Социално търсене в библиотечни бази от данни чрез използване на системата REDIS

Стефка Добрева

Social search in library database by using of sorted sets in Redis: *The publication presents an algorithm for the practical realization of the method of social search for information based on consumer behavior of library information system. Collection and processing of information about consumer behavior through the use of sorted sets in NoSQL Redis are considered as means to achieve the objective. Subsequently, the system will offer tips, which can greatly facilitate the users to find bibliographic information.*

Key words: *Social Search, Redis, Libraries, Information Systems, Bibliographic Information.*

ВЪВЕДЕНИЕ

В библиотечните информационни системи при търсене на библиографска информация всеки потребител започва търсенето чрез въвеждане на ключови думи. Постепенно чрез добавяне или премахване на ключови думи се достига до желания резултат - списък с информационни източници в конкретна предметна област. Този процес се повтаря при всеки потребител на библиографска информация, при което различни потребители, търсещи информация за една обща тема, преминават поотделно през всички етапи на търсене по метода "проба-грешка" без да имат възможност да се възползват от миналия опит на други потребители на библиотечната информационна система.

Възможно е процеса на търсене да се ускори чрез използването на метод, базиран на работата на социалните мрежи. При търсене на информационни източници в библиотечната система, за всеки потребител при запазване на анонимността може да се съхраняват данни за неговото поведение - ключовите думи, които са използвани като критерии при търсенето. В последствие тези данни, разглеждани като скрит модел на Марков чрез динамична мрежа на Бейс, могат да послужат автоматично да се дават препоръки на други потребители за това какви ключови думи да използват във връзка с конкретно тяхно библиографско проучване.

Използването на NoSQL системата Redis може значително са съкрати разработката на подобни системи, тъй като вградената поддръжка на различни абстрактни структури от данни премахва необходимостта от самостоятелното им реализиране и дава възможност разработката да се извърши на качествено ново ниво.

МЕТОДИ ЗА ПОДПОМАГАНЕ НА ТЪРСЕНЕТО

Информационните системи за търсене на библиографска информация могат да използват следните методи, подпомагащи търсенето:

- Методи, базирани на експертен подход. Екип от специалисти определя ключови думи, връзката между тях и ги ранжира по степен на значимост. Предимства - висока релевантност между ключови думи и желан резултат. Недостатъци - бавно изпълнение, трудно обхващане на всички предметни области и висока цена.

- Методи, базирани на текст майнинг. По програмен път, с използване на методите на изкуствения интелект, се извлича информация от голям брой информационни източници. Предимства - процеса се автоматизира, възможно е да се прилагат различни алгоритми. Недостатъци - необходима е голяма база от книги в електронен формат, в идеалния случай - целия информационен ресурс на библиотеката; необходими са скъпо платени ИТ специалисти за поддържането на базата от данни с електронни книги, разработка и поддържане на цялата система.

- Методи, базирани на социалния подход. Изследване на поведението на

множество потребители по отношение на това какви ключови думи използват при търсенето на информационни източници. Предимства - сравнително ниска цена, тъй като се елиминират скъпоструващите компоненти - експертен труд и големи бази от данни с електронни книги. Недостатъци - възможно е умишлено заблуждаване на системата от недоброжелателни потребители, популярно наричани "тролове".

СОЦИАЛНО ТЪРСЕНЕ ЧРЕЗ ИЗПОЛЗВАНЕ НА СИСТЕМАТА REDIS

Както казахме по-горе, методът на социалното търсене се базира на опита и предпочитанията на множество потребители в предходни периоди от време. Използвайки определени зависимости в поведението им, търсещият информация по-бързо може да се ориентира и да избере правилните ключови думи, които да използва при търсенето в библиографската база от данни и по-бързо да достигне до нужните информационни източници.

В основата на алгоритъма лежи натрупване на данни за поведението на множество потребители и изграждане на динамична мрежа на Бейс, която тук ще представим като матрица на вероятностите.

Изходната постановка е следната: потребител на библиотечната информационна система се интересува от информационни източници в конкретно определена предметна област. Първоначално системата му задава сесиен идентификатор с цел проследяване на неговото поведение при работа с нея. Сесийният идентификатор е уникален и случайно генериран с цел запазване анонимността на потребителя. При уеббазирани системи сесийният идентификатор е възможно е да се предава чрез т.нар. "бисквитки" или по друг начин, който да асоциира потребителя с всяка заявка, получавана от системата.

Потребителят въвежда определени ключови думи за търсене, в резултат на което се извежда списък с резултат от търсенето. Тъй като в повечето случаи списъкът е много голям, се налага потребителят да прецизира ключовите думи, които задава като критерии при търсенето. Този цикъл се повтаря, докато потребителя не получи желан от него резултат. Поредицата въвеждани ключови думи във времето от един потребител представлява верига на Марков и осигурява натрупването на данни за логически свързани помежду си ключови думи, които са избрани на база индивидуални субективни предпочитания. Разбира се, потребителят може по погрешка или умишлено да въвежда ключови думи, несвързани с интересуващата го предметна област, но натрупването на данни за поведението на много на брой потребители, ще елиминира значението на тези нежелани флуктуации, които представляват отклонение от нормалното рационално поведение.

В най-неблагоприятния сценарий, когато поведението на по-голяма част от потребителите е ирационално, много трудно ще могат да се уловят тенденции в тяхното поведение при търсене на информация в определена предметна област. При подобно асоциално поведение на потребителите метода за социално търсене не би дал добри резултати.

Данните за ключовите думи и фрази се записват в NoSQL системата Redis [3] в структура множество по следния начин:

```
SADD SESS_ID1 "word1"
SADD SESS_ID1 "word3" "word1"
...
SADD SESS_ID1 "wordX"
```

Записването на ключови думи и фрази като низове в множество в Redis автоматично премахва дублиранията и например "word1" ще се среща само веднъж в множеството, отговарящо на ключа SESS_ID1. Премахването на дублиранията се прави с цел да не може един потребител да окаже влияние върху значението и важността на едни ключови думи и фрази за сметка на други, тъй като това в

последствие би изкривило списъка с предложения в полза на тези дублиращи се ключови думи и фрази. Премахването на дублиранията е една от основните разлики спрямо популярните модели "Bag-of-words" [2] и "Term frequency–inverse document frequency" [4]. В резултат на натрупването на данни за поведението на потребителите, на всеки сесиен идентификатор ще съответства множество с различен брой елементи. Например, нека да имаме следните примерни данни, представени във формат JSON [5]:

{
SESS_ID1: [word1, word3, word4, word6],
SESS_ID2: [word2, word4, word5],
SESS_ID3: [word1, word6],
SESS_ID4: [word1, word3, word5, word6],
SESS_ID5: [word4, word6]
}

, където:

SESS_ID - случайно генериран сесиен идентификатор на потребител търсещ информация в определена предметна област;

word - уникален низ в множеството, асоциирано с определен сесиен идентификатор, представляващ ключова дума или фраза (комбинация от ключови думи, които да се срещат в точно определен ред). Множеството е представено като масив, защото в JSON липсва структура множество. За улеснение сме представили елементите в масива сортирано, въпреки че в множеството в Redis няма определен ред.

Следващият етап от обработката е преобразуване на тези данни в хеш от сортирани множества в Redis, което на практика представлява алгоритмична реализация на Теоремата на Бейс (1) за изчисляване на вероятността за настъпване на дадено събитие, след като вече е известна част от информацията за него [1].

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \quad (1)$$

От множеството с ключови думи и фрази на един потребител, всеки елемент става ключ, а в съответното му сортирано множество се добавят останалите елементи, като тежестта им се увеличава с единица. Ако елементите за пръв път се добавят в сортираното множество на Redis, то тежестта им е единица. Елементът, който е ключ, не се добавя в сортирания списък. При преобразуването сесийните идентификатори отпадат. Процесът се повтаря последователно за всеки елемент от множеството, отговарящо на един сесиен идентификатор.

За илюстрация на алгоритъма ще го изпълним няколко пъти. За елементите на първия сесиен идентификатор SESS_ID1 ще имаме следните команди на Redis:

ZINCRBY word1 1 "word3"
ZINCRBY word1 1 "word4"
ZINCRBY word1 1 "word6"
ZINCRBY word3 1 "word1"
ZINCRBY word3 1 "word4"
ZINCRBY word3 1 "word6"
ZINCRBY word4 1 "word1"
ZINCRBY word4 1 "word3"
ZINCRBY word4 1 "word6"
ZINCRBY word6 1 "word1"
ZINCRBY word6 1 "word3"
ZINCRBY word6 1 "word4"

В резултат структурата от данни в Redis представена във формат JSON би била следната:

{
"word1":{"word3":1, "word4":1, "word6":1},
"word3":{"word1":1, "word4":1, "word6":1},
"word4":{"word1":1, "word3":1, "word6":1},
"word6":{"word1":1, "word3":1, "word4":1}
}

За елементите на втория сесиен идентификатор SESS_ID2 ще имаме следните команди на Redis:

ZINCRBY word2 1 "word4"
ZINCRBY word2 1 "word5"
ZINCRBY word4 1 "word2"
ZINCRBY word4 1 "word5"
ZINCRBY word5 1 "word2"
ZINCRBY word5 1 "word4"

В резултат структурата от данни в Redis представена във формат JSON би била следната:

{
"word1":{"word3":1, "word4":1, "word6":1},
"word2":{"word4":1, "word5":1},
"word3":{"word1":1, "word4":1, "word6":1},
"word4":{"word1":1, **"word2":1**, "word3":1, **"word5":1**, "word6":1},
"word5":{"word2":1, "word4":1},
"word6":{"word1":1, "word3":1, "word4":1}
}

С удебелен шрифт сме маркирали новите моменти спрямо предходния междинен резултат.

За елементите на третия сесиен идентификатор SESS_ID3 ще имаме следните команди на Redis:

ZINCRBY word1 1 "word6"
ZINCRBY word6 1 "word1"

В резултат структурата от данни в Redis представена във формат JSON би била следната:

{
"word1":{"word3":1, "word4":1, **"word6":2**},
"word2":{"word4":1, "word5":1},
"word3":{"word1":1, "word4":1, "word6":1},
"word4":{"word1":1, "word2":1, "word3":1, "word5":1, "word6":1},
"word5":{"word2":1, "word4":1},
"word6":{**"word1":2**, "word3":1, "word4":1}
}
и т.н.

Процесът се повтаря, докато се изчерпят данните за всички сесийни идентификатори. Като краен резултат структурата от данни в Redis, представена във формат JSON, би била следната:

{
"word1":{"word3":2, "word4":1, "word5":1, "word6":3},
"word2":{"word4":1, "word5":1},
"word3":{"word1":2, "word4":1, "word5":1, "word6":2},
"word4":{"word1":1, "word2":1, "word3":1, "word5":1, "word6":2},
"word5":{"word1":1, "word2":1, "word3":1, "word4":1, "word6":1},
"word6":{"word1":3, "word3":2, "word4":2, "word5":1}
}

Теглото на елементите от сортираното множество (например на ключа word1, елемента word6 има тежест 3) показва колко пъти ключовата дума или фраза е била в едно и също множество заедно с ключа на сортираното множество на което принадлежи, т.е. от колко различни потребители е била използвана заедно с ключа. Теглата ще варират от единица до броя на сесийните идентификатори. Теглото ще е единица, ако само един потребител е използвал ключа и елемента в своите търсения, и ще е равно на броя на сесийните идентификатори, в случай че ключа и елемента са били използвани от всички потребители.

Така преобразувани данните в Redis сравнително лесно може да се използват за извеждане на препоръчителен списък с ключови думи и фрази, в зависимост от поведението на потребителя, търсещ библиографска информация.

Сценарият за работа е следния: потребител първоначално въвежда ключова дума word1. В резултат на изпълнение на командата на Redis

ZREVRANGEBYSCORE word1 -inf +inf

, се връща множество от ключови думи или фрази, подредени в низходящ ред по тегло - по-тясно свързаните с ключовата дума или фраза - по-напред в списъка:

word6 (тегло 3 - трима потребители са използвали и word1 и word6)
word3 (тегло 2 - двама потребители са използвали и word1 и word3)
word5 (тегло 1)
word4 (тегло 1)

Елементите с еднакви тегла (какъвто е случая с word4 и word5) се извеждат в обратен азбучен ред и за съжаление обратното не може да се зададе в Redis чрез опция на командата ZREVRANGEBYSCORE.

В резултат на изведените съвети потребителят решава да добави като критерий при търсенето и ключова дума word6. Извеждаме сечението от сортираните множества, асоциирани с word1 и word6 (2).

$$word1 \cap word6 = \{x \mid x \in word1 \wedge x \in word6\} \quad (2)$$

Със следните команди на Redis създаваме ново сортирано множество, сечение от двете сортирани множества и връщаме новото множество от ключови думи или фрази, подредени в низходящ ред по тегло:

ZINTERSTORE tmp_sess_idN 2 word1 word6
ZREVRANGEBYSCORE tmp_sess_idN -inf +inf
След като сортираното множество на word1 е било
"word1":{"word3":2, "word4":1, "word5":1, "word6":3}
, а това на word6 е било
"word6":{"word1":3, "word3":2, "word4":2, "word5":1}
, то новото сортирано множество ще е
"tmp_sess_idN":{"word3":4, "word4":3, "word5":2}
, при което изведените предложения към потребителя ще бъдат:
word3 (тегло 4)
word4 (тегло 3)
word5 (тегло 2)

При този начин на работа теглата на елементите, срещащи се едновременно и в двете сортирани множества, се сумират. Процесът би могъл да се продължи и да се извършат последващи сечения между новото сортирано множество и други ключови думи.

ЗАКЛЮЧЕНИЕ

Представената реализация на метода на социално търсене чрез използване на възможностите на NoSQL системата Redis може значително да облекчи потребителите на библиотечните информационни системи при търсенето на

библиографска информация. Използването на абстрактни структури от данни, като множество и сортирано множество на Redis, съществено опростява реализацията на подобни проекти, като същевременно крайното съхраняване на данните в структурата хеш от сортирани множества е оптимизирано за бързодействие по отношение на извеждане на съвети към потребителите. Въз основа на ключовите думи или фрази, използвани от потребителя, бързо и лесно може да се изведе списък с други подходящи ключови думи или фрази подредени по степен на релевантност.

ЛИТЕРАТУРА

[1] Каракулаков М., Р. Мирянов Теория на вероятностите и математическа статистика, Унив. изд. Наука и икономика, В., 2011, с. 136.

[2] Bag-of-words model, <http://en.wikipedia.org/wiki/Bag_of_words_model, 10.08.2014>

[3] REmote DIctionary Server - REDIS, <http://redis.io/, 10.08.2014>

[4] Term frequency–inverse document frequency, <http://en.wikipedia.org/wiki/Tf-idf, 10.08.2014>

[5] The JSON Data Interchange Format: Standart ECMA-404, 2013. <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf, 10.08.2014>

За контакти:

докторант Стефка Добрева, Катедра "Информационни системи и технологии", Университет по библиотекознание и информационни технологии, тел.: 0884 070 322, e-mail: stefka.dobreva@gmail.com

Докладът е рецензиран.