АЛГОРИТМ СЛИЯНИЯ ДУБЛЕТНЫХ БИБЛИОГРАФИЧЕСКИХ ЗАПИСЕЙ

К.А. Косолапов[1], В.А. Серебряков[2], К.Б. Теймуразов[2], О.Н. Шорин[3]

1 Московский государственный университет им. М.В. Ломоносова
2 ФГБУН Вычислительный центр им. А.А. Дородницына Российской академии наук
3 ФГБУ «Российская национальная библиотека»

Абстракт

В ходе реализации системы публикации библиографических записей с использованием технологий семантической паутины возникла задача создания эффективного алгоритма по поиску нечетких дубликатов среди большого количества записей. В работе рассматривается использование функции из семейства locality-sensitive hashing с дополнительными оптимизациями для выявления дублетных библиографических записей. Выявленные таким образом записи сравниваются друг с другом с помощью меры Жаккара для принятия окончательного решения о дублетности и осуществления слияния.

Введение

В Российской государственной библиотеке и Российской национальной библиотеке реализуется проект по публикации библиографических записей с использованием технологий семантической паутины. Основной целью проекта является создание программной системы, которая бы в автоматическом режиме собирала библиографические записи из различных библиотек, осуществляла бы связывание данных и публиковала бы их в Linked Open Data. Наличие открытого доступа к крупнейшему массиву данных, ориентированному для использования не только человеком, но и автоматизированными средствами, позволит создать новые высокоинтеллектуальные онлайн-сервисы, которые окажут значительное влияние на развитие культуры и книжной отрасли.

Описываемая система создается как часть Национальной электронной библиотеки (НЭБ), которая обеспечит свободный, равный и всеобщий доступ граждан нашей страны к документной информации историко-культурного, научного и образовательного назначения через сеть Интернет, предоставляемый на основе единой общенациональной системы создания и эффективного использования цифровых библиотечно-информационных ресурсов и сервисов. Основным отличием от проектов подобного рода, например, сводного каталога электронных ресурсов (СКЭР) является наличие единого интерфейса для работы с документами из различных библиотек. Предыдущие проекты предоставляли только ссылки на электронные ресурсы, для получения доступа к которым необходимо было регистрироваться в каждой

конкретной библиотеке, устанавливать и настраивать специализированное программное обеспечение, применяемое в данной библиотеке.

Для создания модульной системы публикации данных, способной без значительных усилий подключать новых участников, работы были разбиты на несколько этапов [2]:

1) Разработка онтологии предметной области на базе существующих решений.
2) Осуществление интеграции с автоматизированными библиотечными информационными системами.
3) Осуществление конвертации библиографических записей из форматов MARC21 и Rusmarc в унифицированный формат - MODS.
4) Решение вопроса о хранении сконвертированных данных.
5) Выявление дублетных записей, полученных из различных библиотек, и осуществление взаимного обогащения данных из этих записей.
6) Выбор данных для связывания и публикации данных в Linked Open Data.
7) Реализация модуля визуализации полученного результата.

Данная система имеет распределенную структуру: библиотеки выступают поставщиками библиографических записей, которые аккумулируются, хранятся и обрабатываются на центральном сервере. В мире существует ряд проектов, направленных на интеграцию библиотечных данных и использующих схему аккумуляции данных на центральном сервере из распределенных источников. Среди них можно выделить такие проекты как сводный каталог библиотек России (СКБР), объединенный мировой каталог WorldCat, европейская цифровая библиотека Europeana. Тщательный анализ используемых в этих проектах решений является залогом успешного функционирования сервиса по семантической интеграции библиографических записей.

Постановка задачи

Выполняя свои уставные функции, библиотеки создают библиографические записи на экземпляры, хранящиеся в их фондах. Общее количество записей, хранящихся только в двух крупнейших библиотеках страны – РГБ и РНБ, составляет несколько десятков миллионов. Поскольку обе эти библиотеки являются получателями обязательных экземпляров, то большая часть их фондов совпадает. Как следствие, в библиотечных системах хранятся записи на одни и те же произведения – дублетные библиографические записи. Важно отметить, что в РНБ для описания книг используется формат, разрабатываемый и поддерживаемый Национальной службой развития системы форматов – Rusmarc, а в РГБ используется американский формат описания произведений – MARC21. Нельзя не упомянуть, что в этих библиотеках применяются различные стандарты на полноту заполнения описания – какие поля являются обязательными, а какие – дополнительными, а также на строение отдельных полей.

Стоит подчеркнуть, что сводный каталог библиотек России (СКБР) является частью каталога НЭБ. Также в каталог НЭБ попадают записи из региональных библиотек, которые зачастую являются участниками СКБР. Таким образом, можно утверждать, что количество дублетных библиографических записей, попавших в каталог НЭБ, велико.

Для выявления дублетных библиографических записей, полученных из разных источников и заполненных с использованием различных стандартов, требуется создание эффективного алгоритма. При получении нескольких десятков миллионов записей из нескольких библиотек задача сравнения всех записей со всеми для выявления дублетных записей на одни и те же книги становится неприемлемо затратной – $O(n^2)$. Возникает необходимость создания алгоритма, позволяющего существенным образом сузить множество библиографических записей – потенциальных кандидатов на дублетность. Большинство подобных алгоритмов основано на разбиении всего множества данных на кластеры, внутри которых содержатся подобные друг другу записи. Необходимо проанализировать существующие алгоритмы выявления записей на один и тот же объект с учетом присутствия ошибок и аббревиатур, различий стандартов заполнения полей.

Обзор существующих решений

Априорно известно, что при получении библиографических записей из разных библиотек, часто будет возникать ситуация, когда на одну и ту же сущность будет существовать несколько записей. Эти записи будут отличаться как по формату, так и по полноте заполнения, поскольку в различных учреждениях процессы каталогизации отличаются друг от друга. Например, состав дополнительных элементов, точек доступа может быть разным, системы классификации и предметизации, использование аббревиатур также могут разниться от одной организации к другой. К тому же, запись может содержать ошибки, опечатки, вызванные банальной человеческой оплошностью и невнимательностью.

Вопросы интеграции библиографических записей из различных источников с последующим их объединением и обогащением давно находятся в фокусе внимания ученых. Одними из основоположников этого направления являются Ivan P. Feleggi и Alan B. Sunter [4], которые разработали математическую модель, позволяющую разделить множество записей на несколько кластеров. В кластер попадают записи, которые в терминах той или иной метрики располагаются недалеко друг от друга. Для выявления дублетных записей достаточно сравнить записи, входящие в состав одного кластера, что значительно снижает количество сравнений.

Jeremy A. Hylton [5] развил идеи Feleggi и Sunter, распространив их на проблему не только выявления дублетных записей, но и создания на их основе обогащенной записи, содержащей объединенную информацию из нескольких записей, с последующим удалением дублетных записей, содержащих менее

полную информацию. В его работе показано, что процесс разбиения записей на кластеры должен предваряться процедурой нормализации – набором правил, применение которых приводит библиографические записи к некоему единообразному виду. К таким правилам нормализации можно отнести удаление избыточных пробельных символов, приведение строк к одному регистру, замена общепризнанных аббревиатур и правил написания на единообразный. Например, даты, записанные в разных форматах: Sept. 1987, 09.1987 – заменялись на какой-то один формат.

Поскольку зачастую дополнительная информация, указанная в записи в виде ссылок, может быть недоступна, например, получить полный текст произведения по ссылке из описания невозможно из-за ограничений, накладываемых авторским правом, то процесс выявления дублетных записей может ориентироваться только на информацию, содержащейся в самой записи. Таким образом, целый класс алгоритмов, использующих дополнительную информацию, не может быть нами использован. Мы можем использовать только основные поля записи – «Автор» и «Название», т.е. по сути оперировать строками.

Одними из наиболее распространенных метрик для расчета расстояния между строками являются меры Хемминга, Евклида, Левенштейна, Джаро-Винклера, Рэтклиффа-Обершелпа. В работе Д.Н.Рубцова и В.К.Барахнина [1] показано, что для простейшего случая, когда сравнение записей осуществляется только по полям «Автор» и «Название», достаточно использование одного из методов динамического программирования, предложенного Хиршбергом. Данный метод обладает высокой эффективностью и относительно простой реализацией.

Также в вышеупомянутой работе описан ряд исключений, когда несколько записей при формальной практически полной идентичности содержат информацию о различных объектах. В частности, к таким исключениям можно отнести описания на отдельные тома многотомных изданий.

Выбор данного решения был обусловлен тем, что количество библиографических записей в используемой системе не столь велико. В нашем же случае количество библиографических записей составляет несколько десятков миллионов, что делает невозможным оперирование строками, так как сложность описанного алгоритма составляет O(mn), где m – максимальная длина строки, а n – число уже имеющихся записей.

Алгоритм слияния дублетных библиографических записей

В отличие от строк, поиск и сравнение среди большого количества натуральных чисел можно организовать более эффективно. Достаточно разместить их в отсортированном массиве. Тогда, используя алгоритм двоичного поиска, можно найти необходимое число за O(logn) операций сравнения.

Для перевода строковых значений в числовые используются функции хэширования. Однако, большинство алгоритмов хэширования не подходят для нашей задачи, так как изначально они создавались для равномерного отображения пространства строк на пространство чисел. Это означает, что при незначительном изменении аргумента функции хеширования – строки, значение функции – число может измениться разительно. В нашем же случае необходимо добиться обратного результата – незначительное изменение аргумента должно приводить к незначительному изменению результата функции хэширования.

Подобные алгоритмы хэширования составляют семейство locality-sensitive hashing (LSH) функций. Одно из определений семейства таких функций дано в работе P.Indyk и R.Motwani [6]. Семейство F функций $h: M \to S$ называется (R, cR, P_1, P_2)-чувствительным, если для порога $R>0$, коэффициента $c>1$ и любых двух точек $p,q \in M$ выполнены два условия:

1. Если расстояние $d(p, q) \leq R$, то $h(p)=h(q)$ с вероятностью не меньше P_1.
2. Если расстояние $d(p, q) \geq cR$, то $h(p)=h(q)$ с вероятностью не более P_2.

Альтернативное определение семейства таких функций дано в работе M.Charikar [3]: для sim – функции подобия объектов из множества P: $sim: PxP \to [0,1]$, то схема LSH – это семейство хэш-функций H, имеющих распределение D, таких что при выборе функции $h \in H$ согласно распределению D:

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

Одним из наиболее часто используемых представителей из семейства LSH-функций является функция SimHash. Шаги алгоритма, реализующего данную функцию, 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:
   $hash(s)=s[0]*31^{n-1}+s[1]*31^{n-2}+\ldots+s[n-1]$
4. Для каждого бита, полученного хеша, соответствующий ему элемент массива изменяется следующим образом: 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.

Как можно заметить, данная схема хеширования не зависит от порядка слов в исходном наборе данных, например, «синтаксис и семантика» и «семантика и синтаксис» будут иметь одинаковые хеши.

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, описанный в работе Bingfeng Pi, Shunkai Fu, Weilei Wang и Song Han [7]. В данной работе предлагается делить полученный хэш hash на равные части $hash_1$, $hash_2$, …, $hash_n$. Тогда поиск близких хэшей можно осуществить с некоторой оптимизацией: среди уже имеющихся хешей находятся такие, в которых совпадает хотя бы одна из частей. При разбиения хэша на 4 равные части подобная оптимизация позволяет найти близкие хэша в худшем случае за $4*2^{24}$ операции сравнения.

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 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 $hash_1$-$hash_2$-$hash_3$-$hash_4$, найдутся все записи, в которой первая и вторая части равны $hash_1$-$hash_2$, первая и третья части равны $hash_1$-$hash_3$ и т.д.

Таким образом, используя вышеописанный алгоритм для поиска близких хешей для полей «Автор» и «Название», мы отберем множество записей – претендентов на дублетность. Для каждого претендента мы осуществляем сравнение строк с использованием меры Жаккара: $J(A, B)=|A \cap B|/|A \cup B|$ - соотношение количества совпадающих элементов к объединению. Возникает вопрос о том, что взять за единичный элемент, из которого состоят строки. На первый взгляд кажется, что наиболее логичным решением является разделение на отдельные слова.

Однако в случае, если слова в исходных строках будут близки, но не идентичны, например, «воздушный поток» и «возд. поток», то близость слов «воздушный» и «возд.» будет утеряна. Чтобы это предотвратить, исходной строке ставится в соответствие неупорядоченный набор признаков. Существует несколько подходов для создания таких признаков. Самым распространенным является поиск n-грамм.

N-грамма – последовательность n символов или слов. Например, для словосочетания «синтаксис и семантика» биграммами (n-грамма для n равного 2) с точки зрения слов являются «синтаксис и», «и семантика». С точки зрения символов все биграммы этой строки это: «си», «ин», «нт», «та», «ак», «кс», «си», «ис», «с », « и», «и », « с», «се», «ем», «ма», «ан», «нт», «ти», «ик», «ка». За счет того, что в исходной задаче требуется идентификация небольших строк, располагающихся в полях библиографических записей, имеет смысл использовать подход, трактующий биграммы с точки зрения символов,

поскольку на небольших по объему данных он генерирует наборы большого размера.

Для записей, претендующих на совпадение, посчитаем меры Жаккара для множеств биграмм, полученных из полей «Автор» и «Название», и возьмем среднее арифметическое. Если полученное значение будет выше заданной планки, а указанные записи не подпадают под исключения, описанные в работе Д.Н.Рубцова и В.К.Барахнина [1], то установим соответствие между этими записями.

Текущая реализация алгоритма выявления дублетных библиографических записей обрабатывает 10 тыс. новых записей на хранящихся в РГБ 15 млн. записях за 34 секунды. В РНБ хранится около 10 млн. библиографических записей. Таким образом, выявление всех дублетных записей РНБ и РГБ займет 34 тыс. секунд, т.е. около девяти с половиной часов. В процессе своей работы текущая реализация алгоритма создает дубликаты всех оригинальных записей, а также хранит таблицы хэшей для авторов и названий. Для хранения хэшей 25 млн. записей требуется 32*2*25 млн. бит, что приблизительно равняется 190 Мб. Место, необходимое для хранения оригинальных записей, напрямую зависит от количества найденных дублетов в массивах РГБ и РНБ. По оценкам экспертов около 70% записей в РНБ и РГБ являются дублетами. Таким образом, количество оригинальных записей составляет 18 млн. Т.к. в среднем библиографическая запись занимает 1 Кб, то для хранения оригинальных записей потребуется около 17 Гб.

Для совпавших записей мы производим операцию слияния. Слияние 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 другой, мы берем свойство из той записи, в которой оно присутствует. В случае присутствия простого свойства в обеих записях, мы в обогащенную запись берем свойство из более старой записи.

Заключение

Для выявления дублетных библиографических записей на множестве, имеющим большую мощность, недостаточно алгоритмов, оперирующими функциями сравнения строк – содержимым полей. Если провести предварительную конвертацию строк в числа с использованием функции из семейства LSH, дополнив алгоритм поиска по получившимся значениям оптимизацией с использованием разбиения хэша на части, можно существенно сократить количество операций сравнения. Выявив таким образом претендентов на дублетные библиографические записи, достаточно сравнить их с использованием меры Жаккара на множестве биграмм, полученных из строк полей «Автор» и «Название», для установления дублетности. Полученный

алгоритм является оптимизацией известных алгоритмов поиска нечетких дубликатов для случаи большого количества исходных данных. Созданная на его основе программная система обладает высокой производительностью. Последующие исследования могут быть направлены на создание масштабируемого алгоритма, способного разделять библиографические записи на несколько потоков и параллельно их обрабатывать.

ЛИТЕРАТУРА

1. Рубцов Д.Н. Выявление дубликатов в разнородных библиографических источниках / Д. Н. Рубцов, В. Б. Барахнин // Вестн. НГУ. Сер. Информ. технологии – 2009. – Т. 7 – № 3 – 86–93с.
2. Серебряков В.А. Проблемы семантической интеграции библиотечных данных / В. А. Серебряков, О. Н. Шорин // Библиотековедение – 2014. – № 5 – 41–47с.
3. 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.
4. Fellegi I.P. A Theory for Record Linkage / I. P. Fellegi, A. B. Sunter // J. Am. Stat. Assoc. – 1969. – Т. 64 – № 328 – 1183–1210с.
5. Hylton J. Identifying and merging related bibliographic records / J. Hylton // Tech. Report. Massachusetts Inst. Technol. Cambridge, MA, USA – 1996.
6. Indyk P. Approximate Nearest Neighbors: towards removing the curse of dimensionality (prelim) , 1999. – 27с.
7. 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.