УЕБ БАЗИРАНА УСЛУГА ЗА АВТОМАТИЧНО ИЗВЛИЧАНЕ НА МОДЕЛ НА ПОТРЕБИТЕЛСКИ ИНТЕРЕСИ

Александър Чомаков, Иван Койчев, Александър Григоров

СУ „Св. Кл. Охридски”, Факултет по математика и информатика
sasho.chom@gmail.com, koychev@fmi.uni-sofia.bg, alexander_grigorov@fmi.uni-sofia.bg

***Резюме**: В настоящата статия се разглежда един подход за автоматизирано извличане на модел на потребителски интереси, реализиран като Уеб базирана услуга. Целта е изграждането на система, която да създава и предоставя на други системи модели на потребителите. Системата е изградена като услуга, която ще може да бъде използвана от различни външни системи. В статията се разглеждат основните методи на персонализация, съществуващите технологии за търсене на семантично сходство, функционалността на предлаганата услуга и нейната програмна реализация.*

***Ключови думи**: моделиране на потребителя, адаптивни системи, извличане на информация, уеб услуги*

1. Въведение

Всеки потребител, или група от потребители, има различни нужди и интереси и взаимодейства по различен начин с приложенията. Налага се приложенията да се персонализират и да удовлетворяват изискванията на всеки един потребител. Персонализацията изисква правилна интерпретация на съдържанието, извличане на семантично важна информация и мета данни, и филтриране на получените резултати според потребителските интереси.

Потребителите рядко полагат допълнителни усилия, които да дадат точна и подробна информация за тях (дали харесват нещо или не, избор между няколко обекта, списък с интересни обекти). В същото време очакват да получават информация, съответстваща на интересите им. Решението на този проблем е с помощта на различни техники, следящи и анализиращи поведението на потребителя, да се събира съществена за него информация, която впоследствие да бъде използвана за създаването на персонален модел на интересите му [5]. Този модел се използва за филтрирането на съдържание, която се подава или препоръчва на потребителя.

Персонализираните препоръки могат да бъдат разглеждани в 2 основни категории: базирани на съдържанието и социални (базирани на прилики между отделните потребители на системата). Вторите се правя по-лесно и се използваш широко в комерсиалните системи. Първите изискват значително по-сериозен анализ на съдържанието, което е харесал потребителя, с цел

извличане на явен списък с интересите му. До сега такива системи бяха създавани основно за конкретни системи. Настоящата статия представя един нов подход за моделирани на потребители интерес предоставяйки моделирането на потребителите като услуга. Това дава възможност, няколко приложения съвместно да събират и използват информация за потребителя.

Според [6] има 3 основни метода за събиране на информация при създаването на потребителски модели в зависимост от участието на потребителя в изграждането му:

- **Неявен (implicit)** – скрито (неявно) за потребителя събиране на информация за модела му, следейки посещаваните от него страници.
- **Явен (explicit)** – потребителите явно въвеждат детайлна информация за модела си. Изискването на такъв род информация от потребителите в повечето случаи се посреща с нежелание от тях.
- **Хибриден** – обединява двата гореописани метода. Неявно се събират данни за поведението на потребителя, но във всеки момента той има достъп до модела си и може директно да го промени. Този метод е „по-щадящ” потребителя, защото не изисква подробна информация от него, която да бъде акуратно променяна във времето. Вместо това потребителя трябва само явно да одобрява или не предложените му от системата интереси, които са неявно събрани и асоциирани за неговия профил. Този метод е приложен и при реализацията на услугата за моделиране на потребители.

Целта на нашите изследвания е изграждането на система, която да създава и предоставя на други системи модели на потребителите. Системата е изградена като услуга, която ще може да бъде използвана от различни външни системи. Всяка външна система ще има собствен идентификатор и ще може да управлява собствена колекция от потребители (и моделите им). Така услугата ще може да се ползва от различни приложения в зависимост от техните конкретни предназначения. Идеята за изграждането на тази услуга се породи от нуждите на проекта „Умна книга” [1], който акцентира върху създаването на среда, която максимално да улеснява читателя в търсенето на полезната за него информация. Предлага подобрени възможност за търсене в текста, за социално четене, семантично обогатяване и извличане на обобщена информация за книги. В една такава среда, за да може да бъде предоставена информация, съответстваща на нуждите и очакванията на всеки потребител, е необходимо създаването на персонализирани модели за тях.

Чрез услугата за моделиране на потребители се реализира метод за препоръки на документи, базиран на съдържанието им. Услугата е до известна степен универсална, тъй като може да се използва от различни системи през стандартен интерфейс. Не е от значение дали ще се използва за книги,

новини, статии, музикални произведения. Правилно извлеченото съдържание на съответния (наречен в общия случай) документ, се анализира за термини. Така може да бъде открита връзка с други документи.

2. Съществуващите технологии за търсене на семантично сходство

Системите, предлагащи огромни хранилища от информация в определена приложна област (филми, музика, книги и т.н.) предлагат определени възможности за търсене и дават възможност на потребителите да открият търсеното от тях съдържание. Все по-често обаче потребителите очакват да открият търсеното от тях с минимални усилия и бързо губят интерес ако не успеят. Използването на концептуално нов подход към потребителите е наложително за всяка една система, която иска да задържи потребителите си. За тази цел подобна система трябва да може да създава, управлява и следи промяната в модела на потребителя и съобразно конкретното му състояние да може да му предостави правилната информация в точния момент [2]. Ежедневната работа на потребителите с много такива системи изисква от всяка една от тях да може да предоставя необходимите данни (резюмирани за конкретния потребител) за съответната приложна област [3].

2.1. Предизвикателства при извличането на потребителски интереси

Такива адаптивни системи, които да предоставят правилната информация на правилните потребители в правилния момент, се сблъскват с няколко основни проблема, които трябва да преодолеят:

– **Твърде много информация**

Съвременните потребители са „залети” от информация и трудно могат да преценят коя е достоверна и надеждна. Дори когато търси нещо конкретно, потребителят попада на документи (книги), които намира с помощта на системата, но много често в крайна сметка се оказва, че не е това, което е търсел. В редица такива случаи потребителите губят ценното си време в запознаване с книги, които не са им нужни.

– **Персонализация**

Повечето стандартни системи не правят препоръки на база профилна информация за потребителя, а по-скоро на база най-продавани книги (албуми) за последния месец, последно издадени, актуални автори. Тези препоръките се правят въз основа на общи статистики за системата, рекламни интереси, маркетингови изисквания и т.н.

За разлика от тях, интелигентните системи използват информацията, събирана за всеки един потребител за правене на лични препоръки без да се използва обща статистика за поведението на всички потребители.

– **Прозрачност при събирането на данни за потребителя**

Много системи събират данни за потребителите си (история на закупените артикули, явно позитивно или негативно мнение за друг, продължително разглеждане на определен артикул и т.н.) без да известяват потребителите си за това. По този начин се нарушават правни и етични норми. Още повече, самият потребител може да не желае да бъде събирана и използвана каквато и да е информация за него.

– **Поверителност и прозрачност на препоръките**

Както вече разгледахме, в повечето стандартни системи препоръките стават на общ за всички потребители принцип и са до известна степен произволни (наскоро издадено заглавие, всички книги от автора на текущото издание и т.н.).

За разлика от тях, интелигентните системи използват еднакви алгоритми за препоръки за всички потребители. Препоръките са фактически обосновани и може да бъде обяснено защо е направена точно тази препоръка (наличие на определени термини в съдържанието, вземайки в предвид тежести на дадени характеристики).

– **Проблемът с добавянето на нов обект (документ)**

Повечето системи за препоръки, използват социалният подход и при тях добавянето на нов документ е сериозен проблем. В момента на добавянето му, липсва достатъчно информация за него. Събирането на информация за документа, кои потребители го харесват, отнема време. При препоръките, базирани на съдържание, този проблем не съществува. При добавянето на нов документ съдържанието му се анализира и се извличат неговите ключови думи и съответно да бъде препоръчан на потребители с такива интереси.

В следващите секции ще разгледаме по-подробно с какви средства могат да бъдат преодолени тези проблеми, както и каква конкретна причина поражда необходимостта от реализация на услугата за моделиране на потребители.

2.3. Избор на технология за търсене.

Изборът на правилната технология за търсене на семантична информация в текст, е ключа към успеха на едно подобно начинание. Има няколко различни решения в тази област, които предлагат различен подход, съответно различни резултати и производителност при съответните ограничителни условия. Ще разгледаме основните и най-често използваните.

Apache Solr е бърза платформа за търсене с отворен код. Предлагащ голяма скалируемост, SOLR предлага услуги като търсене в голям обем от текст, динамична клъстеризация, индексиране на резултатите, търсене по атрибути (фасети), интеграция с релационни бази данни. Написан на Java, SOLR използва Tomcat за сървлет контейнер и може да се стартира

самостоятелно. Той използва и надгражда Lucene библиотеката на Java за търсенето в големи документи и индексиране. Използването на услугата става посредством REST базирани интерфейси.

Apache Lucene е библиотека за извличане на информация от текст с отворен код. Първоначално написана на Java, Lucene има имплементации на редица други езици като C++, Python, PHP и други. Lucene разглежда документите като съвкупност от полета, съдържащи текст.

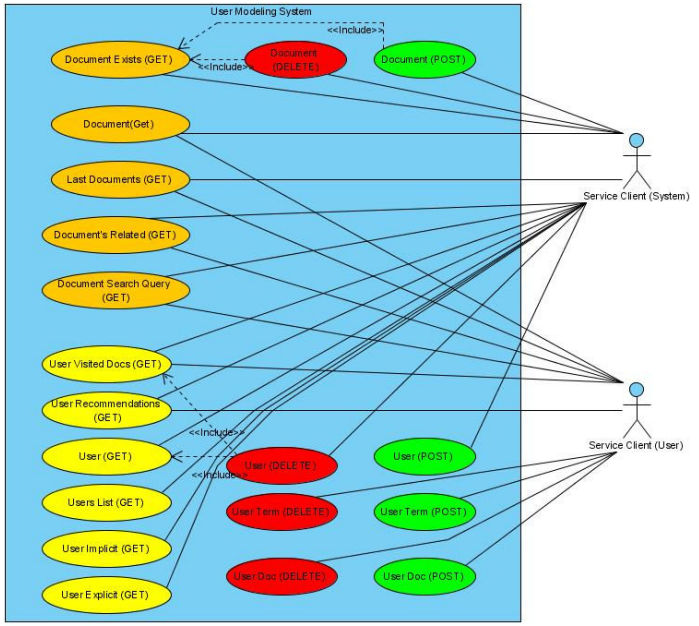
Sphinx е безплатен инструмент за търсене (engine), който акцентира върху индексиране на съдържанието от БД. Поддържа различни типове БД. Написан на C++, и предлага богати възможности за индексиране на текстово съдържание, лесна интеграция посредством публични интерфейси (API).

Използването на **бази данни** за търсене в текст може да бъде полезно в много ситуации при ограничен обем от данни. С нарастването на обема данни обаче производителността рязко пада, налага се непрекъснато преизчисляване на индексите, което е бавна и тежка операция. Също така базите данни не предлагат подобен род вградени услуги, като анализатори, които да филтрират индексираното съдържание или модули, които да ползват алгоритми за data mining за поддържане на резултати, базирани на тях.

Направен бе подробен **сравнителен анализ** на технологиите за търсене в дълбочина на текста. От гледна точка на резултатите, които предоставят, първите 3 подхода дават достатъчно добри резултати за целите на проекта [4]. Поддържането на висока скалируемост е задължително условие за реализацията на подобен род приложение. От гледна точка на имплементацията Lucene предлага възможности за коригиране и настройване на самите алгоритми за търсене и прилагането на собствени такива, за да може приложението да върне максимално точни резултати за съответната приложна област. Lucene обаче обвързва разширяването на тези възможности с писане на Java, в противен случай се губи производителност, която е ключова за приложението. Също така липсват и всякакъв род анализатори и други подобни помощни инструменти, спомагащи правилното анализиране на съдържанието. Без тях приложението губи идеята си, а имплементацията им би отнела много време и усилия, като производителността едва ли би достигнала нивата на тези, давани от SOLR и Sphinx. Изборът се свежда до 2 алтернативи – SOLR или Sphinx. Докато SOLR е под лиценза на Apache2 и може да се ползва свободно за приложения, Sphinx е под лиценза на GPLv2 и ако се реши да се разширява или вгражда в комерсиално приложение е необходимо да се закупи лиценз за комерсиално ползване. При разработване на сериозни Java приложения, търсещи в големи масиви от данни, Solr е по-добрия вариант.

3. Функционалност на услугата

За представяне на функционалността на приложението ще използваме диаграми от тип случай на употреба, които да демонстрират функционалността на приложението посредством актьори, случаи на употреба и определени зависимости между тях. В приложението ще има 2 актьора, които взаимодействат с услугата по определен начин. За да бъде по-ясно коя част от услугата се управлява от приложението, и коя от крайния потребител, използваме различни актьори в диаграмата, макар че в общия случай и двата актьора представляват външно приложение, което се възползва от функционалността на услугата. Консуматорът на услугата, който в общия случай ще бъде някаква външна система, предоставяща управление на набор от потребители и документи, ще се възползва от пълната функционалност на услугата (управление на документи от колекцията, отделните потребители, модели и индекси за тях). Другият актьор в тази схема е крайният потребител. Той не взаимодейства директно с услугата, а го прави посредством външното приложение. Посредством услугата потребителят управлява и извлича данните за себе си и своя модел (Фигура 2).



Фигура 2.. Случаи на употреба за услугата за моделиране на потребители.

Случаите на употреба са разделени в 4 категории в зависимост от типа на извършените операции и това коя част от услугата засяга:

- Добавяне на данни, които ще бъдат анализирани от услугата. Външната система може да добавя документ към колекцията и да добавя потребител към системата, за който в последствие ще бъде създаден потребителски модел. Крайният потребител от друга страна, може да добавя термини към модела си, както и документи, които представляват интерес за него.
- Изтриване на данни (за документи, потребители, интересни за потребителя документи, термини за документи).
- Извличане на данни за модела и обща потребителска информация („маркирани” документи, общ списък с потребители, явни и неявни интереси на потребители, препоръки направени от услугата).
- Извличане на данни за документите и индексна информация (информация за термини).

Тъй като потребителите не търсят винаги едно и също, техните **интереси се менят във времето** [7]. Въпрос на гледна точка е кой е най-удачният избор за филтриране на най-съществените потребителски интереси, тъй като в някои системи потребителите имат по-постоянни интереси, а в други интересите им се менят постоянно и не е необходимо да се пази история за тях. В нашата конкретна услуга последните търсения на потребителя трябва да имат по-голяма тежест от тези, които са били по-назад във времето [7]. Но също не е коректно да се помнят само последните интереси на потребителя и старите да се изтриват. Това е така, понеже потребителите имат т.нар. постоянни интереси, които не трябва да бъдат „забравяни”.

Тук ще се възползваме от резултатите постигнати в [7 и 5] и ще разчитаме на това, че новите интереси са свързани с предишни такива. Използва се т.нар. забравяща функция [7]., която отчита времето, в което е маркиран даден интерес. На последните търсения се задава фиксирано тегло (пр. 1 или 100), а всеки по-стар интерес използва пропорционално тегло спрямо първоначално зададеното (пр. с 0.02 или 2 по-малко от предходното). Така се взимат последните интереси на потребителя с различна тежест. За да бъдат взети под внимание и стари интереси на потребителя, могат да бъдат зададени брой допълнителни интереси от по-старите (тези, които са с дата, по-стара от тази на последните интереси, за които вече зададохме тежест), както и с каква тежест да бъдат взети те. Взимането на тези интереси става произволно, като по този начин вероятността повече от постоянните интереси на потребителя да попаднат в тази група е по-голяма. На база на тези данни се формира заявка, която да търси документи, отразяващи интересите на потребителя.

В рамките на дадена сесия разглеждаме документите, които потребителят е харесал, определяме категориите, в които те попадат, разширяваме контекста на потребителя и му правим препоръки на база на този контекст.

4. Програмна реализация

За изграждането на услугата за моделиране на потребители използваме няколко готови библиотеки, класове и програмна рамка (framework) за работа. За да демонстрираме използването на услугата е изградено и примерно приложение, което използва създадената услуга. За имплементиране на услугата от сървърна страна е използван програмния език PHP.

Използван е също CodeIgniter (PHP framework), който е базиран на архитектурния шаблон Модел-Изглед-Контролер (MVC - Model-View-Controller). Също така съдържа вградени класове за изграждане на допълнителен абстрактен слой за работа с БД. За изграждане на примерното приложение, което ще наречем за краткост UMSC (User Modeling Service Consumer) е използван също CodeIgniter.

Тъй като в примерното приложение ще имаме и операции, които се изпълняват на компютъра на потребителя, ще имаме нужда от език за работа при клиента. Най-често срещаният и подходящ за подобни операции е JavaScript, но функционалността на езика се проявява по различен начин в различните браузъри. Също така ще имаме нужда от бързина, лесно управление на асинхронни Ajax заявки и резултатите от тях, управление на събития, достъпване и промяна на DOM дървото на документа. По посочените критерии ще се спрем на jQuery – най-често използваният език за малки и средни проекти. За него са налични много безплатни компоненти, реализиращи конкретна функционалност.

Отделно от споменатите програмни рамки и библиотеки, използваме и няколко независими самостоятелни инструменти, които улесняват създаването на услугата за моделиране на потребители и примерното приложение, което я ползва:

- TankAuth – библиотека за аутентикация на CI, позволяваща Blowfish bcrypt базирано хеширане на паролата.
- Solr-php-client (библиотека с отворен код за индексиране и търсене на документи в колекция на Apache Solr).
- Rest имплементация за CodeIgniter за създаване на интерфейс (API), от което да може да се ползва услугата.

За изграждането на примерно приложение, което използва услугата, са избрани някои готови продукти (Apache Solr, Apache Solr PHP Client, CodeIgniter Framework, Tank Auth библиотека за аутентикация, REST имплементация, JavaScript jQuery framework).

За реализацията на услугата за моделиране използваме **REST подхода**. REST налага специфични архитектурни принципи, които определят как съдържанието се изпраща през HTTP протокол към различни клиенти. Те могат да са писани върху различни платформи и на различни езици. REST предоставя HTTP операции, аналогични на заявките към бази от данни. Голямата четворка заявки в БД (CRUD (create, read, update, delete) – създаване, четене промяна и изтриване) съответства на 4-те HTTP операции (PUT, GET, POST, DELETE). Тези операции са използвани при добавяне на нови документи, потребители, извличане на данни за списъци от документи, промяна на модели, изтриване на съществуващи на потребители и т.н.

За достъп до услугата се изисква HTTP аутентикация. Този тип аутентикация използва HTTP протокола за изпращане на данни (потребителско име и парола) към сървъра (в случая нашата услуга).

Както вече споменахме TankAuth библиотеката се грижи за сигурността на потребителите в примерното приложение, което използва услугата. Библиотеката има вградени защити срещу атаки посредством rainbow таблици (таблици, които служат за обръщане на хеш функцията с цел откриване на оригиналната парола) и brute force атаки (има механизми за забавяне на проверката, което обезсмисля подобен род атаки).

Заключение

Чрез услугата за моделиране на потребители се реализира метод за препоръки на документи, базиран на съдържанието им. Услугата е до известна степен универсална, тъй като може да се използва от различни системи през стандартен интерфейс. Не е от значение дали ще се използва за книги, новини, статии, музикални произведения. Правилно извлеченото съдържание на съответния (наречен в общия случай) документ, се анализира за термини. Така може да бъде открита връзка с други документи, независимо от вида на документа (текст на песен, книга, дори субтитри на филм и т.н.). На практика възможностите на услугата са разширени и позволяват бързо добавяне на нов документ в системата.

Услугата поддържа за момента 2 езика (макар и с различни възможности) и е базирана до известна степен на възможностите на Solr сървъра и на неговото бързодействие в индексирани колекции. С добавянето на нова функционалност към новите версии към Solr сървъра, тези възможности могат да бъдат интегрирани като подобрения и в услугата. За различните езици (в текущата версия български и английски) услугата използва различни инстанции на Solr сървъра (различни колекции от документи, различни индекси, различни филтри, маркировачи и стратегии за изрязване на корените на думите за езиците, за които е необходимо).

Системата е прозрачна по отношение на препоръките, тъй като дава възможност показването им към потребителя да става само с явното му одобрение. Цялата неявно синтезирана и събрана информация за модела на потребителя (термини и коефициенти за тях) му се предоставя за преглед и редакция. Той може да одобри, откаже или игнорира всяка препоръка, като само явно одобрените ще бъдат използвани. По този начин всяка една препоръка, направена на потребител, може да бъде обоснована и възпроизведена с възможностите на услугата и става с неговото изрично съгласие. Освен това потребителят може да управлява модела си във всеки един момент като включва или изключва части от него в зависимост от конкретните си текущи интереси.

Благодарности

Настоящите изследвания са частично финансирани от НФ„НИ” по проект „Умна книга”, Договор D002-111/15.12.2008.

Литература

1. Ivan Koychev, Darina Dicheva, Roumen Nikolov. Smartbook : Semantics inside. In: Serdica, Journal of Computing, 2010, 263-278. Bulgarian Academy of Science, Institute of Mathematics and Informatics.
2. Gerhard Fischer. User Modeling in Human-Computer Interaction. Contribution to the 10th Anniversary of the journal “User-Modeling and User-Adapted Interaction (UMUAI)”, 2010.
3. Michael Pzzani, Daniel Billsus. Content-based Recommendation Systems. Rutgers University, New Brunswick; FX Palo Alto Laboratory, Palo Alto. In: The Adaptive Web In The Adaptive Web, Vol. 4321, pp. 325-341, 2007
4. Anthony Arnone, Neal Richter. Search Engine Rodeo, summer 2007, Last update Sep 2008. http://www.cs.montana.edu/~richter/Search_Engine_Rodeo.pdf
5. Ingo Schwab, Alfred Kobsa, Ivan Koychev. Learning User Interests through Positive Examples Using Content Analysis and Collaborative Filtering. In: journal “User-Modeling and User-Adapted Interaction (UMUAI)”, 2001
6. Sarah McBurney, Nick Taylor, Howard Williams, Eliza Papadopoulou. Giving the User Explicit Control over Implicit Personalisation. proceedings of AISB '09 http://www.aisb.org.uk/convention/aisb09/Proceedings/PERSIST/FILES/McBurneySM.pdf
7. Koychev, I. Schwab, I. Adaptation to Drifting User's Interests. In Proc. of ECML2000 Workshop: Machine Learning in New Information Age, Barcelona, Spain, 2000. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.1455

WEB BASED SERVICE FOR USER INTERESTS MODELLING

Alexander Chomakov, Ivan Koichev, Alexander Grigorov

Sofia University, Faculty of Mathematics and Informatics
sasho.chom@gmail.com, koychev@fmi.uni-sofia.bg, alexander_grigorov@fmi.uni-sofia.bg

***Abstract**: This paper discusses an approach for automatic retrieval of users' interests models implemented as a web service. Our goal is building a system that creates user models and exposes them to other systems. The system is built as a web service that can be used by external systems. The paper discusses the basic personalization methods, the existing technologies for searching semantic similarity, the functionality of the web service and its implementation.*