О функции поиска в электронной библиотеке

© В.А. Резниченко, Г.Ю. Проскудина
Институт программных систем НАН Украины, г. Киев
reznich@isofts.kiev.ua, gupros@isofts.kiev.ua

Аннотация

Работа посвящена описанию функции поиска в электронной библиотеке. На примерах реально действующей информационной системы рассмотрены все аспекты функции поиска, включая синтаксис запроса поисковой библиотеки программ Lucene. Рассмотрены также модели поиска, положенные в основу работы Lucene, и описаны особенности ранжирования документов.

1 Введение

В рамках развития вопросов, связанных с всесторонним описанием современных систем электронных библиотек [1], наряду с описанием информационной составляющей предпринимается попытка максимально полно описать также ее функциональную составляющую.

Функциональная часть электронных библиотек (ЭБ) представляет собой различные возможности и услуги, предоставляемые пользователям ЭБ. Это наиболее объемная и наиболее открытая часть, поскольку охватывает всю обработку ресурсов, а также действия пользователей в ЭБ. Функции в ЭБ можно условно разделить на пять классов: доступа к ресурсам; управления ресурсами; совместной работы; управления и настройки ЭБ [2].

Данная работа посвящена описанию функции поиска, которая представляет класс функций доступа к ресурсам и является существенным компонентом обнаружения объектов (ресурсов) в любой библиотечной системе.

Учитывая возрастающие пользовательские требования к поисковым возможностям, а также тот факт, что современные популярные поисковые веб-системы (в частности, Google) в значительной степени уже установили базовые требования, пользователи ожидают примерно того же и от библиотечных приложений. Поэтому разработчики популярных библиотечных систем могут использовать уже готовые приложения или библиотеки программ.

Хорошим примером здесь может служить добавление функций индексирования и поиска в библиотечные приложения, базирующееся на библиотеке программ Lucene [3]. В этой работе на примерах приводится описание поисковых возможностей библиотечной системы, работающей с ее использованием.

Труды 12ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» – RCDL2010, Казань, Россия, 2010

2 Lucene

Lucene – высокопроизводительная и масштабируемая библиотека Java-программ (один JAR-файл размером менее 1 Мб, не имеющий зависимостей) или набор инструментов для осуществления информационного поиска, которая не предназначена для использования в качестве автономного продукта, не содержит поисковых роботов (веб-краулеров), фильтров документов и поискового интерфейса пользователя. Тем не менее, в качестве подтверждения популярности Lucene можно привести целый ряд полнофункциональных приложений поиска, которые были интегрированы либо построены на ее основе. В дополнение к тем организациям, что перечислены на странице Wiki Lucene[1], можно привести и другие крупные и известные многонациональные организации, работающие с использованием Lucene. Она предоставляет возможности поиска в системе DSpace, одной из наиболее широко распространенных систем электронных библиотек.

О популярности Lucene также говорит и тот факт, что хотя она написана на Java, существуют многочисленные способы доступа (порты) к функциональности Lucene из других программных сред (C/C++, C#, Ruby, Perl, Python, PHP и др.).

Lucene выполняет две функции – индексирования и поиска. Она использует любые данные, из которых можно извлечь текст. Lucene не заботится об источнике данных, его формате и даже его языке. Это означает, что можно индексировать и производить поиск данных, хранящихся в файлах, например, веб-страницах на удаленных веб-серверах, документов, хранящихся в локальный файловой системе, простых текстовых файлов, документов Microsoft Word, XML, HTML, PDF-файлов или файлов любого другого формата, из которого можно извлечь текстовую информацию.

Поиск представляет собой процесс нахождения слов в индексе, с помощью которого находятся документы, содержащие эти слова. Индекс – это специально разработанная структура данных, хранящаяся в файловой системе в виде набора индексных файлов и являющаяся инструментом поиска. Качество поиска, как правило, описывается с помощью показателей

[1] http://wiki.apache.org/lucene-java/PoweredBy

точности и полноты. Полнота показывает, насколько хорошо система поиска находит соответствующие документы, а точность указывает на то, насколько хорошо система отфильтровывает нерелевантные документы. Тем не менее, здесь необходимо учитывать и ряд других факторов, например, скорость и возможность быстрого поиска в текстах большого объема; поддержку одно- и многословных запросов, запросов фраз, групповых символов, нечетких запросов, ранжирование и сортировку результатов поиска. Большое значение также имеет дружественный синтаксис для ввода таких запросов. Lucene предлагает целый ряд этих и других функций.

3 Поиск

Поиск – важный компонент выявления объектов в какой-либо библиотечной системе. Исходя из высоких пользовательских требований к поисковым машинам, можно утверждать, что предоставление как можно большего числа поисковых возможностей – важная характеристика большинства современных библиотечных систем. Поиск должен предоставлять возможность находить в системе информационные ресурсы, их свойства и/или содержание.

Интерфейс поисковых средств должен строиться таким образом, чтобы он был интуитивно понятным пользователям. Как правило, используется механизм построения поисковых запросов. В простейшем случае это фиксированное множество поисковых запросов, в которые заносятся значения параметров, задаваемых пользователем. Там, где возможно, значения поисковых терминов выбираются из заданных списков (например, язык документа, формат документа, перечень используемых метаданных, дата создания документа).

Как правило, системы ЭБ предоставляют следующие варианты поиска:

- простой или стандартный поиск;
- расширенный поиск;
- профессиональный поиск.

Пользователь имеет возможность самостоятельно выбирать тот или иной поиск.

3.1 Простой или стандартный поиск

Стандартный поиск – простейший вариант поиска, он представляет некоторые минимальные поисковые возможности, которыми может легко овладеть большинство пользователей и которые обладают достаточной полнотой. Поиск осуществляется во всех индексах, которые построены в системе, т. е. во всех описательных полях и во всех текстах. Большинство поисковых запросов формулируется с использованием возможностей именно этого вида поиска.

В современных системах ЭБ можно сузить пространство поиска, а также осуществлять многошаговый поиск. Сужение пространства поиска предусматривает возможность отделить подмножество информационных разыскиваемых ресурсов. Критериями такого отделения могут быть следующие:

- выбор ресурсов определенного вида (книги, журналы, диссертации и др.);
- выбор ресурсов, которые отвечают определенным значениям метаданных (дата создания, автор, организация, язык документа, формат представления документа, др.);
- использование иерархических словарей поисковых терминов, предметных рубрикаторов, тезаурусов или классификаторов.

Многошаговый поиск предусматривает то, что поиск ведется среди тех ресурсов, которые были найдены в результате предыдущего поиска. В результате такого поиска пользователю предоставляется возможность последовательно находить более релевантные ресурсы. Важно отметить, что интерфейс пользователя должен предлагать ясный путь для последующих поисков или действий.

Результаты поиска могут быть отсортированы по автору; названию; дате выпуска; степени релевантности; другим характеристикам.

В интерфейсе порядок сортировки должен быть явно высвечен, по умолчанию, как правило, по степени релевантности.

Язык поиска в ЭБ имеет необходимую для поиска релевантных документов полноту, что обеспечивает достаточную гибкость, выразительность и наглядность. Для этого он должен удовлетворять следующим требованием:

- поиск документов по их полным текстам и по описательным данным (метаданным);
- поисковые термины должны состоять из отдельных поисковых слов или фраз;
- отсечение окончания или начала слов в поисковых терминах;
- использование групповых символов (заменяющих один или несколько символов: знак вопроса (?) может представлять какой-либо одиночный символ; звездочка (*) используется для представления какого-либо символа или группы символов);
- использование булевой логики (как правило, логических связок И-ИЛИ-НЕ);
- поиск по словоформам и синонимам поисковых терминов, а также с учетом морфологии языка (поиск слова во всех его морфологических формах), на котором формулируются поисковые термины;
- чувствительность или нечувствительность к регистру символов;
- поиск по близости размещения слов в тексте;
- поиск по фонемному звучанию поисковых терминов.

Основу поискового языка составляют поисковые термины. Существует два вида терминов: слова и фразы, использование которых опишем подробнее.

3.1.1 Поиск слов

1) Поиск отдельных слов – осуществляет нахождение наличия того или иного слова среди всех описательных полей, а также в тексте статей. Так, на-

пример, если использовать в качестве простого поиска слово Резниченко, то получим результат [2], где не все найденные статьи имеют в качестве автора Резниченко, среди найденных есть статьи с автором Резніченко (написание фамилии на украинском языке), а также статья, в которой только один автор – Стадник. Подчеркнем, что в простой поиск вовлекаются все описательные поля, а также полные тексты статей. И, скорее всего, это поисковое слово встречается в текстах (например, в списке литературы тех статей, где автором является Резніченко). Так как поиск проводится по всем описательным полям, то, например, используя в качестве поискового слова УДК – 004.82, можно получить все статьи данной библиотеки с этим УДК. Но при этом следует помнить, что 004.82 может встречаться в других полях, например, в аннотации или тексте самой статьи. Но наибольшей будет вероятность того, что в найденных статьях слово 004.82 встречается в УДК.

2) Слова, по которым не производится поиск, – стоп-слова. Механизм поиска игнорирует некоторые слова, которые часто встречаются в языке, однако являются бессмысленными с точки зрения поиска. К ним относятся, например, в английском языке: a, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, the, to, was. Если указать в качестве поискового слова какое-либо из этих слов, то ничего не будет найдено. Для украинского и русского языков может быть аналогичное требование. Обычно список стоп-слов поисковых машин состоит из наречий, союзов, предлогов и т. д. Но, если употребительное слово существенно для запроса пользователя, можно включить стоп-слово в поисковый запрос, используя знак +.

3) Поиск по нескольким словам. В поисковом запросе можно указать несколько слов. В этом случае производится поиск статей, которые содержат какие-либо из указанных слов. Нет каких-либо ограничений на количество слов в поисковом запросе.

4) Использование в словах групповых символов. Групповой символ – это специальный символ, который замещает один или несколько символов в поисковом слове. Существует два групповых символа: знак вопроса (?) и звездочка (*). Знак вопроса замещает какой-либо одиночный символ в слове в том месте, где он расположен. Символ * используется для представления какого-либо символа или группы символов. Символ ? можно использовать, например, если спрашивающий не знает точного написания поискового слова. Например, если он точно не знает, как пишется английское слово dynamic или dinamic, то можно указать в поисковом запросе d?namic. Другой вариант возможного использования – если фамилия автора в разных статьях указывается на украинском либо на русском языке, например, Анісімов и Анисимов. Если нужно найти все статьи этого автора, не учитывая языка, то в поисковом запросе указываем Ан?с?мов. Разумеется, что этого можно достичь в случае, когда написание фамилии (или другого поискового слова) отличается только отдельными буквами в одних и тех же позициях слова. Можно использовать несколько подряд расположенных символов ?. Так, например, если использовать поисковое слово алекс??в, то будут найдены статьи с авторами Алексеев, Алексеєв и Алексєєв. Символ * используется в том случае, когда в слове известна лишь некоторая последовательность символов, а другие – не известны. Например, спрашивающий помнит, что фамилия точно начинается на Антон, но следующие буквы не помнит. В этом случае нужно использовать поисковое слово Антон*, и будут найдены все те статьи, в которых фамилия автора начинается на нужные символы, например, Антонюк, Антонцева, Антонов, Антонова (и некоторые другие, в которых поисковое слово находится в других описательных полях или в текстах статей). Можно одновременно использовать символы * и ?. Например, поисковое слово Р?зн*е?ко является корректным, здесь отыскиваются все те статьи, которые соответствуют его содержанию. Заметим, что не разрешается использовать символы ? и * в начале слова. В этом случае и во всех других, когда поисковый запрос сформулирован неправильно, выводится сообщение, что не правильно сформулирован поисковый запрос.

5) Поиск по близости звучания слова. Предоставляется такая оригинальная возможность, как поиск по близости звучания того слова, которое указывает запрашивающий. Для этого нужно указать символ тильда (~) в конце слова. Так, например, если указать поисковое слово семантический~, то будут найдены статьи, которые содержат однокоренные слова, например, семантические, семантическому, семантическими и т. д., а также такие близкие по звучанию слова, такие, как генетический, статический, механические, органический, электрические, математический, электрический и т. д. Также можно указать дополнительный (факультативный) параметр, показывающий меру близости звучания слов и находящийся в интервале 0 – 1. Например, семантический~0.4. Чем больше число, тем большая понадобится схожесть звучания. Например, с использованием выражения семантический~0.8 в тестируемой ЭБ были найдены статьи только с однокоренными словами, но не были найдены другие однокоренные слова, которые не так близки по звучанию, например, семантическому, семантического. Если мера близости звучания не указывается, то по умолчанию используется значение 0.5. Следует отметить, что поиск по близости звучания слова выполняется с использованием алгоритма нахождения минимального расстояния редактирования, описание которого можно найти, например, в [4].

3.1.2 Поиск фраз

Фраза – это последовательность слов, располо-

[2] Все представленные в работе примеры проверены в библиотечной системе http://dspace.nbuv.gov.ua:8080/dspace

женных в двойных кавычках. Например, фразами являются база данных, исчисление предикатов, семантическая сеть. При использовании фразы осуществляется поиск статей, в которых содержится указанная фраза, т. е. именно такая последовательность слов. Например, если указать фразу knowledge base, то можно получить результат, который содержит украиноязычные и русскоязычные статьи. Статья может иметь дополнительное англоязычное название или англоязычную аннотацию, или эта фраза встречается в тексте статьи, например, среди перечня использованной литературы.

Во фразах нет смысла использовать групповые символы для слов. Их использование не рассматривается как ошибка, но спрашивающий не получает ожидаемого результата. Здесь нет смысла использовать поиск по близости звучания всей фразы. Его использование не рассматривается как ошибка, но результат будет таким же, как и при поиске по самой фразе.

1) Использование слов и фраз. В поисковом запросе можно указывать одновременно слова и фразы. Например:

библиотека поиск «база данных» oai-pmh «интегрированный каталог».

В результате такого поискового запроса будут найдены статьи, которые содержат любые из перечисленных слов или фраз.

2) Поиск с использованием расстояния между словами. Предоставляется возможность поиска по словам, которые расположены на расстоянии, которое не превышает указанного числа. Для этого в конце фразы размещается символ тильда ~, за которым идет целое число, указывающее расстояние. Например, если указать поисковое выражение «научных системы»~4, то получим следующий результат:

– «Создание научных архивов с помощью системы EPrints»;

– «Создание научных электронных библиотек с помощью системы DSpace».

В случае же использования выражения «научных системы»~3 получим только статью:

– «Создание научных архивов с помощью системы Eprints».

Отметим, что англоязычные стоп-слова не учитываются при определении расстояния между словами. Расстояние 0 указывает, что слова расположены рядом. Порядок расположения двух слов в фразе является существенным. Фраза может содержать несколько слов. При этом имеется в виду следующее: расстояние между первым и последним словами не должно превышать указанного расстояния, не учитывая указанные промежуточные слова. Например, если поисковый запрос имеет вид «представлення бібліотечних за онтологій»~6, то в результате получим статью:

– «Представлення та відображення бібліотечних предметних класифікацій за допомогою інструментів онтологій».

Действительно, если между словами представлення и онтологій удалить слова бібліотечних и за, то между ними расположено 6 слов. Если указать расстояние 5, то эта статья не будет найдена. Отметим, что порядок промежуточных слов не является существенным.

3) Поиск по важности слов или фраз. Когда перечисляются слова или фразы, предоставляется возможность указать, какие из них являются более важными (релевантными). Важность слов и фраз влияет на порядок расположения статей в результате поиска. Сначала следуют статьи с наиболее важными словами/фразами, а потом с менее важными. Для указания меры важности в конце слова/фразы нужно поместить символ ^, а за ним число, которое указывает степень важности. Например, нужно найти статьи со словами parallel и programming и придать слову parallel большую релевантность, то поисковый запрос будет следующим: parallel^4 programming. В качестве меры релевантности можно использовать неотрицательные целые числа и десятичные дроби в интервале 0-1. По умолчанию все слова/фразы имеют меру релевантности 1. Релевантность можно указывать для многих слов и фраз, например, база данных^20, информационная система^10, библиотека^5, протокол oai-pmh. Слова/фразы нужно располагать в порядке уменьшения их меры релевантности.

4) Обязательное наличие слов или фраз. При перечислении слов или/и фразы можно указать на те, которые обязательно должны встретиться в статье. Для этого нужно перед словом/фразой указать символ +. Например, в запросе: +библиотека +научная электронная статьи, которые ищутся, обязательно должны содержать слова библиотека и научная и могут содержать слово электронная. А в запросе: +«база данных» библиотека статьи, которые ищутся, обязательно должны содержать фразу "база данных", и могут содержать слово библиотека. При использовании слов их можно употреблять с групповыми символами ? и *, например: +библио* +электрон* +наук*.

3.2 Расширенный поиск

Функция расширенного поиска предоставляет пользователям все возможности простого поиска: полнотекстовый поиск; поиск всех описательных метаданных; поиск выбранных полей метаданных (набор полей; как правило, пользователь определяет 3 – 4 поля, объединяя их булевыми операторами), а также предоставляет дополнительные возможности по формулировке поисковых запросов: выбрать пространство поиска; указать поля поиска и их значение; использовать логические операторы в запросе.

В качестве значений всех полей, кроме поля Язык, можно использовать все те возможности, которые описаны в подразделе 3.1. Для поля Язык используются стандартные двухсимвольные представления языков согласно стандарту ISO 639-1. Например: uk – украинский; ru – русский; en – английский.

Использование логических операторов. Расши-

ренный поиск предоставляет возможность объединять поисковые выражения логическими операторами И(AND), ИЛИ(OR) и НЕ(NOT).

3.3 Профессиональный поиск

Профессиональный поиск предусматривает, что пользователь хорошо знаком с синтаксисом поискового языка и может сформулировать поисковый запрос с использованием этого синтаксиса (формулирование такого запроса проводится в тех полях, в которых формулируется простой поиск). Приведем несколько основных правил его использования.

1) Можно формулировать любые выражения, которые допустимы в простом поиске.

2) Для указания того, что поисковое выражение принадлежит тому или другому полю, указываются имя поля, символ : и поисковое выражение. Если это выражение состоит из более чем одного слова и/или фразы, то оно берется в круглые скобки. Названия используемых полей следующие:· author – автор; title – название; keyword – ключевое слово; abstract – аннотация; sponsor – спонсор; identifier – идентификатор. Можно одновременно использовать выражения с полями и без полей. Примеры использования поисковых выражений с полями и без них:

```
author:Резн?ч*;
title:(электр* катал* поиск);
abstract:(библ* "электронный каталог" семантический~0.4);
система title:баз* abstract:дан*.
```

3) Логические операторы (И, ИЛИ, НЕ) записываются между поисковыми выражениями, которые могут быть уточнены именами полей. Если выражение состоит из более чем одного слова/фразы, то его берут в круглые скобки. Если нужно указать порядок вычисления этих операторов, то также используются круглые скобки. В противном случае их вычисление проводится слева направо. Примеры:

```
база И данные ИЛИ ansi НЕ прогр*;
author:Резн?ч* И title:(электр* катал* поиск) ИЛИ (система title:баз*);
(+библ* научн*) НЕ (+электр* семантика~) И (база10);
(jakarta ИЛИ apache) И website.
```

4) Использование специальных символов. Синтаксис поискового языка использует несколько символов в качестве специальных. К ним относятся +, –, &, ||, !, (,), {, }, [,], , ", ~, *, ?, :, \. В общем случае их нельзя использовать в поисковых выражениях. Но все же существует возможность их использования. Для этого перед таким символом нужно поставить символ \. Например, если в запросе нужно использовать выражение (1+1):2, то его записывают следующим образом: \(1\+1\)\:2.

4 Дополнительные поисковые функции

ЭБ могут предоставлять также вспомогательные возможности для осуществления функций поиска, например, такие, как:

- настройка параметров поиска;
- сохранение результатов поиска для последующего использования;
- сохранение текстов запросов и их повторное использование самостоятельно или в составе других запросов;
- представление результатов поиска в разных форматах;
- помощь пользователям при использовании поисковых средств для повышения эффективности поиска.

5 Особенности ранжирования документов в Lucene

В настоящее время различают три общие модели поиска [3, 5].

1) Булева модель, когда документы при поиске делятся на две группы – либо соответствующие, либо несоответствующие запросу, при этом никакие их оценки не вычисляются. Так как в этой модели нет оценок релевантности документа запросу, то выдается все множество документов, соответствующих запросу, без какого-либо ранжирования.

2) Векторная модель, когда и запросы, и документы моделируются векторами весов n-мерного пространства:

$$V(d) = (w_1, ..., w_n),\ V(q) = (v_1, ..., v_n),$$

где n – общее количество различных термов (слов) во всех документах коллекции, каждый уникальный терм – измерение, w_i и v_i – соответственно веса i-го терма в документе d и запросе q, веса могут вычисляться как tf-idf (term frequency – inverse document frequency, частота терма – обратная частота документа) [5].

Релевантность или подобие между запросом и документом вычисляются расстоянием между этими векторами: чем ближе они расположены, тем больше документ d соответствует запросу q. В векторной модели часто используется косинусная оценка релевантности q и d:

$$\text{cosineSim}(q,d) = \frac{\mathrm{V}(q)\cdot\mathrm{V}(d)}{|\mathrm{V}(q)\|\mathrm{V}(d)|},$$

где $\mathrm{V}(q)\cdot\mathrm{V}(d)$ – скалярное произведение двух векторов, а $|\mathrm{V}(q)\|\mathrm{V}(d)|$ – произведение их длин.

Следует отметить, что векторная модель специально не требует, чтобы веса были обязательно tf-idf. Но как показал практический опыт, использование таких весов дает высокоточный поиск. Поэтому Lucene использует tf-idf – функцию, прямо пропорциональную частоте вхождения терма в документ и обратно пропорциональную числу документов коллекции, содержащих этот терм.

3) И, наконец, вероятностная модель, где вычисляется вероятность того, что документ является хорошим соответствием запросу, с использованием полного вероятностного подхода.

Lucene при реализации функции поиска комбини-

рует векторную и булевую модели. Подход заключается в том, что отбор документов осуществляется в соответствии с булевой моделью, а их ранжирование – в соответствии с векторной моделью.

Уравнение для $\text{cosineSim}(q,d)$ можно рассматривать как скалярное произведение нормализованных векторов весов, в том смысле, что деление вектора $\text{V}(q)$ на его длину есть его нормализация к единичному вектору.

Lucene уточняет оценку векторной модели $\text{cosineSim}(q,d)$ как с точки зрения качества поиска, так и удобства ее вычисления.

1) Нормализация $\text{V}(d)$ к единичному вектору может быть проблематичной в том смысле, что таким образом удаляется информация о длине документа. Чтобы избежать этой проблемы, используется множитель, учитывающий его длину, который приводит $\text{V}(d)$ к вектору длиной, равной или большей единицы: $\text{docLenNorm}(d)$.

2) При индексации документа пользователи могут указать, что одни документы важнее, чем другие, путем присвоения документу показателя важности (т. е. одни документы имеют предпочтение перед другими при прочих равных условиях). А это значит, что и оценка каждого документа получит дополнительный множитель важности документа $\text{dBoost}(d)$.

3) Особенностью модели документа Lucene является то, что документ рассматривается как совокупность полей (полей метаданных). В связи с этим каждый терм запроса относится к какому-то конкретному полю. Нормализация длины документа представляет собой нормализацию длин полей документа. Наконец, помимо того, что имеется множитель важности документа, существуют также и множители важности отдельных его полей (например, 0.5 для поля autor, 0.3 – для title и 0.2 – для body).

4) Одно и то же поле может присутствовать в документе многократно (например, в случае, когда документ имеет несколько авторов), поэтому важность этого поля равна произведению множителей важности отдельных его экземпляров в документе.

5) Во время поиска пользователи могут задать важность для каждого запроса, подзапроса и каждого терма запроса, поэтому вклад каждого терма запроса для оценки документа умножается на важность этого терма запроса $\text{qBoost}(q)$.

6) Документ может соответствовать некоторым термам запроса и не содержать при этом все его термы (это справедливо для некоторых запросов), поэтому имеет смысл повышать оценку релевантности тех документов, которые содержат больше поисковых термов. Для этого в оценку вводят множитель $\text{coord}(q,d)$.

На основании вышеизложенного и предполагая для упрощения, что индекс создается для одного поля, получим *концептуальную формулу для оценки релевантности поиска Lucene*:

$$\text{s}(q,d) = \text{coord}(q,d)\cdot\text{qBoost}(q)\cdot\frac{\text{V}(q)\cdot\text{V}(d)}{|\text{V}(q)|}\cdot$$

$$\cdot\,\text{docLenNorm}(d)\cdot\text{dBoost}(d).$$

Данная концептуальная формула упрощена в том смысле, что, во-первых, принимает во внимание документ, а не его поля и, во-вторых, важность, как правило, определяется не для запроса, а для термов запроса.

Вкратце опишем, как Lucene реализует эту формулу на практике:

$$\text{s}(q,d) =$$

$$= \sum_{\text{t in q}} (\text{tf}(t \text{ in } d)\cdot\text{idf}(t)^2\cdot\text{boost}(t.\text{field in } d)\cdot$$

$$\cdot\,\text{lenNorm}(t.\text{field in } d))\cdot\text{coord}(q,d)\cdot\text{qNorm}(q),$$

где $\text{tf}(t \text{ in } d)$ – функция, прямо пропорциональная частоте терма в документе, т. е. числу вхождения терма в документ;

$\text{idf}(t)$ – обратная частота документа, содержащего терм t; это мера того, насколько уникальным является терм; более общие термы имеют низкое значение idf, редкие термы – высокое значение;

$\text{boost}(t.\text{field in } d)$ – важность поля (и документа в целом); важность можно вводить как статически, повышая важность некоторых полей (и документов) при индексировании данного документа, так и динамически, т. е. непосредственно на момент поиска;

$\text{lenNorm}(t.\text{field in } d)$ – нормализованное значение поля, учитывающее общее число термов в поле; это значение вычисляется на этапе индексирования документа и хранится в индексе (в его нормах); почти все поисковые машины (включая Lucene) автоматически (на этапе индексирования) повышают важность более коротких по длине полей; интуитивно это имеет смысл, поскольку, если у нас совпадут слово или два в очень длинном документе (поле), это менее релевантно (не так важно), чем если слова совпадут в документе (поле), скажем, длиной 3 – 4 слова;

$\text{coord}(q,d)$ – множитель, который зависит от числа термов запроса, найденных в данном документе (см. выше);

$\text{qNorm}(q)$ – значение евклидовой нормы (длина вектора) запроса, учитывает сумму квадратов весов каждого терма запроса; норма вычисляется на момент начала поиска; этот множитель, как правило, не влияет на ранг документа (поскольку для конкретного запроса имеет одинаковое значение для всех найденных документов), однако это значение может сохраняться для того, чтобы иметь возможность сопоставить один и тот же документ двум разным запросам.

В дополнение к явным множителям последнего уравнения на основании запроса могут быть вычислены и другие множители (как составляющие множителя $\text{qNorm}(q)$. Сами запросы в некоторых случаях могут влиять на ранг документа, например, они

могли бы повысить важность документа, когда в запросе есть повторяющиеся термы, но только не один, а несколько, поскольку один повторяющийся терм, используемый для поиска, повысил бы все подобранные документы одинаково. При наличии повторяющегося терма в булевом запросе некоторые документы могут соответствовать одному терму, но не другому, позволяя множителю важности различать такие документы. По умолчанию множитель важности для запроса равен 1.0.

При реализации данной оценки релевантности документа поисковому запросу для эффективного вычисления некоторые компоненты (как, например, $\mathrm{lenNorm}(t.\mathrm{field in} d)$, вычисляются и агрегируются заранее, еще на этапе создания индекса.

Так вычисляется оценка $s(q,d)$, назначение которой – измерять релевантность (подобие) между запросом и каждым документом, соответствующим этому запросу. Оценка вычисляется для каждого документа d, содержащего каждый терм t в запросе q. Чем выше эта оценка, тем лучше документ соответствует запросу, т. е. тем выше его релевантность. По умолчанию Lucene возвращает документы в порядке убывания этого показателя, подразумевая, что верхние документы лучше соответствуют запросу.

6 Заключение

В данной работе на многочисленных примерах реально действующей системы ЭБ рассмотрены все аспекты функции поиска, включая синтаксис запроса популярной библиотеки программ для информационного поиска Lucene.

Также в работе рассмотрены модели поиска, положенные в основу работы Lucene, и описаны особенности ранжирования документов концептуально и практически.

Литература

[1] Кудим К.А., Резниченко В.А., Проскудина Г.Ю. Концептуальная модель электронной библиотеки // XI Всерос. науч. конф. «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL'2009, 17 – 21 сентября 2009, Россия, г. Петрозаводск. – Петрозаводск, 2009. – С. 23-31.

[2] Candela L., Castelli D., Dobreva M., Ferro N., Ioannidis Y., Katifori H., Koutrika G., Meghini C., Pagano P., Ross S., Agosti M., Schuldt H., Soergel D. The DELOS digital library reference model foundations for digital libraries. IST-2002-2.3.1.12. Technology-enhanced learning and access to cultural heritage. Version 0.98, December 2007. – http://www.delos.info/files/pdf/ReferenceModel/DELOS_DLReferenceModel_0.98.pdf.

[3] Hatcher E. Gospodnetic O., McCandless M. Lucene in action. Second edition. Manning publications, 2009. – 399 p.

[4] Jurafsky D., Martin J. Speech and language processing. Second edition. – Pearson Education International, USA, New Jersey, 2009. – P. 107-111.

[5] Manning C., Raghavan P., Schütze H. An introduction to information retrieval. – Cambridge University Press, 2009. – 581 p.

On the search function in the digital library

V.A. Reznichenko, G.Yu. Proskudina

This paper describes the search function in the digital library. We illustrate all aspects of search functionality, including query syntax of the search library Lucene, that used in the actual information system. Also we consider underlying information retrieval models in Lucene, and describe how Lucene score document matches to a query.