



TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

JONNE PETTERI PIHLANEN
SUOSITTELIJAJÄRJESTELMÄN RAKENTAMINEN APACHE SPAR-
KILLA

Diplomityö

Examiner: ????

Examiner and topic approved by the
Faculty Council of the Faculty of

xxxx

on 1st September 2014

ABSTRACT

JONNE PETTERI PIHLANEN: Building a Recommendation Engine with Apache Spark

Tampere University of Technology

Diplomityö, xx pages

September 2016

Master's Degree Program in Signal Processing

Major: Data Engineering

Examiner: ????

Keywords:

The amount of recommendation engines around the Internet is constantly growing.

This paper studies the usage of Apache Spark when building a recommendation engine.

TIIVISTELMÄ

JONNE PETTERI PIHLANEN: Building a Recommendation Engine with Apache Spark

Tampereen teknillinen yliopisto

Diplomityö, xx sivua

syyskuu 2016

Signaalinkäsittelyn koulutusohjelma

Pääaine: Data Engineering

Tarkastajat: ????

Avainsanat:

PREFACE

Thanks to my wife, Noora, for pushing me forward with the thesis when my own interest was completely gone. Without you this would never have been ready.

Tampere,

Jonne Pihlanen

SISÄLLYS

| | |
|--|----|
| 1. Johdanto | 1 |
| 2. Suositteijajärjestelmät | 3 |
| 2.1 Suositustekniikat | 5 |
| 2.1.1 Memory-based collaborative filtering | 6 |
| 2.1.2 Model-based collaborative filtering | 9 |
| Bibliography | 10 |

LIST OF ABBREVIATIONS AND SYMBOLS

| | |
|----------------------------|--|
| Spark | Fast and general engine for large-scale data processing |
| Information retrieval (IR) | Activity of obtaining relevant information resources from a collection of information resources. |

1. JOHDANTO

Suosittelujärjestelmiä on menestyksellisesti käytetty auttamaan asiakkaita päätöksenteossa. Itse asiassa ne ovat jatkuvasti läsnä jokapäiväisessä elämässämme. Mikäli asiakas tekee ostoksia, katsoo elokuvaa Netflixistä, selailee Facebookia tai yksinkertaisesti lukee vain uutisia. Periaatteessa kaikki elämämme osa-alueet sisältävät jonkinlaista suosittelua. Ihmiset voivat kuitenkin tehokkaasti suositella vain niitä asioita, jotka ovat itse henkilökohtaisesti kokeneet. Tälläin suosittelijajärjestelmistä tulee hyädyllysiä, sillä ne voivat mahdollisesti tarjota suosituksia tuhansista erilaisista tuotteista.

Suosittelu voidaan jakaa kahteen pääkategoriaan: tuotepohjaiseen ja käyttäjäpohjaiseen suositteluun. Tuotepohjaisessa suosittelussa tarkoituksena on etsiä samankaltaisia tuotteita, sillä käyttäjä saattaa haluta mieluummin samankaltaisia tuotteita myös tulevaisuudessa. Käyttäjäpohjaisessa suosittelussa käyttäjän ajatellaan olevan kiinnostunut tuotteista, joita samankaltaiset käyttäjät ovat ostaneet. Käyttäjäpohjainen suosittelu yrittää siis etsiä samankaltaisia käyttäjiä, jotta voidaan suositella näiden käyttäjien ostamia tuotteita.

Apache Spark on sovelluskehys, joka mahdollistaa hajautettujen ohjelmien rakentamisen. Hajautettu ohjelma tarkoittaa, että ohjelman suoritus on jaettu useiden käsittelysolmujen kesken. Suositteluongelma voidaan mallintaa hajautettuna soveltuksena, jossa kaksi matriisia, käyttäjät ja tuotteet, prosessoidaan iteratiivisella algoritmilla, joka mahdollistaa ohjelman suorittamisen rinnakkain.

Apache Spark on rakennettu Scala ohjelmointikielellä. Scala on monikäyttäinen, moniparadigmainen ohjelmointikieli, joka tarjoaa tuen funktionaaliselle ohjelmoinnille sekä vahvan tyyppityksen. Tyässä käytetään Scalaa, joten lyhyt johdanto ohjelmointikieleen tarjotaan.

Tämä työ on rakentuu seuraavista osista. Luku kaksi kuvailee suosittelujärjestelmiä. Luvussa kolme keskustellaan Apache Sparkista, avoimen lähdekoodin järjestel-

mästä, joka mahdollistaa hajautettujen ohjelmien rakentamisen. Luku neljä esittää toteutuksen suosittelijajärjestelmälle. Luvussa viisi käydään läpi tulokset. Lopuksi luvussa kuusi esitellään johtopäätökset.

2. SUOSITTELIJAJÄRJESTELMÄT

Suosittelulla tarkoitetaan tehtävää, jossa tuotteita suositellaan käyttäjille. Kaikista yksinkertaisin suosittelun keino on vertaisten kesken, täysin ilman tietokoneita. Ihmiset voivat kuitenkin tehokkaasti suositella vain niitä asioita, jotka ovat itse henkilökohtaisesti kokeneet. Tällöin suosittelijajärjestelmistä tulee hyödyllisiä, sillä ne voivat mahdollisesti tarjota suosituksia tuhansista erilaisista tuotteista. Suositelijajärjestelmät ovat joukko tekniikoita ja ohjelmistoja jotka tarjoavat suosituksia mahdollisesti hyödyllisistä tuotteista. Tuote tarkoittaa yleistä asiaa jota järjestelmä suosittelee henkilölle. Suosittelemajärjestelmät on yleensä tarkoitettu suosittelemaan tietyn tyyppisiä tuotteita kuten kirjoja tai elokuvia. [9]

Suosittelijajärjestelmien tarkoitus on auttaa asiakkaita päätöksenteossa kun tuotteiden määrä on valtava. Tavallisesti suositukset ovat räätälöityjä, millä tarkoitetaan että suositukset eroavat käyttäjien tai käyttäjäryhmien välillä. Suositukset voivat olla myös räätälöimättömiä ja niiden tuottaminen onkin usein yksinkertaisempaa. (VIITE?) Räätälöimätöntä suosittelua on esimerkiksi yksinkertainen top 10 lista. Järjestäminen tehdään ennustamalla kaikista sopivimmat tuotteet käyttäjän mieltymysten tai vaatimusten perusteella. Tämän suorittamiseksi suosittelijajärjestelmän on kerättävä käyttäjältä tämän mieltymykset. Mieltymykset voivat olla nimenomaisesti ilmaistuja tuote-arvioita tai ne voidaan tulkita käyttäjän toiminnasta kuten klikkauksista tai sivun katselukerroista. Suosittelemajärjestelmä voisi esimerkiksi tulkita tuotesivulle navigoinnin todisteeksi mieltymyksestä sivun tuotteista. [9]

Suosittelijajärjestelmien kehitys alkoi melko yksinkertaisesta havainnosta: ihmiset tapaavat luottaa toisten suosituksiin tehdessään rutiininomaisia päätöksiä. On esimerkiksi yleistä luottaa vertaispalautteeseen valitessaan kirjaa luettavaksi tai luottaa elokuvakriitikoiden kirjoittamiin arvioihin. Ensimmäinen suosittelijajärjestelmä yritti matkia tätä käytöstä käyttämällä algoritmeja suosituksien löytämiseen yhteisöstä aktiiviselle käyttäjälle, joka etsi suosituksia. Tämä lähestymistapa on olennaisesti yhteistyösuodattamista ja idea sen takana on että jos käyttäjät pitivät saman-

kaltaisista tuotteista aikaisemmin, he luultavasti pitävät samoja tuotteita ostaneiden henkilöiden suosituksia merkityksellisinä. [9]

Verkkokauppojen kehityksen myötä syntyi tarve suosittelulle vaihtoehtojen rajoittamiseksi. Käyttäjät kokivat aina vain vaikeammaksi löytää oikeat tuotteet sivustojen suurista valikoimista. Tiedon määrän räjähdysmäinen kasvaminen internetissä on ajanut käyttäjät tekemään huonoja päätöksiä. Hyödyn tuottamisen sijaan vaihtoehtojen määrä oli alkanut heikentää kuluttajien hyvinvointia. Vaihtoehdot ovat hyväksi, mutta vaihtoehtojen lisääntyminen ei ole aina parempi. [9]

Viimeaikoina suosittelijajärjestelmät ovat osoittautuneet tehokkaaksi lääkkeeksi kärsivällä olevaa tiedon ylimääräongelmaa vastaan. Suosittelijajärjestelmät käsittelevät tätä ilmiötä tarjoamalla uusia, aiemmin tuntemattomia tuotteita jotka ovat todennäköisesti merkityksellisiä käyttäjälle tämän nykyisessä tehtävässä. Kun käyttäjä pyytää suosituksia, suosittelujärjestelmä tuottaa suosituksia käyttämällä tietoa ja tuntemusta käyttäjistä, saatavilla olevista tuotteista ja aiemmista tapahtumista suosittelijan tietokannasta. Tutkittuaan tarjotut suositukset, käyttäjä voi hyväksyä tai hylätä ne tarjoten epäsuoraa ja täsmällistä palautetta suosittelijalle. Tätä uutta tietoa voidaan myöhemmin käyttää hyödyksi tuotettaessa uusia suosituksia seuraaviin käyttäjän ja järjestelmän vuorovaikutuksiin. [9]

Verrattuna klassisten tietojärjestelmien, kuten tietokantojen ja hakukoneiden, tutkimukseen, suosittelijajärjestelmien tutkimus on verrattain tuoretta. Suosittelijajärjestelmistä tuli itsenäisiä tutkimusalueita 90-luvun puolivälissä. Viimeaikoina mielenkiinto suosittelujärjestelmiä kohtaan on kasvanut merkittävästi. Esimerkkinä suuren profiilin verkkosivustot kuten Amazon.com, YouTube, Netflix sekä IMDB, joissa suosittelujärjestelmillä on iso rooli. Oma lukunsa ovat myös vain suosittelujärjestelmien tutkimiseen ja kehittämiseen tarkoitetut konferenssit kuten RecSys ja AI Communications (2008). [9]

Suosittelujärjestelmällä voidaan ajatella olevan kaksi päätarkoitusta. Ensimmäinen on avustaa palveluntarjoajaa. Toinen on tuottaa arvoa palvelun käyttäjälle. Suosittelujärjestelmän on siis tasapainoteltava sekä palveluntarjoajan että käyttäjän tarpeiden välillä. [9] Palveluntarjoaja voi esimerkiksi ottaa suosittelujärjestelmän avuksi parantamaan tai monipuolistamaan myyntiä, lisäämään käyttäjien tyytyväisyyttä, lisäämään käyttäjien uskollisuutta tai ymmärtämään paremmin mitä käyttäjä haluaa [9]. Lisäksi käyttäjillä saattaa olla seuraavanlaisia odotuksia suosittelujärjestelmältä. Käyttäjä saattaa haluta suosituksena tuotesarjan, apua selaamiseen tai

mahdollistaa muihin vaikuttamisen. Vaikuttaminen saattaa olla pahantahtoista. [9]

GroupLens, BookLens ja MovieLens olivat uranuurtajia suosittelujärjestelmissä. GroupLens on tutkimuslaboratorio tietojenkäsittelyopin ja tekniikan laitoksella Minnesotan Yliopistossa, Twin Cities:issä, joka on erikoistunut suosittelujärjestelmiin, verkkoyhteisöihin, mobiili ja kaikkialla läsnä oleviin teknologioihin, digitaalisiin kirjastoihin ja paikallisen maantieteen tietojärjestelmiin. [2] BookLens on on GroupLensin rakentama kirjojen suosittelujärjestelmä [3]. MovieLens on GroupLensin ylläpitämä elokuvien suosittelujärjestelmä [8]. Uranuurtavan tutkimuksen lisäksi nämä sivustot julkaisivat aineistoja, joka ei ollut yleistä. [2]

2.1 Suositustekniikat

Suosittelujärjestelmällä täytyy olla jonkunlainen ymmärrys tuotteista, jotta se pystyy suosittelemaan jotain. Tämän mahdollistamiseksi, järjestelmän täytyy pystyä ennustamaan tuotteen käytännöllisyys tai ainakin verrata tuotteiden hyödyllisyyttä ja tämän perusteella päättää suositeltavat tuotteet. Ennustamista voidaan luonnostella yksinkertaisella ei-personoidulla suosittelualgoritmilla joka suosittelee vain suosituimpia elokuvia. Tätä lähestymistapaa voidaan perustella sillä, että tarkemman tiedon puuttuessa käyttäjän mieltymyksistä, elokuva josta muutkin ovat pitäneet on todennäköisesti myös keskivertokäyttäjän mieleen, ainakin enemmän kuin satunaisesti valikoitu elokuva. Suositettujen elokuvien voidaan siis katsoa olevan kohtuullisen osuvia suosituksia keskivertokäyttäjälle. [9]

Tuotteen i hyödyllisyyttä käyttäjälle u voidaan mallintaa reaaliarvoisella funktiolla $R(u, i)$, kuten yleensä tehdään yhteisösuodatuksessa ottamalla huomioon käyttäjien antamat arviot tuotteista. Yhteisösuodatus suosittelijan perustehtävä on ennustaa R :n arvoa käyttäjä-tuote pareille ja laskea arvio todelliselle funktiolle R . Laskiessaan tätä ennustetta käyttäjälle u tuotejoukolle, järjestelmä suosittelee tuotteita joilla on suurin ennustettu hyödyllisyys. Ennustettujen tuotteiden määrä on usein paljon pienempi kuin tuotteiden koko määrä, joten voidaan sanoa että suosittelijajärjestelmä suodattaa käyttäjälle suositeltavat tuotteet. [9]

Suosittelijajärjestelmät eroavat toisistaan kohdistetun toimialan, käytetyn tiedon ja erityisesti siinä kuinka suositukset tehdään, jolla tarkoitetaan suosittelualgoritmia [9]. Tässä työssä keskitytään vain yhteen suosittelutekniikoiden luokkaan, yhteisölliseen suodatuksen, sillä tätä menetelmää käytetään Apache Sparkin MLlib kirjas-

tossa.

Yhteisöllistä suodatusta käyttävät suosittelijajärjestelmät perustuvat käyttäjien yhteistyöhön. Niiden tavoitteena on tunnistaa kuvioita käyttäjän mielenkiinnoista voidakseen tehdä suunnattuja suosituksia [1]. Tämän lähestymistavan alkuperäisessä toteutuksessa suositellaan aktiiviselle käyttäjälle niitä tuotteita joita muut samankaltaiset mieltymyksen omaavat käyttäjät ovat pitäneet aiemmin [9]. Käyttäjä arvostelee tuotteita. Seuraavaksi algoritmi etsii suosituksia perustuen käyttäjiin, jotka ovat ostaneet samanlaisia tuotteita tai perustuen tuotteisiin, jotka ovat eniten samanlaisia käyttäjän ostoshistoriaan verrattuna. Yhteisösuodatus voidaan jakaa kahteen kategoriaan, jotka ovat tuotepohjainen- ja käyttäjäpohjainen yhteisösuodatus. Yhteisösuodatusta on eniten käytetty ja toteutettu tekniikka suositusjärjestelmissä [5] [9] [4].

Yhteisöllinen suodatus analysoi käyttäjien välisiä suhteita ja tuotteiden välisiä riippuvuuksia tunnistaaakseen uusia käyttäjä-tuote assosiaatioita [6]. Päätelmä siitä, että käyttäjät voisivat pitää samasta laulusta koska molemmat kuuntelevat muita samankaltaisia lauluja on esimerkki yhteisöllisestä suodatuksesta [10].

Koska yhteisöllisessä suodatuksessa suosittelu perustuu pelkästään käyttäjän arvosteluihin tuotteesta, yhteisöllinen suodatus kärsii ongelmista jotka tunnetaan nimillä uusi käyttäjäongelma ja uusi tuoteongelma [5]. Ellei käyttäjä ole arvostellut yhtään tuotetta, algoritmi ei kykene tuottamaan myöskään yhtään suositusta. Muita yhteisöllisen suodatuksen haasteita ovat kylmä aloitus sekä niukkuus (sparsity). Kylmällä aloituksella tarkoitetaan sitä, että tarkkojen suositusten tuottamiseen tarvitaan tyypillisesti suuri määrä dataa. Niukkuudella tarkoitetaan sitä, että tuotteiden määrä ylittää usein käyttäjien määrän. Tästä johtuen suhteiden määrä on todella niukka, sillä useat käyttäjät ovat arvostelleet tai ostaneet vain murto-osan tuotteiden kokomäärästä. [1]

2.1.1 Memory-based collaborative filtering

Memory-based or neighborhood collaborative filtering

In memory-based methods the user-item ratings stored in the system are directly accessed to predict ratings for new items. This can be done in two ways known as user-based or item-based recommendation.

The following sections describe user-based collaborative filtering and item-based collaborative filtering.

Item-based collaborative filtering

Item-based collaborative filtering (IBCF) starts by finding similar items from the user's purchases [5]. Next step is to model the preferences of a user to an item based on ratings of similar items by the same user [9]. The following snippet presents the idea in IBCF for every new user.

Program 2.1 *Item-Based Collaborative Filtering algorithm [5]*

1. For each two items, measure how similar they are in terms of having received similar ratings by similar users

```
val similarItems = items.foreach { item1 =>
items.foreach { item2 =>
val similarity = cosineSimilarity(item1, item2);
}
}
```

2. For each item, identify the k-most similar items

```
val itemsSorted = sort(similarItems)
```

3. For each user, identify the items that are most similar to the user's purchases

```
users.foreach { user =>
user.purchases.foreach { purchase =>
val mostSimilar = findSimilarItem(purchase)
}
}
```

Amazon.com, the biggest Internet retailer in the United States, has previously been

using item-to-item collaborative filtering method. In their implementation the algorithm builds a table containing similar items by finding ones that users tend to purchase together. The algorithm then finds items similar to each of the user's purchases and ratings, combines those items, and returns the most popular or correlated items. [7]

User-based collaborative filtering

User-based collaborative filtering (UBCF) starts by finding the most similar users, rate items purchased by similar users, pick top rated items. The similarity in taste of two users is calculated based on the similarity in the rating history of the users [9].

The steps for every new user in user-based collaborative filtering are as follows:

Program 2.2 *User-Based Collaborative Filtering algorithm [5]*

1. Measure how similar each user is to the new one. Like IBCF, popular similarity measures are correlation and cosine.

```
case class Similarity(userId1: Int, userId2: Int, score: Int)
```

```
val newUser: User = User("Adam", 31, purchases)
```

```
val similarities = users.map { user =>
  Similarity(newUser.id, user.id, cosineSimilarity(user, newUser))
}
```

2. Identify the most similar users. The options are: Take account of the top k users (k-nearest_neighbors) Take account of the users whose similarity is above a defined threshold

```
val mostSimilarUsers = similarities.filter(_.score > 0.8)
```

3. Rate the items purchased by the most similar users. The rating is the average rating among similar users and the approaches are:

Average rating

Weighted average rating, using the similarities as weights

```
val ratedItems = mostSimilarUsers.map { user =>
  user.purchases.map { purchase =>
    val purchases = mostSimilarUsers.map { usr =>
      usr.purchases.filter (_id === purchase.id)
    }
    purchases.sum() / purchases.size
  }
}
```

4. Pick the top-rated items.

```
val topRatedItems = ratedItems.take(10)
```

2.1.2 Model-based collaborative filtering

In contrast to memory-based systems, which use the stored ratings directly in the prediction, model-based approaches use these ratings to learn a predictive model. The general idea is to model the user-item interactions with factors representing latent characteristics of the users and items in the system, like the preference class of users and the category class of items. This model is then trained using the available data, and later used to predict ratings of users for new items. [9]

Alternating Least Squares is an example of model-based collaborative filtering algorithm, it is presented in the next chapter.

BIBLIOGRAPHY

- [1] C. Aberger, “Recommender: An analysis of collaborative filtering techniques,” 2014. [Online]. Available: <http://cs229.stanford.edu/proj2014/Christopher%20Aberger,%20Recommender.pdf>
- [2] C. C. Aggarwal, *Recommender Systems*. Springer International Publishing, 2016.
- [3] BookLens. [Online]. Available: <https://booklens.umn.edu/>
- [4] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370. [Online]. Available: <http://dx.doi.org/10.1023/A:1021240730564>
- [5] S. K. Gorakala and M. Usuelli, *Building a Recommendation Engine with R*, 1st ed. Packt Publishing, 2015.
- [6] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” 2009. [Online]. Available: [https://datajobs.com/data-science-repo/Recommender-Systems-\[Netflix\].pdf](https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf)
- [7] G. Linden, B. Smith, and J. York, “Amazon.com recommendations,” *IEEE INTERNET COMPUTING*, pp. 76–79, 2003. [Online]. Available: <http://www.cin.ufpe.br/~idal/rs/Amazon-Recommendations.pdf>
- [8] MovieLens. [Online]. Available: <https://movielens.org/info/about>
- [9] F. Ricci, L. Rokach, B. Shapira, and P. B. Kanto, *Recommender Systems Handbook*, 1st ed. Springer, 2011.
- [10] S. Ryza, U. Laserson, S. Owen, and J. Wills, *Advanced Analytics with Spark*. O’Reilly Media, Inc., 2015.