Matias Jalava

Mobiilisovelluksen sisällön suosittelu tekoälyn avulla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

17.5.2022

Tiivistelmä

Tekijä:	Matias Jalava
Otsikko:	Mobiilisovelluksen sisällön suosittelu tekoälyn avulla
Sivumäärä:	39 sivua
Aika:	17.5.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mobile Solutions
Ohjaajat:	Lehtori Peter Hjort Teknologiajohtaja Tatu Ulmanen

Insinöörityön tarkoituksena oli toteuttaa suosittelujärjestelmä yhdistäen koneoppimisen ja suosittelujärjestelmien toimintatapoja. Suosittelujärjestelmä kehitettiin osaksi tilaajan mobiilisovellusta, jossa sen tarkoituksena oli tuoda sovelluksen käyttäjälle personoituja suosituksia perustuen käyttäjien katseluhistoriaan.

Suosittelujärjestelmä tuli osaksi käytetyn tavaran mobiilisovelluksen etusivua, jossa on listattuna sovelluksessa myynnissä olevia tuotteita. Työn tavoitteena oli korvata etusivun tuotteiden noudon satunnaisuus personoiduilla suosituksilla. Koska puhelinsovellusta, jonka osaksi järjestelmä tuli, ei julkaistu työn aikana, suosittelujärjestelmä piti testata generoidulla datalla ja kehittää niin, että jatkokehittäminen on tulevaisuudessa helppoa.

Tekoälymalli, eli suosituksia luova osa, luotiin Tensorflow-kirjaston metodeilla. Järjestelmä koostettiin sisältämään neuroverkko katseluaikojen ennustamiseen ja metodit käyttäjien ja tuotteiden profiloimiseen 32-ulotteisiin vektoreihin. Tuotteiden noutovaihe, joka hakee käyttäjäkohtaisesti X-määrän tuotteita koko valikoimasta, toteutettiin palauttamaan ne tuotteet, joiden pistetulot pyydettyyn käyttäjään olivat pienimmät. Tuotteiden pisteytys, eli vaihe, jossa arvioitiin noudettujen tuotteiden sopivuus käyttäjälle, mahdollistettiin neuroverkon avulla. Neuroverkko ennusti käyttäjän ja tuotteen välisen katseluajan yksi kerrallaan, minkä jälkeen kohteet järjesteltiin suurin aika ensin.

Järjestelmän toimivuutta mitattiin vertaamalla sen suosituksia käyttäjien oikeisiin näyttökertoihin, vertaamalla käyttäjien ja tuotteiden upotusvektoreita ja tarkkailemalla mallin harjoitusmetriikoita. Työn testeistä selvisi, että järjestelmä onnistuu molemmissa sen tehtävissä, eli tuotteiden noudossa ja pisteytyksessä. Työ tehtiin melko itsenäisesti aihetta opiskellen, yritysten ja erehdysten kautta. Suurin osa tekoälyn toimintatavoista testattiin Python-notebook-ympäristössä. Kun jokin osa todettiin toimivaksi, se lisättiin järjestelmän rajapintasovellukseen. Tuloksena oli tehtäväänsä sopiva suosittelujärjestelmä, joka suositteli testikäyttäjille osuvia tuotteita.

Avainsanat: suosittelujärjestelmät, tekoäly, koneoppiminen, Tensorflow, Python, tietojenkäsittely

Abstract

Author:	Matias Jalava
Title:	Recommending mobile application content with AI
Number of Pages:	39 pages
Date:	17 May 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Mobile Solutions
Supervisors:	Peter Hjort, Senior Lecturer Tatu Ulmanen, CTO

The purpose of the final year project was to create a recommendation system powered by artificial intelligence. The recommendation system was developed to be a part of a secondhand marketplace mobile application, where its role will be to bring personalized recommendations to users based on their watch history.

The recommendation system was developed to be a part of the application's front page, which lists random products that are currently being sold, its role being to personalize the page's content. Because the application which the system was developed for was not released, the system had to be developed and tested based on generated data.

The artificial intelligence model, the role of which is to create recommendations, was created using TensorFlow's libraries. The model was developed to include a neural network to predict watching times and methods to create 32-dimensional user- and product-embeddings. These methods enabled the system to handle two common tasks of recommendation systems: retrieval, and ranking. At retrieval-phase, the system retrieves X number of products from all products that are closest to queried user based on dot-product. Ranking was done using the neural network of the model to predict the watching times between queried user and given product.

The system was tested by comparing its recommendations to the watch history of queried user. Based on the tests, the recommendation system handles both retrieval and ranking well. The project was done quite independently via self-studying and testing different methods in Python-notebook environment. When a feature was deemed functional, it was added to the system. The product was a functional recommendation system which worked as intended.

Keywords: recommendation systems, AI, Tensorflow, Python, machine learning, data science

Sisällys

Lyhenteet

1 Johdanto 1

2 Suosittelujärjestelmät 1

2.1 Hakuvaihe 2

2.1.1 Yhteisöllinen suodatus 5

2.1.2 Sisältösuodatus 6

2.2 Suositusten pisteytys ja järjestely 7

3 Tensorflow ja koneoppiminen 9

3.1 Harjoitus ja mittaaminen 10

3.2 Mallien rakenne 12

4 Suosittelujärjestelmä mobiilisovellukseen 16

4.1 Tiedon haku ja käsittely 17

4.2 Tekoälymalli 21

4.3 Testaaminen 24

4.4 Julkaisu 29

5 Reflektointi 33

6 Yhteenveto 34

Lähteet 35

Lyhenteet

MSE – Mean Squared Error

RMSE – Root Mean Squared Error

TF – Tensorflow

NN – Neural Network

WSGI – Web Server Gateway Interface

1 Johdanto

Sovellusten ja verkkosivujen, joiden toiminta perustuu suuren ja jatkuvasti muuttuvan sisältömäärän tuomiseen käyttäjälle, kuten YouTube (1), Spotify (2; 3) ja Netflix (4), on jatkuvasti kehitettävä tapoja tuoda sisältöä käyttäjille. Jokaisella suurella verkkopalvelulla on erittäin suuret määrät dataa sekä käyttäjistä että tarjoamistaan kohteista. Yksi kysymys, joka yhdistää kaikkia näitä palveluita, onkin se, miten käyttäjät löytävät haluamaansa sisältöä loputtomalta tuntuvasta valikoimasta. YouTubea, Spotifyta ja Netflixiä yhdistää se, että ne käyttävät omaan tarkoitukseensa kehitettyjä suosittelualgoritmeja tuodakseen käyttäjilleen osuvinta mahdollista sisältöä.

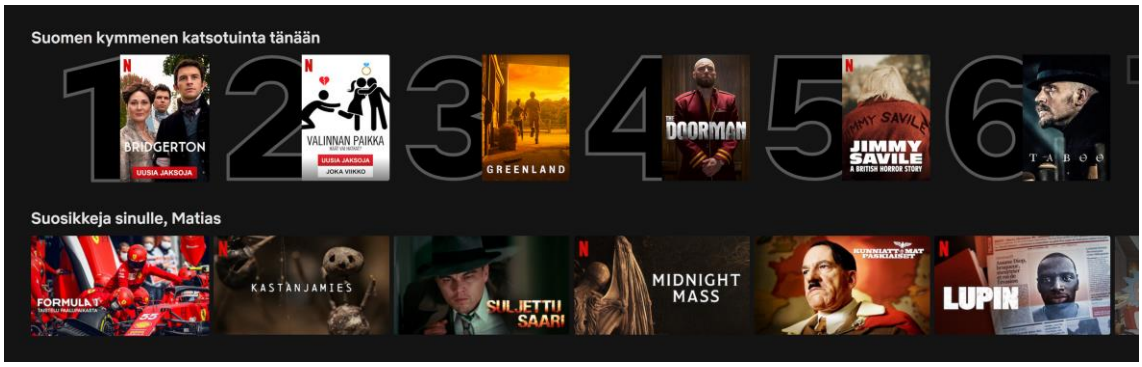
Tässä työssä perehdytään suosittelujärjestelmien ja Tensorflow-koneoppimiskirjaston perusteisiin ja luodaan suosittelujärjestelmä tuottamaan puhelinsovelluksen etusivulle tuotesuosituksia käyttäjien nähtäväksi. Työn on tilannut Jemma-niminen startup-yritys, jonka samannimisen sovelluksen tarkoituksena on kannustaa käyttäjiään käymään kauppaa omistamillaan tavaroilla.

Jemman etusivu on yksinkertainen lista palvelussa myynnissä olevia tuotteita. Tämän työn tavoitteena on kehittää sovelluksen etusivulle sopiva suosittelujärjestelmä, joka innostaa käyttäjiä selaamaan sovelluksen tarjontaa tarjoamalla käyttäjäkohtaisia, osuvia suosituksia. Työ toteutetaan Python-ohjelmointikielellä Tensorflow-koneoppimiskirjaston metodeja ja suosittelujärjestelmien menetelmiä hyödyntäen.

2 Suosittelujärjestelmät

Suosittelujärjestelmät ovat suosituksia luovia ohjelmistoja. Suositusten tarkoituksena on auttaa käyttäjää sovellusta koskevissa päätöksentekoprosesseissa, kuten mitä elokuvia katsoisi, mitä uutisia lukisi tai mitä tuotteita ostaisi. Yleensä suosittelujärjestelmät suosittelevat yhdenlaisia kohteita, kuten elokuvia, kirjoja tai uutisia. (5, s. 1.) Suositukset voivat olla yleisiä suosituksia, kuten 10 suosituinta elokuvaa, tai käyttäjälle personoituja, kuten edellisiin katseluihin

rinnastettavia elokuvia (kuva 1). Yleiset suositukset ovat huomattavasti personoituja suosituksia yksinkertaisempia luoda, minkä takia niitä ei yleensä käsitellä suosittelujärjestelmien tutkimuksissa. (5, s. 2.) Tämäkin työ keskittyy täysin personoitujen suosittelujärjestelmien tekniikoihin.



Kuva 1. Netflixin suosittelulistoja. Ylempänä yleinen ja alempana personoitu. Yleinen perustuu täysin sarjojen ja elokuvien katsojamäärään vuorokauden aikana. Personoidut suositukset perustuvat siihen, että käyttäjä on katsonut toiminta- ja jännityselokuvia sekä -sarjoja. (6.)

Suosittelujärjestelmän toiminnan voi jakaa kolmeen päävaiheeseen:

- kohteiden nouto
- haettujen kohteiden pisteytys
- pisteytettyjen kohteiden uudelleenjärjestely.

Kohteiden nouto on vaihe, jossa järjestelmä käyttää tarkoitukseensa sopivaa suodatusmenetelmää hakemaan suuresta kohdemäärästä tarkan ryhmän käyttäjälle sopivia kohteita. Noudon jälkeen kohteiden sopivuus käyttäjälle pisteytetään. Koska pisteytysvaiheessa kohteita on jo huomattavasti vähemmän, voi tämä vaihe olla paljon hitaampi suorittaa, eli suositukset voidaan pisteyttää hyvinkin tarkasti. Viimeisen vaiheen, eli uudelleenjärjestelyn, tehtävä on käydä pisteytetyt kohteet läpi ja järjestellä kohteet siten, että parhaat suosituskohteet ovat ensimmäisinä suositeltavina. (7.)

2.1 Hakuvaihe

Suosittelujärjestelmän noutovaiheeseen on pääasiassa kaksi eri toimintatapaa:

- yhteisöllinen suodatus, collaborative filtering (8; 9)
- sisältösuodatus, content-based filtering (8; 10).

Yhteisöllisessä suodatuksessa kohdekäyttäjälle suositellaan kohteita, joita muut kohdekäyttäjää vastaavat käyttäjät ovat katsoneet (9; 11). Sisältösuodatuksessa verrataan käyttäjän katsomien kohteiden ominaisuuksia ja näytetään vain kohteita, joilla on samoja ominaisuuksia (10; 11).

Molempia suodatusmenetelmiä yhdistää sama kysymys: Miten tiedetään, mitkä kohteet ovat käyttäjälle mielenkiintoisia? Mielenkiinnon osoituksia, eli arvosteluita, voidaan muodostaa käyttäjien jättämien tietojälkien perusteella. Arvosteluja on sekä eksplisiittisiä ja implisiittisiä. Eksplisiittiset arviot ovat mitä vain käyttäjän tietoisesti jättämiä arvosteluja:

- tykkäykset – binäärinen arvostelu, jossa käyttäjät arvioivat kohteen mielenkiintoiseksi tai ei-mielenkiintoiseksi
- arvostelut – käyttäjät arvioivat kohteen ennalta määritellyllä numeroasteikolla huonoimmasta parhaaseen
- kommentit – kommentein jätetyistä arvioista voi saada erittäin tarkan kuvan käyttäjien mielipiteistä, mutta niiden muuttaminen koneellisesti käytettävään muotoon on raskasta muihin eksplisiittisiin tapoihin verrattuna. (5, s. 77.)

Implisiittiset arvot ovat seurantadataa, kuten klikkauksia, katseluaikoja tai katselukertoja. Toisin kuin eksplisiittiset arvot, implisiittiset eivät tarvitse tallentuakseen mitään aktiivista käyttäjän toimintaa (5, s. 77).

Implisiittisesti saatua dataa on hyvä käyttää hakuvaiheessa, sillä sitä on huomattavasti eksplisiittistä enemmän ja siitä saa tarkan kuvan käyttäjän mieltymyksistä, vaikkei hän arvioisi mitään mitä katsoo (12). Esimerkiksi Netflix käyttää näitä arvoja tehdessään suosituksia: Mitä ohjelmia tai elokuvia käyttäjä on katsonut ja kuinka paljon? (4.) Toisaalta eksplisiittisten arvojen etuna on niiden yksinkertaisuus. Jätetyt arvostelut tykkäyksinä tai numeroina kertovat järjestelmälle melko suoraan, mitkä kohteet merkitsevät käyttäjälle muita enemmän, sillä hän on nähnyt vaivaa niiden arvioimiseen (5, s. 77).

Hakuvaiheen datankäsittelytapa perustuu siihen, että mallin luontivaiheessa jokaiselle käyttäjälle ja kohteelle annetaan upotusarvot. Upotusarvot ovat moniulotteisia vektoreita, jotka esittävät käyttäjiä tai kohteita. Jokainen vektorin piste merkitsee eri asiaa. Esimerkiksi kuvassa 2 käyttäjävektorin ensimmäinen, ja ainoa, arvo esittää mieltymystä lasten elokuviin ja kohdevektorin ensimmäinen arvo esittää, onko elokuva lapsille.

Kuva 2. Yhteissuodatuksen tiedonkäsittelymenetelmä. Merkki käyttäjän ja elokuvan välissä merkitsee, että käyttäjä on katsonut elokuvan. Käyttäjien ja elokuvien upotusarvot ovat yksiulotteisia vektoreita välillä -1 ja 1. Käyttäjien upotusarvo kertoo käyttäjien mieltymykset lasten vai aikuisten elokuville. Elokuvien upotusarvot kertovat niiden kohdeyleisön ikäryhmän. (9.)

Upotusvektorit mahdollistavat kohteen ja käyttäjän välisien suhteiden laskemisen. Pistetulo, eli vektoreiden välinen etäisyys, on yksi näistä metodeista. Pieni pistetulo kertoo isommasta todennäköisyydestä, että käyttäjä pitää kohteesta. (1.)

2.1.1 Yhteisöllinen suodatus

Yhteisöllisen suodatuksen vahvuus on se, ettei sen tarvitse tietää kohteiden ominaisuuksista mitään, mutta se voi silti luoda tarkkoja kohdesuosituksia muiden käyttäjien toiminnan perusteella. Yhteisöllisen suodatuksen avulla tehdyt suositukset voivat olla myös positiivisella tavalla yllättäviä, sillä se ei ole rajoitettu suosittelemaan tarkkojen kohdeominaisuuksien mukaan. Yhteisöllisen suodatuksen tuomat suositukset voivat parhaimmillaan auttaa kohdekäyttäjää löytämään uusia mielenkiinnon kohteita (5, s. 111).

Koska yhteisöllisen suodatuksen malli luo suosituksensa perustuen kaikkiin sen luontivaiheessa olemassa oleviin arvosteluihin käyttäjien ja kohteiden välillä, se ei osaa suositella kohteita, jotka on lisätty sen alustuksen jälkeen (13; 5, s. 111).

Jos yhteisöllisen suodatuksen suosittelumallin voisi alustaa uudelleen aina, kun suositeltava data muuttuu, olisi edellä mainittu ongelma helposti ratkaistu. Ongelma on siinä, että malli on suunniteltu suuria tietomääriä varten ja siinä yhteydessä uudelleen alustaminen koko tietomäärällä on erittäin raskasta. Yksi ratkaisu hallita uutta dataa yhteisöllisen suodatuksen mallissa on nimeltään WALS-yhtälö eli Weighted Alternative Least Squares (kaava 1). Yhtälön avulla voidaan luoda kohteelle upotusarvo ilman, että mallia tarvitsee alustaa uudelleen.

$$\min_{v_{i_0} \in \mathbb{R}^d} \| \mathrm{A}_{i_0} - \mathrm{U}_{V_{i_0}} \| \tag{1}$$

i_0 on kohde, jolle upotus lasketaan
A on arvostelumatriisi, johon upotus lisätään
U on syötettävän kohteen arvostelumatriisi
V_{i_0} on upotusarvo, jonka yhtälö palauttaa. (13.)

WALS-yhtälöllä verrataan uutta kohdetta muihin, jo olemassa oleviin, ja annetaan sille upotusarvo perustuen muihin upotusarvoihin. Se toimiikin vain, jos uudella kohteella on käyttäjäarvioita. Muiden kohteiden tai käyttäjien upotusarvot

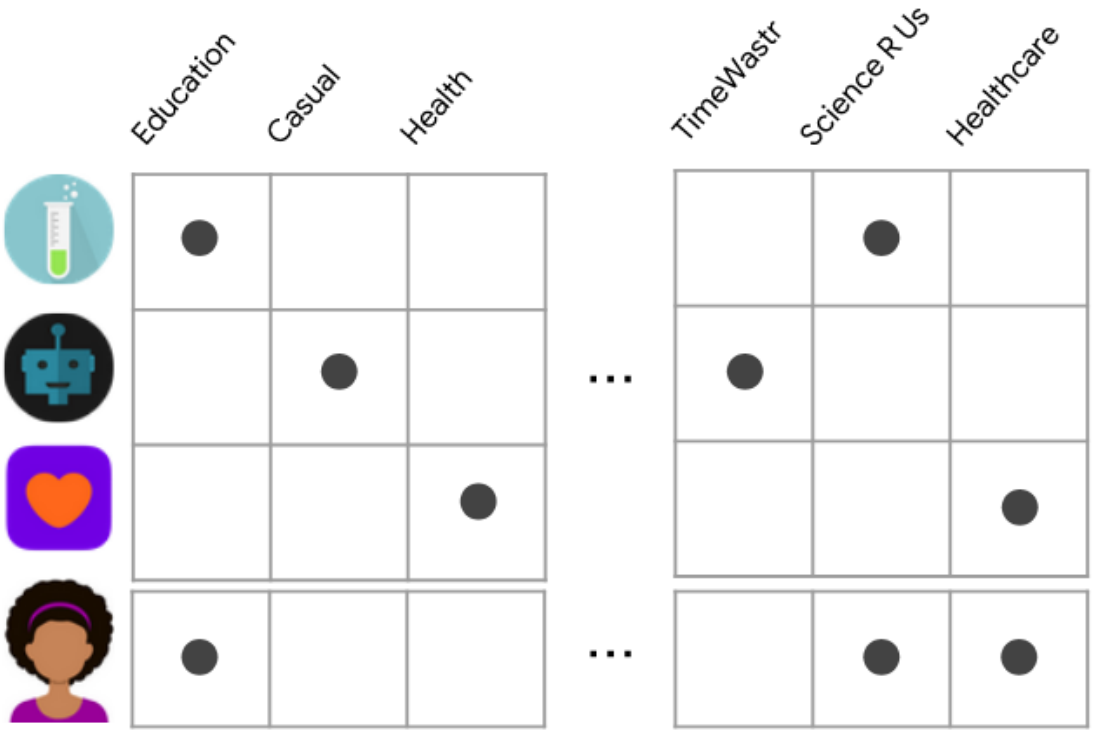
eivät muutu WALS-yhtälön myötä. Tämä tarkoittaa sitä, että upotusarvoa tehdessä järjestelmä ei ota huomioon käyttäjien vaihtuvia mielenkiinnonkohteita tai kohteiden muuttuneita ominaisuuksia. Sen vuoksi yhteisöllisen suodatuksen malli on hyvä alustaa kokonaan uudelleen, vaatimusten mukaan, viikoittain, päivittäin tai jopa tunneittain. (13.)

2.1.2 Sisältösuodatus

Puhtaasti sisältösuodatukseen perustuva järjestelmä ei suosittele kohteita sen mukaan, ovatko muut käyttäjät katsoneet niitä. Sen sijaan se suosittelee niitä kohteita, jotka ovat ominaisuuksiltaan samankaltaisia muiden kohteiden kanssa, joita käyttäjä on aiemmin arvostellut (5, s. 73). Koska sisältösuodatuksen järjestelmän suositukset perustuvat vain käyttäjän arvostelemien kohteiden ominaisuuksiin, siinä on kaksi merkittävää etua yhteisölliseen suodatukseen verrattuna:

- Se ei tarvitse mitään käyttäjätietoja kohteiden suositteluun.
- Se voi suositella käyttäjälle kohteita, vaikka ne eivät kiinnostaisi yhtäkään muuta käyttäjää. (10.)

Sisältösuodatuksen metodeja käyttävä järjestelmä voi olla erittäin ennalta-arvattava. Vaatii paljon hienosäätöä, jotta sen tuottamat suositukset pysyisivät käyttäjälle mielenkiintoisina tai yllättävinä. (10; 5.) Kuvassa 3 on esitetty sisältösuodatuksen toimintamalli, jossa käyttäjälle suositellaan puhelinsovelluksia.



Kuva 3. Sisältösuodatuksen toimintamalli. Käyttäjä on käyttänyt tiede-, terveys- ja opiskelusovelluksia paljon, joten järjestelmän tavoitteena on suositella näitä ominaisuuksia omaavia sovelluksia. (10.)

Mukautuvimman suosittelujärjestelmän saa, kun yhdistää yhteissuodatuksen ja sisältösuodatuksen ominaisuuksia hybridimalliin. Hybridimallin tarkoituksena on täyttää toisen mallin heikoimmat kohdat toisen mallin metodeilla, kuten ottaa yhteissuodatuksessa huomioon kohteiden ominaisuuksia arviointien lisäksi. (5.)

2.2 Suositusten pisteytys ja järjestely

Suosittelujärjestelmissä pisteytykseen hyödynnetään usein koneoppimisen mallia, joka, perustuen käyttäjän jättämiin arvosteluihin, ennustaa käyttäjän arvostelun kohteelle. Järjestelemällä listan siten, että parhaan ennustusarvon saanut kohde on ylimpänä ja huonoimman alimpana, saadaan lista kohteista, jonka ensimmäiset kohteet ovat järjestelymallin mukaan käyttäjälle osuvimpia. (15.) (Kuva 4.) Tekoälymallin kehittäminen sisältää useita vaiheita järjestelmästä riippuen (16, s. 10). Tämän työn kannalta sen kolme tärkeintä vaihetta ovat

- datan valmistelu

- mallin harjoittaminen
- mallin evaluointi.

Datan valmistelu on erittäin merkittävä osa tekoälymallin kehittämistä. Pisteytysmallissa tärkeintä on arvioida, mitä arvosteluarvoa käytetään käyttäjän ja kohteen välillä. Jos harjoituksessa käytetään eksplisiittisiä arvoja, pisteytysmalli pisteyttää käyttäjän tietoisesti hyvin arvostelleita kohteita vastaavat kohteet parhaiten. Kuvassa 4 pisteytysmalli on harjoitettu käyttäjien eksplisiittisten, 0–5 numeroasteikolla annettujen, arvostelujen avulla.

```
predictions = []
for movie_title in unique_movie_titles:
    _,_,predicted_rating = ranking_model({
        "user_id": np.array(['3']),
        "movie_title": np.array([movie_title])
    })
    predictions.append({
        'movie_title': movie_title,
        'predicted_rating': predicted_rating.numpy()
    })

pd.DataFrame(sorted(predictions, key=lambda x: x['predicted_rating'], reverse=True))
```

✓ 4.5s

	movie_title	predicted_rating
0	b'It Happened One Night (1934)'	[[4.058612]]
1	b'Rear Window (1954)'	[[4.019231]]
2	b'Wonderful, Horrible Life of Leni Riefenstahl...	[[4.015783]]
3	b'Citizen Kane (1941)'	[[4.006154]]
4	b'Third Man, The (1949)'	[[4.0036573]]

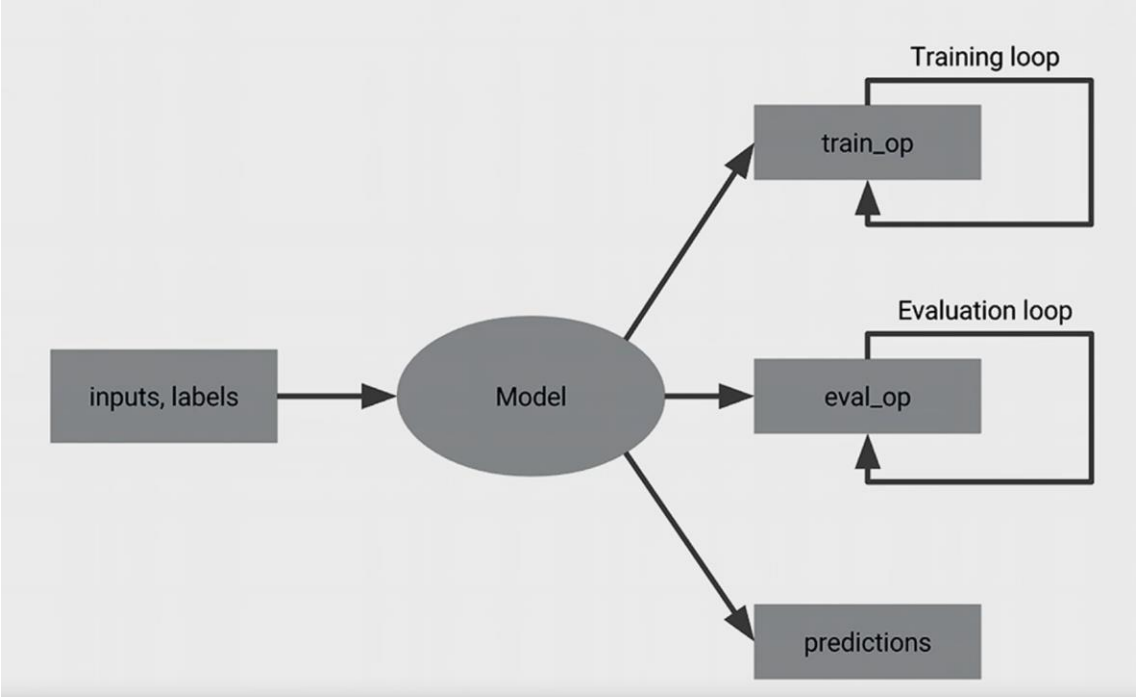
Kuva 4. Järjestelymallin avulla luotu lista pisteytetyistä suosituksista suurin arvosteluennustus ensimmäisenä (15).

Kohteiden pisteytystä varten koneoppimisen malli kehitetään siten, että joka harjoitusvaiheen sukupolvella, koneoppimisen termein epochilla, malli pyrkii ennustamaan käyttäjän ja kohteen välisen arvosteluarvon. Sukupolvien välissä malli vertaa ennustettua arvoa oikeaan arvosteluarvoon ja seuraavassa sukupolvessa korjaa virhettä oikeaan suuntaan. Kun ennalta määritetty määrä sukupolvia on takana, tekoälymallin ennusteiden tarkkuus evaluoidaan, minkä jälkeen, jos tuloksiin ollaan tyytyväisiä, sitä voidaan käyttää ennustusten luontiin ja suositusten järjestelyyn. (16, s. 10.)

3 Tensorflow ja koneoppiminen

Tensorflow on Googlen kehittämä ja vuonna 2015 julkaisema avoimen lähdekoodin koneoppimisen alusta, joka sisältää paljon erilaisia kirjastoja moniin koneoppimisen tehtäviin (17). Tensorflow'n kirjastoja on monelle eri ohjelmointikielelle, kuten Javalle, C++:lle, JavaScriptille ja Pythonille. Tässä työssä Tensorflow'n kirjastoja käytetään Pythonilla.

Tässä työssä tekoälymallit ovat Tensorflow'n Keras-rajapinnan Model-luokkia (18). Yleisesti määriteltynä Model on joko metodi, joka laskee jotain Tensor-datatyypillä, tai X-määrä muuttujia, joita voi päivittää harjoituksen kautta (19). Kaikki Tensorflow'n mallit, jotka harjoitetaan valvotusti, toimivat samalla tavalla: mallille syötettävä data on jaettu syöte- ja nimikedataan, input ja label. Syötedata sisältää datan, jolle malli tulee tekemään ennustuksia. Nimikedata, labels, kertoo, minkä kaltaisia ennustuksia mallin tulee syötetystä datasta palauttaa. (20.) Kuvassa 5 on esitetty yleinen Tensorflow'n mallin toimintarakenne.



Kuva 5. Yleisluontoinen visualisointi Tensorflow-mallin toimintalogiikasta (21).

Mallit voidaan harjoittaa myös ei-valvotusti, eli ilman nimikedataa. Ei-valvotussa harjoituksessa käytetään koneoppimisen metodeja analysoimaan ja

ryhmittelemään, käytännössä nimeämään, syötedatan ominaisuudet. (22.) Tässä työssä kohteiden ja käyttäjien luokittelu on toteutettu mallilla, jonka harjoitus on ei-valvottu. Sen häviömetriikalla, top k categorical accuracy, voi seurata ei-valvotun mallin harjoitusta.

3.1 Harjoitus ja mittaaminen

Harjoitusvaiheen tavoitteena on saada malli oppimaan sille syötetyn datan tunnistus siten, että se voi tehdä ennustuksia datalle, jota se ei ole ennen nähnyt. Mallin ennustuksien tarkkuutta mitataan mittausmetodeilla. Niillä voidaan seurata ennusteiden kehittymistä harjoitussukupolvien yli. Tässä työssä mallin tarkkuutta mitataan kahdella eri mittaustavalla:

- Top K categorical accuracy – Kuinka monta käyttäjän arvostelemaa kohdetta on kierroksen aikana haetussa K-pituisessa kohdelistauksessa. Malli toimii oikein, jos arvo nousee sukupolvien myötä. (23.)
- Root Mean Squared Error – Kertoo mallin tuottamien ennustuksien ja oikeiden arvojen erotuksien neliöiden keskiarvon. Malli toimii oikein, jos arvo laskee sukupolvien myötä. (24.)

Top K categorical accuracy -mittausmetodia käytetään noutomallin harjoituksen seurannassa. Jokaisella harjoituksen sukupolvella malli tuottaa K-määrän suosituksia ja metodin avulla tarkistaa virheen, kuinka paljon käyttäjän arvostelemia tuotteita suosituksissa on. Esimerkiksi top-5 categorical accuracy’n arvo 0,2 kertoo sen, että käyttäjän arvioima kohde on viidessä ensimmäisessä mallin palauttamassa suosituksessa 20 prosentin todennäköisyydellä (25; 23).

Root Mean Squared Error, RMSE-menetelmää, käytetään neuroverkkoennustusmallin harjoituksen seurannassa. Sen tulo on kaikkien ennustuksien ja nimikearvojen erotuksien neliöiden keskiarvo juuressa. Kaava 2 kuvaa RMSE:tä. Mitä pienempi RMSE:n arvo on, sitä parempi malli on ennustamaan käyttötarkoitukseensa sopivia arvoja.

$$RMSE = \sqrt{\sum \frac{(P_i - O_i)^2}{n}} \tag{2}$$

P on ennustettu arvo
O on nimikearvo
i on indeksi
n on kohteiden määrä. (26; 24.)

Harjoittaessa tekoälymallia sille tulee määritellä vähintään sukupolvien määrä ja oppimisnopeus. Jos sukupolvia on liikaa tai malli on liian monimutkainen, se voi päätyä ylisovitetuksi (27). Ylisovittunut, overfit, malli muistaa harjoitusdatan liian tarkasti eikä osaa luoda ennustuksia sen ulkopuolisesta datasta (28; 27). Ylisovittumista voi estää monilla eri keinoilla:

- vähentämällä tekoälymallin harjoituskierroksia
- vähentämällä mallin monimutkaisuutta, kuten poistamalla neuroneita tai neuroverkon kerroksia
- muuttamalla mallin oppimisnopeutta
- regularisoinnilla
- dropout-arvoa käyttämällä. (27.)

Oppimisnopeus, learning rate, määrittää, kuinka herkästi malli korjaa sukupolvien välisiä virheitä. Jos korjauksen määrä on liian iso, malli korjaa painoja sukupolvien välissä liikaa eikä opi sopivia painoarvoja. Päinvastoin liian pienellä korjauksen määrällä malli ei korjaa kerrosten välisiä painoja sukupolvien välissä tarpeeksi. (29.)

Dropout on neuroverkoille tarkoitettu tapa estää ylisovittamista. Jos neuroverkolle on määritetty dropout-arvo, se kytkee satunnaisesti arvon määrän neuroneita pois päältä jokaisella harjoituskierroksella. Koska jokaisella harjoituskierroksella eri neuronit ovat pois päältä, ei ole suurta riskiä, että yksittäiset neuronit vahvistuisivat liikaa. Dropoutin avulla neuroverkko ei tukeudu liikaa yksittäisten neuroneiden opittuihin arvoihin eikä ylisovitu herkästi. (30.)

Regularisoinnin, suomeksi vakinaistamisen, tavoitteena on vähentää harjoituksessa opittuja painoarvoja, jotka ovat liian tarkkoja harjoitusdatalle (32).

LASSO-regressio (kaava 3) ja Ridge-regressio (kaava 4) ovat kaksi Tensorflow'n yleistä vakinaistamismetodia (31).

$$\lambda * |W| \tag{3}$$

$$\lambda * W^2 \tag{4}$$

λ on Tensorflow'n asiayhteydessä vakinaistamisen määrä
W on mallin luoma painoarvomatriisi

Mallin harjoituskierroksen aikana ne luovat mallin luomista painoarvoista rangaistusarvon, joka lisätään mallin häviömetodiin. Jos rangaistusarvo, ja sitä kautta häviömetodin arvo, on suuri, malli korjaa opittuja painoja enemmän pois opitusta kuin tavallisesti. (32; 33.)

3.2 Mallien rakenne

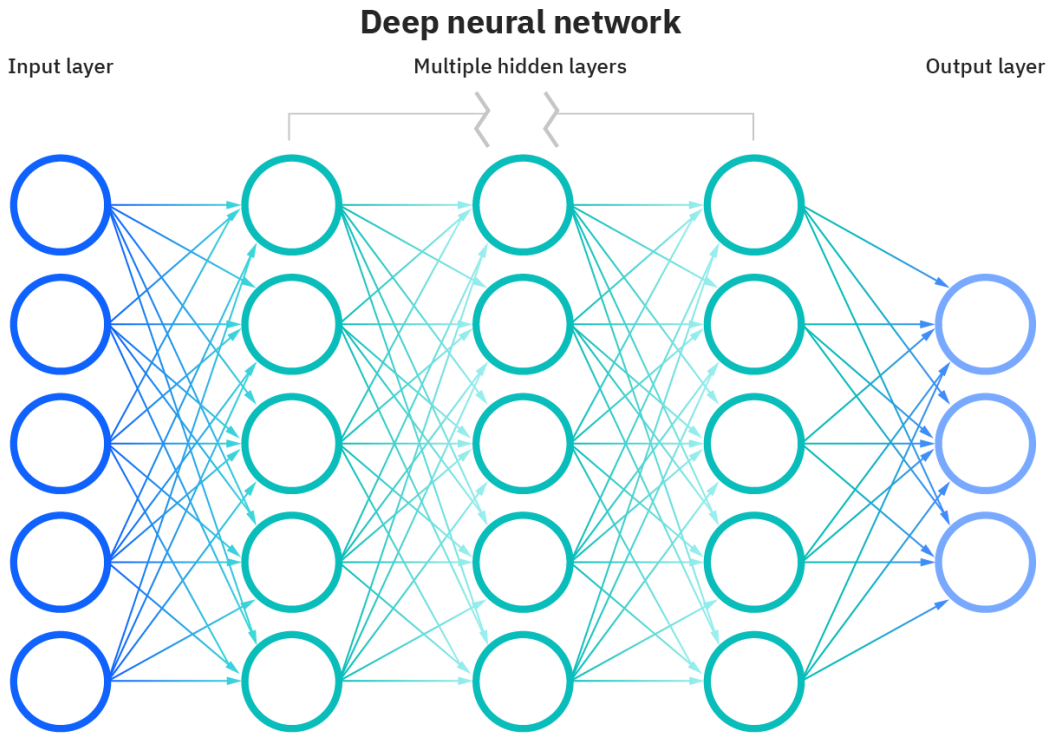
Tekoälymallit koostuvat lähes poikkeuksetta kerroksista, jotka palauttavat mallille syötetyn datan toivotussa muodossa. Kerrokset voivat koostua mistä vain metodeista, jotka hyväksyvät ja palauttavat Tensor-datatyypin arvoja. (19; 34.) (Kuva 6.)

Kuva 6. Esimerkki Tensorflow-mallin kerroksista ja datan kulkusuunnasta. Kuvassa kuvantunnistusmallin rakenne (21).

Tensorflow'n Layers-kirjasto sisältää kerroksia moniin eri tehtäviin. Tässä työssä käytetään neljää eri kerrosta:

- Dense – täysin kytketty neuroverkkokerros
- IntegerLookup – datanvalmistelukerros, joka muuttaa sille syötetyn harvan numerolistauksen jatkuvaksi numerolistaukseksi
- StringLookup – kuten IntegerLookup, mutta tekstidatalle
- Embedding – upotuskerros, joka luo sille syötetyistä positiivisista numeroarvoista X-ulotteisia vektoreita.

Täysin kytketyt kerrokset, Dense-layers, koostuvat nimensä mukaisesti neuroneista, joista kaikki ovat yhteydessä toisiinsa (35). Neuroverkon sisältämän tekoälymallin koulutusvaiheessa malli säätelee neuroneiden välisiä yhteyksiä siten, että haluttuihin tuloksiin johtaneet neuroniyhteydet vahvistuvat ja päinvastoin virheellisiin tuloksiin johtaneet yhteydet heikentyvät (29) (kuva 7).



Kuva 7. Täysin kytketty neuroverkko, jonka jokaisessa kerroksessa on pisteiden määrä neuroneita (36).

Yksittäinen neuroverkon neuroni laskee syötedatansa painokertoimilla aktivointifunktion arvon ja syöttää sen seuraaville neuroneille (35) (kaava 3). Aktivointifunktioita on erilaisia, mutta kaikkien tavoite on sama: määritellä miten neuroverkko oppii ymmärtämään harjoitusdataansa (37). Harjoituksen aikana malli päivittää aktivointifunktion gradientin perusteella neuronin painokertoimet, jotka muuttavat neuronin palautusta seuraavalla harjoituskierroksella (38). Kaava 5 esittää, miten neuronin palautus lasketaan.

$$output = activation((input \cdot kernel) + bias) \tag{5}$$

$input$ on neuronin syötedata
$kernel$ sisältää opitut painoarvot
$bias$ on mallin luoma arvo, jolla se voi vaikuttaa neuronin aktivointiin

ReLU, Sigmoid ja Tanh ovat yleisiä aktivointimetodeja. ReLU, Rectified Linear Unit, on nopein aktivointimetodi. Se palauttaa joko neuronille syötetyn arvon tai nollan, jos arvo on sen alapuolella. (38.) (Kaava 6.)

$$f(x) = \max(x, 0) \tag{6}$$

x on funktiolle syötettävä arvo

ReLU:n suurin ongelma on kuolleiden neuronien riski. Jos neuroni palauttaa nollan, on sen gradienttikin nolla, eli neuronin painoarvoja ei päivitetä seuraavalle kierrokselle ja neuroni jää hyödyttömäksi. (39.) Sigmoid on aktivointimetodi, joka palauttaa aina arvon 0:n ja 1:n väliltä. Se on ReLU:a hitaampi suorittaa, mutta sopii tiettyihin tapauksiin ReLU:a paremmin, kuten todennäköisyyden ennustamiseen. (40; 38.) (Kaava 7.)

$$\delta(x) = \frac{1}{1+e^{(-x)}} \tag{7}$$

Koska Sigmoid palauttaa aina arvon nollan ja yhden välillä, on mahdollista, että Sigmoidin palauttama gradientti on niin lähellä nollaa, että gradientin korjaaminen harjoitusten välillä on käytännössä mahdotonta, eli neuronista tulee erittäin

lineaarinen. Tätä ongelmaa kutsutaan nimellä katoavan gradientin ongelma, vanishing gradient problem. (40.) Tanh, hyperbolinen tangentti, on samanlainen aktivointimetodi kuin Sigmoid, mutta palauttaa arvon miinus yhden ja yhden välistä (38). Tanh-metodillakin on riski katoavan gradientin ongelmaan.

Hyvä tapa korjata aktivointimetodien ongelmia on vaihtaa tarkoitukseen epäsopiva aktivointimetodi toiseen, joka toimii paremmin mallin käsittelemän datan kanssa (40).

Upotuskerrokset, Embedding-layers, muuttavat niille syötetyt numeroarvot niitä vastaaviksi vektoreiksi (41). Tässä työssä käytetään upotuskerroksien luomia 32-ulotteisia upotusvektoreita profiloimaan sekä käyttäjät että kohteet. Kohteiden noutovaihe käyttää upotusvektoreita tuomaan käyttäjälle suosituksia perustuen esimerkiksi vektoreiden väliseen etäisyyteen (luku 2.1.1). Kuvassa 8 on esitetty, minkälaisen upotusvektorin suosittelumalli on luonut mieskäyttäjästä 3.

```
user_embedding = recommender.user_model({
    'user_id': np.array([3]),
    'user_gender': np.array(['male'])
})

user_embedding
[38]  ✓ 0.6s
...  <tf.Tensor: shape=(1, 32), dtype=float32, numpy=
     array([[ 0.12874627, -0.243021  , -0.8834963 , -0.21243191,  0.537373  ,
             -0.35991514,  0.28392082, -0.9143469 ,  0.39357972,  0.54288805,
             -0.02166152, -0.39694592,  0.64907575,  0.22477846, -0.44461036,
              0.14298259, -0.6283138 , -0.4119809 , -0.7286689 ,  0.07410888,
             -0.30756328, -0.46390462,  0.08429161,  0.41143793, -0.36167225,
             -0.7881881 ,  1.0375428 ,  0.57810754, -0.36274886,  0.28510892,
              0.06402567,  0.29895476]], dtype=float32)>
```

Kuva 8. Suosittelumallin luoma upotusvektori käyttäjästä 3, jonka sukupuoli on mies (41).

Koska upotuskerrokselle syötettävän datan tulee olla numeerisessa muodossa, on yleistä, että sille syötettävä data käsitellään IntegerLookup- ja StringLookup-kerroksien, tai vastaavien, avulla (22). Ne muuttavat syötetyn datan jatkuviksi

numerolistauksiksi (42). Kuvassa 9 on esimerkki, miten lookup-kerrokset muokkaavat niille syötetyn datan upotuskerrokselle sopivaksi.

```
ids     = [1, 2, 5, 7, 10, 39]
genders = ['male', 'female', 'other']

str_lookup = tf.keras.layers.StringLookup(vocabulary=genders, mask_token=None)
id_lookup  = tf.keras.layers.IntegerLookup(vocabulary=ids, mask_token=None)

print('genders: ', str_lookup(genders).numpy())
print('ids ',      id_lookup(ids).numpy())
✓ 0.6s

genders:  [1 2 3]
ids  [1 2 3 4 5 6]
```

Kuva 9. Lookup-kerrokset muuttavat niille syötetyn datan jatkuvaksi numerolistaukseksi, joka on sekä upotuskerrokselle oikeassa muodossa että tehokas käsiteltävä (42).

Kaikki edellä mainitut kerrokset yhdistettynä niitä käyttävä tekoälymalli on valmis profiloimaan sille syötettyjä kohde- ja käyttäjätietoja upotuskerroksien avulla sekä luomaan arvosteluennustuksia tiiviin neuroverkon avulla (7).

4 Suosittelujärjestelmä mobiilisovellukseen

Insinöörityössä kehitettiin uusi tekoälysovellus tilaajayrityksen, Jemman, aiemman suosittelujärjestelmän tilalle, joka loi suosituksia yhteisöllisen suodatuksen menetelmin käyttäen käyttäjien katseluaikaa arvioina. Vanhan järjestelmän suurimmat heikkoudet olivat samat kuin yhteisöllisessä suodatuksessa yleisesti: se ei tiennyt tuotteista, joilla ei ollut näyttökertoja, eikä osannut suositella uusia tuotteita ennen uudelleenkoulutusta. Käyttäjä- tai tuotetietojen lisääminen siihen järjestelmään olisi myös ollut erittäin vaikeaa, ellei mahdotonta.

Yhtenä työn tavoitteena oli mahdollistaa tuotteiden ja käyttäjien lisätietojen huomioiminen. Käyttäjien lisätiedot voivat olla esimerkiksi sukupuoli, mielenkiinnon kohteet ja lasten määrä. Ensimmäinen lisättävä käyttäjän tieto oli sukupuoli, koska se on yksinkertainen tekstiarvo, jonka lisääminen ja testaaminen oli

muihin käyttäjätietoihin verrattuna yksinkertaista. Tuotteiden lisätietoja ovat esimerkiksi tuotemerkki ja tuoteryhmä, mutta tässä vaiheessa kehitystä niiden lisätiedot jätettiin huomiotta. Koska järjestelmää kehitetään jatkossa aktiivisesti käyttäjäpalautteen ja sovelluksen tarpeiden mukaan, oli tärkeää, että jatkokehittäminen, kuten käyttäjä- ja tuotetietojen lisääminen, on uudessa suosittelujärjestelmässä helppoa ja järjestelmän toiminta ja rakenne ovat selkeät.

Toteutettavan suosittelujärjestelmän tuli osata kaikki suosittelujärjestelmän tehtävät, eli suositusten haku, pisteytys ja järjestely siten, että järjestelmän ennustusten mukaisesti kiinnostavimmat tuotteet tarjotaan käyttäjälle ensimmäisenä. Kaikkiin suosittelujärjestelmän tehtäviin käytettiin Tensorflow Recommenders -kirjaston metodeja, jotka palvelevat suoraan työn tarkoitusta (19).

Koska suositusten haku tehdään rajapintakutsulla, suosittelujärjestelmäsovellus toteutettiin Flask-rajapintaohjelmistokehystä käyttäen. Flask on nopea ja erittäin suosittu Pythonille kehitetty rajapintaohjelmistokehys, joka mahdollistaa helposti muokattavan ja skaalattavan rajapintasovelluksen kehityksen ja julkaisun. (43.)

Sovelluksen tärkeimmät rajapintakutsut ovat /train ja /recommend, harjoitus ja suosittelu. Harjoituskutsussa haetaan ja käsitellään tekoälymallin käyttämä data ja harjoitetaan tekoälymalli. Se palauttaa tekoälymallin harjoitushistorian ja -yhteenvedon, joka sisältää factorized top k:n ja RMSE:n mittausarvot harjoitussukupolvilta ja viimeisimmältä sukupolvelta. Suosittelukutsu palauttaa listan ID:itä, jotka ovat tekoälymallin suosituksia pyydetylle käyttäjälle. Kutsuja lisättiin ja muokattiin tarpeiden mukaan järjestelmän kehittämisen ohella.

4.1 Tiedon haku ja käsittely

Suositusten luontia ja pisteytystä varten suosittelujärjestelmä tarvitsee dataa kahdesta eri rajapinnasta, neljällä erillisellä kyselyllä:

- käyttäjät – sovelluksen käyttäjien sukupuolet
- tykkäykset – käyttäjien antamat tykkäykset tuotteille

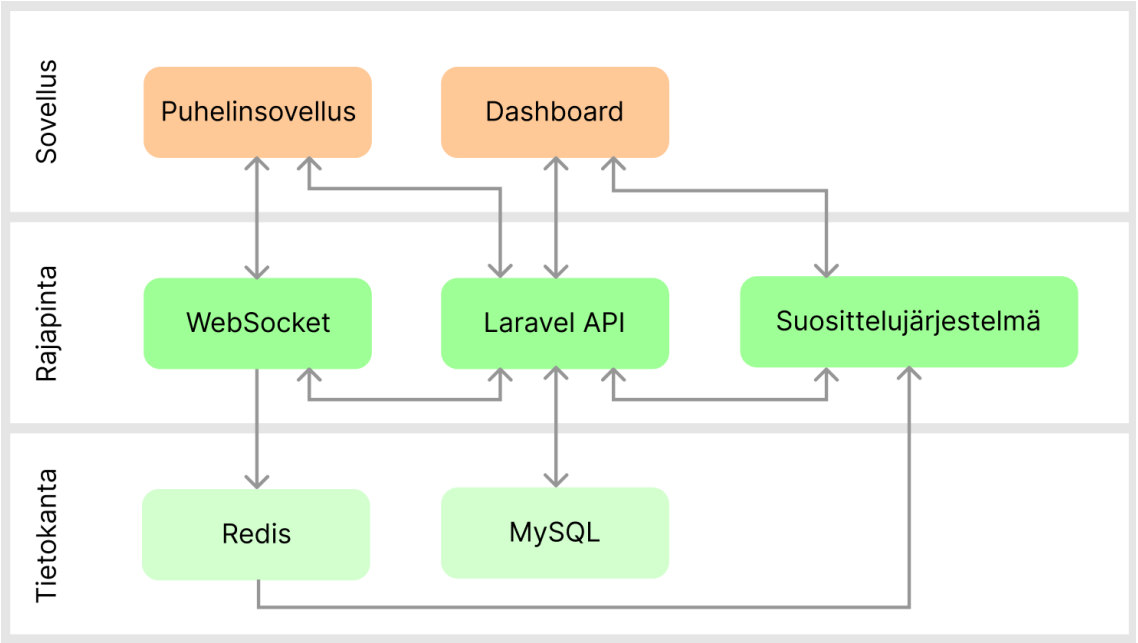
- tuotteet – kaikki sovelluksen olemassa olevat tuotteet
- katseluajat – käyttäjien käyttämä aika tuotenäkymässä.

Jemma-järjestelmän käyttäjä-, tykkäys- ja tuotedata on varastoitu MySQL-tietokantaan, jonka rajapintasovellus toimii Laravel-ohjelmistokehyksessä. Laravel on PHP:lle kehitetty rajapintaohjelmistokehys, joka mahdollistaa helpon rajapintasovellusten kehittämisen ja kommunikoinnin mm. tietokantapalvelimien kanssa (44).

Aika-arvodata on varastoitu, erillisenä MySQL-rajapintasovelluksesta, Redis hash -tietokantaan. Redis on välimuistia käyttävä tietokantajärjestelmä, eli in memory data structure store (45). Välimuistin käyttö mahdollistaa erittäin nopean tiedonhaun verrattuna perinteisempiin kiintolevyratkaisuihin (46). Koska Rediksen hash-tietokannan avaimet ovat tekstityyppiä, käyttäjien aikadata on varastoitu avaimilla muodossa ”käyttäjä-ID:tuote-ID” views-nimiseen hash-tauluun (45). Avaimien takana oleva arvo on katseluaika millisekunteina.

Kun käyttäjä on kirjautunut sisään Jemma-puhelinsovellukseen, on sovellus jatkuvassa WebSocket-yhteydessä Jemman WebSocket-palvelimeen. Se huolehtii kahdesta asiasta: ilmoitusten lähettämisestä ja, tämän työn kannalta tärkeämpänä, katseluaikojen tallentamisesta. Katseluajat tallennetaan erikseen muusta Jemman käyttämästä datasta, koska implisiittisesti jätettyä katseluaikadataa tulee olemaan enemmän kuin mitään käyttäjien eksplisiittisesti syöttämää dataa. Pyrkimyksenä on pitää Jemman MySQL-tietokanta irtonaisena implisiittisestä seurantadatasta.

Kun käyttäjä avaa myynnissä tai vuokrattavissa olevan tuotteen koko näyttöön puhelinsovelluksessa, aikaseurantaa tekevä sovelluksen sisäinen aikaseuranta-luokka ottaa sen hetken aikaleiman muistiin. Kun käyttäjä poistuu näkymästä, aikaseuranta-luokka laskee poistumisajan ja aloitusajan erotuksen ja tieto lähetetään WebSocket-yhteyden kautta Redis-tietokantaan. Kuvassa 10 on esitetty datan kulku Jemman sovellusten ja rajapintojen sekä tietokantapalvelimien välillä.



Kuva 10. Jemma-järjestelmän rakenne. Suosittelujärjestelmä kommunikoi Jemman API:n ja Redis-kannan kanssa.

Suosittelujärjestelmäsovelluksen tiedonhaku tehdään aina tekoälymallin koulutuksen kanssa samaan aikaan, vähintään kerran päivässä. Koska tiedonhaun ja mallin koulutuksen nopeus ei näy puhelinsovelluksen käyttäjille mitenkään, niiden nopeudella ei ole käytännössä mitään väliä. Suosittelujärjestelmän tulee olla nopea vasta suosittelumetodissa, jossa palautuksen nopeus vaikuttaa suoraan puhelinsovelluksen etusivun päivitysnopeuteen.

Harjoitusvaiheessa tekoälysovellus teki neljä eri kyselyä kaikelle tarvitsemalleen datalle. Kun kaikki sovelluksen tarvitsema data oli noudettu, Redis-kannasta haettuun aikadataan yhdistettiin käyttäjien sukupuolet ja tykkäykset. Lopputuloksena oli taulu, joka sisälsi jokaisen katseluaikatiedon käyttäjä- ja tuote-ID:n, käyttäjän sukupuolen ja tiedon tykkäyksestä (kuva 11). Yhdistetyn käyttäjätietotaulun lisäksi sovellus teki tuotetaulun, joka sisältää kaikki sovelluksen tuotteet, nekin, joita kukaan ei ole vielä nähnyt tai arvostellut. Esimerkkikoodi 1 näyttää, miten tieto parsittiin Pandas dataframe -muotoon, jotta sen käsittely olisi mahdollisimman yksinkertaista. Jokaisen sovelluksessa tehtävän rajapintakyselyn vastaus muutettiin samalla tavalla dataframe-muotoon.

```
data = Redis.hgetall('views')
user_ids, classified_ids, times = [], [], []
for key in data:
    user_ids.append(int(key.split(':')[0]))
    classified_ids.append(int(key.split(':')[1]))
    times.append(int(data[key]))

df = pd.DataFrame({
     'user_id': user_ids,
     'classified_id': classified_ids,
     'time': times
})
```

Esimerkkikoodi 1. Katseluaikojen tiedon haku Redis-tietokannasta ja datafra-men luominen.

Koska katseluajat saattavat erota toisistaan huomattavasti ja ääriarvot vääristäisivät sovelluksen normaalikäytöstä saatavat arvot, yli 30 sekunnin ajat muutettiin 30 sekuntiin. Tämän jälkeen ajat normalisoitiin desimaaliksi nollan ja yhden välille, jotta käsiteltävä data olisi kauttaaltaan samassa suuruusluokassa. Syötettävän datan normalisoinnilla nopeutettiin mallin harjoitusta ja vähennettiin epätoivottuja ilmiöitä, kuten pyöristysvirheitä tai yksittäisten ominaisuuksien liiallista korostumista. Kuva 11 esittää katseluaika-taulua, jossa on yhdistettynä kaikki käyttäjän data katseluaikaan.

index	user_id	classif…	time	has_lik…	user_ge…
0	1088	3549	0.410647…	0	other
1	1159	2059	0.150006…	0	female
2	1143	3766	0.487514…	0	male
3	1085	3567	0.995842…	1	other
4	1172	3243	0.232627…	0	other
5	2	3769	0.420458…	0	male
6	1143	3450	0.879579…	1	male

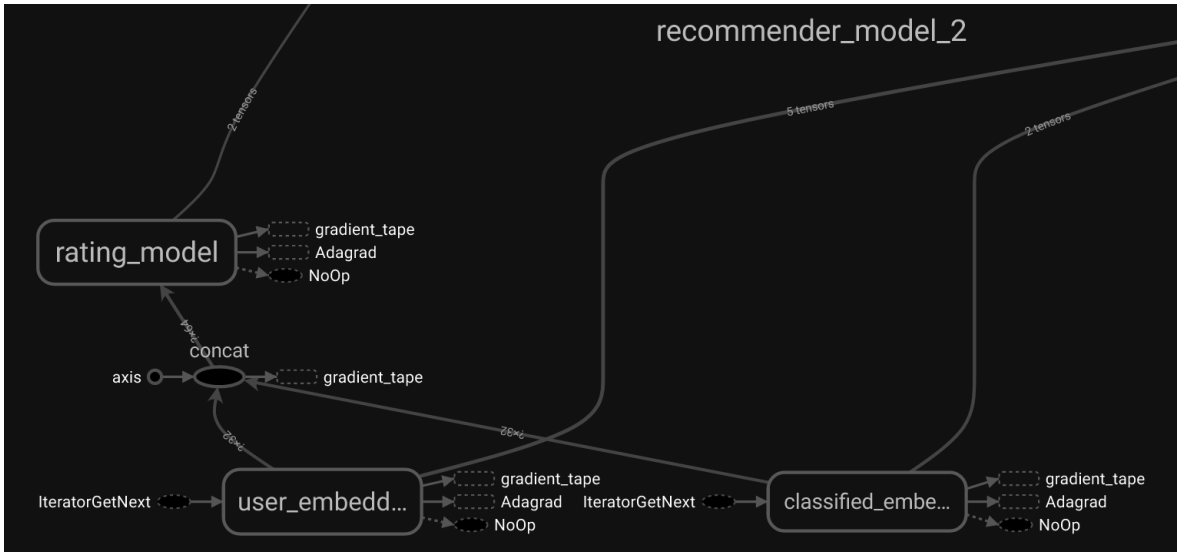
Kuva 11. Suosittelumallin käyttämä data yhdistettynä ja normalisoituna yhteen tauluun.

Käyttäjän implisiittistä katseluaika-arvostelua käytettiin sekä suosittelujärjestelmän haku- että pisteytysvaiheessa. Hakuvaiheessa katselukertaa, eli binääristä tietoa aika-arvon olemassaolosta, käytettiin positiivisena mielenkiinnon osoituksena käyttäjältä tuotteelle. Pisteytysvaiheessa aika-arvoa käytettiin arvioimaan

käyttäjän mielenkiinto tuotteelle. Pisteytyksessä myös käyttäjän jättämät tykkäykset olivat isossa painoarvossa. Jos käyttäjä tykkäsi tuotteesta, eli antoi koko näytön näkymässä tuotteelle sydämen, käyttäjän katseluaika tuotteelle kaksinkertaistettiin. Näin käyttäjän tykkäykset vaikuttavat siihen, mitä tuotteita hän näkee ensimmäisenä.

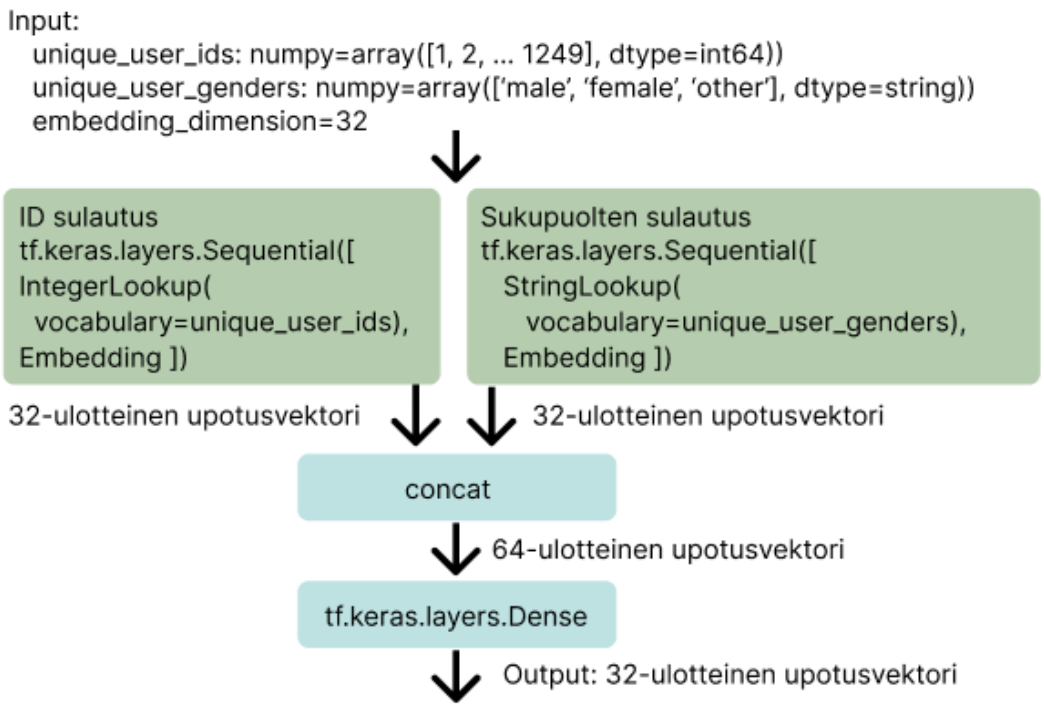
4.2 Tekoälymalli

Insinöörityössä tehty tekoälymalli kehitettiin sisältämään tuote- ja käyttäjäupotukset sekä pisteytystä tekevän neuroverkko-pisteytysmallin (kuva 12). Harjoituksen aikana se sääteli jokaisen osan painoarvoja siten, että pisteytysmalli ennusti tarkasti käyttäjän ja tuotteen välisen katseluajan ja toisiaan vastaavien käyttäjien ja tuotteiden upotukset olivat lähellä toisiaan. Harjoituskierrosten seurantaan malli käytti RMSE- ja Top K categorical accuracy -metodeja.



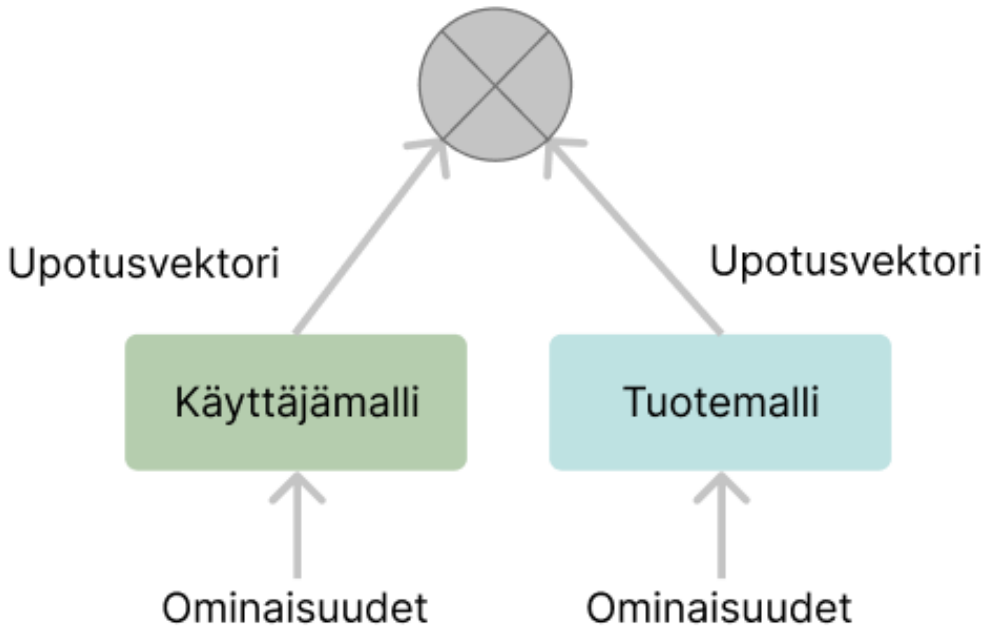
Kuva 12. Koko suosittelumalli. Se sisältää sulautuskerrokset käyttäjille ja tuotteille sekä neuroverkon, rating_model, pisteytysennustuksia varten.

Käyttäjä- ja tuoteupotukset olivat omia mallejaan, joiden syöttödata on niille valitut ominaisuudet, tässä tapauksessa vain käyttäjien sukupuolet ID:n lisäksi. Mallit palauttavat niille määritellyt upotusvektorit. Koska upotukset luotiin omina malleinaan, niiden jatkokehittäminen on jatkossa yksinkertaista. Kuva 13 esittää käyttäjämallin rakennetta.



Kuva 13. Käyttäjämalli, joka profiloi käyttäjän ID:n ja sukupuolen 32-ulotteiseksi upotusvektoriksi, jota voi hyödyntää suoraan noutomallissa (Lähteen 49 pohjalta).

Mallin noutovaiheen toteutustapaa kutsutaan two-tower retrieval modeliksi eli kahden tornin noutomalliksi. Kahden tornin noutomallissa toinen torni, query-tower, esittää käyttäjää, jolle suosituksia haetaan, ja toinen torni, candidate-tower, esittää kohdetta, jota noudetaan. (49.) Kahden tornin noutotavassa molemmat tornit ovat saman mallisia upotusvektoreita. (Kuvat 13 ja 14.) Kahden tornin suositukset perustuvat käyttäjää ja tuotteita esittävien vektoreiden väliseen etäisyyteen, eli mitä lähempänä käyttäjän vektori on tuotteen vektoria, sitä varmemmin tuote on käyttäjälle sopiva (8).



Kuva 14. Kahden tornin suosittelumallin rakenne. Sekä käyttäjien että tuotteiden ominaisuudet voivat olla mitä vain nimistä väreihin, kunhan ne ovat oleellisia suosittelumallin asiayhteydessä. Tornien yhdistymiskohta voi esittää mitä tahansa vektoreiden etäisyydenlaskentametodia, tässä tapauksessa pistetuloa. (Lähteen 49 pohjalta.)

Järjestelmän noutovaihe toteutettiin BruteForce-metodilla, joka laskee pyydetyn käyttäjäupotuksen ja jokaisen järjestelmässä olevan tuoteupotuksen etäisyyden, eli pistetulon, ja palauttaa ne tuotteet, jotka ovat kaikista lähimpänä (49). BruteForce on melko tehoton noutometodi, mutta se oli työn tekohetkellä ainoa, jota voi käyttää Windows-pohjaisessa ympäristössä. Jos BruteForce osoittautuu testattaessa liian hitaaksi, sen voi vaihtaa vain Linux-ympäristössä toimivaan ScaNN-metodiin, sillä tekoälyn tuotantopalvelin on Linux-pohjainen. ScaNN on Approximate Nearest Neighbour -noutometodi, joka on noin 10 kertaa nopeampi kuin BruteForce. Nopeutensa lisäksi sen palautukset ovat lähes yhtä tarkkoja kuin BruteForcen. (47; 50.)

Tekoälymallin neuroverkko oli vastuussa noutovaiheen palauttamien tuotteiden pisteytyksestä. Neuroverkko koostui kolmesta kerroksesta, joista kaksi koostui ReLU-aktivointimetodeja käyttävistä neuroneista ja viimeinen yhdestä lineaarisesta neuronista, joka palautti ennusteen käyttäjän katseluajasta. Harjoituksessa neuroverkon syötedata oli käyttäjän ja tuotteen upotusvektorit ja

nimikedatana katseluaika, eli malli opetettiin ennustamaan käyttäjä- ja tuoteupotusten välistä katseluaikaa.

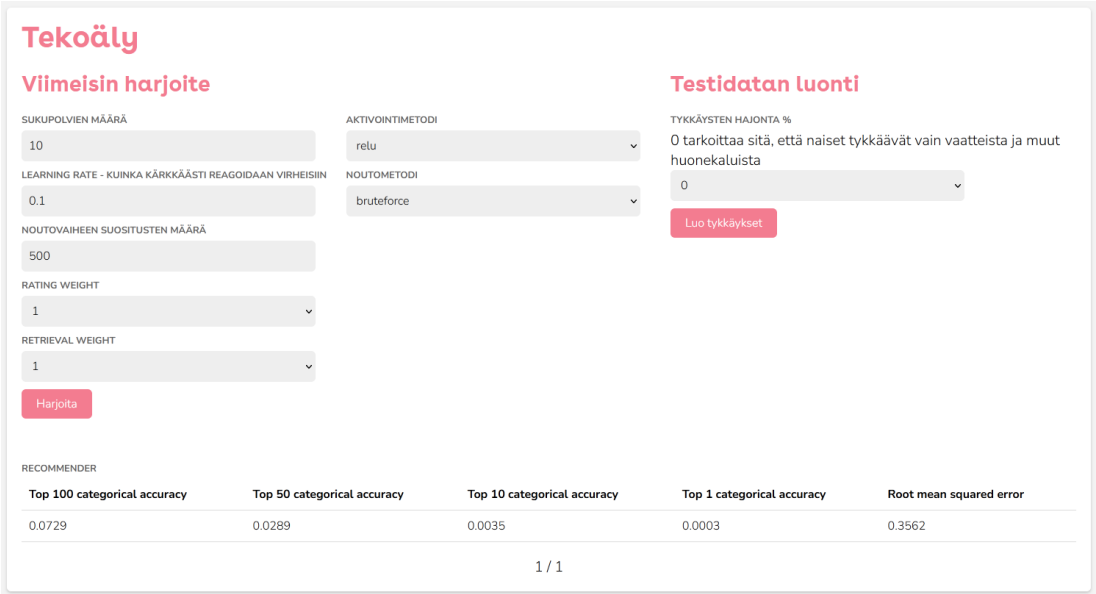
4.3 Testaaminen

Koska Jemma-sovellusta ei ollut vielä julkaistu, mallin testaaminen piti tehdä manuaalisesti generoidulla datalla. Testaamista varten kehitettiin metodit tuotteiden ja näyttökertojen lisäämiselle. Sovellukseen lisättiin 1000 tuotetta tuoteryhmittäin, kolmeen eri tuoteryhmään:

- naisten vaatteet
- työkalut
- huonekalut.

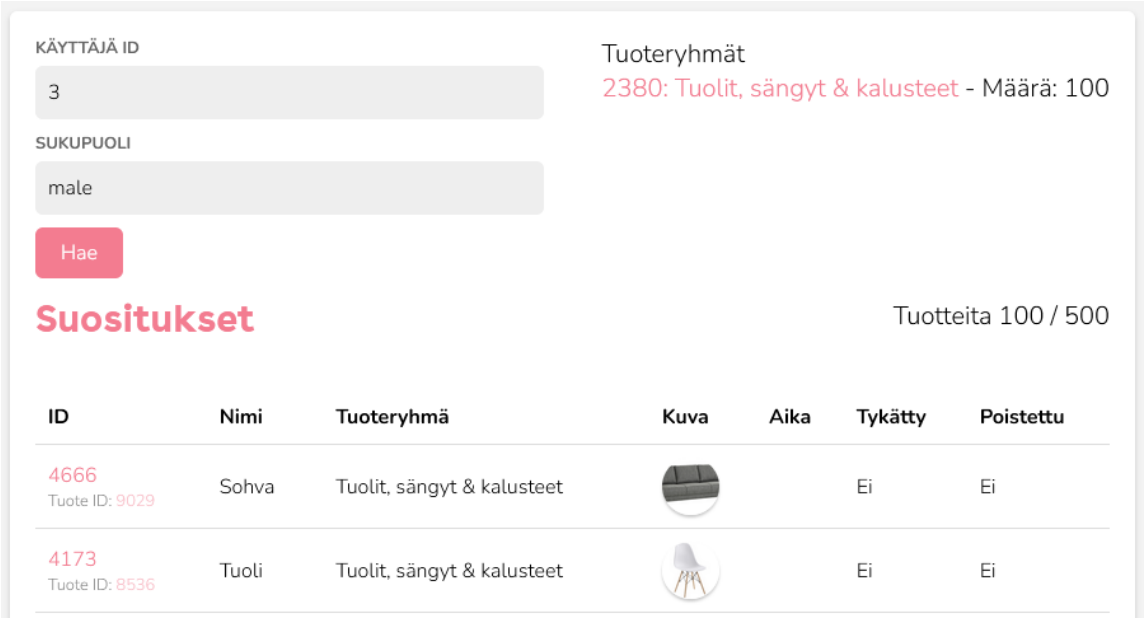
Jokaiselle sovelluksen käyttäjälle luotiin 50–200 näyttökertaa ja jokaiselle näyttökerralle tykkäys 50 prosentin todennäköisyydellä satunnaisesti, jos näyttökerran aika ylittää 10 sekuntia. Näyttökerrat osoitettiin käyttäjän sukupuolen mukaan tiettyjen tuoteryhmien tuotteille. Jotta käyttäjät olisi testatessa mahdollisimman helppo erottaa toisistaan, katselukerrat jaettiin siten, että naiset katsoivat naisten vaatteita ja miehet tai muunsukupuoliset katsoivat huonekaluja. Työkalut jäivät kokonaan huomiotta, eli suosittelumallin ei tulisi suositella niitä kenellekään. Katselukertojen luontimetodissa oli hajautusparametri, jolla voi vaihtaa mielenkiinnon kohteet sukupuolien välillä päikseen parametrin mukaisessa vaiheessa näyttökertojen luontia. Sen avulla voitiin kokeilla, suositteleeko malli miehille myös naistenvaatteita n. puolet ajasta, jos näyttökertojen lisäys oli tehty 50 prosentin hajautuksella.

Jemmalla on käytössä React-ohjelmistokehyksellä luotu verkkosovellus, jonka avulla voidaan seurata ja hallita kaikkea puhelinsovellukseen liittyvää tietoliikennettä sekä mm. sopimus-, käyttäjä- ja tuotedataa. Tekoälysovelluksen tarkkailua varten Dashboard-verkkosovellukseen lisättiin näkymä, jossa voitiin luoda mielenkiinnonkohdedata halutulla hajautuksella, harjoittaa tekoälymalli eri parametreilla ja tarkkailla mallin harjoitusdataa. (Kuvat 15, 16 ja 17.)



Kuva 15. Tekoälynäkymän testidatan luonti ja harjoituksen parametrien muokkaus. Alarivillä mallin results-data, joka kertoo mallin pisteytys- ja noutomallien tarkkuuden top k categorical accuracy- ja RMSE-mittausmetodeilla.

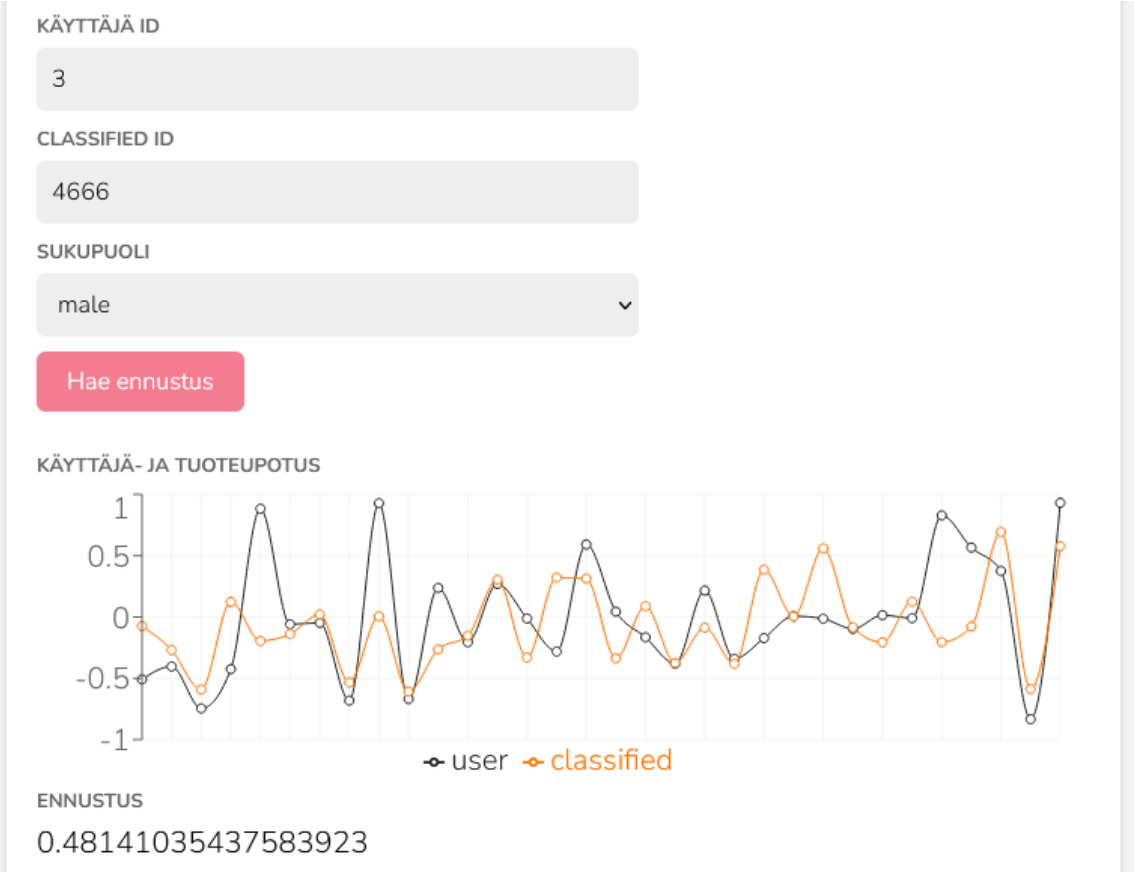
Kutsuttaessa tekoälysovelluksen suosittelumetodia se palauttaa vain listan ID:itä. Saatuaan tuote-ID:t Dashboard tekee sovelluksen palvelimelle kyselyn, jolla se hakee kaikki tuotteet, jotka täsmäävät palautettuihin ID:ihin. Näin nähtiin, mitä tuotteita malli suosittelee ja mihin tuoteryhmään ne kuuluvat. Ensimmäisen testin testausdata generoitiin nollan prosentin hajautuksella, eli testi onnistuu, jos miehille suositellaan vain huonekaluja ja naisille vain vaatteita. Kuva 16 esittää ensimmäisen testin palautusta.



Kuva 16. Suositusten haku käyttäjä-ID:n ja sukupuolen perusteella. Mieskäyttäjä 3 on katsonut vain huonekaluja.

Ensimmäinen testi onnistui, sillä malli suositteli tuotteita vain siitä tuoteryhmästä, jota käyttäjä oli katsonut. Testi todisti sen, että mallin harjoitetut upotusvektorit käyttäjille ja tuotteille täsmäävät kohtuullisen hyvin ja kahden tornin noutomalli toimii alustavasti.

Mallin ennustusten tarkkailuun Dashboard-näkymään lisättiin ikkuna, jolla voi tarkkailla syötetyn käyttäjän ja tuotteen upotusvektoreiden eroa ja niille ennustettua aikaa. Ikkuna on esitetty kuvassa 17, jossa haettiin edellisen testin käyttäjän ja ensimmäisen suositellun tuotteen upotukset.

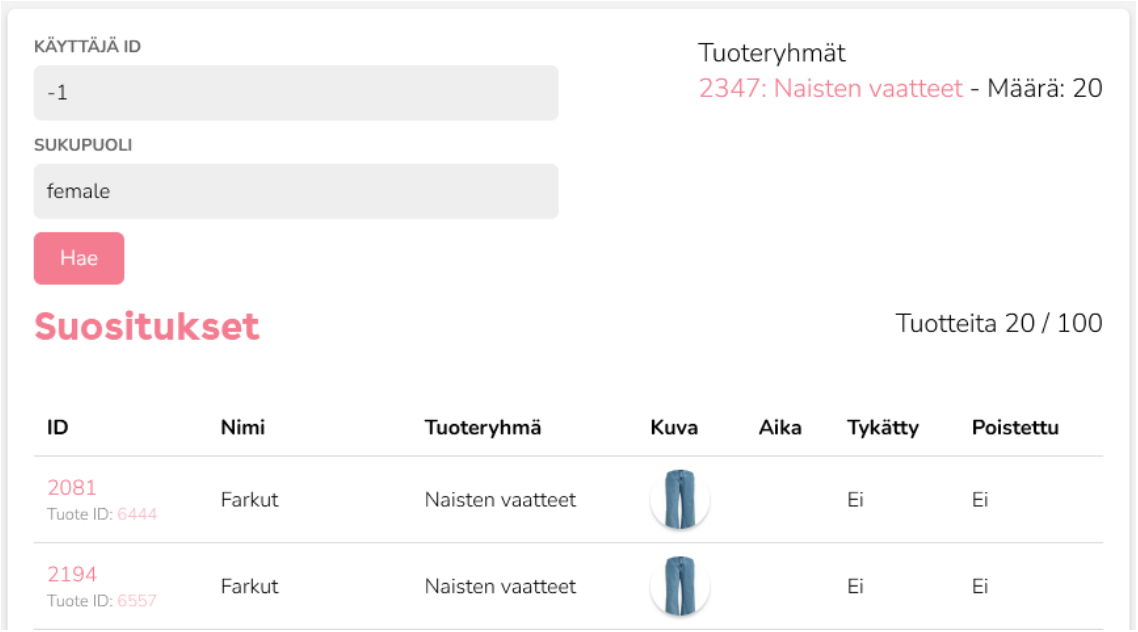


Kuva 17. Ennustus- ja hakumallin testaus. Tuote 4666 on mallin tuottamissa suosituksissa ensimmäisenä katseluajalla 0,481. Upotus-visualisoinnissa käyttäjän kuvaaja on musta ja tuotteen oranssi.

Suosittelujärjestelmää testattiin keinotekoisella datalla vielä kahdella eri tavalla:

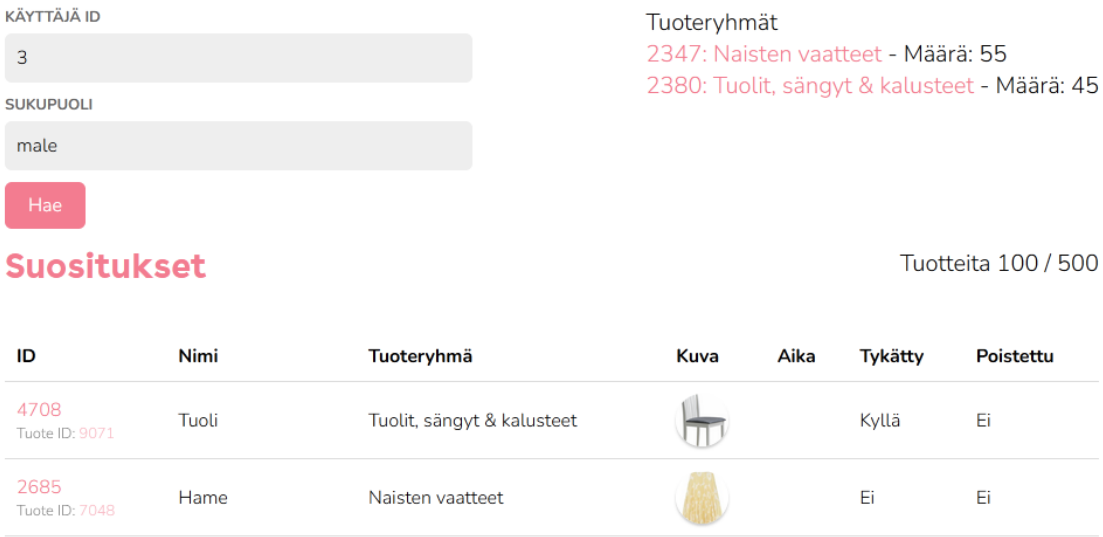
- Mitä malli suosittelee tuntemattomalle käyttäjälle?
- Miten malli luo suositukset katseludatalla, joka on generoitu hajautuksella?

Sovelluksen asiayhteydessä tuntemattomat käyttäjät ovat niitä käyttäjiä, jotka ovat rekisteröityneet Jemma-sovellukseen vasta sen jälkeen, kun malli on viimeksi koulutettu, eli niiden ID:ille ei ole tehty upotusarvoja. Mallin pitäisi toimia niin, että tuntemattomille käyttäjille suositellaan tuotteita puhtaasti sukupuolen perusteella. Kuvassa 18 ovat testin tulokset.



Kuva 18. Suositukset naiskäyttäjälle, jota järjestelmä ei tunne, eli suositusten hakuun käytetään vain käyttäjän sukupuolta.

Jos katseludata on tehty 50 prosentin hajautuksella, eri tuoteryhmien tuotteiden upotusvektoreiden ei kuuluisi erota toisistaan kovin merkittävästi, sillä kaikki käyttäjät ovat katsoneet molempia tuoteryhmiä keskimäärin yhtä paljon. Kutsuttaessa suosituksia hajautetulla datalla harjoitetulle mallille, suositusten pitäisi olla n. 50 prosentin verran huonekaluja, loput naisten vaatteita. Kuvassa 19 ovat esitettynä hajautetulla datalla harjoitetun mallin suositukset.



Kuva 19. Suositukset mieskäyttäjälle, joka on katsonut kalusteita ja naisten vaatteita 50 prosentin hajonnalla.

Jokainen testi vastasi odotuksia: malli suositteli testidatan mukaisia tuotteita sekä tunnetuille että tuntemattomille käyttäjille. Seuraava vaihe oli julkaista suosittelujärjestelmä tuotantopalvelimelle ja lisätä se osaksi puhelinsovelluksen etusivua.

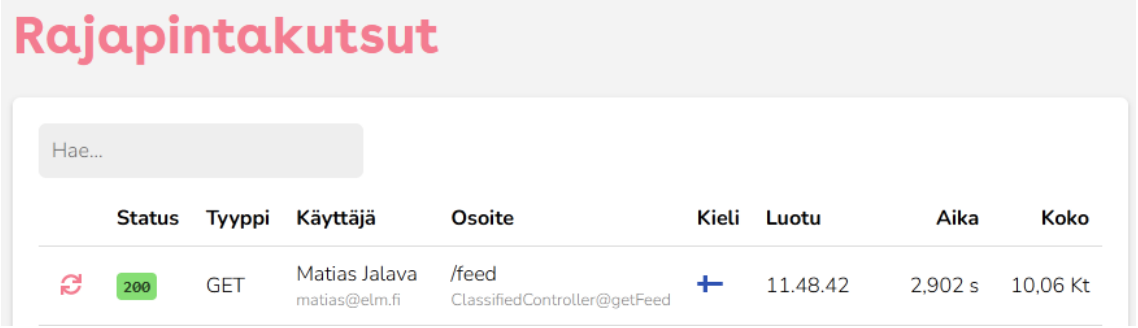
4.4 Julkaisu

Suosittelujärjestelmä julkaistaan omalle Linux-pohjaiselle palvelimelleen, jossa se pidetään jatkuvasti käynnissä omana palvelunaan. Kun sovellus on palvelimella käynnissä, pitäisi kaiken manuaalisen hallinnan, lähdekoodin päivitystä lukuun ottamatta, onnistua tavallisilla rajapintakutsuilla. Sovelluksen julkaisu Linux-palvelimelle toteutetaan Gunicorn-HTTP-palvelimella, joka selkeän dokumentaationsa ansiosta on yksinkertainen tapa julkaista Flask-rajapintasovellus. (48.) Kun rajapintasovellus on käynnissä, se reititetään Nginx-välityspalvelimen avulla Jemman määrittämään julkiseen HTTP-osoitteeseen, josta suosittelujärjestelmän metodeja voi kutsua.

Suosittelujärjestelmässä ei vielä ole tapaa lisätä upotuksia uusille käyttäjille tai tuotteille ilman uudelleen harjoitusta. Tässä vaiheessa Jemma-sovelluksen elinkaarta se ei ole ongelma, sillä mallin harjoittamiseen ei mene juuri minuuttia pidempään. Jotta malli olisi aina kohtuullisen tarkasti ajan tasalla, Jemman rajapintasovellukseen lisätään ajoitettu harjoituskomento, joka ajetaan joka kolmas tunti. Myöhemmin tiheän harjoitustahdin tilalla voidaan harkita mm. WALS-metodin käyttöönottoa.

Puhelinsovelluksen käyttäjille suositukset näkyvät heti etusivulla. Kun käyttäjä kirjautuu sisään, sovellus kutsuu palvelimen getFeed-kutsua, joka sisäisesti kutsuu tekoälypalvelimelta metodia ”recommend”. Suosittelukysely palauttaa enintään kolme tuntia vanhat suositukset, minkä avulla sovelluksen palvelin tarjoaa puhelinsovelluksen etusivulle tuotteet siinä järjestyksessä, jonka tekoälymalli antoi. Koska mallin toiminta on jo vahvistettu ainakin käytännössä toimivaksi, suurin kysymys tässä vaiheessa on suositusten noutoaika: Onko getFeed tarpeeksi nopea tuomaan tuotteet käyttäjän näytölle vai pitääkö

suosittelujärjestelmän toimintaa tehostaa? Kaikkien puhelinsovelluksen GET-kutsujen tulisi kestää enintään 700 millisekuntia sovelluksen sulavan toiminnan takaamiseksi. Kuvassa 20 ovat suosittelujärjestelmää käyttävän etusivukutsun tiedot.



Kuva 20. Etusivun suosituskutsun aika kyselyn luonnista siihen, että palautettava data on päässyt laitteelle. Kutsuun menee n. kolme sekuntia.

Käyttäjän syötteen saanti kestää n. kolme sekuntia, joka on aivan liian hidas pyynnölle, joka tehdään heti sovelluksen käynnistyessä. Ilman lisättyä tekoälyä pyyntöön menee n. 700 ms aikaa, eli suosittelujärjestelmä hidastaa pyyntöä huomattavasti (kuvat 20 ja 21).

Yksi ratkaisu suosittelujärjestelmän palautusajan nopeuttamiseen voi olla aiemmin mainitun ScaNN-metodin käyttöönotto suositusten noutoon (50). Sen lisäämisessä ilmeni kaksi ongelmaa:

- ScaNN-metodia käyttävän järjestelmän testaaminen on mahdotonta Windows-pohjaisessa kehitysympäristössä, eikä Linux-pohjaisen kehitysympäristön käyttöönottoon ole aikaa.
- Vaikka suositukset noudettaisiin ScaNN-metodilla, pitäisi ne silti järjestellä yksitellen neuroverkon avulla, mikä myös vie aikansa.

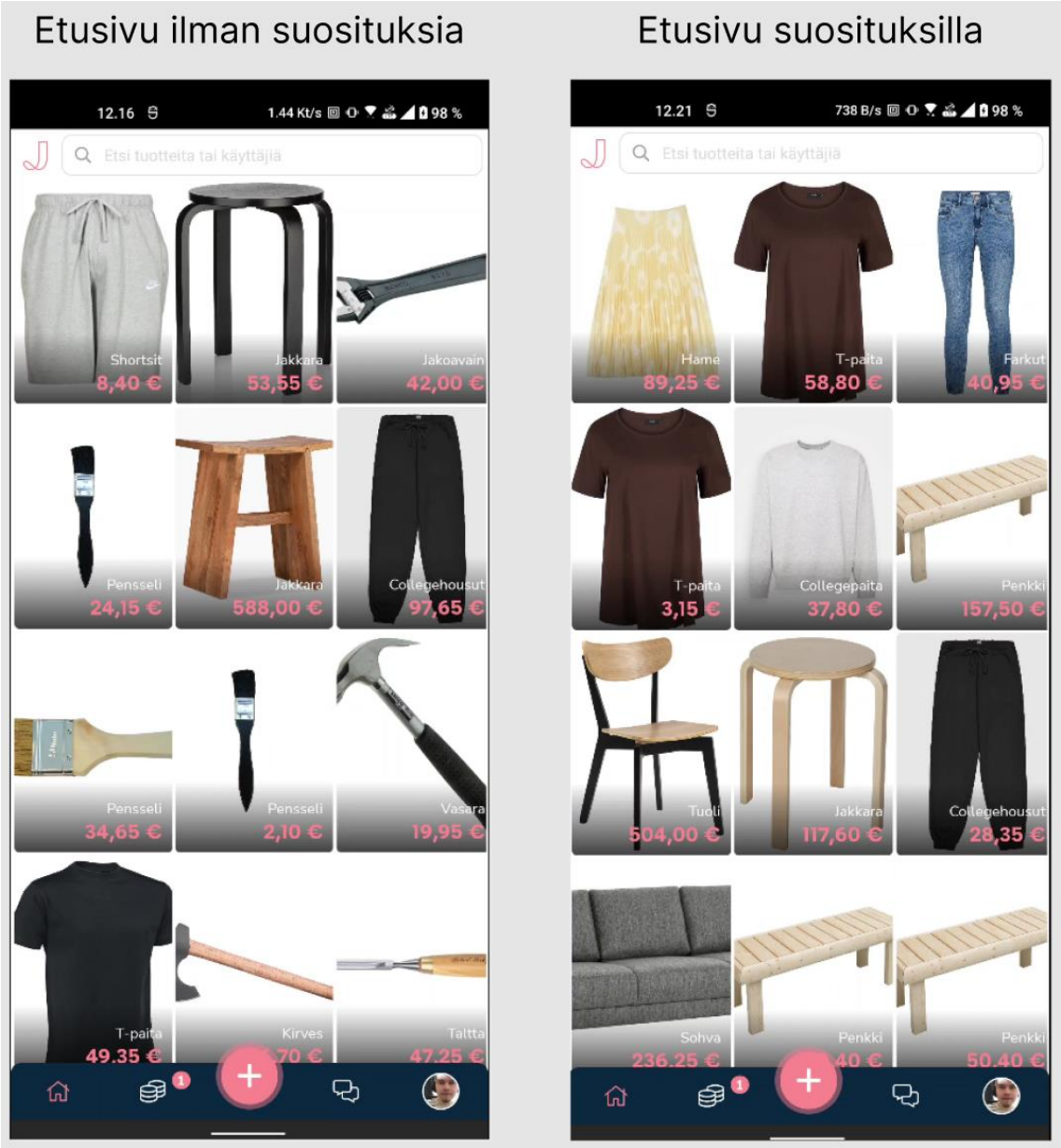
Tässä kehitysvaiheessa nopein ratkaisu on muuttaa suosittelujärjestelmän toimintaa siten, että mallin harjoitusvaiheen jälkeen jokaiselle käyttäjälle luodaan suositukset automaattisesti ja tallennetaan ne samalla Linux-palvelimella toimivaan Redis hash -tietokantaan. Kun suosittelukutsu tehdään, järjestelmä hakee sisäisestä Redis-kannasta kutsutun käyttäjän suosituslistan. Jos valmiiksi

tallennettuja suosituksia ei löydy, suosittelujärjestelmä kutsuu suosittelumallin haku- ja pisteytysmetodit tavalliseen tapaan ja palauttaa niiden antamat suositukset.

	Status	Tyyppi	Käyttäjä	Osoite	Kieli	Luotu	Aika	Koko
	200	GET	Matias Jalava matias@elm.fi	/feed ClassifiedController@getFeed		12.00.52	757 ms	10,32 Kt
	200	GET	Matias Jalava matias@elm.fi	/feed ClassifiedController@getFeed		12.00.29	1,44 s	10,40 Kt

Kuva 21. Rajapintakutsun kesto suosittelujärjestelmän optimoinnin jälkeen järjestelmän kanssa ja ilman järjestelmää. Ylempi kyselyistä on ilman järjestelmää, eli suosittelujärjestelmä kaksinkertaistaa kyselyn pituuden.

Suositusten hakeminen Redis-kannasta puolittaa kyselyn keston, joka on tässä vaiheessa hyväksyttävä nopeus. Siltä varalta, että suosittelujärjestelmässä ilmenee jatkossa ongelmia tai se toimii liian hitaasti, Jemman palvelimelle lisätään ympäristömuuttuja, jonka avulla voidaan estää sen käyttö etusivun dataa haettaessa. Ilman suositusjärjestelmää etusivulle haetaan vain satunnaisia myynnissä olevia tuotteita. Kuvassa 22 on Jemma-sovelluksen etusivu ilman suosituksia ja suositusten kanssa.



Kuva 22. Jemma-puhelinsovelluksen etusivu suosittelumallin kanssa ja ilman sitä.

Kuten luvussa 4.3 mainittiin, mallin testeistä ilmeni, että suosittelujärjestelmä ei suosittele niitä tuotteita käyttäjän etusivulle, joilla ei ole mallin harjoituksen aikaan näyttökertoja. Kun sovellukseen saadaan enemmän tuotteita ja käyttäjiä, tulee etusivu eroamaan huomattavasti työssä näytetyistä esimerkeistä ja olemaan personoitu käyttäjäkohtaisesti.

5 Reflektointi

Puhelinsovelluksessa, jonka osaksi suosittelujärjestelmä kehitettiin, on erittäin kattavasti tuoteryhmiä vaatteista toimistotarvikkeisiin. Ihmiskäyttäjät eivät anna robottimaisesti tykkäyksiä vain tuotteille, joita katsovat yli 10 sekuntia eivätkä selaa vain yhden tuoteryhmän tuotteita riippuen omasta sukupuolestaan. Tässä työssä malli harjoitettiin tuotteilla kolmesta eri tuoteryhmästä, joissa jokainen naiskäyttäjä katsoi yhtä tuoteryhmää ja muut sukupuolet katsoivat toista. Oikeassa käyttötarkoituksessaan malli tulee työskentelemään paljon monimutkaisemman datan kanssa, eikä mallin toimivuutta monimutkaisemman datan kanssa testattu kunnolla tässä työssä. Kuitenkin perustuen siihen, miten nyt tehty suosittelujärjestelmä luo käyttäjä- ja tuoteupotukset ja ennustaa niiden perusteella katseluajat, tulee järjestelmä erittäin todennäköisesti toimimaan monimutkaisessakin ympäristössä.

Koska suosittelujärjestelmä hidasti puhelinsovelluksen etusivun latausta yli 2 sekuntia, muutettiin sen toimintaa siten, että suositukset tallennetaan heti mallin harjoituksen jälkeen tekoälypalvelimen paikalliseen Redis-tietokantaan. Ratkaisu oli hyvä, sillä se yli kaksinkertaisti suosittelujärjestelmän vastausnopeuden. Se myös toimii odotetusti testivaiheen 70 käyttäjälle ja 100 suositukselle käyttäjää kohden, mutta sitä ei testattu suuremmilla tietomäärillä. Sovelluksen kasvaessa siitä saattaa syntyä ongelmia.

Jos työ aloitettaisiin nyt alusta, ensimmäisenä suunniteltaisiin, mitkä käyttäjä- ja tuoteparametrit ovat suositusten kannalta merkityksellisiä. Järjestelmän testaus toteutettaisiin myös monimutkaisemmalla datalla, joka vähintään perustuisi oikeaan käyttäjädataan. Työssä jäi epäselväksi, kuinka tarkasti järjestelmä profiloi käyttäjät ja haettavat kohteet eli kuinka tarkasti työn suosittelujärjestelmä tulee personoimaan tarjoamiaan suosituksia.

Testauksen puutteista huolimatta työ onnistui sen pääasiallisessa tarkoituksessa. Suosittelujärjestelmä toimii omana osanaan sovelluskokonaisuudessa ja tarjoaa käyttäjäkohtaisia tuotesuosituksia. Myös suosittelujärjestelmän ajoitettu

uudelleenharjoitus ja sen tarkkailun mahdollistava hallintapaneelinäkymä ovat tärkeitä ominaisuuksia sovelluksen tulevaisuuden kannalta. Työn tilaaja oli tyytyväinen sen tuloksiin ja suosittelujärjestelmän luomiin mahdollisuuksiin, joten työn tavoitteisiin päästiin (51).

6 Yhteenveto

Insinöörityössä perehdyttiin suosittelujärjestelmien ja tekoälymallien toimintaan ja toteutettiin suosittelujärjestelmä osaksi puhelinsovelluskokonaisuutta. Työn suurimmat haasteet liittyivät aiheen monimutkaisen kokonaisuuden johdonmukaiseen jäsentämiseen ja oikean käyttäjädatan puutteeseen.

Työn tavoitteena oli kehittää sovelluksen etusivulle sopiva suosittelujärjestelmä, joka innostaisi käyttäjiä selaamaan sovelluksen tarjontaa mahdollisimman paljon tarjoamalla käyttäjäkohtaisia, osuvia suosituksia. Työn tavoitteisiin päästiin, vaikka suosittelujärjestelmä jäi odottamaan jatkokehitystä. Mikä tärkeintä, työn tilaajat Jemma Oy:ssä olivat tyytyväisiä työn tuloksiin ja valmiita jatkamaan järjestelmän kehitystä tarpeiden mukaan.

Työn aihe ja vaatimukset olivat erittäin laajat. Työ aloitettiin jo n. puoli vuotta ennen sen nykyistä tilannetta toteuttamalla kevyt suosittelujärjestelmä esimerkkitoteutuksia seuraten, mutta jatkokehityksen myötä lähes kaikki järjestelmän toimintatavat muutettiin. Koska sekä tekoäly- että suosittelujärjestelmät olivat tekijälle uusi aihe, tekoälymallien ja suosittelujärjestelmien toimintamallien opiskelu vaati ison osan järjestelmän kehittämisajasta.

Suosittelujärjestelmän toimintaa ei päästy testaamaan oikeilla käyttäjillä, mutta sen toiminnan pystyi varmistamaan generoidulla datalla. Koska järjestelmä tuli erilliseksi osakseen sovelluskokonaisuutta, sen jatkokehittäminen, muuttaminen tai poistaminen ei muuta radikaalisti mitään puhelinsovelluksen toiminnasta.

Lähteet

1. Yi, Xinyang; Yang, Ji; Hong, Lichan; Zhiyuan, Derek; Lukasz, Heldt; Kumthekar, Aditee; Zhao, Zhe; Wei, Li & Chi, Ed. 2019. Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations. Google, Inc.

2. Edmundson, Annie. 2021. The Rise (and Lessons Learned) of ML Models to Personalize Content on Home (Part I). Verkkoaineisto. Spotify R&D. <https://engineering.atspotify.com/2021/11/the-rise-and-lessons-learned-of-ml-models-to-personalize-content-on-home-part-i/>. Luettu 17.3.2022.

3. Edmundson, Annie. 2021. The Rise (and Lessons Learned) of ML Models to Personalize Content on Home (Part II). Verkkoaineisto. Spotify R&D. <https://engineering.atspotify.com/2021/11/the-rise-and-lessons-learned-of-ml-models-to-personalize-content-on-home-part-ii/>. Luettu 17.3.2022.

4. Alvino, Chris & Basiliko, Justin. 2015. Learning A Personalized Homepage. Verkkoaineisto. Netflix Technology Blog. <https://netflixtechblog.com/learning-a-personalized-homepage-aa8ec670359a>. Luettu 17.3.2022.

5. Ricci, Fransceco; Rokach, Lior; Shapira, Bracha & Kantor, Paul B. 2011. Recommender Systems Handbook. Springer.

6. Netflix. 2022. Verkkoaineisto. Netflix. <https://www.netflix.com/>. Luettu 17.3.2022.

7. Recommendation Systems Overview. 2020. Verkkoaineisto. Google Developers. <https://developers.google.com/machine-learning/recommendation/overview/types>. Luettu 28.3.2022.

8. Candidate Generation Overview. 2020. Verkkoaineisto. Google Developers. <https://developers.google.com/machine-learning/recommendation/overview/candidate-generation>. Luettu 23.3.2022.

9. Collaborative Filtering. 2021. Verkkoaineisto. Google Developers. <https://developers.google.com/machine-learning/recommendation/collaborative/basics>. Luettu 14.2.2022.

10. Content-Based Filtering. 2021. Verkkoaineisto. Google Developers. <https://developers.google.com/machine-learning/recommendation/content-based/basics>. Luettu 14.2.2022.

11. Glauber, Rafael & Loula, Angelo. 2019. Collaborative Filtering vs. Content-Based Filtering: differences and similarities. State University of Feira de Santana.

12. Hu, Yifan; Koren, Yehuda & Volinsky, Chris. 2008. Collaborative Filtering for Implicit Feedback Datasets. AT&T Labs, Yahoo! Research.

13. Collaborative Filtering Advantages & Disadvantages. 2021. Verkkoaineisto. Google Developers. <https://developers.google.com/machine-learning/recommendation/collaborative/summary>. Luettu 14.4.2022.

14. Multi-task Recommenders. 2022. Verkkoaineisto. TensorFlow. <https://www.tensorflow.org/recommenders/examples/multitask>. Luettu 14.2.2022.

15. Recommending Movies: Ranking. 2022. Verkkoaineisto. TensorFlow. <https://www.tensorflow.org/recommenders/examples/basic_ranking/>. Luettu. 14.2.2022.

16. Harrison, Matt. 2019. Machine Learning Pocket Reference – Working with Structured Data in Python. E-kirja. O'Reilly.

17. An end-to-end open-source machine learning platform. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/>. Luettu 24.3.2022.

18. Why Tensorflow. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/about/>. Luettu 11.4.2022.

19. Introduction to modules, layers, and models. 2022. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/guide/intro_to_modules/>. Luettu 21.4.2022.

20. What is a difference between a feature and a label? 2019. Verkkoaineisto. Stack Overflow. <https://stackoverflow.com/a/40899529/>. Luettu 24.4.2022.

21. Wicke, Martin. 2017. TensorFlow High-Level APIs: Models in a Box. Verkkoaineisto. TensorFlow Dev Summit 2017. <https://youtu.be/t64ortpgS-E/>. Katsottu 21.4.2022.

22. Unsupervised Learning. 2020. Verkkoaineisto. IBM Cloud Education. <https://www.ibm.com/cloud/learn/unsupervised-learning/>. Luettu 26.4.2022.

23. How do I understand Top-N accuracy in machine learning in a simple way? 2020. Verkkoaineisto. Quora. <https://www.quora.com/How-do-I-

understand-Top-N-accuracy-in-machine-learning-in-a-simple-way/>. Luettu 21.4.2022.

24. tf.keras.losses.MeanSquaredError. 2022. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/api_docs/python/tf/keras/losses/MeanSquaredError/>. Luettu 21.4.2022.

25. Recommending movies: retrieval. 2022. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/recommenders/examples/basic_retrieval/>. Luettu 14.2.2022.

26. How to Interpret Root Mean Square Error (RMSE). 2021. Verkkoaineisto. Statology.org. <https://www.statology.org/how-to-interpret-rmse/>. Luettu 21.4.2022.

27. Overfitting. 2021. Verkkoaineisto. IBM Cloud Education. <https://www.ibm.com/cloud/learn/overfitting/>. Luettu 24.4.2022.

28. Custom training: walkthrough. 2022. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/tutorials/customization/custom_training_walkthrough/>. Luettu 24.4.2022.

29. Tensorflow Playground. Verkkoaineisto. Tensorflow. <https://playground.tensorflow.org/>. Luettu 18.4.2022.

30. Shivastava, Nitish; Hinton, Geoffrey; Krizhevsky, Alex; Sutskever, Ilya & Salakhutdinov, Ruslan. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15.

31. tf.keras.regularizers.Regularizer. 2022. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/api_docs/python/tf/keras/regularizers/Regularizer/>. Luettu 26.4.2022.

32. Kamalov, Firuz & Leung, Ho Hon. 2020. Deep learning regularization in imbalanced data. Canadian University Dubai; UAE university.

33. Regularization in Tensorflow. 2020. Verkkoaineisto. Tensor Examples. <https://tensorexamples.com/2020/08/02/Regularization-in-TensorFlow.html/>. Luettu 26.4.2022.

34. tf.keras.layers.Layer. 2022. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/api_docs/python/tf/keras/layers/Layer/>. Luettu 21.4.2022.

35. tf.keras.layers.Dense. 2022. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense/>. Luettu 14.4.2022.

36. Neural Networks. 2020. Verkkoaineisto. IBM Cloud Education. <https://www.ibm.com/cloud/learn/neural-networks/>. Luettu 16.5.2022.

37. Brownlee, Jason. 2021. How to choose an Activation Function for Deep Learning. Verkkoaineisto. Machine Learning Mastery. <https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/>. Luettu 24.4.2022.

38. Must-Know TensorFlow Activation Functions. 2021. Verkkoaineisto. TF Certification. <https://www.tfcertification.com/blog/must-know-tensorflow-activation-functions/>. Luettu 21.4.2022.

39. What is the "dying ReLU" problem in neural networks? 2015. Verkkoaineisto. StackExchange. <https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks/5734#5734/>. Luettu 24.4.2022.

40. Sigmoid Activation (logistic) in Neural Networks. 2022. Verkkoaineisto. OpenGenus IQ. <https://iq.opengenus.org/sigmoid-logistic-activation/>. Luettu 24.4.2022.

41. tf.keras.layers.Embedding. 2022. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/api_docs/python/tf/keras/layers/Embedding/>. Luettu 14.4.2022.

42. tf.keras.layers.StringLookup. 2022. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/api_docs/python/tf/keras/layers/StringLookup/>. Luettu 24.4.2022.

43. Flask. Verkkoaineisto. The Pallets Project. <https://palletsprojects.com/p/flask/>. Luettu 26.4.2022.

44. Meet Laravel. Verkkoaineisto. Laravel. <https://laravel.com/docs/8.x#meet-laravel/>. Luettu 26.4.2022.

45. Introduction to Redis. Redis. Verkkoaineisto. <https://redis.io/docs/about/>. Luettu 19.4.2022.

46. Kuppusamy, K. S. 2021. An Introduction to Redis, the In-Memory Data Structure Store. Verkkoaineisto. Open Source For You. <https://www.opensourceforu.com/2021/03/an-introduction-to-redis-the-in-memory-data-structure-store/>. Luettu 19.4.2022.

47. Wang, Ruoxi; Sun, Phil; Shivanna, Rakesh & Kula, Maciej. 2020. Tensorflow Recommenders: Scalable retrieval and feature interaction modelling. Verkkoaineisto. Tensorflow Blog. <https://blog.tensorflow.org/2020/11/tensorflow-recommenders-scalable-retrieval-feature-interaction-modelling.html/>. Luettu 19.4.2022.

48. Flask. Deployment Options. Verkkoaineisto. Gunicorn. <https://flask.palletsprojects.com/en/2.1.x/deploying/wsgi-standalone/#gunicorn/>. Luettu 26.4.2022.

49. tfrs.tasks.Retrieval. 2021. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/recommenders/api_docs/python/tfrs/tasks/Retrieval> . Luettu 17.3.2022.

50. tfrs.layers.factorized_top_k.ScaNN. 2022. Verkkoaineisto. Tensorflow. <https://www.tensorflow.org/recommenders/api_docs/python/tfrs/layers/factorized_top_k/ScaNN/>. Luettu 17.3.2022.

51. Farzan, Daniel & Ulmanen, Tatu. Toimitusjohtaja & teknologiajohtaja, Jemma Oy. Etätapaaminen 5.5.2022.