Пловдивски Университет

Факултет
"Математика и Информатика"

Дипломна работа

Система за препоръки на мобилни приложения

Дипломант:

Венелин Вълков

фак. № 0901261093

Научен ръководител:

гл. ас. д-р Ангел Голев

кат. „Компютърни технологии“

03. 07. 2013 г.

Съдържание

Резюме

Въведение

Глава 1

Създаване на системата

1.1 Проучване

Системи за препоръки (Recommendation systems) (СП) са интересна алтернатива на алгоритми за търсене, тъй като те намират обекти, които може да нямат общо с термина за търсене.

Със създаване на СП се занимава Машинно обучение (Machine learning) (МО) (отрасъл от Компютърни науки (Computer science) (КН)). Тези системи, предоставят списък от препоръки, използвайки Кооперативно филтриране (Collaborative filtering) (КФ) или Филтриране, базирано на съдържание (Content-based filtering) (ФБС).

КФ подход, при който се изгражда модел на съществуваща история за потребителя и решения, взети от подобни потребители. Този модел се използва за предвиждане на какви нови обекти може да се харесат на потребителя. [Melville]

ФБС използва серии от абстрактни характеристики на обектите, за да препоръча подобни обекти. [Mooney]

Пример[1] за използване на двата подхода са *Last.fm*[2] и *Pandora Radio*[3]

- Pandora използва характеристиките на песен или артист за да избере радио станция, която предоставя песни с подобни характеристики. Мнението на потребителя се използва за определеня на тежест на различните характеристики на радио стацията. Pandora е пример за ФБС
- Last.fm създава виртуална радио станция на базата на историята на потребителя (какви песни и групи е слушал). Тя се сравнява с историята и предпочитанията на други потребители като по този начин, системата предоставя нови песни. Last.fm е пример за КФ

Двата подхода имат слаби и силни страни. КФ се нуждае от голямо количество от информация за да направи качествени препоръки. ФБС има нужда от малко данни за да започне работата си, но е лимитиран до първоначално подадени данни.

1.1.1 Преглед на КФ

КФ е подход за създаване на СП, който се налага в практическите имплементации на подобни системи. Основно предимство на метода е, че не разчита на анализиране на съдържание от машина и поради тази причина има възможност за точно препоръчване на сложни обекти (напр. филми), като не е необходимо разбиране на самия обект.

Използват се различни алгоритми за оценяване сходността на два обекта в СП. Два от най-използваните са к-близки съседи (k-nearest neighbours) (КБС) и Свързаност на Пиърсън (Pearson Correlation) (СП).

КБС е един от най-простите от всички МО алгоритми. Обектът е класифициран спрямо мажоратирен вот от неговите съседи, като той е поставен между най-близкия клас от възможните K.

СП е мярка за линейна свързаност между две променливи в интервала [-1:+1].

[1] `http://en.wikipedia.org/wiki/Recommender_system`
[2] `http://www.last.fm/`
[3] `http://www.pandora.com/`

Проблеми с КФ

КФ има три основни проблема: "студен старт скалируемост и рядкост

- студен старт системата изисква много информация за да предостави добри препоръки. В началото на нейното съществуване, обикновено, такава не е налична.
- скалируемост много СП оперират върху милиони потребители и обекти. Нужна е голяма изчислителна мощ за да се предоставят точни препоръки
- рядкост броят на обектите, обикновено, е в пъти по-голям от този на потребителите. Дори и най-активните потребители оценяват само малко подмножество от обектите. Това води до малък брой оценки за отделните обекти

1.1.2 Съществуващи СП

Amazon

Amazon[4] използва СП за препоръчване на нови продукти на потребителите си. Предоставя такива, които смята, че ще са интересни за тях. Използва се ФБС.

Интересно за Amazon е, че има над 29 милиона потребителя и няколко милионен каталог от продукти. Размер данни, който затруднява голяма част от алгоритмите за намиране на препоръки. Разработчиците на Amazon се справят с този проблем като използват офлайн създаване на таблици със сходни продукти. В резултат на това, алгоритъма се грижи само да извлече данни от таблицата, когато те са нужни.

Youtube

Youtube[5] използва СП за да предостави на потребителите си видео записи на интересни за тях теми. Системата се опитва да максимизира броя видеота,

[4] `http://www.amazon.com/`
[5] `http://www.youtube.com/`

които потребителя гледа и времето което прекарва в сайта. Използва се хибридна версия между КФ и ФБС.

Лимитиращи фактори са взети под предвид. Системата предоставя само определен брой видео клипове от същия потребител. Използват се уникални за потребителя предпочитания, неговата история, брой изгледани видеота и време по което са гледани за да се увеличи възможността за добра препоръка.

1.1.3 Android и пазарът за мобилни приложения

Android[6] е мобилна **OS!** (**OS!**) базирана на Линукс, създадена предимно за мобилни устройства, като таблети и телефони. Тя се разработва от Google, които през 2005 я закупуват от малка компания [Elgin]. С появата на Android се основава и Open Handset Alliance[7], организация която се грижи за развитието на мобилните технологии.

Нарастващ брой приложения

Android нараства бързо след 2009 г., когато представлява само 2.8%[8] от пазара за мобилни устройства. В края на 2010 г. представя 33%[9] от него. В края на 2012 г., Android е на челно място с 75% [?]. Активирани са над 900 милиона Android устройства [?].

Броят приложения нараства заедно с популярността на Android.

1.2 Моделиране

За създаване на гореописаната система са нужни два големи отделни компонента:

- *Сървър* обработва данни за отделните приложения и потребители на системата. Грижи се за събиране, запазване, предоставянето и анализиране-

[6] `http://www.android.com`

[7] `http://www.openhandsetalliance.com/`

[8] `http://appleinsider.com/articles/09/08/21/canalys_iphone_outsold_all_windows_mobile_phones_in_q2_2009.html`

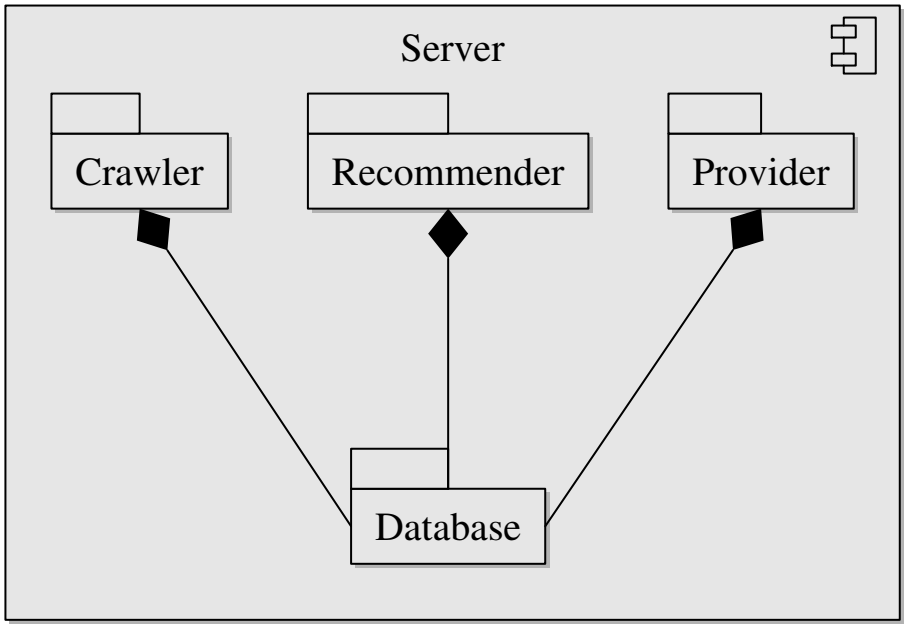
[9] `http://www.canalys.com/newsroom/google%E2%80%99s-android-becomes-world%E2%80%99s-leading-smart-phone-platform`

то на препоръки за клиентската част. Предоставя Приложим интерфейс за програмиране (Application Programming Interface) (ПИП) за работа с данните, които съхранява. От своя страна, този компонент може да бъде разделен на следните, по-малки такива:

- Database е компонент, който предоставя услуга за съхранение на данни. Той играе ролята на косвен(implicit) интерфейс с който да работят останалите компоненти от сървърната част.
- Spider се грижи за събиране на информация за различните приложения от уеб страници. Данните се запазват за по-нататъшен анализ.
- Recommender е сърцето на системата. Подобно на примера на Amazon, този модул предварително създава препоръки и ги запазва за употреба от други части на системата. Използва данни събрани от предишния модул, както и такива предоставени от потребителите на системата.
- Provider предоставя ПИП за комуникация със сървърната част. Той е единствения начин за обмяна на информация. Негова цел е да остане независим от клиенти и същевременно да предостави лесен и бърз начин за работа.

Фигура 1.1 показва UML диаграма на сървърния компонент.

- *Клиент* предоставя, може би, най-важната част от системата - това с което потребителя взаимодейства. Скрити за него остава комуникацията със сървъра. В този модул се разглеждат следните подмодули:
  - GUI се грижи за представяне на препоръките, инсталиране на приложения и оценяването им. Този модул трябва да предоставя идентично изживяване за потребителя, независимо от устройството с което той разполага, т.е. трябва да е високо адаптивен.
  - Watcher събира допълнителна информация за потребителя, която да спомогне за създаване на по-точни препоръки. Инсталирани приложения, време за което се посещават, брой посещения са част от съби-



Фигура 1.1: UML диаграма на сървърния компонент

раните данни. Модулът спомага за идентифициране на потребителя, което премахва нуждата от регистрация.

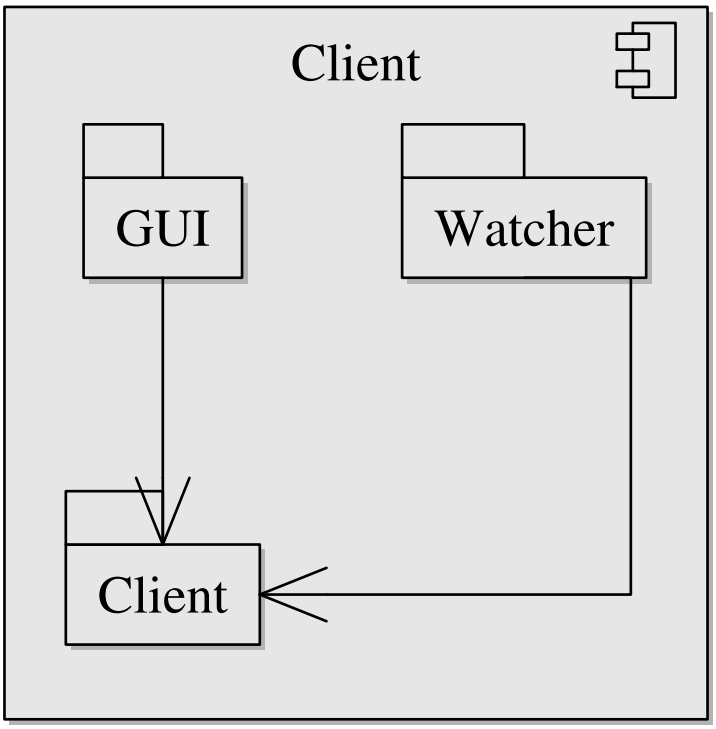
– Client комуникира със сървърната част, като предоставя информация за потребителя и извлича препоръки. Важно за модула е да извършва функциите си по напълно прозрачен начин за потребителя.

Естествен въпрос, който може да се породи в подобна ситуация е "Защо клиентът изпълнява толкова малка част от тежките изчислителни процеси?". При по-детайлно вглеждане в хардуера на днешните мобилни телефони от висок клас(напр. *Samsung Galaxy S4*[10]), става ясно, че съществува модел с 4-ядрен централен процесор и два гигабайта вътрешна памет. Това е съпоставима изчислителна мощ на преносим компютър от преди три години. Разработчиците на мобилни приложения нямат достъп до пълния капацитет на устройството. Допълнителните ядра се грижит по-скоро за това телефона да остане използваем и не предоставят пълен контрол върху хардуера [Gupta]. Казаното до тук не оставя възможността за създаване на хибридни системи, в които клиента да допринася към по-тежките

[10] `http://www.gsmarena.com/samsung_i9500_galaxy_s4-5125.php`

изчислителни процеси.

Фигура 1.2 показва UML диаграма на клиента.



Фигура 1.2: UML диаграма на клиента

1.3 Използвани технологии

1.3.1 Сървърен модул

MongoDB

Python

scrapy

mongoengine

bottle

1.3.2 Клиент

Android SDK

retrofit

1.4 Реализация

Глава 2

Ръководство за потребителя

Заключение

Резултати

Приноси

Проблеми по време на разработка

Бъдещо развитие

Списък на фигурите

Списък на алгоритмите

Приложение А

Използвани съкращения

СП Системи за препоръки (Recommendation systems)

МО Машинно обучение (Machine learning)

КН Компютърни науки (Computer science)

КФ Кооперативно филтриране (Collaborative filtering)

ФБС Филтриране, базирано на съдържание (Content-based filtering)

КБС к-близки съседи (k-nearest neighbours)

СП Свързаност на Пиърсън (Pearson Correlation)

ПИП Приложим интерфейс за програмиране (Application Programming Interface)

ЦП Централен процесор (Central Processing Unit)

Приложение Б

Библиография

[Elgin] Ben Elgin. Google buys android for its mobile arsenal. 2005.

[Google] Google.

[Gupta] Tushar Gupta. Multi-threading android apps for multi-core processors. 2013.

[IDC] IDC.

[Melville] Prem Melville and Vikas Sindhwani. Recommender systems, encyclopedia of machine learning. 2010.

[Mooney] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. 2000.