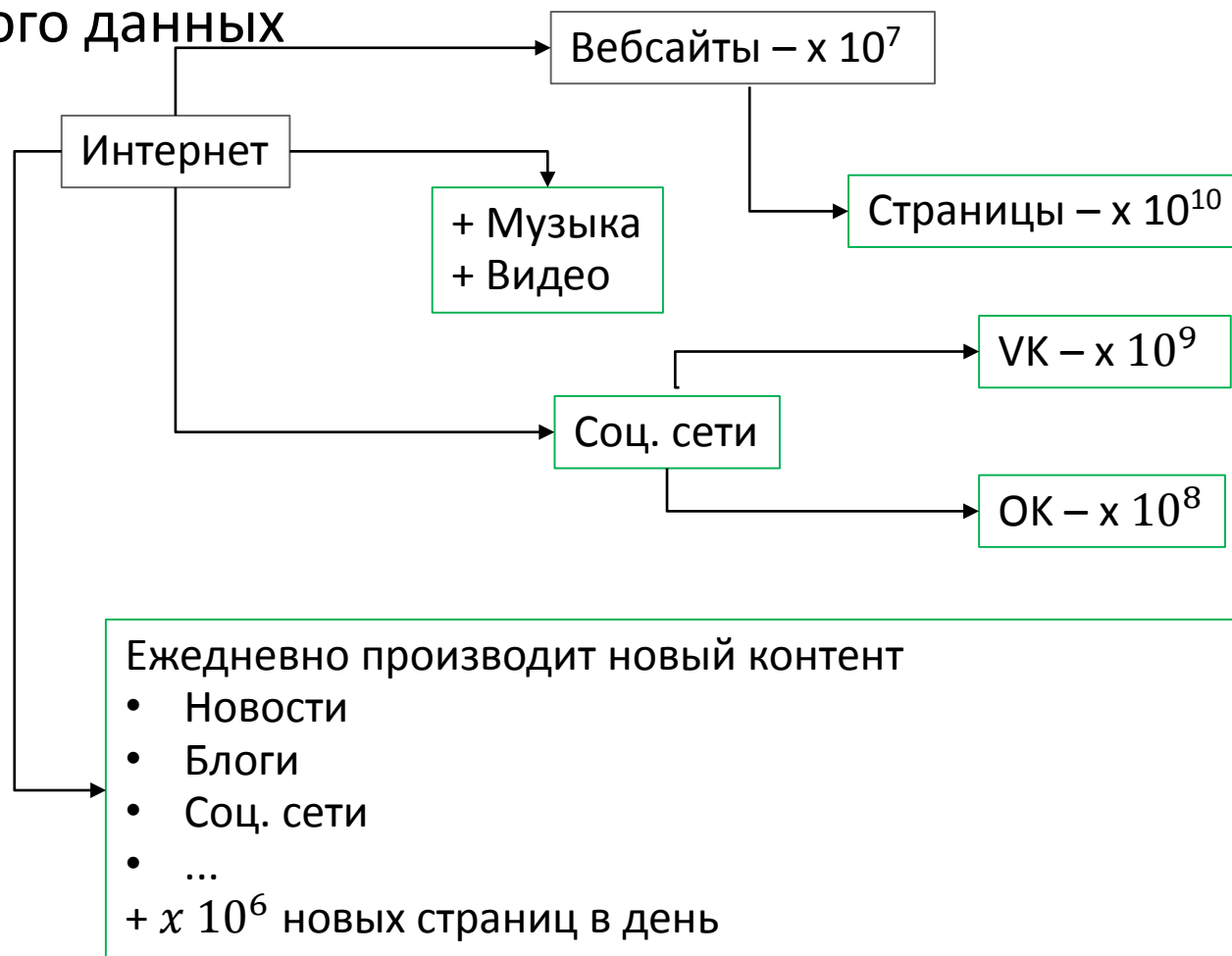
Почему мы не можем скачать всё?

Почему мы не можем скачать всё?

Слишком много данных

Почему мы не можем скачать всё?

Слишком много данных



Почему мы не можем скачать всё?

Слишком много данных:

30×10^6 - запросов в день || ~ 500 rps

Ограничения поисковой машины == ограничения по производительности:

- Время ответа на запрос 1 сервера < 60 мл. сек.
- Макс. документов на сервер $\sim 10 \times 10^6$
- Для индекса : $\sim 5 \times 10^9$ страниц потребуется ~ 400 машин (x 2 – для резервирования)

Почему мы не можем скачать всё?

Слишком много данных

При этом:

Качество поиска напрямую зависит от количества или качества страниц в индексе

Почему мы не можем скачать всё?

Слишком много данных

Мы не можем бездумно наращивать мощности:
Для размещения 10×10^9 страниц потребуется уже ~ 1000
серверов в одной реплике

Репликация сёрверов $\times 2$

Репликация данных в хранилище $\sim \times 3$

Почему мы не можем скачать всё?

Слишком много данных

Мы не можем бездумно наращивать мощности:
Для размещения 10×10^9 страниц потребуется уже ~ 1000
серверов в одной реплике

И прирост в качестве - порядка 5% ☹

Дадим спайдеру мозги

Бездушный wget качает всё.

Наш идеальный спайдер будет качать только те страницы, которые нужны пользователям.

Задача

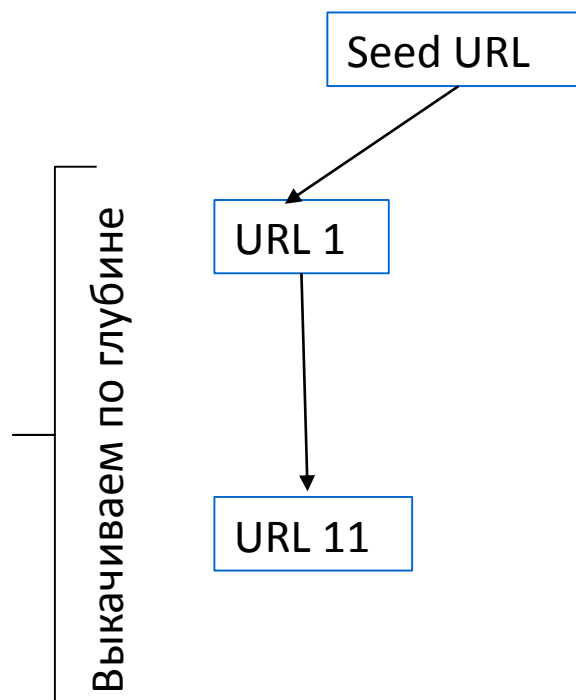
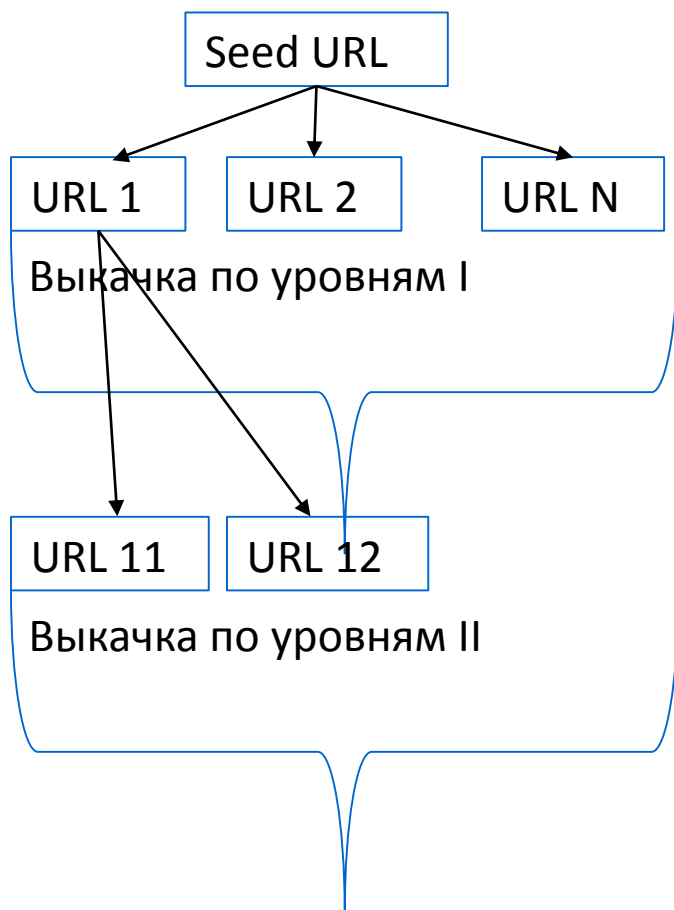
«Running a web crawler is a challenging task»

Sergey Brin and Lawrence Page, 1998

Алгоритмы обкачки

Обход сайта

Алгоритмы обкачки. Поиск в ширину/ глубину



Без приоритета!!!

Алгоритмы обкачки. Fish Search

- «игра в жизнь» с «рыбками» в главной роли
- Есть точка входа. d – минимальная глубина обкачки
- Скачиваем страницу. Запускаем агентов по каждой исходящей ссылке и пересчитываем d для каждой страницы:
 - Родительский документ и текущая страница релевантны:
 $d(\text{child}) = d$
 - Нерелевантны: $d(\text{child}) = d - 1$
- В итоге «рыбки» умирают

Алгоритмы обкачки. PageRank/HITS.

Авторитетность источников.

- PageRank – индекс цитируемости страницы. Чем больше цитируют, тем больше вес ссылки
- HITS – индекс цитируемости с учетом авторитетности источника.

Алгоритмы обкачки. Naïve best First

- Использует предположение, о взаимной релевантности страниц.
 - $A \rightarrow B$ – значит темы A и B одинаковые
- Вес B оценивается через вес A по отношению к теме T.
- Вес оценивается через схему TF-IDF, через близость к некоторой теме.

Что выбрать?

- Обходы
 - **X** Поиск в ширину
 - **X** Поиск в глубину
 - **+** Fish Search
 - **+** Shark Search
- Подсчет весов
 - **+** Наивный (Naïve best First)
 - **+** Page Rank
 - **+** HITS
 - ...

Фокусировка поискового робота

Сводится к построению очереди на обкачку и ее регулярной балансировке

Фокусировка поискового робота

Фокусируем робота через:

- Приоритет (priority-based)
- Структура (structure-based)
- Контекст (context-based)
- Поведение пользователей (behavioral-based)
- Обучение (learning-based)

Фокусировка: priority-based

Дано:

- Скачанная страница
- Метрика релевантности текста (например, тематические словари)

Фокусировка: priority-based

Алгоритм:

1. Скачали страницу
2. Оценили ее релевантность – число
3. Все ссылки с нее – с этим приоритетом
4. Очередь ссылок упорядочивается по их приоритетам

Фокусировка: structure-based

Учитываем структуру страницы:

- Заголовок важнее обычного текста
- Приоритет: не просто вхождения тематических слов, но и куда именно они входят

Фокусировка: structure-based

Division Score:

- У нас есть тематические словари. Тематика – Division. У каждой тематики – свой коэффициент.
- Каждая ссылка принадлежит определенной тематике (например, по родительской странице).
- Division score – отношение пересечения слов ссылки и всей тематики к полному словарю темы

Фокусировка: context-based

У ссылок тоже есть текст.

Текст ссылки определяет ее релевантность.

+ PageRank/HITS

Фокусировка: behavioral-based

Предыдущие подходы – релевантность каким-то тематикам.

Фокусировка: behavioral-based

Предыдущие подходы – релевантность каким-то тематикам.

Поиск – для пользователей!

Фокусировка: behavioral-based

Предыдущие подходы – релевантность каким-то тематикам.

Поиск – для пользователей!

Давайте учтем их интересы.

Фокусировка: behavioral-based

Давайте учтем интересы пользователей:

- Какие запросы задают
- Какие сайты посещают

Фокусировка: learning-based

До сих пор работали с уже обкачанной страницей.

Можно научиться предсказывать приоритет страницы.

Фокусировка: learning-based

Например, ML

Features:

- Релевантность родительской страницы
- Количество входящих ссылок (и их тексты)
- Форма урла (?)

Очередь на обкачку

- У каждого урла есть приоритет
- Приоритеты сравнимы между собой
- Очередь упорядочена

Очередь на обкачку

- У каждого урла есть приоритет
- Приоритеты сравнимы между собой
- Очередь упорядочена

Считаем, что так мы будем в первую очередь скачивать страницы, которые **интересны** пользователю.

Какие страницы интересны?

Что мы можем ранжировать

- С заранее определёнными темами
 - Знаем запросы пользователей, и качаем страницы релевантные этим запросам, взвешивание по релевантности
- С большим индексом цитируемости
 - Знаем, на что ссылаются и взвешиваем их по количеству входящих ссылок
- С хорошей пользовательской посещаемостью
 - Раз ходят пользователи, значит это нужные страницы. Можно взвешивать на количество посещений

Какие страницы интересны?

С заранее определёнными темами:

- Хорошо для популярных запросов
- Плохо для низкочастотных запросов
- Много ресурсов: скачать, оценить, принять решение

Какие страницы интересны?

С большим индексом цитируемости:

- Хорошо для популярных ресурсов
- Плохо для новых сайтов (с нерелевантными входящими ссылками)
- Линкофермы(!)

Какие страницы интересны?

С большим индексом цитируемости:

Как ИЦ коррелирует с популярностью страниц у реальных пользователей?

Какие страницы интересны?

С хорошей пользовательской посещаемостью:

- Страницы точно нравятся пользователям
- Мало данных – проще работать

Какие страницы интересны?

С хорошей пользовательской посещаемостью:

- Страницы точно нравятся пользователям
- Мало данных – проще работать

QLink – query-link

Ранк – количество переходов. Нормируем по всем переходам на сайт.

Полнота покрытия

У Qlink малая полнота покрытия (~5-10%)

Полнота покрытия

У Qlink малая полнота покрытия (~5-10%)

Экстраполируем эти данные

Полнота покрытия

У Qlink малая полнота покрытия (~5-10%)

Экстраполируем эти данные:

1. Для всего сайта
2. По ссылкам ($A \rightarrow B: ql(B) = coef * ql(A)$)
3. Для сегмента сайта

Что такое сегмент

Например:

Host: aldebaran.ru

Path: /kid/krapiv/krapiv[0-9]*\$

Query: *

Что такое сегмент

Предполагаем, что урлы, похожие по форме, имеют схожее содержимое.

В контексте QLink:
чем больше QLink в сегменте, тем выше вероятность, что и неизвестные еще урлы из сегмента понравятся пользователям

URL

RFC 1738, RFC 3986

Нас интересует схема `http – address`:

- `http://<host>:<port>/<path>?<query>#<fragment>`
- `<host>:<port>` - одинаковы для всего сайта
- `fragment` выбрасываем (анкеры в ajax)
- Нас интересует `<path>` и `<query>`
 - `path = segment * ["/" segment]`
 - `segment = * [uchar | ";" | ":" | "@" | "&" | "="]`
- `<query>` состоит из пар `name=value` разделенных `&`
- Порядок следования пар в `<query>` не важен (на нормальных сайтах)

Кластеризация урлов

Сколько урлов надо взять, чтобы разбить на сегменты с выраженными особенностями?

Слишком мало: неточные и крупные сегменты

Слишком много (и слишком строгие условия): много мелких сегментов, распыление QLink

Сколько урлов брать?

Сегмент == тематика

α – вероятность встретить урл из тематики

N – размер сэмпла

Какова вероятность найти менее k урлов из тематики?

$$p_{N,k}(\alpha) = \sum_{i=1}^k \binom{i}{N} \alpha^i (1 - \alpha)^{(N-i)}$$

$$P_{1000,10}(0.01) \approx 0.58$$

$$P_{1000,10}(0.02) \approx 0.01$$

$$P_{1000,10}(0.03) \approx 2 \times 10^{-5}$$

石庭(sekitei)



石庭(sekitei)

1. Отбираем N случайных урлов с сайта

石庭(sekitei)

1. Отбираем N случайных урлов с сайта
2. Создаем признаки для урлов
 - Какие?

石庭(sekitei)

1. Отбираем N случайных урлов с сайта
2. Создаем признаки для урлов
 - Длина урла
 - Количество query-параметров
 - Сегменты пути
 - Query-параметры
 - Регулярки O_O

石庭(sekitei)

1. Отбираем N случайных урлов с сайта
2. Создаем признаки для урлов
 - Длина урла
 - Количество query-параметров
 - Сегменты пути
 - Query-параметры
 - Регулярки O_O
3. Отбираем признаки по частотности: αN
4. Кластеризуем:
 - Jaccard distance measure
 - Stack clustering

$$K(a, b) = \frac{|A \cap B|}{|A \cup B|}$$

Отбираем N случайных урлов

Насколько случайных?

Отбираем N случайных урлов

1. Сколько урлов? Примерно 1к

$$p_{N,k}(\alpha) = \sum_{i=1}^k \binom{i}{N} \alpha^i (1 - \alpha)^{(N-i)}$$

$$P_{1000,10}(0.01) \approx 0.58$$

$$P_{1000,10}(0.02) \approx 0.01$$

$$P_{1000,10}(0.03) \approx 2 \times 10^{-5}$$

2. Насколько случайные? Известные:неизвестные – 1:1

Признаки урлов

`somesite.com/path/to/url.html?a=1&b=2`

1. Количество сегментов:

- `/path/to/url.html` – 3

Признаки урлов

`somesite.com/path/to/url.html?a=1&b=2`

1. Количество сегментов:
 - `/path/to/url.html` – 3
2. Query:
 1. Количество параметров: 2
 2. Список параметров: `a+b`
 3. Наличие пары параметр-значение: `a=1`

Признаки урлов

`somesite.com/path/to/url.html?a=1&b=2`

1. Количество сегментов:
 - `/path/to/url.html` – 3
2. Query:
 1. Количество параметров: 2
 2. Список параметров: `a+b`
 3. Наличие пары параметр-значение: `a=1`
3. Конкретные сегменты в пути:
 1. На 1-ой позиции `path`
 2. На 2-ой позиции `to`
 3. ...

Конкретные сегменты в пути

Давайте использовать регулярки

Конкретные сегменты в пути

Давайте использовать регулярки

/path/12345/day_576/image.jpg

path -> "path", "[^/]+"

12345 -> "12345", "[0-9]+"

day_576 -> "day_576", "[^/]_576", "day_[0-9]+", "[^/]+_[0-9]+"

Image.jpg -> "image.jpg", "[^/]+.jpg", "[^/]+"

Пример

Создаем признаки для каждого адреса (пример):

http://www.sports.ru/tags/1365242.html?p=57&type=photo

↑ ↖ ↗
сегмент путь запрос


| № | Название признак |
|---|------------------------------------|
| 1 | 2 Сегмента |
| 2 | Запрос состоит из двух параметров |
| 3 | 0-й сегмент пути: tags |
| 4 | 1-й сегмент пути: 1365242\.html |
| 5 | 1-й сегмент пути; [0-9]+\. |
| 6 | 1-й сегмент пути: [^/]+\. |
| 7 | В запросе есть параметр p=57 |
| 8 | В запросе есть параметр type=photo |

Пример

Отсекаем признаки по частотности αN

- Отбираем признаки для sport.ru
- $\alpha = 0.1; N = 1000$

| | Н (частота) | Признак |
|---|-------------|---------------------------------|
| 1 | 759 | Пустой запрос |
| 2 | 379 | В пути ровно два сегмента |
| 3 | 328 | 0-й сегмент: fantasy |
| 4 | 321 | 1-й сегмент пути: [^/]+\..html |
| 5 | 315 | 1-й сегмент пути: [0-9]+\..html |
| 6 | 266 | 1-й сегмент пути: football |
| 7 | 249 | В пути ровно 4 сегмента |



Не берем признаки с частотой
меньше: $0.1 * 1000 = 100$

Кластеризация

- Используем любой алгоритм, который позволяет нам найти кластера по выделенным признакам
 - Принадлежность сегменту определяем через пространство признаков
- Формируем регулярные выражения в формате PCRE для найденных кластеров.
 - Принадлежность сегменту определяем по регулярным выражениям описывающим кластер.

Что делать с остатком?

Кластеризация

- Используем любой алгоритм, который позволяет нам найти кластера по выделенным признакам
 - Принадлежность сегменту определяем через пространство признаков
- Формируем регулярные выражения в формате PCRE для найденных кластеров.
 - Принадлежность сегменту определяем по регулярным выражениям описывающим кластер.
- Урлы вне сегментов – тоже сегмент.

Пример. Случай регулярных выражений

1. **`^/wiki/File:[^/]+\.`**`jpg$`

Регулярное выражение,
описывающее кластер

`/wiki/File:Spongilla_lacustris.jpg`

2. **`^/wiki/[^\.]+\.`**`jpg$`

`/wiki/Image:Deve.jpg`

3. **`^/wiki/Category:[^/]+$`**

`/wiki/Category:Roman-era_historians`

4. **`^/wiki/Talk:[^/]+$`**

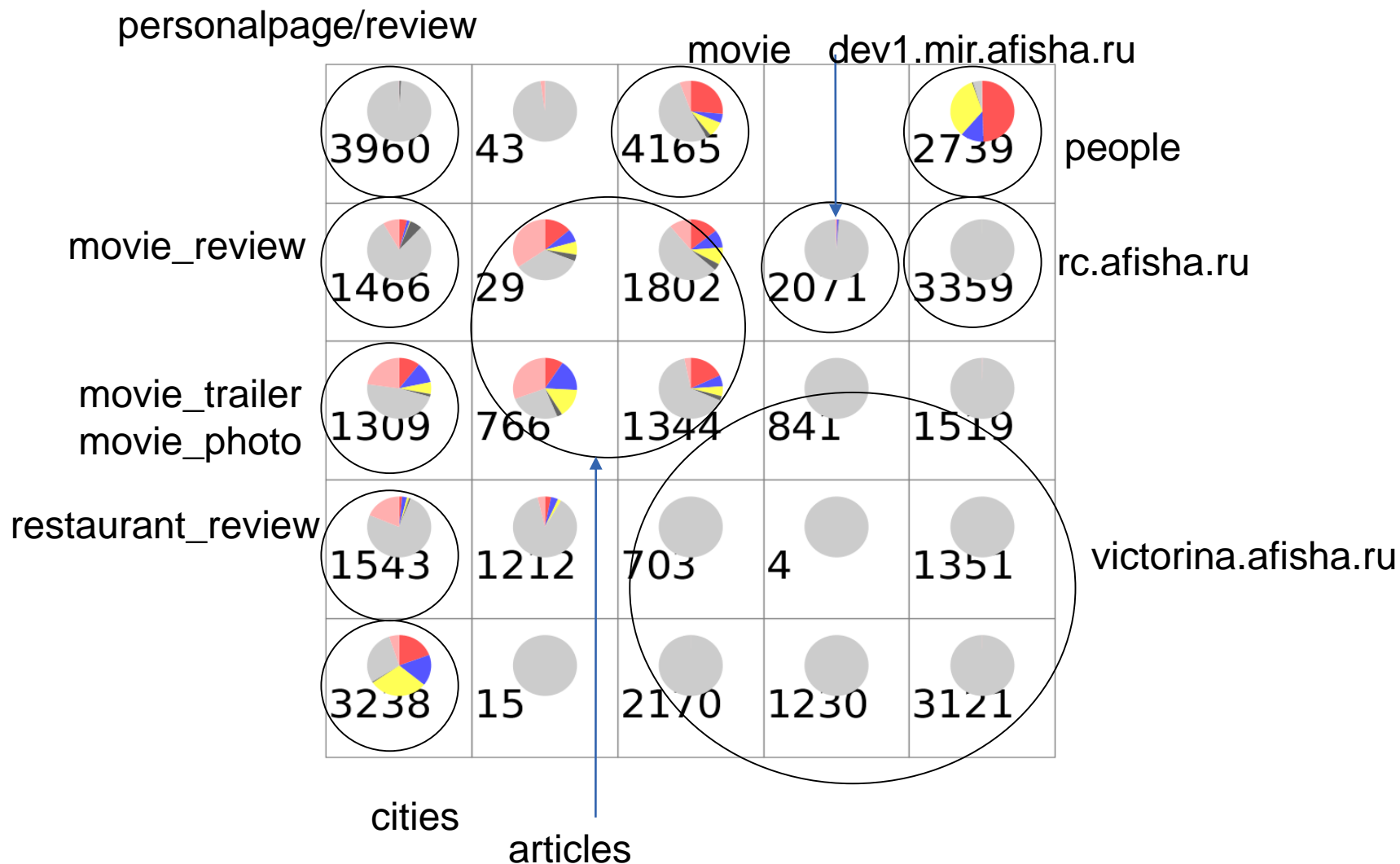
`/wiki/Talk:North_Light`

...



Кластеризация

afisha.ru



Что делать с сегментами?

- QLink'и позволяют оценить сегмент
- Хороший сегмент качаем активнее
- Используем эту информацию в построении индекса

Что делать с сегментами?

Давайте качать и показывать людям только сегменты с QLink'ами!

Что делать с сегментами?

Давайте качать и показывать людям только сегменты с QLink'ами!

Нельзя – вырожденные случаи + есть то, что мы не знаем

Статический ранк (Static Rank)

Ранк – число.

Получаем из:

1. Sekitei
2. Антиспам
3. Ссылочные (Indegree, PR и т.д.)

Строим модель: gradient boosting decision trees

Предугадываем, сколько QLink получим.

Модели:

- Индивидуальные для больших сайтов
- Общая для всех остальных

Что дальше?

Мы научились оценивать урлы и упорядочивать их по значимости.

В очереди на выкачку лежат урлы с разных сайтов.

Как соотносятся оценки между сайтами?

Надо ли качать весь сайт?

Квотирование

Мы ограничены в возможностях хранения и индексации

Размер квоты

Квотирование

Мы ограничены в возможностях хранения и индексации

Размер квоты:

- Всем поровну

Квотирование

Мы ограничены в возможностях хранения и индексации

Размер квоты:

- Всем поровну
- По посещаемости

Квотирование

Мы ограничены в возможностях хранения и индексации

Размер квоты:

- Всем поровну
- По посещаемости
- По сегментам

Как оценить качество?

Как оценить качество?

Мы утверждаем, что умеем предсказывать появление QLink в результатах выкачки (не ниже определенного уровня)

Оцениваем, сколько в итоге оказалось

Как оценить качество?

Цель: собрать индекс фиксированного размера

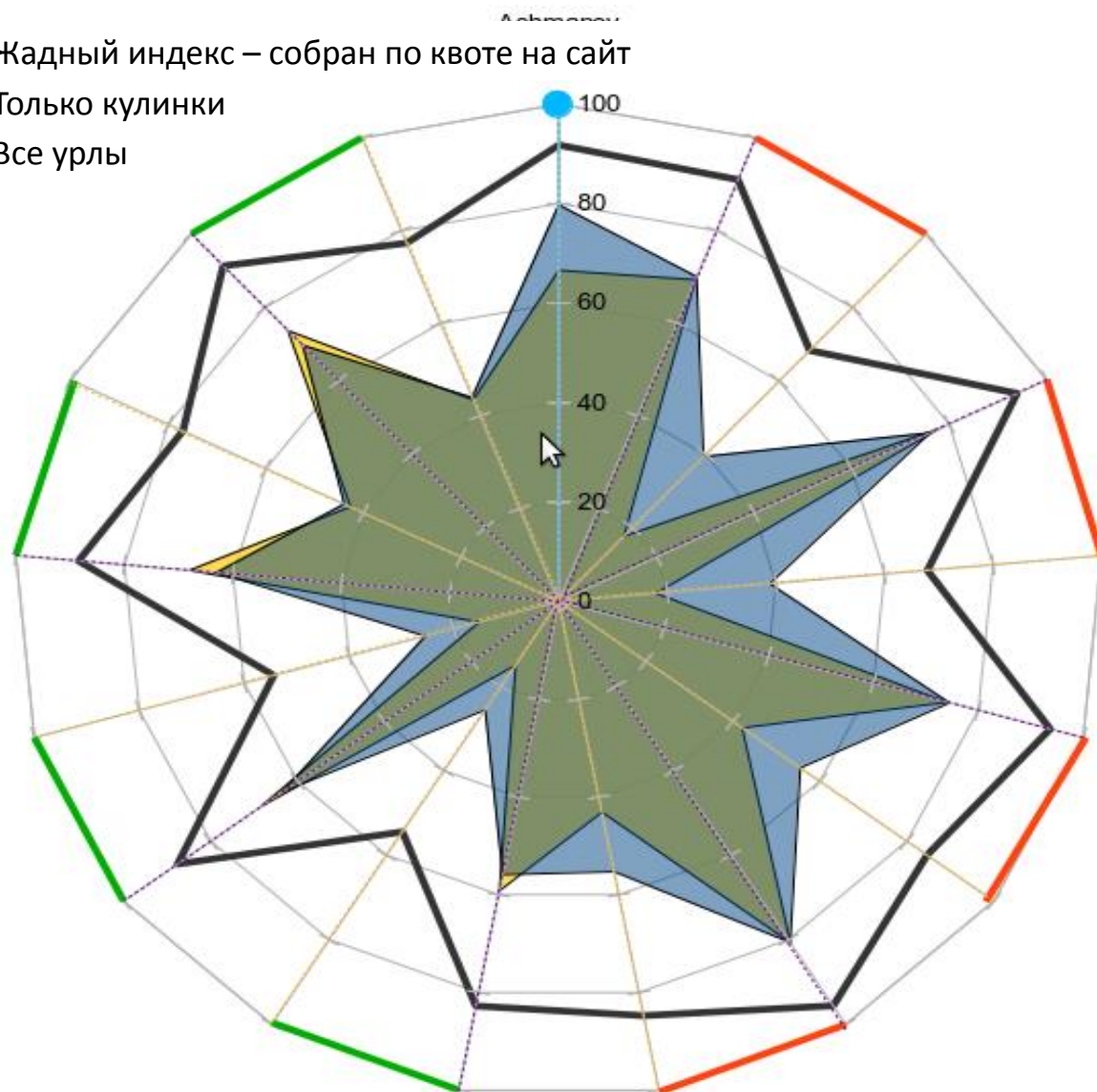
Сайты:

- «хостинги» – домены 2 уровня, чьи поддомены – посещаемые и крупные, часто независимые друг от друга, сайты. Пример: livejournal.com
- «большие сайты» – поддомены не-хостингов, которые по характеру могут быть выделены в отдельный сайт. Пример: mail.ru – не «хостинг», но можно выделить tu.mail.ru
- все остальные

| Алгоритм Секитей | Жадный алгоритм - посещаемость |
|--------------------------------------|--------------------------------|
| MIN_QUOTA ~ 100 | MIN_QUOTA ~ 100 |
| QUOTA = #PagesWithQlinks * MIN_QUOTA | QUOTA = F(#Visits) * MIN_QUOTA |
| Квота по камням | Квота для сайта |

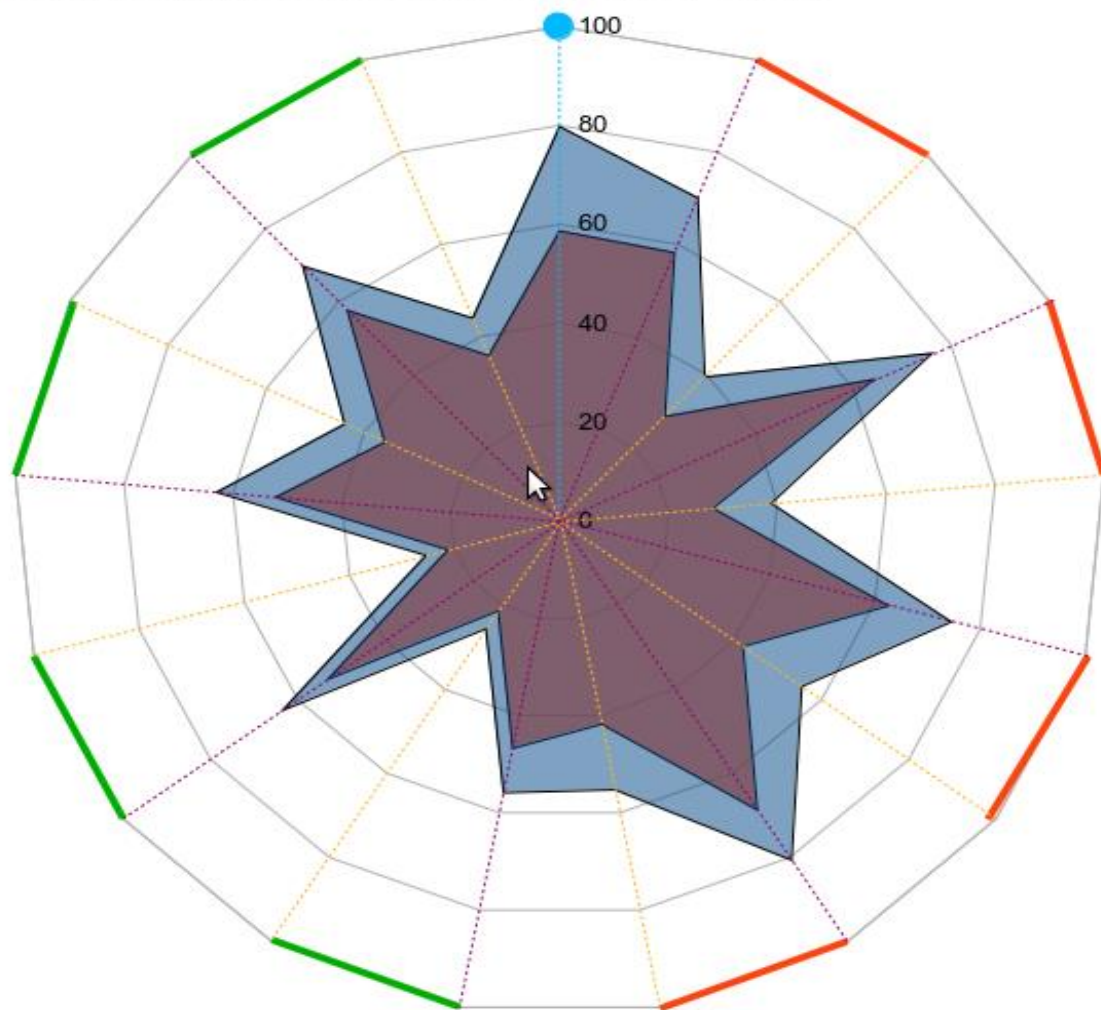
Baseline

- Жадный индекс – собран по квоте на сайт
- Только кулинки
- Все урлы



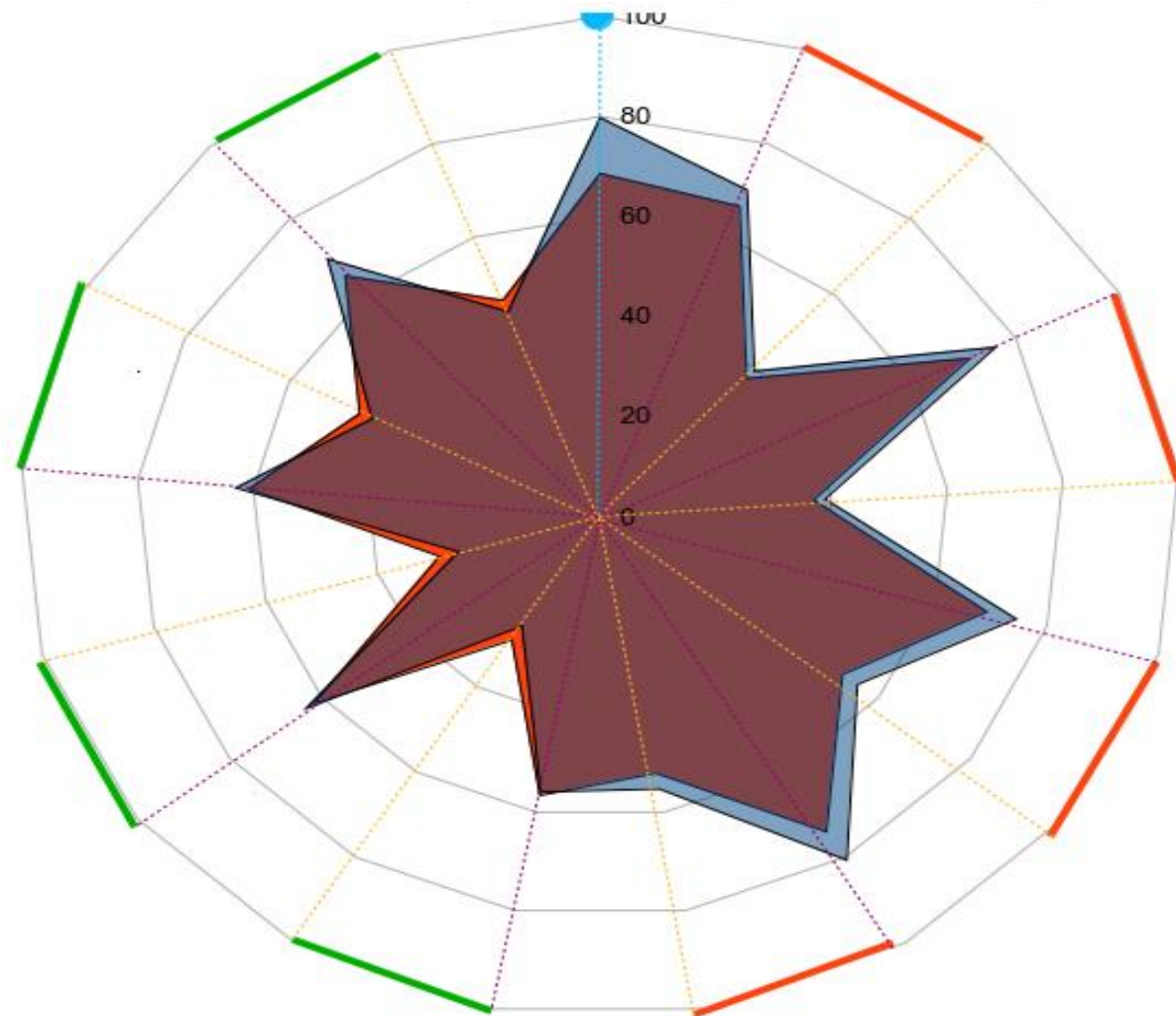
Квота по посещаемости

- Жадный индекс
- Жадный индекс, одинаковый размер, меньшая квота



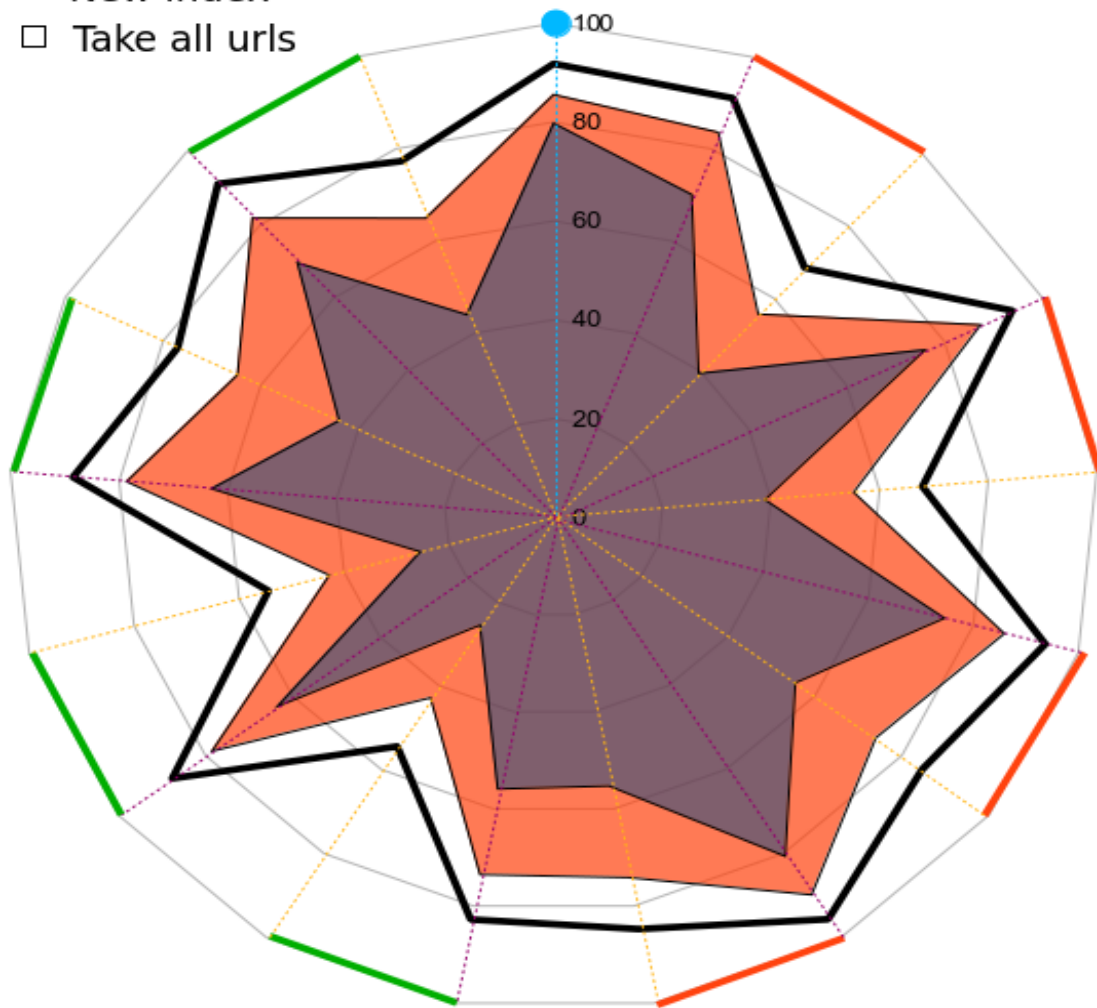
Квота sekitei (с уменьшенным индексом)

- Жалный индекс
- Жадный индекс с новой квотой (в два раза меньший размер)



Квота sekitei vs посещаемость

- Old index
- New index
- Take all urls



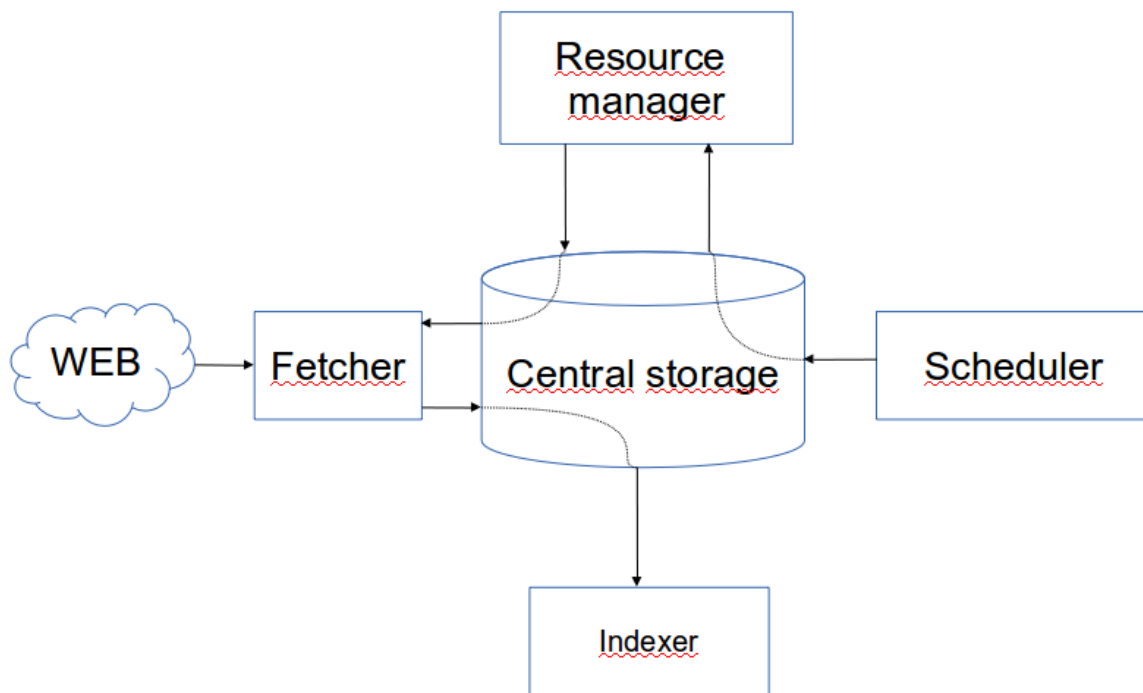
Как всё работает

Fetcher – просто качает (чуть сложнее wget'а)

Resource manager – доп.данные

Scheduler – квотирование, очереди на обкачку – батчи

Indexer – индексация урлов, построение поискового индекса



ДЗ

- С той же почты, которая указана на портале
- 10 попыток на нос (у вас есть открытые тесты – тестируйтесь на них, а не на демоне)
- sekitei1_infosearch@mail.ru и sekitei2_infosearch@mail.ru

ДЗ

- Python 2.7
- Реализуется модуль `extract_features`, который экспортирует функцию `extract_features` с параметрами
 - Вход – файл с хорошими урлами
 - Вход – файл с обычными урлами
 - Файл, в который будут записаны результаты
- Шаблон прилагается в архиве
- Запуск проверки `python ./check-features.py`
- Смотрим результаты.

ДЗ

- Всего 5 сайтов
- Каждый сайт это 20К обычных ссылок и 2К хороших ссылок
- Три открыты – обучающая выборка.
- Два скрыты - тестовая выборка.
- Для сайтов нужно сделать алгоритм “Сад камней” - выделение признаков.
- Мониторьте блог: архив с заданием + подробная инструкция + сроки сдачи (2 недели от даты появления задания до дедлайна)

Вопросы?