KARELIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Ville Kontturi

IBM BLUEMIX - RETRIEVE AND RANK -DEMO

Opinnäytetyö
Huhtikuu 2017

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Huhtikuu 2017
Tietotekniikan koulutusohjelma

Tikkarinne 9
80220 JOENSUU
(013) 260 600

Tekijä(t)
Ville Kontturi

Nimeke
IBM Bluemix - Retrieve and Rank Demo

Toimeksiantaja
Olapcon Oy

Tiivistelmä

Tässä työssä perehdytään ajankohtaisesti kiinnostavaan sekä vauhdilla nousevaan koneoppimisen (deep learning) suuntaukseen toteuttamalla aihealueeseen liittyvä sovellus käyttäen IBM:n Bluemix verkkoportaalista saatavilla olevaa Retrieve and Rank -alustaa.

Työssä annetaan lukijalle vankat pohjatiedot koneoppimisesta, sekä kerrotaan kuinka Retrieve and Rank -palvelualustan käyttöönotto Java-ohjelmointikielellä tapahtuu.

Opinnäytetyön yhteydessä toteutettu sovellus käyttää Wikipediassa olevan yhden aihealueen (kategorian) sivuja, jotka opetetaan Retrieve and Rank -sovellukselle. Lopullisessa työssä sovellukselle pystyy esittämään selkokielisiä kysymyksiä englanniksi, joihin keinoäly etsii ja palauttaa parhaiten vastaavaa Wikipedia artikkelia.

Sovellukseen jäi jatkokehitettävää lisäämällä opetus- ja arviointikysymyksien lukumäärää, jolloin sovelluksen tuottamien vastauksien vertailu olisi lähempänä oikean maailman toteutuksien kanssa. Lisäksi sovelluksen eri käyttöasetuksien vaikutusta lopputulokseen voisi tutkia tarkemmin.

Kieli
suomi

Sivuja 33
Liitteet 0
Liitesivumäärä

Asiasanat
Koneoppiminen, keinoäly, Bluemix, Retrieve and Rank, Neuroverkot, Java, Eclipse

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
April 2017
Degree Programme in Information Technology

Tikkarinne 9
80220 JOENSUU FINLAND
+358 13 260 600

Author (s)
Ville Kontturi

Title
IBM Bluemix - Retrieve and Rank Demo

Commissioned by
Olapcon Oy

Abstract
This thesis orients itself to current, interesting and fast rising software development trend about Machine learning (deep learning) by implementing a themed application using IBM's Retrieve and Rank service available in Bluemix online could portal.

This work will give its reader a solid base information about machine learning and describes how the Retrieve and Rank service is setup and started with Java programming language.

The application developed during the Thesis process reads data from Wikipedia articles (of one category) and teaches the Retrieve and Rank -service to operate with this data. With a web interface a user can type questions in plain language and the Retrieve and Rank -service search and returns the Wikipedia pages which answers the question best.

In the application, there remains further development by increasing teaching and evaluation question count. Then the query comparison would give results much closely to real world applications. In addition, the effects of different use settings into the end results could be studied more closely.

In the end the work describes the lessons learned.

Language
Finnish

Pages 33
Appendices 0
Pages of Appendices

Keywords
Machine learning, AI, Artificial Intelligence, Bluemix, Retrieve and Rank, Neural networks, Java, Eclipse

Sisältö

1 Johdanto

Tässä työssä tarkoituksena on saattaa lukijalle pohjatiedot koneoppimisesta sekä sen taustalla olevien algoritmien peruslogiikasta. Algoritmien toimintaan ei pureuduta kovin syvälle, sillä nykypäivänä käytettävissä olevien itseoppivien sovelluksien algoritmit ovat piilotettuja sekä vaativat yleensä korkeaa algoritmiosaamista.

Lukiessaan tätä työtä lukijan pitäisi saada peruskäsitys, miksi koneoppiminen on tärkeä tulevaisuuden kehitysala, sekä missä asioissa sitä on järkevää ja mahdollista hyödyntää.

Työ tehty Olapcon Oy:lle, joka hyödyntää työssään paljon IBM:n tarjoamia tuotteita (esimerkiksi Cognos). Työtä oli tarkoitus esitellä työn antajan toimesta Helsingissä 19.10.2016, joten työn valmistumiselle oli asetettu tämä aikaraja.

Projektissa tutustutaan keinoälyyn käyttämällä IBM:n Retrieve and Rank -palvelualustaa Bluemix-verkkoportaalista, jonka käyttöönoton tärkeimmät vaiheet järjestyksessään olivat

1. käytettävän tiedonlähteen valinta ja lataaminen,
2. tiedon jäsentäminen xml-muodosta palveluun ladattavaksi,
3. opetus- ja arviointikysymyksien laadinta jäsennetyistä tiedoista,
4. koneälyn (neuroverkot) ja Retrieve and Rank -sovelluksen toimintaperiaatteen opetteleminen,
5. Retrieve and Rank -palvelualustan käynnistäminen, tiedon lataaminen ja toimintakuntoon saattaminen,
6. käyttöliittymän tekeminen ja palvelun tuloksien tutkiminen.

2 IBM

IBM:n alku juontaa juurensa 1900-luvun taitteeseen, jolloin monia IBM:n toiminnan kannalta tärkeitä tuotteita lanseerattiin. IBM:n virallisena perustamisena voidaan pitää vuotta 1911, jolloin Computing- Tabulating- Recording Company (C-T-R) perustettiin. C-T-R nimi muutettiin nykyiseen IBM muotoon vasta vuonna 1924 [1].

Nykyään IBM toimii yli 170 maassa, IBM:llä työskentelee noin 400 000 työntekijää, ja IBM on ilmoittanut vuonna 2008 tehneensä ennätystuloksensa 103,6 miljardia dollaria [2] ja vuonna 2015 IBM:n tulot olivat 81,741 miljardia dollaria [3].

IBM:n vuoden 2015 vuosikatsauksesta nousevat esille alueet, joihin IBM aikoo toimintansa painottaa; asiantuntijakonsultointiin, palvelinliiketoimintaan, sekä ohjelmistotarjontaan. Näiden pohjalta IBM on nykyään yritys, joka luottaa ja panostaa vahvasti kolmeen ohjenuoraan strategioissaan:

- Kognitiivisiin ratkaisuihin
    - Kognitiivisuus, kehittyneet analytiikat ja avaintieto sisällytetään kaikkiin yrityksen johtaviin ratkaisuihin.
- Pilvialustaan
    - Uudet ratkaisut rakennetaan IBM Cloud -pilvipalvelimeen ja IBM:n tuotteet tulevat olemaan pilvi-yhteensopivia.
    - IBM kehittää ensiluokkaista palvelutarjontaa, kehittäjäympäristöä ja kaikkein turvallisinta hybridipilvialustaa.
- Keskittyminen teollisuuteen
    - Kaikki tuotteet ja sovellutukset kehitetään teollisuuden ja ammattilaisten käyttöä varten.

[3]

Vuonna 2008 IBM aloitti keskustelun älykkäämmän maailman luomisesta (Smarter Planet), jonka ideana oli mahdollistaa maailman kaupunkien parempi, ekologisempi ja tehokkaampi hyödyntäminen sensoreiden, anturien ja tietokoneiden

tuottaman tiedon pohjalta. Tämä keskustelu jatkuu edelleen tuottaen uusia ja parempia ratkaisuja älykkäämmän ja paremman maailman luomiseksi [4].

Vuoden 2008 jälkeen maailmaan on syntynyt uusi käsite IoT (Internet of Things), joka kuvastaa lukemattomien yhteen liitettyjen älykkäiden asioiden muuttumista systeemien systeemiksi, joka tuottaa jatkuvasti tietoa [5].

Seitsemän vuotta älykäs planeetta -keskustelun jälkeen IBM on lanseerannut konseptin Cognitive business yrityksille. IBM:n mukaan nykymaailman yrityksiltä menee valtavasti resursseja sekä rahaa asioihin, jotka voitaisiin ratkaista koneoppimisen keinoin. Kyseisen konseptin mukaan oppiva systeemi koostuu siitä, että se kykenee

- ymmärtämään rakenteetonta tietoa (tieto ei ole ennestään luokiteltua),
- perustelemaan tuottamalla hypoteeseja, ottaen huomioon olemassa olevat perustelut ja suositukset,
- oppimaan asiantuntijoiden harjoituksessa, jokaisesta vuorovaikutuksesta, sekä jatkuvasta tietovirrasta.

[3] [6]

Tässä konseptissa avainasemassa on IBM:n Watson, pilvipalveluna tarjottava rajapinta, josta löytyy keinoälytuote lähes kaikkien yritysten palveluiden automatisointiin [5].

3 Koneoppiminen

Rob Schapiren mukaan koneoppiminen on oppimista toimimaan paremmin aikaisemman tiedon pohjalta. Tarkoituksena on siis löytää itseoppiva / -kehittyvä algoritmi, joka pystyy itsenäisesti ratkaisemaan halutun ongelman erikseen tarjottavan testiaineiston pohjalta. Koneoppiminen onkin keinoälyn suunnittelun tärkein osa-alue [7].

Koneoppiminen jaetaan yleensä kahteen eri osa-alueeseen:

- Valvottuun koneoppimiseen, sekä
- ei-valvottuun koneoppimiseen.

Näistä ensimmäinen tarkoittaa koneälyn oppimista esivalmistellun testiaineiston pohjalta, missä testiaineisto on satunnainen ja pieni otanta oikeista ja vääristä vastauksista koko käytettävästä aineistosta. Esimerkiksi numerontunnistus on yksi yleisimmistä koneoppimisen alueista, missä koneälylle annettavaan testiaineistoon ihminen on antanut jokaiselle numerokuvalle oikean vastauksen 0–9 väliltä.

Ei-valvotun koneoppimisen päätarkoitus on löytää tietoaineistosta toistuvia kaavoja ja yhteyksiä ilman lisämateriaaleja. Näiden kaavojen ja yhteyksien avulla on tarkoitus tehdä lopulta järkeviä päätelmiä tai ennustuksia aineistosta. Hyvä esimerkki ei-valvotun koneoppimisen järjestelmästä on tiedonpakkausalgoritmi, jonka tarkoitus on pakata lähtöaineisto pienempään tilaan, siten että alkuperäinen tiedosto pystytään edelleen palauttamaan. Algoritmin toiminta kuvataan siten että sille annetusta lähtöaineistosta tuotetaan täsmälleen sama lopputulos, ja kun näiden välisessä piilotetussa kerroksessa on vähemmän arvoja (neuroneita) kuin lähtö- tai loppuaineistossa tällöin piilotetun kerroksen voi katsoa onnistuneesti pakkaamaan aineisto pienempään tilaan. [8]

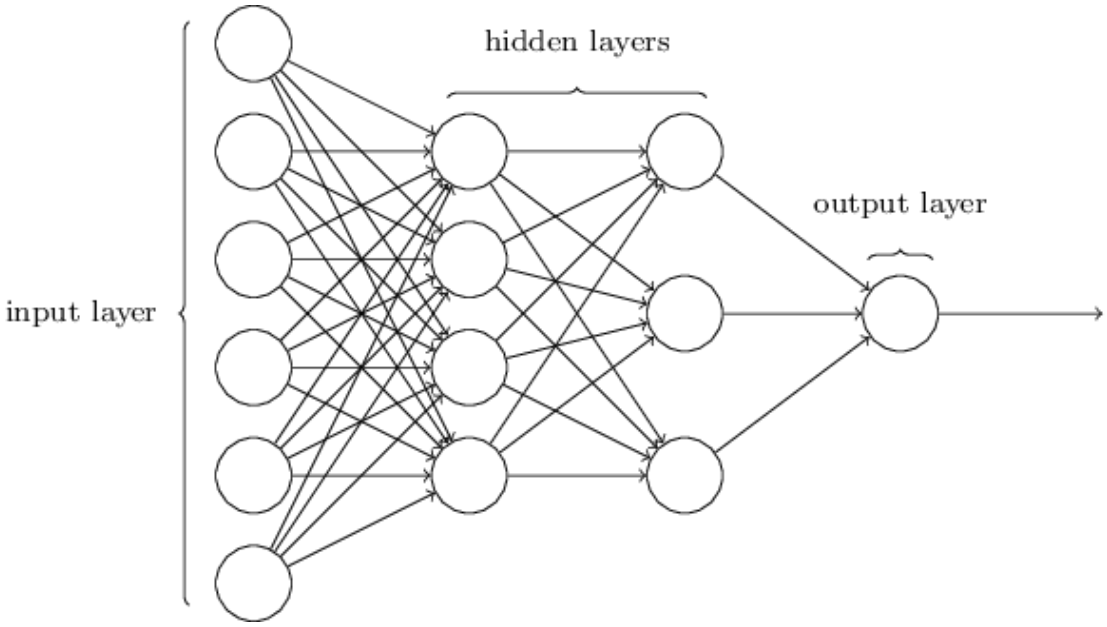
3.1 Neuroverkot

Koneoppiminen tapahtuu neuroverkon avulla, jonka toimintaperiaate on saanut inspiraationsa siitä, miten ihmismielen hermoverkoston ajatellaan prosessoivan informaatiota [9]. Yksi ensimmäisistä neuroverkkoajatuksen kehittäjistä oli suomalainen Teuvo Kohonen jo vuonna 1982 kehittäessään mallin itseorganisoituvalle kartalle, joka on käytännössä ohjaamattomaan oppimiseen perustuva neuroverkkomalli [10].

Michael Nielsen kuvaa neuroverkon toimintaperiaatteita kattavasti Neural Networks and Deep Learning -verkkokirjassaan. Hänen mukaansa neuroverkko rakentuu useista neuroneista, joista jokainen neuroni vastaanottaa yhden tai useamman syötteen (input), mutta tarjoaa vain yhden ulostulon (output). Kuvassa 1

näkyvässä neuroverkossa jokainen syötekerroksen jälkeinen ympyrä edustaa tällaista yksittäistä neuronia [11]. Kuvan 1 neuroneista näyttää lähtevän useita ulostuloja, mutta jokaisesta yhdestä kuvan neuronista lähtevä nuoli sisältää saman ulostuloarvon.

Yleisimmässä neuroverkon muodossa jokainen yksittäinen neuroni lähettää ulostuloarvonsa jokaiselle seuraavan tason neuronille, kuten kuvassa 2 on esitetty.



Kuva 1. Yksinkertainen neuroverkko [11].

Kuvasta 1 on myös nähtävissä neuroverkon rakenteesta kolme osiota:

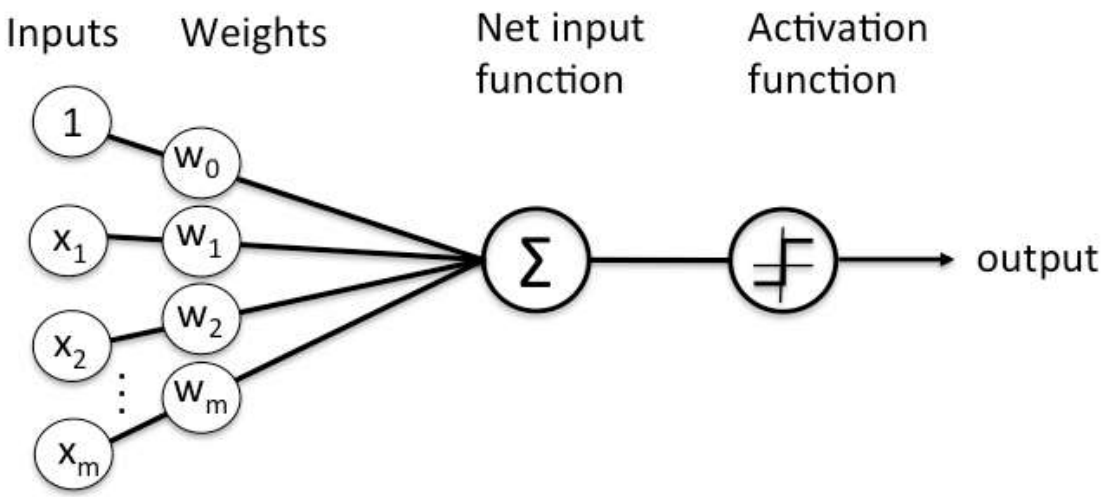
- syötekerros (input layer),
- ulostulokerros (output layer),
- piilotetut kerrokset (hidden layers).

Syötekerros tarkoittaa järjestelmään annettavia lähtöarvoja (esimerkiksi kuvatiedoston pikseli-arvoja). Syötekerroksesta on hyvä huomioida, että vaikka syötteet on kuvattu ympyröillä, ne eivät ole neuroneita vaan algoritmiin annettuja syötearvoja. Muilla tasoilla jokainen ympyrä edustaa kuvan 2 mukaista neuronia.

Piilotetun kerroksen tarkoituksena on tallettaa esivalmistellun testiaineiston avulla algoritmiin abstrakti kuvaus siitä aineistosta, jota algoritmin on tarkoitus lopulta käyttää. Yleistetymmin piilotettu kerros yrittää kuvastaa tai mallintaa lopullista aineistoa, samalla tavalla kuin ihminen voi visualisoida tai mallintaa reaalimaailmaa mielessään; esimerkiksi kiikun keinahtelua tai väreilevää veden pintaa [12]. Tästä syystä onkin erittäin tärkeää muodostaa harjoitusaineisto siten, että se kuvastaa mahdollisimman tarkasti lopullista, järjestelmän käyttämää todellista aineistoa. Michaelin mukaan piilotettu kerros, joka voi sisältää yhden tai useamman kerroksen (kuten kuvassa 1 olevat kaksi piilotettua kerrosta), tarkoittaa pelkästään sitä, että se ei ole syöte- tai ulostulokerros [11].

Neuronin periaate

Ymmärtääkseen neuroverkon toimintaperiaatteen ensimmäiseksi pitää ymmärtää yksittäisen neuronin toimintaperiaate.



Kuva 2 Neuronin toimintaperiaate **[13]**

Kuvassa 2 näkyy, miten neuroni saa alkuarvona yhden tai useamman arvon (x_1, x_2, … x_m), erillisen kynnysarvon (w_0), sekä alkuarvoihin vaikuttavan painoarvon (w_1, w_2, … w_m). Yksittäisen neuronin toimintaperiaate on ehkä helpointa mieltää summaavaksi lineaarifunktioksi

$y = kx + b$, missä

$k = w_1, w_2, \ldots w_m,$

$b = 1 * w_0$ ja

$x = x_1, x_2, \ldots x_m.$

Jokainen alkuarvo kerrotaan siis omalla painoarvolla, jotka sitten summataan yhteen kynnysarvon kanssa ja syötetään lopuksi aktivointifunktiolle. Aktivointifunktion tarkoituksena on muuttaa neuronin summaama tulos oikein / väärin vastaukseksi (eli arvoksi 0 ja 1:n väliltä). Aktivointifunktio määrittää siten arvon seuraavan tason neuroneille, jotka vastaanottavat arvoja yleensä kaikilta tai useammalta edellisen tason neuronilta tehden saman toiminnan saamilleen arvoille. [13]

Kirjoittaessani jatkossa painoarvoista, viittaan tällä neuronin paino-, sekä kynnysarvon tulokseen.

Neuroverkon toiminta

Kun neuroverkossa on neuroneja useita pysty-, että vaakatasossa (kuten kuvassa 1), kykenee verkko tallettamaan erittäin monimutkaisia tietorakenteita (algoritmeja) ja siten myös ennustamaan syötearvoista monimutkaisia lopputuloksia (esimerkiksi tunnistamaan kuvasta numeroita). [13]

Verkon toimintaperiaate on yksinkertaisuudessaan musta-laatikko. Laatikolle annettaan tietty joukko syötearvoja ja laatikko antaa vastukseksi ennalta tiedettyjä vastauksia, esimerkiksi numerontunnistuksessa vastauksena kuvapikseleihin arvon 0 ja 9 väliltä tai kuvantunnistuksessa kategorian jota kuva esittää, kuten ihminen, eläin, kasvo, käsi, silmä, auto, pyörä, ja niin edelleen. Ajankohtainen esimerkki neuroverkon käytöstä löytyy Suomen poliisin käyttämästä automaattisesta rekisterikilven tunnistusjärjestelmästä, jonka on kehittänyt Hallinnon tietotekniikkakeskus. Kyseinen järjestelmä kulkee nimellä RELLU-projekti. [14] [15]

Neuroverkon toimintaan laittaminen on yksinkertaisuudessaan verkossa olevien neuronien saamien syötearvojen painoarvojen painottamista, toisin sanoen painoarvon muuttamista. Että verkon voisi ottaa käyttöön pitäisi kyseisten neuroneiden optimaaliset, tai lähes optimaaliset painoarvot saada laskettua.

Tämä painoarvojen korjaaminen tapahtuu yleensä pitkän iteratiivisen prosessin tuloksena, missä opetusaineistossa olevia harjoitussarjoja toistetaan järjestelmälle uudelleen, yhteensä tuhansia tai jopa miljoonia kertoja. Jokaisella toistokierroksella jokaisen verkossa olevan neuronin yksittäisiä painoarvoja muutetaan aavistus oikean vastauksen suuntaan. Tätä prosessia jatketaan, kunnes järjestelmän tuottama virhe laskee hyväksyttävälle tasolle. Jotta painoarvoja pystyttäisiin neuroneihin muuttamaan, pitää harjoitusvaiheen neuroniverkon antaman tuloksen ja järjestelmästä halutun lopputuloksen erosta pystyä jotenkin laskemaan jokaisen neuronin aiheuttama virhe verkon tuottamaan lopputulokseen nähden. Tämän virheen laskentaan on kehitelty useita erilaisia virheen minimointi-funktioita. Painoarvojen säätämisprosessista käytetään yleensä nimeä takaisinjohtaminen (backpropagation).

Tämän jälkeen järjestelmään voidaan alkaa esittää harjoittamattomia syötearvoja, että saadaan arvioitua neuroverkon toimintakyvykkyyttä. Jos näissä ennalta harjoittamattomissa syötearvoissa ei ilmene suuria virheitä, voidaan verkon katsoa olevan riittävän toimintakuntoinen käyttöönotettavaksi. Näin ollen opetettu neuroverkko tuottaa neuroneihin tallennettujen painoarvojen perusteella ennustuksen siitä mikä olisi oikea vastaus (esimerkiksi onko pikseleinä annetussa kuvassa numero 3 vai 5).

Takaisinjohtaminen

Takaisinjohtaminen on järkevintä ajatella iteratiiviseksi prosessiksi, missä edellisellä tasolla (layer) laskettua virhettä käytetään apuna nykyisen tason painoarvojen muuttamisessa. Kun kaikkien neuronien painoarvot on korjattu, aloitetaan uusi kierros seuraavan harjoituskappaleen osalta.

Neuroverkon opettamisessa verkolle annettaville syötearvoille on etukäteen tiedossa haluttu lopputulos, josta voidaan laskea järjestelmän tuottaman ja järjestelmästä halutun lopputuloksen välinen erotus. Tämä erotus syötetään virheen minimointi -funktioon, joka kertoo, onko yksittäisen neuronin tuloksen virhe isompi vai pienempi kuin oikea tulos. Kun verkkoa kuljetaan siten lopusta alkuun, kyseisen virheen avulla pystytään laskemaan virheen arvo myös edeltävissä tasoissa ja neuroneissa. Juuri tämä virhearvon laskenta mahdollistaa neuronien painoarvojen muuttamisen [11].

Takaisinetenemisessä on tärkeää muistaa, että koska verkkoon syötetään toisistaan hyvin erilaisia syötteitä, joiden odotetaan myös tuottavan erilaisia vastauksia, pitää neuroneiden painoarvoja muuttaa kerrallaan vain hyvin pienen arvon verran. Tämä sen takia, että seuraavan kierroksen syötearvoilla voidaan päivittää painoarvoja uudelleen, yleensä vielä eri suuntaan kuin edellisessä kierroksessa. Kun näitä kierroksia tehdään harjoitusaineiston pohjalta sadoista – miljooniin kertoihin, pitäisi algoritmin tuottaman virheen lopulta päätyä hyväksyttävälle tasolle, olettaen että valitun neuroverkon piilotettu kerros on tarpeeksi suuri pystyäkseen kuvaamaan aineistoa. [15]

3.2 Koneoppimisen sovellutusperiaate

Riippuen käyttötarpeesta koneoppimiselle on kehitetty useita erilaisia algoritmeja, jotka soveltuvat kukin parhaiten juuri tietynlaiselle tiedoille. Esimerkiksi algoritmi, jonka tarkoituksena on laatia ennustuksia säästä ei voi olla sama, kuin algoritmi joka etsii vastausta asiakkaan vaivaan lääkärien aikaisemmin hoitamien potilaiden tapauskansioista. Toisen pitää tehdä laskelmia numeraalisen datan pohjalta, kun toisen pitää etsiä vastausta kirjallisesta aineistosta. Koska tämä työ käyttää Retrieve and Rank -sovellusta, pureudutaan tässä Learning to Rank -algoritmeihin, joka edustaa esimerkeistä jälkimmäistä.

IBM ei paljasta Retrieve and Rank -sovellusalustan käyttämiä koneälyalgoritmeja, mutta kyseisen palvelun dokumentaatiossa on mainittu, että palvelussa on useita Learning to Rank -nimisiä koneoppimisen algoritmeja, joista sovellus

sitten valitsee parhaiten soveltuvat algoritmit käytetyn opetusdatan perusteella [17].

Learning to Rank -algoritmit ovat avainosassa hakukoneiden optimoinnissa ja parempien hakutulosten tuottamisessa. Kuten edellisen kappaleen lääkärien aikaisempien hoidettujen tapauskansioiden etsimisen esimerkissä Learning to Rank -käyttölogiikka voidaan yksinkertaisella tasolla jakaa kahteen osioon

- soveltuvien dokumenttien etsimiseen tietokannasta (Retrieve),
- löydettyjen dokumenttien listaamiseen arvojärjestykseen (Rank). [18]

Näiden lisäksi tässä työssä esiteltävää sovellusta kehitettäessä havaittu, että lopullisen sovelluksen käyttölogiikkaan kuuluu lisäksi seuraavat kaksi tärkeää vaihetta

- kysymyksen parametrisoiminen ja
- tulosten esittäminen arvojärjestyksessä.

Kysymyksen parametrisoinnissa annetusta kysymyksestä erotetaan avainsanat täytesanoista. Samalla sanat voidaan pisteyttää niiden merkitsevyyden perusteella. Parametrisointi tuottaa siis sanoja, jotka parhaiten kuvaavat mitä halutaan tietää. Esimerkiksi jos kysymyksessä esiintyy sana ”kuka”, tällöin on tarkoitus tuottaa vastaukseksi henkilö tai vastaavasti sanaan ”missä”, odotetaan vastaukseksi paikkaa. Tämän lisäksi kysymyslauseesta voidaan karsia tarpeettomat täytesanat kuten erikoismerkit ja alistuskonjunktiot. [19]

Parametrisoinnin merkitys näkyy heti seuraavassa vaiheessa, kun saaduilla hakusanoilla haetaan tietokannasta kaikki liittyvät dokumentit (Retrieve). Jos hakusanat sisältävät turhia täytesanoja, tulee haku kestämään pidempään, sekä sisältämään dokumentteja jotka eivät liity kysymykseen millään tavalla. Käytännön kokemuksena tästä vaiheesta on tärkeää ymmärtää, että hakutulosten lukumäärä on tärkeä; jos hakutuloksia on liian vähän, kasvaa todennäköisyys, että kysymykseen parhaiten vastaava dokumentti jää hakematta tietokannasta kokonaan, jolloin seuraavalla vaiheella ei ole mahdollisuutta korostaa oikean dokumentin arvoa lopullisissa tuloksissa.

Suuren hakutulosmäärän huonona puolena on, että lopullisen vastauksen saaminen kestää myös kauemmin, sillä dokumenttien haun jälkeen dokumentit syötetään luokittelija -sovellukselle (Rank), joka käyttää esiopetettuja algoritmeja listatakseen dokumentit paremmuus- / arvojärjestykseen. Luokitteluvaihe käyttää luokittelemiseen neuroverkkoon tallennettua algoritmia harjoitusdatasta.

Lopuksi luokittelijan tuottamat dokumentit näytetään käyttäjälle hakutuloksina paremmuusjärjestyksessään. Yleensä tässä vaiheessa esitettävien dokumenttien listaa rajataan siten, että käyttäjälle ei näytetä liikaa tuloksia kerralla sovelluksen käytettävyyden parantamiseksi.

3.3 Opettaminen

Learning to Rank algoritmien opettaminen jaetaan kahteen vaiheeseen, harjoittamiseen ja testaamiseen. Tämän pohjalta voidaan todeta, että Learning to Rank -koneoppiminen on valvottua koneoppimista.

3.3.1 Harjoittaminen

Learning to Rank algoritmeja koulutetaan niin sanotuilla ”kysymys-oikea vastaus” dokumenttipareilla. Algoritmin opettamisessa jokaisen kysymys-oikea vastaus parin kohdalla tehdään erillinen kysely opetuksessa olevaan neuroniverkkoon. Mikäli algoritmin antama vastaus ei ole optimaalinen viilataan neuroverkon piilotetun kerroksen neuronien vektori arvoja siten, että ne siirtyvät osoittamaan aavistuksen verran oikean vastauksen suuntaan. Tätä harjoitustoimenpidettä kutsutaan takaisinjohtamiseksi.

Harjoituksen päätteeksi tätä piilotettuun kerrokseen tallennettua algoritmia käytetään lopullisen järjestelmän kysymyksien tuottamien dokumenttien arvottamiseen (luokitteluun tai järjestämiseen); eli oikean vastauksen ennustamiseen. Mitä

enemmän näitä kysymys-oikea vastaus-pareja on harjoitus vaiheessa harjoitettavalle algoritmille esittää, sen luotettavampia hakutuloksia voi olettaa saavansa lopullisesta järjestelmästä.

Nämä ”kysymys-oikea vastaus dokumenttiparit” voidaan edelleen jakaa kolmeen lähestymiskategoriaan sen mukaan mitä ominaisuutta ne yrittävät minimoida:

- **Pisteitä**
  Algoritmi yrittää ennustaa oikean luokan jokaiselle kysely-dokumentti-parille tarkoituksenaan pienentää vääriä luokittelu-tietoja testiaineistosta. Esimerkiksi valokuvat voidaan kategorioida (ihminen, eläin, auto, talo, rekka-auto, jne.).
- **Pareja**
  Algoritmi yrittää asettaa kaksi kysymys-dokumentti-paria paremmuusjärjestykseen kerrallaan. Tarkoituksena on minimoida väärässä järjestyksessä olevat kysymys-dokumentti-parit. Tällöin parin luokalla ei ole väliä.
- **Listoja**
  Algoritmi on samankaltainen parittamisen kanssa, mutta tässä kysymys-vastaus-parit pyritään laittamaan oikeaan järjestykseen kokonaisuutena, eikä vain kahden kysymys-dokumentti-parin erona, kuten parittamisessa.

[17]

3.3.2 Testaaminen

Testauksen tarkoituksena on testata ja tarkistaa kuinka tarkasti tai hyvin algoritmin opettaminen on onnistunut. Järjestelmälle esitetään uusia kysymyksiä, joihin tiedetään jo vastaukset. Kun järjestelmä antaa ennustetut vastauksensa, tarkistetaan että oikea vastaus löytyy tuloksista, jonka jälkeen tarkistetaan kuinka todennäköisenä järjestelmä pitää oikean vastauksen relevanssia. Jos oikeaa vastausta ei löydy lainkaan vastauksien joukosta on opetusvaiheessa käytettyjä kysymys-dokumentti-pareja syytä miettiä uudelleen, tarkentaa, parantaa, tai lisätä. [21]

Algoritmin opettaminen on hyvä mieltää iteratiiviseksi prosessiksi, eli kun järjestelmän lopullisilta käyttäjiltä saadaan palautetta, esimerkiksi tieto mihin kysymykseen ei ole saatu oikeaa vastausta, voidaan kyseinen kysymys opettaa algoritmille uudestaan ja siten parantaa algoritmin luotettavuutta.

4 Projektin motivaatio

Lopullinen aihe valittiin oikeastaan uutuuden, ajankohtaisuuden, sekä sen nykyhyödyntämisen harvinaisuuden vuoksi. Aihe on opinnäytetyön aiheen antajalle ajankohtainen, sillä tällä hetkellä Suomesta ei juuri löydy sovelluksia, jotka hyödyntäisivät IBM:n Retrieve and Rank palvelua.

Aihe on myös globaalisti ajankohtainen, sillä kone-avusteisten oppimisalgoritmien tarve on jo yksin räjähdysmäisesti kasvavan tietomäärän takia välttämätön siirtymäsuunta. Niclas kirjoitti tivi.fi:n web-artikkelissa vuonna 2011 miten maailmassa olisi ollut digitaalista tietoa arviolta huikeat 295 eksabittiä (~39 600 000 000 Gb). Kyseisessä artikkelissa digitaalisen tiedon määrän arvioitiin kasvavan 23 prosentin vuosivauhtia, mikä tarkoittaisi sitä olevan vuoden 2016 loppuun mentäessä noin 830 eksabittiä [18].

Kuitenkin vuonna 2013 digitaalisen universumin datamäärän on arvioitu olevan 4,4 zettatavua (~ 4,4 biljoonaa gigatavua → 4 400 000 000 000 Gb), joten selkeästi Tivi:n artikkeli ei ole onnistunut ennustamaan datan kasvua edes lähelle sen nykyistä tahtia. EMC:n verkkosivuston mukaan datan määrä kaksinkertaistuu joka toinen vuosi ja vuosien 2013 ja 2020 välillä datan määrä kymmenkertaistuu 4,4 zettatavusta 44 zettatavuun. [19].

Koska dataa on nyt, sekä jatkossa erittäin laajalti käytettävissä, keinot tämän datan hyödyntämiseen ovat rahanarvoisia. EMC:n sivustolla mainitaankin, että maksimoidakseen datan tarjoamat mahdollisuudet, yrityksen on hallittava seuraavat osa-alueet:

- informaatio turvallisuus
- virtuaaliset datapalvelimet
- saumattomat julkiset ja yksityiset pilvipalvelut
- uuden sukupolven data analysoinnit
- uudet varaston hallinta menetelmät ja järjestelmät
- uudet datan hallinta ja prosessointi menetelmät
- automaattinen datan luokittelu, sekä
- kyky toimia reaaliaikaisen tiedon pohjalta.

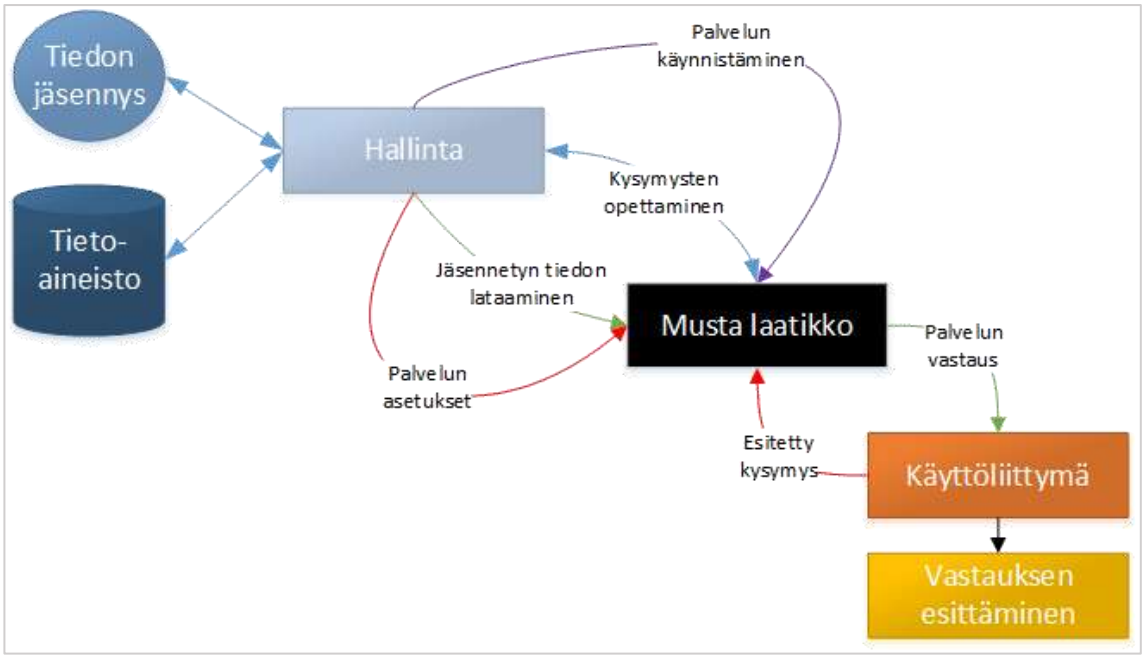
Näistä suurin osa vaatii jonkinlaisen keino-älyn olemassaoloa, sillä pelkän tietomäärän takia esimerkiksi tiedon luokittelemista ei ehditä enää hoitamaan yksin ihmisten toimesta (esimerkiksi kuvien luokitteleminen).

Näistä tiedoista on pääteltävissä, että nyt ja eritoten lähitulevaisuudessa tämän keinoälylogiikan ymmärtäminen ja hyödyntäminen tulee olemaan avainasemassa menestyvien it-järjestelmien kehittäjien työtehtävissä, kun työtehtävät tulevat vääjäämättä vaatimaan tämän kasvavan datamäärän käyttämistä, sekä eritoten sen hyödyntämistä.

5 Projektin kuvaus

Opinnäytetyöprojektin voi kuvata ylätasolla kolmen toiminnon kautta:

- Musta laatikko (IBM Retrieve and Rank palvelu)
- Hallinta (Java)
- Käyttöliittymä (NodeJS)



Kuvio 1 Projektielementit

Musta laatikko edustaa jäsennetyn tiedon tietokantaa, sekä itseoppivaa neuroverkkoa palveluun esitettävien kysymyksien vastaamisessa, eli Retrieve and Rank -rajapintaa.

Hallinta vastaa palvelun esimäärittämisestä, asetuksien asettamisesta ja käytettävän tietoaineiston jäsentämisestä, lataamisesta, sekä opettamisesta Mustalle laatikolle. Lopuksi Hallinta myös käynnistää Retrieve and Rank -palvelun käyttöliittymän käytettäväksi.

Käyttöliittymä vastaa kyselyiden välittämisestä Mustan laatikon rajapinnalle, sekä saatujen vastauksien esittämisestä palvelun loppukäyttäjälle käyttöliittymästä.

Tutustuessa IBM Bluemix - Retreive and Rank -sovelluksen eri käyttöönottotapoihin, päätetty toteuttaa tietorakenteiden lisäys, täyttö, muokkaus, sekä keinoälyn opettaminen Java-ohjelmointikielellä, koska ensinnäkin kieli oli työn tekijälle entuudestaan tuttu, toiseksi IBM:ltä löytyi Java-kielelle hyvä rajapintaohjesivusto [24] ja kolmanneksi Java-kielelle löytyi erittäin kattava Github projekti Ask Professor Languou, joka oli toteutettu Java-kielellä [21]. Ask Professor Languou

-sovellus toimi laajan koodipohjan takia apuna tässä työssä tehdyn sovelluksen laatimiselle.

Muita ohjelmointikieli-vaihtoehtoja palvelun käyttöönotoille olisi ollut cURL, sekä Node.Js, mutta nämä kielet eivät olleet työn tekijälle entuudestaan kovin tuttuja, eivätkä tarjonneet tekijän mielestä tarpeeksi kattavaa jatkokehitysrajapintaa monimutkaisemmalle järjestelmälle.

Retrieve and Rank -sovelluksen käyttöönotto olisi ollut mahdollista myös erillisen graafisen käyttöliittymän avulla, mutta tämän käyttö olisi rajoittanut sovelluksen jatkokehitysmahdollisuuksia suuresti, esimerkiksi siten, että se hyväksyy tietolähteiksi ainoastaan HTML, PDF, tai Microsoft Word formaatteja [26], joten sitä vaihtoehtoa ei edes harkittu.

5.1 Palvelun käytön pääperiaatteet

IBM:n palvelujen käyttö tapahtuu IBM Bluemix pilviportaalissa. Retrieve and Rank -sovellus tarvitsee toimiakseen tietokannan käytettävää aineistoa varten, sekä Apache Lucene informaationhaku ohjelmistokirjaston, joka käyttää tietokantaa, sekä pyörii Apache Solr verkkosovellusrajapinnan alla [27]. Tietokanta, Apache Lucene, sekä Apache Solr tulevat kaikki automaattisesti Retrieve and Rank -sovelluksen käyttöönoton mukana, mutta etenkin näiden kahden jälkimmäisen palvelun tunteminen auttaa huomattavasti Retrieve and Rank -palvelun toimintalogiikan ymmärtämistä. Lopullista toimivaa Retrieve and Rank palvelua on tarkoitus käyttää Node.js:llä toteutetun graafisen web-käyttöliittymän avulla.

Retrieve and Rank -rajapinnan käyttöönottaminen vaatii useita vaiheita, jotka kuvataan alla vain otsikkotasolla, tarkennusta vaativiin vaiheisiin pureudutaan myöhemmissä kappaleissa perusteellisemmin.

1. Kirjaudu Bluemix sovellukseen ja luo käytettävät tunnukset Retrieve and Rank -sovellukseen (service credentials).
2. Päätä sovelluksessa käytettävä pohjadata (opetettava aineisto):

   2.1. hae käytettävä data,
   2.2. jäsennä data lataamista varten (parserointi).
3. Luo palveluun klusteri (cluster).
4. Luo klusteriin kokoelma (collection) ja lataa käytettävä data kokoelmaan (documents).
5. Luo luokittelija (ranker) palveluun ja kouluta se harjoitus-datalla.
6. Tee palvelusta hakuja (retrieve and rank).
   6.1. Tarkastele vastauksia.
7. Lue Bluemixiin Node.js rajapinta, joka tekee kyselyjä luotuun palveluun ja esittää vastaukset pyytäjälle.

[28]

Työn tekijälle annettiin vapaat kädet pohjadatan suhteen, jonka takia aiheeksi valikoitui Wikipedia-kirjastosta kategoria tekninen piirtäminen [29]. Kategoria sisälsi ohjelmointityön teon aikaan yhteensä 75 sivua, jotka jokainen ladattu omaksi dokumentikseen palveluun (tämän dokumentin kirjoittamishetkellä kategoriassa oli 78 sivua). Loppukäyttäjän on tarkoitus pystyä esittämään kysymyksiä tästä aihekategoriasta, josta luokittelija palauttaa mielestään parhaiten kysymykseen osuvat sivut.

5.2 Lucene-tietokannan rakenne

Lucene tallentaa talletettavat asiakirjat käänteisenä indeksinä [23]. Käänteinen indeksointi tarkoittaa sitä, että haettavien asiakirjojen sanat on käännetty listaksi ja jokaista sanaa vastaan ilmoitetaan dokumentin tunniste.

Esimerkiksi seuraavat kaksi lausetta käännetään käänteiseen indeksiin:

1. Tämän asian ymmärtäminen on helpointa esimerkin kautta.
2. Asian ymmärtäminen onnistuu parhaiten esimerkillä.

Tässä meillä on kaksi lausetta, joilla molemmilla on oma id (1 ja 2). Taulukossa 1 on esitetty nämä lauseet käänteisenä indeksinä.

Taulukko 1 Käänteinen indeksi

Termi	**Lause 1**	**Lause 2**
tämän	x	
ymmärtäminen	x	x
on	x	
helpointa	x	
esimerkin	x	
kautta	x	
asian	x	x
onnistuu		x
parhaiten		x
esimerkillä		x

Kuten taulukosta 1 on havaittavissa, lauseet voidaan esittää lauseista muodostetun sanalistan avulla ja jos nyt indeksiin halutaan tehdä haku sanoilla: ”esimerkin ymmärtäminen”, palauttaa tietokanta vastaukseksi:

esimerkin: Lause 1, Lause 2

ymmärtäminen: Lause 1.

Saadun vastauksen perusteella Lause 1 olisi todennäköisesti parempi vastaus, koska se sisältää kaksi osumaa, kun Lause 2 sisälsi vain yhden osuman [31].

Käänteinen indeksointi vie huomattavasti enemmän prosessointitehoa tietojen syöttämisvaiheessa, mutta tuottaa moninkertaisen nopeuden, kun tietokannasta aletaan hakea tietoa.

5.3 Pohjadatan lataaminen tietokantaan.

Retrieve and Rank -sovellus tallettaa dokumentteja seuraavan logiikan mukaan:

- cluster - manages search collections
  - config1
  - config2
  - collection1 - index of documents, issued by a solr_configuration {config1}
    - document1
      - field1

        - field2
    - document2
        - field1
        - field2
- collection2 - index of documents, issued by a solr_configuration {config2}
    - document1
        - field1
        - field1
    - document2
        - field1
        - field2

Lista laadittu Retrieve and Rank Tutorial -sivun perusteella [28]. Kuten listasta luettavissa, tallennetaan sovelluksen käyttämät dokumentit erillisen kokoelman (collection) sisälle. Jokainen dokumentti voi sisältää yhden tai useamman kentän (field), sisältäen tietoa kyseisestä dokumentista.

Nämä rakenteet määritetään palveluun erillisessä solr_config.zip paketissa, joka sisältää xml-tiedostoja, jotka kertovat Lucene-moottorille millaista dataa tietokannan tulee ohjelmalta odottaa. solr_config.zip tiedosto pitää ladata järjestelmään ennen dokumenttien lataamista.

Näiden pohjalta päätetty paloitella Wikipedia-artikkelit siten, että jokainen Wikipedia sivu (artikkeli) on oma dokumenttinsa ja jokaisesta dokumentista tallennetaan tiedot Lucene tietokantaan kenttiin: pageId, title ja body, missä pageId on artikkelin yksilöivä id-numero, title on artikkelin otsikko ja body sisältää dokumentin raakatekstin. Esimerkiksi sivulla https://en.wikipedia.org/?curid=387931, curid 387931 on artikkelin yksilöivä pageId, sivun title on French curve ja body on artikkelin tekstisisältö. Samalla artikkeleista poistettu kuvat, kuvatekstit, sekä alaotsikot, jotka eivät palvelun käytön osalta olleet tarpeellisia.

Wikipedia-artikkeleiden lataaminen onnistuu kätevästi Wikipedian itse tarjoamasta sivujen export-palvelusta [32], johon voi suoraan syöttää haluamansa kategorian nimen, jonka jälkeen halutut sivut voi ladata yhtenä xml-tiedostona.

Ennen kuin artikkelit voitiin ladata järjestelmään, kyseinen xml-tiedosto piti lukea ja parseroida, sekä poistaa tarpeettomat tekstikentät. Parserointia varten ohjelmaan rakennettu oma luokka (XmlParser), joka tekee tämän automaattisesti.

5.4 Harjoitusaineiston luominen ja sovelluksen opettaminen

Kun pohjadata on ladattu Retrieve and Rank palvelun tietokantaan, voidaan palvelun luokittelijan (Ranker) harjoittaminen aloittaa. Harjoittaminen tarkoittaa prosessia, jossa pohjadatasta on olemassa kysymyksiä, joihin jo tiedetään oikeat & väärät vastaukset. Se voisi tarkoittaa esimerkiksi seuraavaa riviä:

Taulukko 2 Harjoituskysymyksen esimerkki

What does aerial viewpoint mean?	2872234	3	21482414	2	16960713	1

Missä ensimmäisenä on kysymys. Seuraavaksi ilmoitetaan Wikipedia artikkelin yksilöivä pageId-tunnus, sekä sivun relevanssi kysymykseen. Eli taulukossa 2 esitettyyn kysymykseen tuottaa parhaimman vastauksen pageId 2872234, kun pageId 16960713, ei tuota kunnollista vastausta.

Projektissa harjoituskysymyksien vastaukset on luokiteltu neljän tason mukaan seuraavasti:

- 0 Sivu ei vastaa kysymykseen laisinkaan.
- 1 Sivussa viitataan kysymyksessä mainittuun asiaan.
- 2 Sivu vastaa osittain kysymykseen.
- 3 Sivu sisältää täydellisen vastauksen kysymykseen.

IBM:n ohjesivuston mukaan näitä harjoituskysymyksiä pitää olla minimissään 49 kappaletta. Kun harjoitusaineisto on valmis, luokittelijan harjoittaminen tapahtuu neljässä vaiheessa. Oletuksena, että järjestelmä on muilta osin toimintakuntoinen ja harjoituskysymykset laadittu:

1. Käsittele harjoituskysymys ja sivu - relevanssi aineisto harjoitus tietojoukoksi.

2. Luo Bluemix portaaliin luokittelija ja lataa harjoitus tietojoukko siihen.
3. Testaa ja vertaa luokittelijan antamia tuloksia normaaliin Lucene tietokantakyselyyn ja käyttäjien olettamiin vastauksiin nähden.
4. Korjaa, muuta ja tarkenna harjoituskysymys ja sivu - relevanssi aineistoa kohdan 3 vastauksien perusteella.

[21]

Tätä mallia toistettava niin kauan, kunnes palvelu antaa riittävän tarkkoja vastauksia.

Edellisen listan kohdassa 1 mainitulla harjoitus-tietojoukolla tarkoitetaan Taulukossa 2 esitetyn kysymys ja sivu-relevanssi aineiston kaltaisten rivien lähettämistä Retrieve and Rank -sovelluksen harjoittajalle, joista jokaiselle lähetetylle riville sovellus palauttaa ominaisuusvektoriarvot. Vastauksena saatavien ominaisuusvektorien lukumäärään voi vaikuttaa lähetyksen yhteydessä erillisen SolrQuery.setRows funktiokutsun avulla. Nämä kysymys-vastausparien vektoriarvot muodostavat lopuksi sovelluksen harjoitustietojoukon. Ominaisuusvektoriarvot on helpointa mieltää Retrieve and Rank neuroverkon neuronien paino ja kynnysarvoiksi. Kun koko harjoituskysymys- ja sivurelevanssiaineisto on käsitelty, sekä palautetut ominaisuusvektoriarvot talletettu erilliseksi csv-tiedostoksi, luodaan toimiva luokittelija Retrieve and Rank palveluun lähettämällä ominaisuusvektorit (harjoitustietojoukko) palvelun syötteeksi.

Käytännössä harjoitustietojoukon muodostus tarkoittaa kysymys – vastaus parien lähettämistä Retrieve and Rank -sovellukselle rivikohtaisina cURL tai REST -kyselyinä. Jokainen tällainen kysely on muotoa

https://gateway.watsonplatform.net/retrieve-and-rank/api/v1/solr_clusters/{solr_cluster_name}/solr/{solr_collection_name}/fcselect&q={query}&wt=json&fl=id,title&rows=30, missä

- {solr_cluster_name} → Retreive and Rank -sovellukseen luodun klusterin nimi.
- {solr_collection_name} → Retrieve and Rank -sovellukseen luodun tietojoukon nimi.
- {query} → Harjoitusaineiston yksittäisen rivin kysymys – vastaus parit.

- rows=30 → Rivimäärä; montako ominaisuusvektori-riviä Retrieve and Rank -sovellus jokaiseen yksittäiseen harjoitusrivi-kyselyyn palauttaa.

Sovelluskehityksessä huomattu, että Retrieve and Rank -alustassa tästä cURL tai REST kyselyn kysymyksestä muodostuu aluksi normaali kysely Lucene-tietokantaan, missä rivimäärän arvo määrittää montako vastausta Lucene-kyselyyn saadaan. Tämän jälkeen Retrieve and Rank -sovellus määrittää saaduille vastauksille vektoriarvot. Jos Lucene-kyselyn palauttamien vastauksien joukossa on oikean vastauksen sisältäviä rivejä antaa sovellus näille riveille suuremman merkitsevyysarvon (ground truth), jonka arvo on kysymys-vastausparien mukainen. Jos taas joku Lucene-kyselystä palautuneesta rivistä ei sisällä oikeaa vastausta annetaan tälle merkitsevyys-arvo 0. Taulukossa 3 esitetty yksi Retrieve and Rank -sovellukselle esitetyn harjoituskyselyn palauttamista vektori-arvoista.

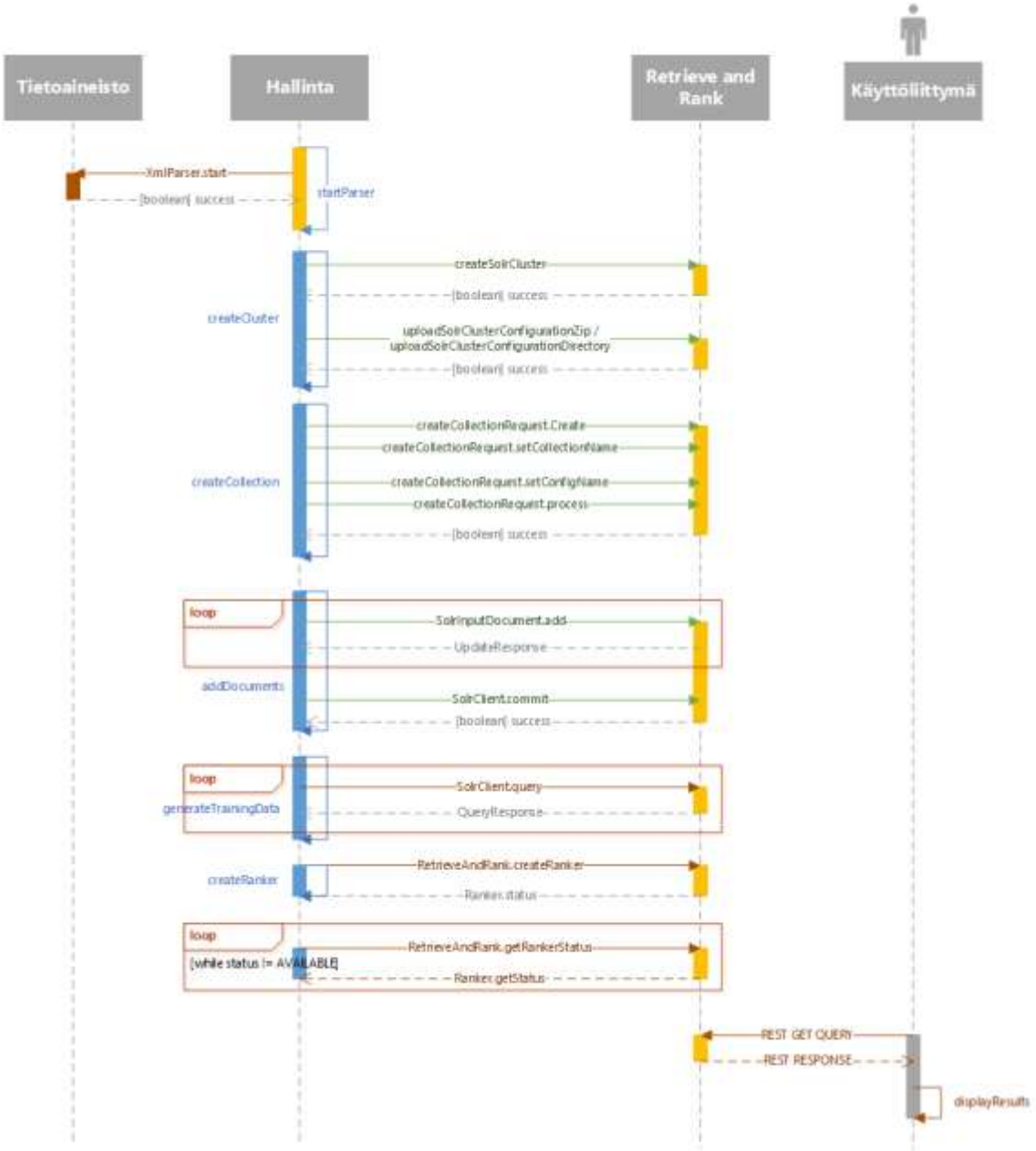
Taulukko 3 Kysymys-parin ominaisuusvektori esimerkki

question_id	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	r1	r2	s	ground_truth
9dffdc7c-73d2-4392-824e-197f2f237eec	5,01	2,51	0,16	2,69	4,95	2,52	0,17	2,70	0,04	0	0	0	0,83	0	0,69	10,00	3

Esimerkiksi, jos kysymykseen on olemassa vain kolme oikea-vastaus pageId:tä, kuten taulukossa 2 ja kyselyn rivimäärä arvona on 30 saadaan kyselyyn vastaukseksi 30 ominaisuusvektoria, joista vain kolmessa on ground_truth arvo suurempi kuin 0.

5.5 Projektityön toimintalogiikka

Tässä kuvataan Retrieve and Rank palvelun pystyttämisestä ja datan latauksesta vastuussa oleva Java toteutus, sekä Retrieve and Rank palveluun kyselyitä tekevä ja käyttöliittymää hallinnoivan Node.js moduuli.



Kuvio 2 Projektin sekvenssi kaavio

Java toteutus

Koska Java toteutus käsittää yli 1800 riviä koodia / kommentteja, ei tässä työssä käydä kaikkia kirjoitettuja koodeja tai moduuleita yksityiskohtaisesti läpi. Koodit ja kommentit on kokonaisuudessaan luettavissa GitHub-versionhallinta ohjelmistokehitysalustasta [33]. Toteutus on pyritty kirjoittamaan siten, että sen uusiokäyttäminen olisi mahdollisimman yksinkertaista.

Sovelluksesta löytyy seuraavat luokkatiedostot:

- Bluemix
  - BController.java
    - Päätiedosto, joka hallinnoi koko sovellusta.
- Data parser
  - PageData.java
    - Yksittäisen Wikipedia sivun ilmentymä.
  - TextNormalizer.java
    - Poistaa Wikipedia xml-tekstistä tarpeettomat html-tagit (esim. <b> ja &npsp;).
  - XmlParser.java
    - Lukee Wikipedia xml-tiedoston ja luo jokaisesta artikkelista PageData ilmentymän taulukkoon.
- Bluemix Objects
  - BCluster.java
    - Vastuussa klusterin luomisesta Bluemix pilvipalveluun.
  - BCollection.java
    - Vastuussa kokoelman luomisesta Bluemix pilvipalveluun.
  - BRanker.java
    - Vastuussa luokittelijan luomisesta Bluemix-pilvipalveluun.
  - SolrQueries.java
    - Hoitaa kaikki kyselyt Bluemix-palveluun. Myös harjoitusaineiston osalta.

Kuten listassa jo mainittu BController.java -tiedostolla on ns. päävastuu. Toisin sanoen palvelua käyttävän henkilön tarvitsee käyttää ainoastaan BController.java -tiedostosta julkaistuja funktioita koko sovelluksen käyttämistä varten. Alle kuvattu kuinka suoraviivainen Retrieve and Rank -palvelun käyttöönotto tapahtuisi BController luokan avulla.

```
// alusta BController luokka ja anna sille parametrina tiedosto, johon on tallennettu esiasetetut yhteysasetukset Retrieve and Rank -palveluun (mm. käyttäjätunnus, salasana, yhteyden url).
BController controller = new BController("/connection.dat");

// käynnistä wikipediasta ladatun xml-tiedoston parserointi.
controller.startParser("src/main/resources/technical_drawing_wikidump.xml");
```

```
// luo Retreive and Rank palveluun klusteri ja lataa solr_config.zip tiedosto.
controller.createCluster();

// luo Retrieve and Rank palveluun kokoelma.
controller.createCollection();

// lisää Retrieve and Rank palveluun dokumentit.
controller.addDocuments();

// luo ladatuista dokumenteista harjoitus-aineiston Trainingdata.csv
controller.generateTrainingData();

// luo Retrieve and Rank palveluun luokittelijan (Ranker), sekä lähettää edellisessä vai-
heessa luodun esiopetusaineiston (Trainingdata.csv -tiedosto) palvelun käytettäväksi.
controller.createRanker();

String queryMsg = "What is French curve?";
// Lucene tietokantaan voi nyt esittää perinteisiä kysymyksiä (ei keinoälyä).
controller.makeStandardQuery( queryMsg );

// tai Retrieve and Rank palveluun voi esittää kysymyksiä luokittelijalle.
controller.searchAndRankQuery( queryMsg );
```

Kuten yllä tehdyistä kyselyistä Retrieve and Rank palveluun voi jo päätellä, voidaan koko toteutus, käyttöliittymä mukaan lukien, tehdä suoraan Java sovelluksessa. Mutta koska työssä käytetyllä sovelluksella oli kiireellisempi aikataulu, päädytty toteuttamaan käyttöliittymä lähes toimivalla Node.js -ohjelmalla (johon saatu aineisto opinnäytetyön antajalta).

Node.js toteutus

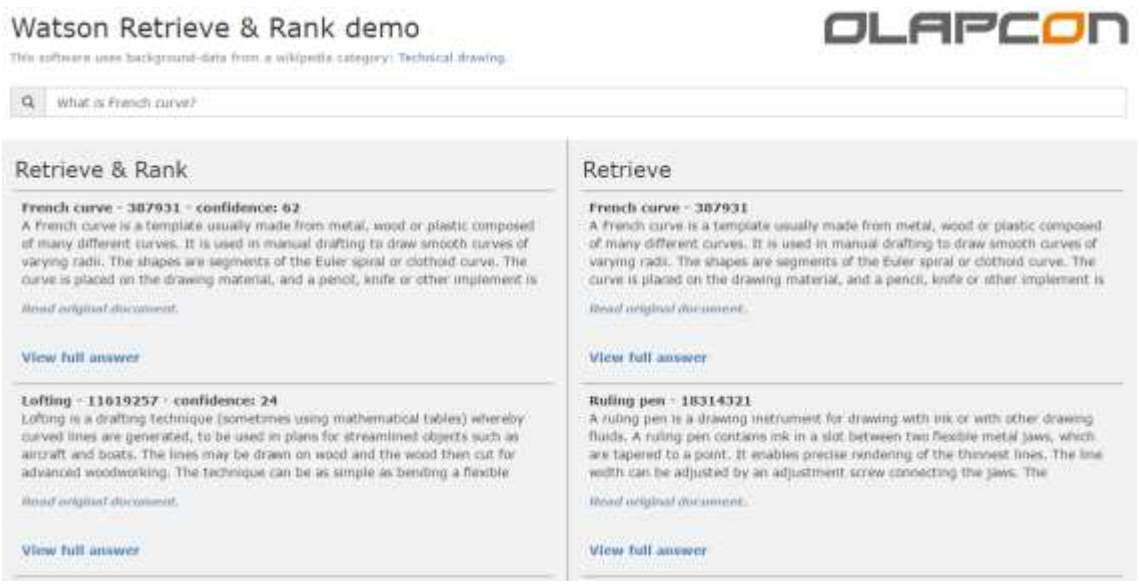
Node.js sovelluksen etuna aiemmin esiteltyyn Java toteutukseen on se, että sen ainoana tarkoituksena on tarjota käyttäjälle käyttöliittymä kysymyksien esittämistä varten, joten koko Node.js toteutuksen voi rakentaa toimimaan erillisenä yksikkönä suoraan Bluemix pilvipalvelussa ja siten vähentää väärinkäytön mahdollisuuksia. Tällä tavalla mm. käyttäjätunnuksien ja salasanojen käyttäminen saadaan piilotettua loppukäyttäjältä.

Kuva 3 Node.js käyttöliittymä

Node.js toteutus toimii kahdella tasolla seuraavasti:

1. REST rajapinta Retrieve and Rank palvelun kutsuja varten.
2. Kuvan 3 mukaisen käyttöliittymä sivun esitykseen verkko-osoitteessa.

Kun käyttäjä kirjoittaa kyselyn kuvan 3 mukaiseen kysymyslaatikkoon lähettää käyttöliittymä Ajax GET pyynnön REST-rajapinnalle, joka suorittaa samalle kysymykselle kaksi hakua. Ensimmäinen haku on normaali Lucene kysely tietokantaan ja toinen kysely on Retrieve and Rank kysely suoraan keinoälysovellutukselle. REST-rajapinta palauttaa molemmat vastaukset kyselyihin, jonka jälkeen sivusto listaa vastaukset vierekkäin kuvan 4 mukaisesti.



Kuva 4 Käyttöliittymän vastauskysely

Kuvaan 4 on otettu mukaan vain kaksi ensimmäistä vastausta. Käyttöliittymä pyytää palvelusta oikeasti yhteensä 25 vastausta, jotka kaikki listataan sivulla allekkain saapumisjärjestyksessään. Kuvassa 4 vasen puoli (Retrieve and Rank)

on keinoälylle tehdyn kyselyn vastaukset ja oikea puoli (Retrieve) on Lucene tietokantaan lähetetyn tavallisen kyselyn vastaukset. Näin käyttöliittymästä on helppo vertailla opetetun vs. puhtaan Lucene-tietokanta kyselyn vastauksien eroja.

5.6 Työn vaikeudet ja ongelmakohdat

Projektin aikana kohdattu useita haasteita, joista isoimmat mutkat selostetaan tässä (niiden vaatiman ajan tai ohjelmoitavuuden perusteella).

Projektista ajallisesti suurimman työmäärän aiheutti tietoaineiston parseroiminen, opetuskysymysten laatiminen ja harjoitus-tietojoukon muodostaminen.

Tietoaineiston parseroiminen

Wikipedian MediaWiki-sivustolta löytyy pitkä lista jo olemassa olevia Wikipedia-parsereita [34]. Wikipedia xml-aineiston parseroimiseen kokeiltu muutamia näistä kirjastoista, joista mm. Sweble Wikitext Parser tuotti ihan luettavia html sivuja, mutta valitettavasti parserin tuottama teksti sisälsi sovelluksen kannalta paljon tarpeetonta tietoa, jonka Retrieve and Rank -sovellukseen lataamiseen ei ollut mitään järkeä. Esimerkiksi lähdeviittaukset (References) tai Galleria-otsikoiden sisältö, joista Galleria sisälsi vain sivulla olevien kuvien otsikkotekstejä. Itse kuvat eivät tule xml-tiedoston mukana edes viittauksina [35]. Koska kertaalleen parseroidun xml-tiedon uudelleenparseroimisessa ei olisi ollut järkeä, tehty projektiin täysin oma xml-parseri, joka tuottaa kerralla ladattavaa tietoa xml-materiaalista.

Opetuskysymysten laadinta

Projektiin sopivan xml-parserin kehittämistä ajallisesti suuremmaksi haasteeksi osoittautui opetuskysymysten laadinta. Retreive and Rank -dokumentaation mukaan sovelluksen vaatima uniikkien kysymyksien minimimäärä on 49 kappaletta [21]. Aineistoon ei löytynyt hakukoneilla valmiita kysymyksiä, joten kysymykset lopulta laadittu manuaalisesti jokaisesta sivusta erikseen. Vaikka aineiston valittu aihealue olikin työn tekijää kiinnostava, oli sivustojen lukeminen ja niistä tehtävien

kysymyksien miettiminen ajallisesti vaativa tehtävä. Kysymyksien piti lisäksi simuloida asiantuntijoiden esittämiä kysymyksiä eli olla riittävän tarkkaan määriteltyjä, että tavallisen tietokantakyselyn ei voisi olettaa niissä onnistuvan.

Harjoitus-tietojoukon muodostaminen

Opetuskysymyksistä luotavien kyselyiden tekeminen Retrieve and Rank -sovellukselle osoittautui yllättävän hankalaksi, sillä sovellukselle olevissa ohjeissa ei ollut tarpeeksi yksityiskohtaisesti kerrottu, miten opetuskysymyksistä muodostetaan koodillisesti harjoitustietojoukko lopullisen Luokittelijan luomista varten.

Ensimmäinen ratkaisuun johtava havainto vaati IBM:n sivuillaan tarjoama python-koodin lukemista ja purkamista toimintalogiikan ymmärtämiseksi [28]. Tämä tärkeä havainto oli, että harjoitustietojoukko muodostetaan lähettämällä cURL-muotoisia REST-kutsuja sovelluksen harjoittajalle, joka palautti vastaukseksi ominaisuusvektoriarvot yhtä kysymys-oikeat vastaukset riviä kohden. Toinen havainto vaati Luokittelija-algoritmin harjoittajalle lähetettävän cURL kutsun testaamista eri lähtöarvoilla. Ongelmana tässä oli, että tavallinen Lucene tietokantakysely alustettiin koodilla

setRequestHandler("select")

ja harjoittelijan cURL-kutsu ohjeistettiin tekemään koodilla

setRequestHandler(”fcselect”),

joka ei toiminut Java-puolella lainkaan. Vasta kun koodi muutettiin muotoon

setRequestHandler("/fcselect")

palautti Retrieve and Rank -sovellus halutut ominaisuusvektoriarvot. Ongelman aiheutti siis puuttunut kauttaviiva (/), josta ei ollut mainintaa tai esimerkkiä missään sovellukselle olemassa olevissa ohjeissa.

6 Tulokset ja johtopäätökset

Tuloksissa esitellään sovelluksesta saatavat vastaukset molemmilla hakuteknii-koilla (tavallisena Lucene-kyselynä, sekä Luokittelija-kyselynä). Haut on tehty sekä harjoitusaineistossa mukana olleilla kysymyksillä, että harjoitusaineistosta

puuttuneilla kysymyksillä. Palvelu palauttaa molempaan kyselyyn maksimissaan 25 tulosta, joista jokainen saa pistearvon 1:n ja 26:n väliltä sen mukaan monentena listalta löytyy oikea vastaus. Arvo 26 tarkoittaa, ettei oikeata vastausta ollut listalla laisinkaan. Eli jokaista kysymystä kohden annetaan pisteet Lucene-haulle, sekä Luokittelija-haulle. Esimerkiksi taulukon 5 tuloksissa kysymys 1:n ja rivimäärä 10:n palauttaman Lucene-kyselyn tulos 6 tarkoittaa, että palvelun palauttamista vastauksista oikea vastaus oli kuudentena listallla ja Luokittelijan tulos 3 tarkoittaa, että oikea vastaus oli listalla kolmantena.

IBM:n ohjeet kertovat, että Luokittelijan tarkkuutta voi parantaa antamalla sovellukselle enemmän opetuskysymyksiä tai muuttamalla harjoitusaineiston muodostusvaiheessa lähetettyjen harjoituskysymyksiin palautettavien ominaisuusvektorien lukumäärää (rivimäärä, jonka maksimiarvo on 100 kpl). [21] Koska lisäkysymyksien laatimiseen ei tässä työssä ollut aikaa, kokeiltu Luokittelijan palauttamien tuloksien muuttumista kolmella eri rivimäärä -arvolla:

- 10 (default),
- 50,
- 80.

Sovellukselle tehty yhteensä 54 kysymys – oikea vastaus pareja, joista 49 otettu Luokittelijan kouluttamista varten ja jäljelle jäävät 5 jätetty palvelun toiminnan tarkistamista varten. Koska järjestelmän toiminnan tarkistamista varten jäi vain 5 algoritmille kokonaan opettamatonta kysymystä, tuloksiin otettu lisäksi harjoitusaineistosta mukaan 11 kysymystä tarkasteluun.

Taulukko 4 Toimivan järjestelmän vertailukysymykset

Kysymyksen numero	Kysymys
Kysymykset harjoitusaineistosta	
1	How to present data about 3D-object?
2	What information is needed in a house blueprints?
3	What is projected tolerance zone?
4	How to illustrate object assembly?
5	What are the instructions to build a wall?
6	What devices are there to calculate area size?
7	What is a good drawing frame?

8	How to record or present old discovered items?
9	How to develop unique letters?
10	What instructions are incorporated in manufacturing documents?
11	What tools can be used to draw perspective drawings?
Kysymykset ei harjoitusaineistosta	
12	How is paper sizes defined in different countries?
13	Is floor plan done in bird view?
14	What describes the methods and work phases of construction plans?
15	What name is used for ink pencils?
16	What special purpose paper templates exists?

Taulukko 5 Tulokset

	Rivimäärä 10		**Rivimäärä 50**		**Rivimäärä 80**	
Kysymyksen numero	**Lucene**	**Luokittelija (luottamus)**	**Lucene**	**Luokittelija (luottamus)**	**Lucene**	**Luokittelija (luottamus)**
1	6	3 (36)	6	3 (34)	6	2 (28)
2	2	1 (43)	2	1 (48)	2	1 (67)
3	5	1 (41)	5	1 (57)	5	1 (99)
4	10	3 (41)	10	3 (39)	10	1 (61)
5	26	4 (26)	26	7 (28)	26	2 (51)
6	4	11 (9)	4	1 (46)	4	6 (17)
7	22	4 (35)	22	9 (8)	22	7 (10)
8	8	10 (14)	8	10 (11)	8	10 (3)
9	26	1 (45)	26	2 (41)	26	1 (57)
10	3	4 (14)	3	4 (26)	3	3 (18)
11	26	26 (0)	26	26 (0)	26	26 (0)
Luokittelija vs. Lucene	**49 %**		**49 %**		**43 %**	
12	1	1 (46)	1	1 (61)	1	1 (100)
13	1	2 (33)	1	2 (42)	1	1 (89)
14	10	4 (18)	6	4 (18)	10	2 (11)
15	3	9 (16)	3	14 (5)	3	13 (3)
16	10	1 (47)	10	20 (0)	10	13 (3)
Luokittelija vs. Lucene	**68 %**		**-95 %**		**-20 %**	

Luokittelija vs. Lucene solun arvot on laskettu jakamalla Luokittelijan saama yhteispistemäärä Lucenen saamalla yhteispistemäärällä. Pienempi luku tarkoittaa Luokittelijan tuottaman pistemäärän olevan pienempi (parempi), kuin tavallisen Lucene tietokantakyselyn. Negatiivinen arvo taas tarkoittaa, että tavallinen Lucene tietokantahaku on sisältänyt tarkempia vastauksia.

Tuloksista tärkeätä huomata, että luokittelija ei saanut kaikissa kysymyksissä korotettua oikeata vastausta listalla aina ensimmäiseksi, edes opetetuista kysymyksistä. Tämä ei tietysti ole algoritmin pääasiallinen tarkoituskaan, vaan tarkoitus on parantaa hakutuloksien järjestystä tavalliseen hakualgoritmiin verrattuna.

Harjoitusaineistosta otetuilla, algoritmille opetetuilla kysymyksillä (1-11), luokittelijan vastaukset vaikuttivat tuottavan sitä tarkempia vastauksia, mitä suurempaa rivimäärää luokittelijan harjoituksessa on ollut käytössä. Kysymykset, jotka eivät olleet mukana harjoitusaineistossa (12 – 16) taas tuottavat tarkempia vastauksia pienemmällä rivimäärällä.

Syy miksi suurempi rivimäärä parantaa ennestään opetettujen kysymyksien vastaustarkkuutta, mutta ei harjoittamattomien kysymysten tarkkuutta johtuu ylisovittamisesta [30]. Toisin sanoen, kun harjoitusvaiheessa pyydämme ominaisuusvektoreille suurempaa rivimäärää, tulee harjoitustietoihin mukaan suurempi määrä vääriä vastauksia sisältäviä sivuja. Tämä auttaa toki sovellusta parantamaan hakutuloksia ennestään opetetuille kysymyksille, mutta vastavuoroisesti tämän takia algoritmi ei kykene yleistämään hakutuloksia ennestään harjoittamattomille kysymyksille. Algoritmin toiminnan tasolla tätä voisi kuvata, että neuronien paino- ja kynnysarvot on tarkennettu liian tarkasti harjoitusaineiston tiedoilla, joten algoritmi ei kykene toimimaan niin tarkasti ennalta opettamattomilla kysymyksillä.

Tämän projektin tuotos on täysin avointa lähdekoodia ja se on ladattavissa GitHub.com:sta (linkki löytyy lähteistä). [31]

7 Päätelmät

Työn ei ollut tarkoitus toteuttaa täydellisesti toimivaa järjestelmää, vaan osoittaa että IBM:n pilvipalveluna tarjoama koneoppimisen Retrieve and Rank -sovelluksen käyttöönotto onnistuu ja on nopeasti toteutettavissa.

Asiakaskäytössä Retrieve and Rank -sovelluksen itseoppiva algoritmi on nykyaikainen ja oletettavasti oikein konfiguroituna myös toimiva ratkaisu. Tärkeintä on muistaa tuottaa paljon opetusmateriaalia, sekä lisäopettaa algoritmia määräajoin aina uudelleen loppukäyttäjien tekemien kysymyksien, sekä ilmoittamien käyttökokemuksien pohjalta.

Koska tässä työssä käytetty sovellukselle ohjeistettua minimimäärää kysymyksiä, ei järjestelmän todellista käytettävyyttä ole pystytty kunnolla tutkimaan tai testaamaan. Tätä varten harjoituskysymyksiä pitäisi olla vähintään kymmenkertainen määrä nyt käytettyyn nähden, sekä testausta varten testauskysymyksiä olisi hyvä olla 20 – 30% harjoituskysymyksiin verrattuna.

Retrieve and Rank kehitys- ja parannusehdotukset

Jos pohjadata on hyvin laaja (määrällisesti, sekä eri aihealueista koostuva), ei Retrieve and Rank sovellusta ole luultavasti mahdollista sovittaa toimimaan riittävän luotettavasti koko datalle kerralla.

Jos käytettävä data on helposti luokiteltavissa aihe-alueisiin, voisi sovelluksen toteutusta miettiä kaksivaiheiseksi. Tarkoittaen sitä, että ensiksi pohjadata luokitellaan ja näihin luokkiin kerätään hakusanoja ja niiden synonyymejä (esimerkiksi aineistossa usein esiintyviä sanoja). Tämän pohjalta lopullisessa järjestelmässä ensimmäiseksi suoritettaisiin käyttäjän tekemästä kyselystä aihealueen (ns. luokan) etsintä. Lisäksi luotettavuuden parantamiseksi olisi viisasta varmistaa käyttäjältä onko määritetty luokka oikea. Tällaisen esivalinnan toteutus olisi tehtävissä esimerkiksi yhdistäen IBM:n Retrieve and Rank sekä Conversion sovellusalustoja. Esivalinnan jälkeen suoritettaisiin virallinen aineiston etsintä halutusta luokasta käyttäjän esittämillä hakusanoilla.

Retrieve and Rank -sovellutus on suunniteltu ilmeisesti juuri tätä silmällä pitäen, sillä jokainen Retrieve and Rank -alustaa tarvitseva pohjadata on mahdollista jäsentää toisistaan erillisiin yksiköihin (sovellus käyttää näistä nimitystä Collection).

Tähän hyvänä esimerkkinä käy Wikipedia-aineisto, joka on nykyisellään jaoteltavissa tuhansiin kategorioihin (todellista lukemaa ei saatavilla verkosta, sillä uusia kategorioita syntyy jatkuvasti).

Tässä työssä ei perehdytty siihen, miten palvelun toimintaan vaikuttaisi, jos kysymyksiin palautettavien vastausten lukumäärä laitettaisiin huomattavasti suuremmaksi. Toteutuksessa Lucene-tietokannasta pyydettiin vain 25 osuvinta sivua, jonka jälkeen sivut järjesteltiin luokittelijan toimesta paremmuusjärjestykseen. Olisikin mielenkiintoista tietää miten vaikuttaisi, jos Lucene-tietokannasta pyydettäisiinkin 100 vastausta, nykyisen 25:n sijaan ja suodatettaisiin käyttäjälle näkyviin vain 25 ensimmäistä / luotettavinta vastausta.

8 Lopuksi

Etsittäessä kirjallista materiaalia Learning to Rank tai Retrieve and Rank -keinoälystä mm. koulun kautta saatavilla olevista tietokannoista Finna ja eLibrary, ei aiheesta löytynyt yhtään hyödyllistä painosta (kirjaa tai lehtiartikkelia). Englanninkielisillä hakusanoilla etsimällä suosituilla verkko-hakukoneilla, löytyi taas lukuisia blogeja, artikkeleita ja tieteellisiä tekstejä, joista suurin osa käsitteli peliohjelmointiin liittyvää tekoälyä. Aiheeseen ei löytynyt myös yhtään opinnäytetyötä (theseus). Suomenkielisen materiaalin puutos ihmetyttää, sillä ensimmäiset askeleet itsenäisesti oppiviin algoritmeihin on otettu jo ennen tietokoneiden olemassaoloa vuonna 1943. [9].

Opinnäytetyöprosessi on ollut erittäin antoisa kokemus ja etenkin työn tekemisen aikana on päästy tutustumaan erinomaisesti tämänhetkisen sovelluskehityksen kärkirintamaan. Nykypäivänä on hyvin todennäköistä, että ohjelmoija joutuu työskentelemään useamman kuin yhden eri ohjelmointirajapinnan / järjestelmän kanssa, joiden yhteensovittaminen tuo aina omat haasteensa. Uskonkin vakaasti, että opinnäytetyön tekemisestä saamani opit tulevat kantamaan minua pitkälle myös työelämässä.

Lähteet

[1] IBM, ”Chronological History of IBM,” 2016. [Online]. Available: http://www-03.ibm.com/ibm/history/history/decade_1880.html. [Haettu 21 12 2016].

[2] IBM, ”IBM basics,” 2016. [Online]. Available: http://www.ibm.com/ibm/responsibility/basics.shtml. [Haettu 27 12 2016].

[3] IBM, ”IBM Annual Report 2015,” 2015. [Online]. Available: http://www.ibm.com/annualreport/2015/assets/img/2016/02/IBM-Annual-Report-2015.pdf.

[4] IBM, ”Smarter Planet,” 2016. [Online]. Available: http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/smarterplanet/. [Haettu 27 12 2016].

[5] IBM, ”IBM builds a smarter planet,” 2016. [Online]. Available: http://www.ibm.com/smarterplanet/us/en/. [Haettu 27 12 2016].

[6] IBM, ”Cognitive Business Operations,” 2016. [Online]. Available: https://www.ibm.com/middleware/us-en/solutions/intelligent-business/process-decisions-sense-respond-learn/#/welcome-to-the-cognitive-era. [Haettu 27 12 2016].

[7] R. Schapire, ”COS 511: Theoretical Machine Learning,” 4 2 2008. [Online]. Available: http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf. [Haettu 30 10 2016].

[8] N. Mccrea, ”An Introduction to Machine Learning Theory and Its Applications: A Visual Tutorial with Examples,” 2016. [Online]. Available: https://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer. [Haettu 30 10 2016].

[9] S. Christos ja S. Dimitrios, ”Neural Networks,” 2016. [Online]. Available: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html. [Haettu 30 10 2016].

[10] T. Kohonen, ”Self-organized formation of topologically correct feature maps,” *Biological Cybernetics,* osa/vuosik. 43, nro 1, pp. 59-69, 1 1982.

[11] M. Nielsen, ”Using neural nets to recognize handwritten digits,” 1 2016. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap1.html. [Haettu 30 10 2016].

[12] I. Vasilev, ”A Deep Learning Tutorial: From Perceptrons to Deep Networks,” 2016. [Online]. Available: https://www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks. [Haettu 30 10 2016].

[13] ”Introduction to Deep Neural Networks,” 2016. [Online]. Available: https://deeplearning4j.org/neuralnet-overview. [Haettu 17 11 2016].

[14] M. Heikkilä, ”Kamerasta apu,” Turun Sanomat, 28 12 2009. [Online]. Available: http://www.ts.fi/teemat/auto+ja+liikenne/98950/Kamerasta+apu+anastettujen+autojen++etsintaan. [Haettu 18 2 2017].

[15] V.-P. Kallberg ja M. Saarinen, ”Katsastamattomien ajoneuvojen tunnistaminen liikennevalvonnassa,” *VTT TIEDOTTEITA - RESEARCH NOTES 2518,* p. 30, - 12 2009.

[16] S. Dufresne, ”Backpropagation neural network software (3 layer),” 22 4 2016. [Online]. Available: http://rimstar.org/science_electronics_projects/backpropagation_neural_network_software_3_layer.htm. [Haettu 17 11 2016].

[17] IBM, ”Retrieve and Rank service documentation,” 2016. [Online]. Available: https://www.ibm.com/watson/developercloud/doc/retrieve-rank/. [Haettu 27 10 2016].

[18] B. B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao ja Z. Zheng, ”Early Exit Optimizations for,” tekijä: *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, New York, 2010.

[19] J. Shindelar, ”Configure Search API Solr to Use Stopwords When Indexing Content,” Drupalize.Me, 11 9 2015. [Online]. Available: https://drupalize.me/blog/201508/configure-search-api-solr-use-stopwords-when-indexing-content. [Haettu 2 6 2017].

[20] B. Itay ja O. Ish-Am, 2012/13. [Online]. Available: http://www.cs.tau.ac.il/~itayberm/L2RProject/L2R.htm. [Haettu 27 10 2016].

[21] IBM, ”Preparing training data,” 2016. [Online]. Available: http://www.ibm.com/watson/developercloud/doc/retrieve-rank/training_data.shtml. [Haettu 21 12 2016].

[22] N. Storås, ”Maailman tiedon määrä selvitettiin,” 14 2 2011. [Online]. Available: http://www.tivi.fi/Uutiset/2011-02-14/Maailman-tiedon-m%C3%A4%C3%A4r%C3%A4-selvitettiin-3183003.html. [Haettu 27 10 2016].

[23] ”Executive Summary: Data Growth, Business Opportunities, and the IT Imperatives | The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things,” 19 2 2014. [Online]. Available: https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm.

[24] IBM, ”Retrieve and Rank - API,” 2016. [Online]. Available: https://www.ibm.com/watson/developercloud/retrieve-and-rank/api/v1/?node#introduction. [Haettu 10 2016].

[25] GitHub, ”Retrieve and Rank application overview,” 18 4 2016. [Online]. Available: https://github.com/watson-developer-cloud/professor-languo.

[26] IBM, ”Using the Retrieve and Rank Web Interface,” IBM, 2017. [Online]. Available: https://www.ibm.com/watson/developercloud/doc/retrieve-rank/ranker_tooling.html. [Haettu 2 6 2017].

[27] IBM, ”Overview of the Retrieve and Rank service - About Apache Solr,” 2017. [Online]. Available: https://www.ibm.com/watson/developercloud/doc/retrieve-rank/index.html#solr. [Haettu 2 6 2017].

[28] IBM, ”Retrieve and Rank - Tutorial,” 2016. [Online]. Available: http://www.ibm.com/watson/developercloud/doc/retrieve-rank/tutorial.shtml. [Haettu 20 12 2016].

[29] Wikimedia Foundation, Inc., ”Category:Technical drawing,” 2016. [Online]. Available: https://en.wikipedia.org/wiki/Category:Technical_drawing. [Haettu 2016].

[30] The Apache Software Foundation, ”Index File Formats,” 21 6 2013. [Online]. Available: http://lucene.apache.org/core/3_5_0/fileformats.html#Overview.

[31] E. Hatcher, ”Introduction to Solr,” 11 9 2011. [Online]. Available: http://www.slideshare.net/erikhatcher/introduction-to-solr-9213241. [Haettu 2016].

[32] Wikipedia, ”Export pages,” 2016. [Online]. Available: https://en.wikipedia.org/wiki/Special:Export. [Haettu 20 12 2016].

[33] V. Kontturi, ”Retrieve and Rank Demo,” Individual, 11 2016. [Online]. Available: https://github.com/mogu84/RetrieveAndRankDemo.

[34] Wikipedia Project, ”Aternative parsers,” 2016. [Online]. Available: https://www.mediawiki.org/wiki/Alternative_parsers. [Haettu 29 12 2016].

[35] Open Source Research Group, ”Sweble,” 2016. [Online]. Available: http://sweble.org. [Haettu 29 12 2016].

[36] M. Nielsen, ”Improving the way neural networks learn,” 1 2016. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap3.html. [Haettu 31 12 2016].