

# Сервис поиска электронных книг

Серверная часть, обеспечивающая хранение информации о книгах, поиск по ней и возможность модифицировать её

Иваницкий Андрей

Санкт-Петербургский Академический Университет РАН

31 мая 2010 г.

Руководитель: Н. М. Пульцин

## План презентации

- 1 Введение
  - Описание проблемы
  - Обзор существующих решений
  - Описание системы
- 2 Постановка задачи
- 3 Поиск по данным
- 4 Интерфейс модификации данных
  - Алгоритм взаимодействия с анализатором
  - Расчёт расстояния между строками
  - Фаза добавления
- 5 Заключение

## Описание проблемы

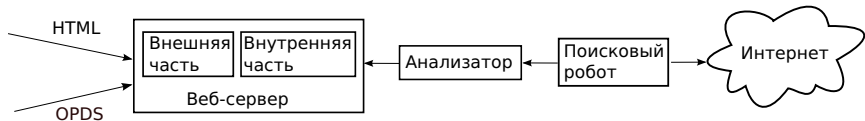
### Проблемы поиска электронных книг

- Проблема единого интерфейса электронных библиотек
- Проблема многих форматов книг

## Обзор существующих решений

- 1 Универсальные поисковые системы (Google, Яндекс)
- 2 Google books
- 3 Проект eBdb
- 4 Amazon Kindle, Sony Reader
- 5 OPDS
- 6 BookServer

## Описание внутренней структуры системы



## Постановка задачи

- ❶ **Поиск по данным**  
Мощный, быстрый и удобный поиск как для пользователя, так и для анализатора.
- ❷ **Интерфейс модификации данных**  
Полный, гибкий и расширяемый интерфейс модификации данных для анализатора.

## Поиск по данным

С помощью Sphinx реализован поисковый механизм

Предоставляет

- 1 Релевантный поиск как по отдельным сущностям, так и по различным их комбинациям;
- 2 Фильтрация результатов поиска по некоторым сущностям (язык книги, тэг);
- 3 Поиск с учётом морфологии языка;
- 4 Простой поиск (простой в использовании);
- 5 Исправление опечаток в запросе;
- 6 Поиск среди авторов по звучанию.

## Алгоритм взаимодействия с анализатором

К каждой сущности добавлены два понятия:

- 1 *индекс доверия (credit)*
- 2 *индекс релевантности (relevance)*

Если для автора *индекс доверия* и *релевантности* оказались **больше** пороговых значений, то **не создаётся** новой сущности. В противном случае **создаётся** новый автор.

Если для книги *индекс доверия* и *релевантности* оказались **больше** пороговых значений и авторы этой книги распознались как уже существующие в базе, то новой сущности **не создаётся**.

В противном случае **создаётся** новая книга.



## Алгоритм: Расчёт расстояния между строками

Две строки  $s_1$  и  $s_2$  разбиваются на слова:

$$s_1 \rightarrow S_1 = \{a_1, \dots, a_n\}, s_2 \rightarrow S_2 = \{b_1, \dots, b_m\}$$

Пусть  $n \leq m$

$$M_{i,j} = D_{Levenshtein}(a_i, b_j), i = 1..n, j = 1..m$$

где  $D_{Levenshtein}$  — расстояние Левенштейна

$$C_{min} = \min_{\alpha_1, \alpha_2, \dots, \alpha_m} \sum_{i=1}^n M_{i, \alpha_i}$$

где  $\alpha_1, \alpha_2, \dots, \alpha_m$  — перестановка чисел от 1 до  $m$ ,  
минимум по всем таким возможным перестановкам

$$C_{full} = C_{min} + (m - n) \times C_{remove}$$

где  $C_{remove}$  — цена добавления/удаления слова.

## Структура запроса

Запрос состоит из двух секций: *define*, *update*

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
  <define>
    ...
  </define>

  <update>
    ...
  </update>
</request>
```

## Секция define

Определяются сущности с *уникальным идентификатором*  $ui$  (в рамках одного запроса)

```
<author ui="1">
  <full_name> Leo Tolstoy </full_name>
</author>

<file ui="2">
  <link>http://example.com</link>
  <type>pdf</type>
  <size>4563214</size>
</file>
```

## Секция update

Обновляются связи между сущностями

```
<book ui="3">
  <authors>
    <author id="343" />
    <author ui="1" />
  </authors>
  <files>
    <file ui="2" />
  </files>
</book>
```

## Заклучение

- 1 Разработана внутренняя часть веб-сервера;
- 2 С помощью Sphinx реализован мощный быстрый поиск;
- 3 Реализован гибкий, расширяемый протокол взаимодействия с анализатором.

Полная работающая система доступна в Интернете по адресу  
<http://ebooksearch.webfactional.com/>

Исходный код проекта в репозитории google  
<http://code.google.com/p/ebooksearchtool/>