Методы и средства интеграции и обогащения библиотечных данных

Косолапов К.А ,Серебряков В.А., Тёймуразов К.Б.,

Шорин О.Н.
Вычислительный центр им. А.А. Дородницына РАН, kbt@ccas.ru
Российская национальная библиотека, shorin@nlr.ru
Росковский государственный университет им. М.В. Ломоносова, kosolapov.ka@gmail.com

Абстракт

В Российской государственной библиотеке и Российской национальной библиотеке реализуется совместный проект, целью которого является публикация библиотечных данных библиотек, входящих в состав Национальной электронной библиотеки, в соответствии с принципами Linked Open Data. Реализация данного проекта позволит получить доступ к библиографической информации, хранящейся в ряде крупнейших библиотек России, в виде, пригодном для машинной обработки. Набор данных состоит из нескольких десятков миллионов записей. В процессе семантической интеграции решен ряд актуальных задач: библиографические записи собраны из различных библиотек, разработан алгоритм выявления дублетных записей и осуществлено их слияние, разработана онтология предметной области, произведена конвертация библиотечных данных из различных MARC-форматов в RDF с использованием разработанных XSLT-шаблонов, осуществлена публикация данных и предоставлена SPARQL точек доступа к ним.

Ключевые слова: Linked Open Data, семантическая паутина, связанные данные, библиографическая запись, Национальная электронная библиотека, слияние дублетов.

Введение

Министерство культуры Российской Федерации реализует проект создания Национальной электронной библиотеки (НЭБ). Основной целью создания НЭБ является обеспечение свободного, равного и всеобщего доступа граждан нашей страны к документ-

ной информации историко-культурного, научного и образовательного назначения через сеть Интернет, предоставляемой на основе единой общенациональной системы создания и эффективного использования цифровых библиотечно-информационных ресурсов и сервисов. В октябре 2015 года доступ к своим ресурсам предоставляют 6 федеральных, 29 региональных библиотек и 1 библиотека ВУЗа. Счетчик, расположенный на центральном портале НЭБ (http://нэб.рф) показывает, что читателям доступен каталог, содержащий более 33 миллионов записей.

Основной целью проекта семантической интеграции библиотечных данных, реализуемого совместно Российской национальной библиотекой, Российской государственной библиотекой и Вычислительным центром Российской академии наук, является публикация библиографических записей НЭБ с использованием принципов Linked Open Data (LOD) [1].

Постановка задачи

Для достижения этой цели необходимо решить следующие задачи:

- Осуществить сбор библиографических записей на центральный портал;
- Произвести конвертация записей в единый формат;
- Выявить дублетные записи и произвести их слияние;
- Создать онтологию предметной области и осуществить конвертацию записей согласно созданной онтологии;
- Решить вопрос о хранении сконвертированных данных
- Произвести выбор данных для связывания в LOD и осуществить публикацию библиотечных данных.

Реализация

В процессе работы были проанализированы протоколы для обмена библиографической информацией, реализованные в современных автоматизированных библиотечных информационных системах (АБИС). Была рассмотрена история развития протоколов Z39.50 и OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting). Протокол Z39.50 используется для перекрестного поиска информации в многочисленных источниках, а протокол OAI-

РМН - для автоматического сбора метаданных из различных АБИС и аккумуляции их на центральном сервере.

Поскольку структура создаваемой системы интеграции библиографических записей представляет из себя распределенную конфигурацию с выделенным центральным сервером, на котором агрегируются записи из различных источников, то наиболее подходящим протоколом для сбора информации является узкоспециализированный протокол, изначально предназначенный для решения именно этой задачи – OAI-PMH. Именно он и используется для агрегации библиографических записей из АБИС различных библиотек в НЭБ.

Для осуществления сбора библиографических записей из библиотек – участников НЭБ были созданы модули интеграции для автоматизированных библиотечных интегрированных систем, используемых в библиотеках: Aleph, Ирбис, MarcSQL и OPAC-Global.

Записи, собираемые из разных библиотек, поступают в двух форматах: MARC21/XML и RUSMARC/XML. На центральном сервере хранятся в формате MODS. Так как форматы MARC21/XML и MODS разрабатываются и продвигаются Библиотекой Конгресса США, то для преобразования одного формата в другой существуют XSLT-шаблоны. Для преобразования файлов из формата RUSMARC/XML в MODS был разработан XSLT-шаблон.

Для выявления дублетных записей в Национальной электронной библиотеке был реализован алгоритм, состоящий из 4 этапов. На первом этапе происходит нормализация библиографических записей:

- Записи конвертируются из формата MARCXML в MODS с использованием XSLT-шаблона.
- Последовательности пробельных символов заменяются на один пробел;
- Все буквы приводятся к нижнему регистру;
- Общепризнанные аббревиатуры и правила написания заменяются на единообразный формат.

Следующим шагом алгоритма является детерминированный поиск дублетов: если в разных записях совпадает ISBN (International Standard Book Number) – уникальный идентификатор

книги, то эти записи описывают один и тот же объект. ISBN представляет собой последовательность цифр, разделенных дефисом или пробелом. Удалив дефисы и пробелы из ISBN, получается целое число, поиск по которым в сбалансированном дереве можно осуществить со сложностью $O(\log n)$.

После детерминированного поиска дублетов происходит вероятностный поиск. Перед разбиением всех записей на кластеры, они упорядочиваются по полю «Год издания». В данном случае используется следующее свойство: если у записей поле «Год издания» не является пустым, и записи являются дублетами, то у этих записей «Год издания» должен совпадать. Таким образом, отобрав претендентов на дублетность, нам достаточно будет сравнить только те записи, в которых «Год издания» либо совпадает, либо отсутствует. Данная оптимизация значительно снижает количество операций сравнения на втором этапе работы алгоритма.

Для разбиения библиографических записей на кластеры используется функция хэширования SimHash, входящая в семейство locality-sensitive hashing (LSH) функций. Определение семейства таких функций дано в работе M.Charikar [2]. Если sim – функции подобия объектов из множества P: $\mathrm{sim}: P \times P \to [0,1]$, то схема LSH – это семейство хэш-функций H, имеющих распределение D, таких что при выборе функции $h \in H$ согласно распределению D:

$$\Pr_{h \in H}[h(p) = h(q)] = \mathrm{sim}(p,q), \forall p,q \in P$$

Основным свойством функций из данного семейства является то, что при незначительном изменении аргумента результат функции хэширования изменяется незначительно.

Шаги алгоритма вычисления хэша следующие:

1. Исходная строка разбивается на слова, между которыми стоят разделительные знаки (пробелы, знаки отступа, перенос строки и т.д.), в результате чего получается массив строк.
2. Создаётся массив целых чисел, заполненный нулями, с размером, равным длине хеша в битах.
3. Для каждого слова вычисляется хеш-функция в виде:
$$hash(s) = s[0] * 31^{n-1} + s[1] * 31^{n-2} + \cdots + s[n-1]$$
4. Для каждого бита, полученного хеша, соответствующий ему элемент массива изменяется следующим об-

разом: элемент массива увеличивается на единицу в случае, если исходный бит равен 1 и уменьшается на единицу в противном случае.

5. На основе полученного массива генерируется результат хеширования – последовательность бит (в данном случае 32 бита, представленные типом int). Используется следующее соответствие элементов массива и хеша: если элемент массива больше нуля, то соответствующий бит равняется единице, иначе нулю.

Поскольку количество записей велико, то сгенерировав хэши для всех записей, появляется проблема эффективного поиска среди этого набора, поскольку в худшем случае для поиска близких хэшей необходимо произвести 2^{32} операций сравнения.

Для снижения количества сравнений был использован метод разбиения хэша на равные части [4]. Сделав предположение о том, что различие двух близких хэшей будет локализировано в половине частей, можно добиться дополнительной оптимизации. Если выбирать хэши, в которых совпадает только $\frac{n}{2}$ из n частей, то при n равном 4 в худшем случае необходимо будет сделать $6 * 2^{16}$ операций сравнения. Таким образом, для хэша, имеющего вид

$$\mathrm{hash}_1 - \mathrm{hash}_2 - \mathrm{hash}_3 - \mathrm{hash}_4,$$

найдутся все записи, в которой первая и вторая части равны $\mathrm{hash}_1 - \mathrm{hash}_2$, первая и третья части равны $\mathrm{hash}_1 - \mathrm{hash}_3$ и т.д.

Используя вышеописанный алгоритм для поиска близких хешей для полей «Автор» и «Название», все исходное множество библиографических записей разбивается на кластеры, которые содержат претендентов на дублетность.

Для всех записей каждого кластера используется метод сравнения строк с использованием n-грамм, в нашем случае биграмм. Например, для словосочетания «синтаксис и семантика» биграммами (n-грамма для n равного 2) с точки зрения символов являются: «си», «ин», «нт», «та», «ак», «кс», «си», «ис», «с », « и», «и », « с», «се», «ем», «ма», «ан», «нт», «ти», «ик», «ка».

Разбив поля библиографических записей на биграммы, можно посчитать расстояние между двумя записями, используя различные метрики. В разработанной системе используется мера Жаккара, которая представляет собой соотношение количества

совпадающих элементов этих множеств к суммарному количеству элементов в этих множествах:

$$J = \frac{|A \cap B|}{|A \cup B|}$$

Для записей, претендующих на совпадение, считаются меры Жаккара для множеств биграмм, полученных из полей «Автор» и «Название», и берется среднее арифметическое. Если полученное значение выше порогового значения, то между этими записями устанавливается соответствие.

На заключительном этапе сравнения происходит проверка на исключения. В частности, разные части одной и той же статьи или книги могут иметь большое значение меры Жаккара, но описывать при этом различные объекты. Для предотвращения таких случаев алгоритм отдельно просматривает строки на наличие вхождений, обозначающих номера частей, например, «[1]», «Часть первая» и т.п.

Для записей, между которыми установлено соответствие, запускается процесс создания обогащенной записи. Обогащение осуществляется по свойствам формата MODS. Для библиографической записи в формате MODS существуют простые и составные свойства. Для составных свойств происходит объединение наборов данных из различных записей. Для простых свойств возможны две ситуации. В случае, когда свойство присутствует в одной записи и отсутствует в других, берется свойство из той записи, в которой оно присутствует.

В случае присутствия простого свойства в нескольких записях, используется подход, описанный Jeremy A. Hylton [3]. В соответствии с этим подходом, для каждого возможного написания свойства подсчитывается количество записей, в которых присутствует такое написание. В обогащенную запись попадает свойство, присутствующее в наибольшем количестве записей. Иногда различные написания присутствуют в одинаковом количестве записей. В таком случае в обогащенную запись попадает свойство, имеющее наибольшую длину, поскольку более длинное свойство содержит в себе больше информации.

Алгоритм выявления дублетных библиографических записей и создания обогащенных на их основе реализован на языке Java с использованием фреймворка Spring MVC, объектно-реляционного

отображения Hibernate и библиотеки XMLBeans. При реализации алгоритма было принято решение хранить на центральном сервере как обогащенные записи, так и первоначальные. Первоначальные библиографические записи могут пригодиться в случае поступления измененной, отредактированной записи из АБИС библиотеки. В этом случае для избежания коллизий гораздо проще заново создать обогащенную запись, нежели выявить разницу между первоначальной и измененной записью и применить эту разницу к обогащенной записи.

Процесс выявления дублетов и слияния записей исполняется на центральном сервере системы. Данные берутся из хранилища, в которое попадают записи, собранные из различных библиотек с использованием протокола OAI-PMH. В результате работы алгоритма обогащенные записи попадают в отдельную базу данных, также располагающуюся на центральном сервере.

Было проведено исследование наиболее распространенных онтологий в контексте возможного их использования для конвертации библиографических записей НЭБ. В процессе исследования выяснилось, что онтологии MODS/RDF, ЕНИП РАН и Europeana Data Model являются избыточными для того набора данных, который присутствует в НЭБ. Была предложена схема, состоящая из двух классов «Автор» и «Произведение», которая основана на использовании схем Dublin Core и FOAF. Использование разработанной схемы обеспечивает совместимость с огромным количеством данных, опубликованных в LOD. Для преобразования данных из формата MODS в RDF/XML, соответствующий разработанной онтологии, был реализован XSLT-шаблон преобразования.

Был разработан алгоритм связывания библиографических записей НЭБ с данными Библиотеки Конгресса США, Британской национальной библиотеки и DBpedia. Для каждого из трех сайтов на языке Java были созданы модули, каждый из которых осуществлял поиск по заданному выражению и возвращал найденные результаты. Поскольку большая часть данных из Библиотеки конгресса США и Британской национальной библиотеки представлена на английском языке, а в НЭБ преобладают данные на русском языке, то для повышения вероятности успешного поиска имена авторов и названия произведений предварительно переводились на английский язык с помощью сервиса Google Translate. Для каждо-

го источника в качестве результатов поиска модуль возвращал множество претендентов на установление связей owl:sameAs и rdfs:seeAlso. Для отсечения заведомо неподходящих кандидатов использовался метод биграмм, который применялся нами для выявления дублетных библиографических записей.

Исходное поисковое словосочетание и полученный результат проходят предварительную обработку, в рамках которой из выражений удаляются излишние пробельные символы, знаки пунктуации, слова приводятся в нижний регистр. После этого сравниваемые строки разбиваются на множества биграмм. Расстояние между строками вычисляется с использованием меры Жаккара, которая представляет собой соотношение количества совпадающих элементов множеств биграмм к суммарному количеству элементов в этих множествах:

$$J = \frac{|A \cap B|}{|A \cup B|}$$

Если полученное значение меры оказывалось выше порогового значения

$$T_1 = 0{,}95,$$

то в RDF/XML файл добавлялся RDF-триплет со связью owl:sameAs между объектами. Если полученное значение меры оказывалось меньше порогового значения T_1 и выше порогового значения

$$T_2 = 0{,}7,$$

то в RDF/XML файл добавлялся RDF-триплет со связью rdfs:seeAlso между объектами. Пороговые значения T_1 и T_2 были установлены экспериментальным путем.

Среди всего массива библиографических записей Национальной электронной библиотеки количество уникальных значений «Автор» и «Название» меньше, чем количество записей. Количество уникальных значений поля «Автор» составляет 19,6% от общего числа записей. Для поля «Название» данное значение составляет 68,6%. Для избежания повторных запросов к сервису Google Translate, сайтам Библиотеки конгресса США, Британской национальной библиотеки и DBpedia алгоритм поиска подобных записей сохраняет в базу данных значение исходного поля, результаты своей работы, а также время, когда данная проверка была сделана.

Таким образом, осуществляется кэширование результатов запросов к онлайн-сервисам.

Для каждого нового поля алгоритм сначала ищет значение этого поля в базе данных. В случае присутствия данного значения в базе данных, берется уже готовый результат работы алгоритма. Если же значение отсутствует в базе данных, то осуществляется поиск с использованием вышеописанного алгоритма. Подобная оптимизация алгоритма позволяет уменьшить количество запросов к онлайн-сервисам на 55,6%.

С помощью разработанной системы связывания данных в автоматическом режиме было осуществлено связывание 33,8% RDF/XML-файлов. До начала работы алгоритма в каждом RDF/XML-файле содержалось в среднем по 4,46 RDF-триплета. После осуществления связывания данных среднее количество RDF-триплетов в RDF/XML-файлах с проставленными связями возросло до 15,16 RDF-триплетов. Распределение среднего количества RDF-триплетов связи в одном RDF/XML-файле в зависимости от ресурса и типа связи приведены в таблице 1.

	owl:sameAs	rdfs:seeAlso
Библиотека конгресса США	4,04	2,51
Британская национальная библиотека	0,25	2,61
DBpedia	0,51	0,73

Таблица 1: Среднее количество RDF-триплетов связи в одном RDF/XML-файле

Был произведен анализ существующих RDF-хранилищ для наиболее оптимального решения предоставления доступа к RDF-триплетам с использованием протоколов SPARQL и HTTP. Рассмотрены системы 4store, Sesame и интегрированная среда Jena. В результате сравнения этих систем сделан выбор в пользу Jena [5].

Для загрузки RDF-триплетов в хранилище TDB, входящий в состав Jena, на языке Java был написан специальный модуль. Данный модуль использует Application Programming Interface (API) для языка Java, имеющийся в Jena.

Доступ к данным по протоколу SPARQL реализован с помощью сервера Fuseki, входящим в состав Jena. Для поддержки протокола HTTP был развернут и настроен веб-сервер с использованием Play Framework. Для интеграции с Fuseki используется механизм SOH – SPARQL Over HTTP, который представляет собой набор специализированных утилит – скриптов, входящих в состав Jena. В качестве результата обращения к веб-серверу по URI является RDF-файл, сериализованный с использованием Turtle. На рисунке 1 показана архитектура системы интеграции данных.

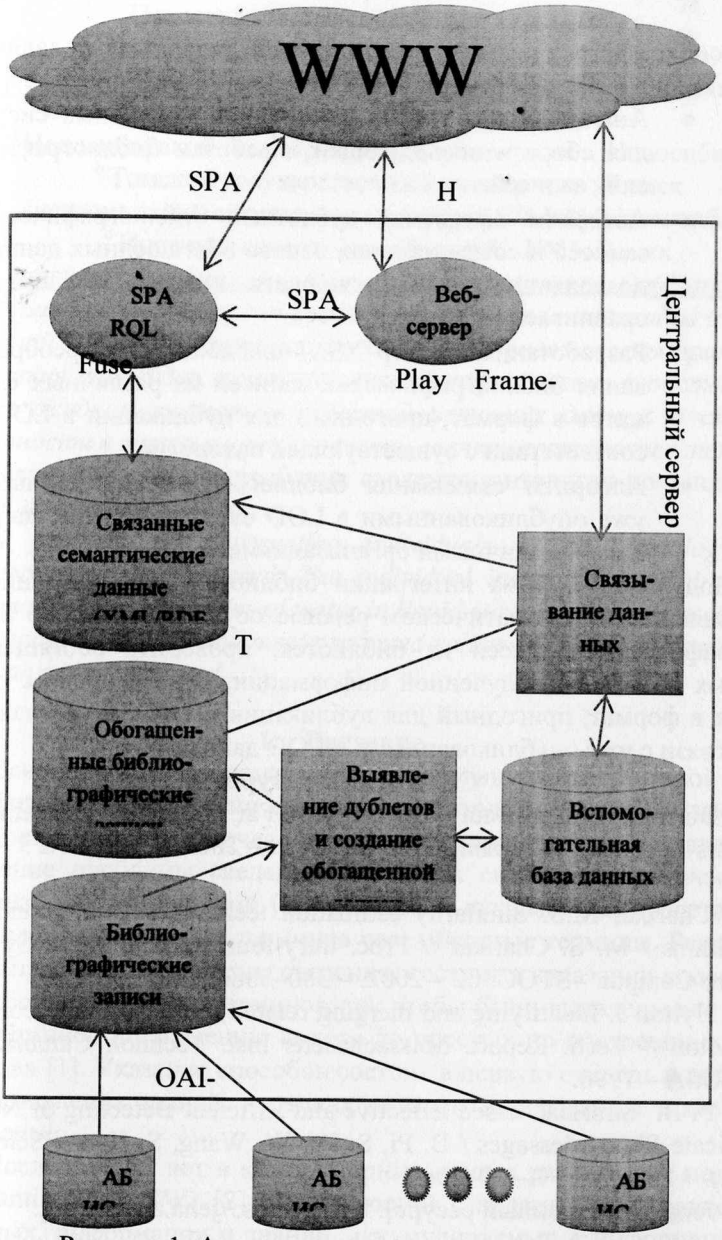

Рисунок 1: Архитектура системы интеграции данных

Результаты

Основными научными и практическими результатами являются:

- Аналитическая модель построения модульных систем для сбора и последующей обработки библиографических записей.
- Алгоритм выявления дублетных библиографических записей и создания на их основе обогащенных данных, позволяющий минимизировать количество попарно сравниваемых записей.
- Разработанный набор XSLT-шаблонов для преобразования библиографических записей из различных форматов в формат, пригодный для публикации в LOD, в соответствии с существующей онтологией.
- Алгоритм связывания библиографических данных с уже опубликованными в LOD сведениями, поставляемыми различными организациями.

Модульная система интеграции библиографических данных, позволяющая в автоматическом режиме осуществлять сбор библиографических записей из библиотек, проводить обогащение данных на основе полученной информации, конвертировать сведения в формат, пригодный для публикации в LOD, и устанавливать связи с уже опубликованными в LOD данными.

Список литературы

1. Bizer C. Linked Data - The Story So Far / C. Bizer, T. Heath, T. Berners-Lee // Int. J. Semant. Web Inf. Syst. – 2009. – Т. 5 – № 3 – 1–22с.

2. Charikar M.S. Similarity estimation techniques from rounding algorithms / M. S. Charikar // Proc. thiry-fourth Annu. ACM Symp. Theory Comput. - STOC '02 – 2002. – 380–388с.

3. Hylton J. Identifying and merging related bibliographic records / J. Hylton // Tech. Report. Massachusetts Inst. Technol. Cambridge, MA, USA – 1996.

4. Pi B. SimHash-based Effective and Efficient Detecting of Near-Duplicate Short Messages / B. Pi, S. Fu, W. Wang, S. Han // Science (80-.). – 2009. – Т. 7 – 20–25с.

5. Jena [Электронный ресурс]. URL: https://jena.apache.org.