Tampereen yliopisto

Minja Survonen

LUONNOLLISESTA KIELESTÄ SQL-KYSELYKSI

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Toukokuu 2023

TIIVISTELMÄ

Minja Survonen: Luonnollisesta kielestä SQL-kyselyksi
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Toukokuu 2023

Relaatiotietokantoja käytetään paljon. Niihin voidaan varastoida tietoa, ja varastoitua tietoa voidaan hakea tietokannasta SQL-kielellä. Mahdollisuus tehdä tietokantakyselyitä luonnollisella kielellä hyödyttäisi heitä, joilla ei ole SQL-osaamista. Mikäli tietokannasta tietoa tarvitsevalla on jokin motorinen vamma, hän voisi hyötyä siitä, että luonnollista kieltä olevan kysymyksen voisi esittää puheena. Tässä kandidaatintutkielmassa tarkasteltiin tapoja muuntaa luonnollista kieltä olevaa tekstiä tai puhetta SQL-kyselyksi. Tämä kandidaatintutkielma on muodoltaan kirjallisuuskatsaus.

Tekstistä voidaan muuntaa SQL-kysely sääntöpohjaisten lähestymistapojen avulla tai koneoppimista hyödyntäen. Sääntöpohjaisissa lähestymistavoissa muunnosprosessi perustuu ennalta määritettyihin sääntöihin. Sen sijaan koneoppimiseen perustuvissa lähestymistavoissa tärkeää on opetusdata, jonka avulla järjestelmä oppii. Yhteen muunnosprosessiin voi sisältyä eri koneoppimisen muotoja.

Puheen muuntamisessa SQL-kyselyksi hyödynnetään samoja tekniikoita. Osa järjestelmistä muuntaa ensin puheen tekstiksi hyödyntäen puheentunnistusta, mutta osassa järjestelmistä muunnetaan puhesignaalit suoraan SQL-kyselyksi ilman tekstivälivaihetta. Puheen muuntamisessa SQL-kyselyksi koneoppimisen rooli on suurempi verrattuna muunnosprosessiin tekstistä SQL-kyselyksi.

Työssä tarkasteltiin keskeisiä lähestymistapoja esimerkkijärjestelmien kautta. Lisäksi käytiin läpi eri muunnosprosesseihin liittyviä haasteita. Eri järjestelmien välillä on eroja siinä, kuinka hyvin muunnosprosessi SQL-kyselyksi onnistuu. Moni järjestelmistä on käyttöalaltaan hyvin suppea, koska ne on rakennettu vain tietyn alan tietokannan tarpeisiin. Lisäksi osa järjestelmistä mahdollistaa vain yksinkertaisten SQL-kyselyiden muodostamisen, kun taas osassa järjestelmistä on mahdollista muuntaa myös monimutkaisempia kyselyitä. Lisäksi tarkasteltiin lähestymistapakohtaisesti juuri kyseiselle lähestymistavalle ominaisia haasteita.

Kandidaatintutkielmassa tarkasteltiin luonnollisen kielen muuntamista SQL-kyselyksi myös yleisellä tasolla. Työssä käsiteltiin sitä, miten erilaiset mittaustavat hankaloittavat eri järjestelmien vertailua. Lisäksi työ sisältää pohdintaa siitä, mitkä ovat muunnosprosessin mahdolliset kehityssuunnat. Myös koneoppimisen merkitys muunnosprosessien kehittymisen kannalta oli tarkastelun kohteena.

Avainsanat: relaatiotietokannat, SQL, luonnollinen kieli, sääntöpohjaisuus, koneoppiminen

SISÄLLYSLUETTELO

1 Johdanto

Relaatiotietokantoja käytetään paljon. Niistä voi hakea tietoa ja niitä voidaan päivittää SQL-kyselykielellä. SQL-kielen käyttö vaatii kuitenkin sen opettelua. Tästä syystä mahdollisuus tehdä tietokantakyselyitä luonnollisella kielellä mahdollistaisi sen, että tietokantakyselyitä voisi tehdä myös ilman SQL-osaamista.

Tietokantakyselyiden tekemistä luonnollisella kielellä on lähestytty rakentamalla sekä teksti- että puhekäyttöliittymiä. Tekstikäyttöliittymissä käyttäjä kirjoittaa luonnollista kieltä olevan kyselyn tekstimuodossa. Puhekäyttöliittymän tapauksessa taas käyttäjä voi puhua kyselyn ääneen. Kuten tekstikäyttöliittymä, myös puhekäyttöliittymä auttaa heitä, joilla ei ole SQL-osaamista. Lisäksi puhekäyttöliittymistä voivat hyötyä henkilöt, joille kirjoittaminen on haastavaa tai mahdotonta motoristen vaikeuksien vuoksi (Lyons ja muut, 2016).

Tärkeitä käsitteitä ovat luonnollinen kieli ja relaatiotietokanta. Luonnollinen kieli tarkoittaa ihmisten välisessä vuorovaikutuksessa syntynyttä ja kehittynyttä kieltä. Luonnolliset kielet ovat ihmisryhmien äidinkieliä, kuten suomi tai englanti. (TEPA-termipankki). Tässä kirjallisuuskatsauksessa luonnollisen kielen käsite viittaa englannin kieleen, jollei toisin mainita. Tämä johtuu siitä, että tarkastelluissa tutkimuksissa, joissa luonnollisesta kielestä muunnetaan SQL-kysely, kielenä on käytetty englantia. Relaatiotietokanta taas on yksi tietokantatyypeistä. Siinä tieto varastoidaan tauluihin, jotka ovat yhteydessä toisiinsa (TEPA-termipankki). Relaatiotietokantojen sisältämää tietoa voi hakea ja muuttaa SQL-kyselykielen avulla.

Tässä kandidaatintutkielmassa käsittelen sitä, miten luonnollisesta kielestä saadaan muunnettua SQL-kysely. Vertailen eri tapoja muuntaa SQL-kysely tekstistä ja puheesta. Tekstiä voidaan muuntaa SQL-kyselyksi eri metodeilla. Tärkeitä lähestymistapoja ovat sääntöpohjainen sekä koneoppimista hyödyntävä lähestymistapa (Kim ja muut, 2020). Kun puheesta muunnetaan SQL-kysely, on mahdollista hyödyntää osittain samoja tekniikoita, joita hyödynnetään muunnosprosessissa tekstistä SQL-kyselyksi. Siksi tarkastelen muunnosprosessia puheesta SQL-kyselyksi vasta sen jälkeen, kun olen käsitellyt muunnosprosessia tekstistä SQL-kyselyksi. Kumpaankin muunnosprosessiin liittyy erilaisia hyviä ominaisuuksia sekä haasteita. Keskeistä eri lähestymistapoja vertailtaessa on luonnollisesta kielestä muunnettujen SQL-kyselyiden oikeellisuus.

Tämä kandidaatintutkielma on muodoltaan kirjallisuuskatsaus. Lähdekirjallisuutta olen etsinyt eri tietokannoista. Suurin osa lähteistä on ACM:stä sekä ProQuestista ja lisäksi olen tehnyt hakuja Andoriin, Springeriin ja Google Scholariin. Osa lähteistä on löytynyt myös tietokannoista löytyneiden artikkeleiden lähdeluetteloiden perusteella. Hakusanoina olen käyttänyt eri yhdistelmiä seuraavista: natural language, SQL, database, text to SQL, speech to SQL, machine learning sekä rule-based. Lisäksi olen käyttänyt

hakusanoina yksittäisiä termejä koneoppimiseen liittyen, kun varsinaisen lähteen lisäksi olen tarvinnut taustatietoja. Lähteiden soveltuvuuden tähän kirjallisuuskatsaukseen olen arvioinut hakuvaiheessa otsikon, avainsanojen sekä tiivistelmän avulla. Lisäksi lähteiden laadun ja tieteellisyyden arvioinnissa olen hyödyntänyt Julkaisufoorumin julkaisukanavahakua. Suurin osa lähteiden julkaisukanavista löytyy Julkaisufoorumin julkaisukanavahausta ja on saanut arvioksi vähintään 0.

Tässä kirjallisuuskatsauksessa tarkastelen luonnollista kieltä olevan tekstin ja puheen muuntamista SQL-kyselyksi. Luvussa 2 käyn läpi sitä, miten luonnollista kieltä olevasta tekstistä muunnetaan SQL-kysely sääntöpohjaisten lähestymistapojen avulla. Luvussa 3 keskityn edelleen luonnollisen kielen tekstin muuntamiseen SQL-kyselyksi, mutta käsittelen koneoppimiseen pohjautuvia lähestymistapoja. Esittelen kolme keskeistä koneoppimista hyödyntävää mallia sekä kaksi tunnettua opetusdatana paljon käytettyä datajoukkoa. Luvussa 4 tarkastelen luonnollista kieltä olevan puheen muuntamista SQL-kyselyksi. Luvussa 5 tarkastelen muunnosprosessien toimivuutta. Lisäksi tarkastelen, mitä ongelmia muunnosprosesseihin liittyy ja onko ongelmia ratkaistu. Lisäksi arvioin eri lähestymistapojen tulevaisuudennäkymiä.

2 Tekstistä SQL-kyselyksi sääntöpohjaisia lähestymistapoja hyödyntäen

Tässä luvussa tarkastelen muunnosprosessia luonnollista kieltä olevasta tekstistä SQL-kyselyksi. Tarkastelussa ovat metodit, joilla luonnollista kieltä muunnetaan SQL-kyselyksi sääntöpohjaisuutta hyödyntäen.

Sääntöpohjaisissa (rule-based) tekniikoissa hyödynnetään *mappausta* (mapping), jossa luonnollisen kielen sanat kuvautuvat SQL-kielen sanoilla tai tietokantaan liittyvillä sanoilla (Kim ja muut, 2020). Tietokantaan liittyviä sanoja ovat esimerkiksi taulujen sekä attribuuttien nimet. Mappausta hyödyntävissä, sääntöpohjaisissa tekniikoissa oletuksena on, että yhdelle luonnollisen kielen sanalle on vain yksi vastine joko SQL:n avainsana tai tietokantaan liittyvä sana (Kim ja muut, 2020). Tämän on havaittu aiheuttavan haasteita, kun yksi luonnollisen kielen sana mappautuukin monelle eri sanalle (Kate ja muut, 2018).

Keskeistä muunnosprosesseissa luonnollisen kielen tekstistä SQL-kyselyksi on muunnosprosessin koostuminen useasta vaiheesta. Näitä vaiheita tarkastelen esimerkkijärjestelmän kautta.

2.1 Muunnosprosessin vaiheita

Katen ja muiden (2018) järjestelmä mahdollistaa luonnollisella kielellä kysymysten kysymisen opiskelijatietokannasta. Heidän mallissaan kysymykset voi esittää tekstinä tai puheena. Kuitenkin puheen ja tekstin muunnosprosessi tässä mallissa on pitkälti samanlainen, koska ensin puhe muunnetaan luonnollista kieltä olevaksi tekstiksi ja vasta tekstiä aletaan prosessoida kohti SQL-kyselyä. Katen ja muiden (2018) mallissa

muunnosprosessi koostuu useasta vaiheesta. Seuraavaksi tarkastelen kyselyn muunnosalgoritmin vaiheita tarkemmin.

Ensin tekstimuotoinen kysely on yksi merkkijono. *Tokenisointivaiheessa* (tokenization) se pilkotaan yksittäisiksi sanoiksi, jotka sijoitetaan erilliseen listaan. Tätä listaa käsitellään muissa muunnosprosessin vaiheissa. (Kate ja muut, 2018.) Tokenisoinnin jälkeen sanalistaa karsitaan vertaamalla listan sanoja toiseen listaan, joka sisältää sanoja, joilla ei ole kyselyn muunnosprosessin kannalta merkitystä. Kyselyn kannalta epäolennaiset sanat poistetaan ja muunnosprosessia jatketaan karsitulla sanalistalla. (Kate ja muut, 2018.) Luonnollisen kielen prosessoinnissa on yleinen toimintatapa tunnistaa yleisiä sanoja, joilla ei välttämättä ole merkitystä yksinään. Näitä sanoja kutsutaan *hukkasanoiksi* (stop word) ja englannin kielessä tällaisia hukkasanoja voisivat olla esimerkiksi artikkelit a, an ja the (Affolter ja muut, 2019). Vaikka Katen ja muiden (2018) mallissa hukkasanat poistettiin, aina näin ei kuitenkaan toimita.

Hukkasanojen poiston jälkeen on analyysivaiheiden vuoro. *Sanastollisessa analyysissa* (lexical analysis) kyselyn sanoja mapataan sanakirjan kanssa ja korvataan niiden synonyymeilla. Tässä vaiheessa pyritään korvaamaan kyselyn sanat sellaisilla, jotka esiintyvät tietokannassa, josta kyselyitä tehdään. Tällaisia sanoja ovat esimerkiksi tietokantataulujen sekä attribuuttien nimet. Lisäksi luonnollisen kielen kyselyn sanoja korvataan SQL-kieleen kuuluvilla synonyymeilla. Esimerkiksi sana ”fetch” korvataan sanalla ”SELECT”. (Kate ja muut, 2018.) Tämä vaihe on tärkeä, jotta tietokantakyselyiden teko mahdollistuu, sillä tietokannasta ei löydy mitään, mikäli käyttää esimerkiksi väärää attribuutin nimeä. Lisäksi sanastollisen analyysin vaihe mahdollistaa sen, ettei käyttäjän tarvitse tietää tarkasti tietokantataulujen tai attribuuttien nimiä.

Seuraavaksi *syntaktisessa analyysissa* (syntactic analysis) käsitellään sanalistaa edelleen. Syntaktisessa analyysissa valitaan kyselyn muoto sen mukaan, mikä avainsana kyselyssä esiintyy. Avainsanoja ovat esimerkiksi SELECT, INSERT ja UPDATE. (Kate ja muut, 2018.) SELECT-avainsanan yhteyteen liitetään attribuuttien nimet ja FROM-avainsanan perään taas tietokantataulun nimi. Syntaktisessa analyysissä määrittyy siis SQL-kyselyn rakenne.

Viimeinen analyysivaihe on *semanttinen analyysi* (semantic analysis). Tämän vaiheen tavoitteena on tunnistaa kyselyssä olevat ehtolauseet. Mikäli ehtolauseita löytyy, ne mapataan sanakirjan kanssa. Tällöin ehtolauseita muokataan SQL-kyselykieleen sopiviksi. Vertailusanat korvataan SQL-kielessäkin käytettävillä vertailuoperaattoreilla, Esimerkiksi ”less than” korvataan <-merkillä. Lisäksi ehto sijoitetaan kyselyssä WHERE-avainsanan jälkeen, ja vertailevat arvot järjestetään niin, että ensin on attribuutti ja vertailuoperaattorin jälkeen arvo, johon verrataan. Tämän jälkeen kysely voidaan suorittaa tietokantajärjestelmässä. (Kate ja muut, 2018.)

Muunnosprosessi luonnollisen kielen tekstistä SQL-kyselyksi sisältää monia haasteita ja muita huomioitavia asioita. Tärkeää on huomioida se, että muunnosprosessin aikaisessa mappauksessa hyödynnettävät sanakirjat ovat tietokantakohtaisia taulujen nimien ja attribuuttien osalta. Toinen tärkeä seikka on kyselyiden oikeellisuus, sillä se vaihtelee eri järjestelmien välillä (Solanki &Kumar, 2022). Tärkeä seikka on myös järjestelmän suoriutuminen monimutkaisista SQL-kyselyistä, jotka sisältävät liitosoperaatioita tai sisäkkäisiä kyselyitä. Haasteena muunnosprosessissa ovat monimerkityksiset sanat. Mikäli yksittäisellä sanalla on monta merkitystä, se voi aiheuttaa haasteita sanastollisessa analyysissa. Seuraavaksi tarkastelen näitä huomioitavia seikkoja tarkemmin.

Katen ja muiden (2018) kehittelemä muunnosprosessin malli mahdollistaa kyselyiden tekemisen luonnollisella kielellä opiskelijatietokannasta. Se on siis tarkoitettu oppilaitoksen henkilökunnan käyttöön, eikä mallia siis voi suoraan ottaa käyttöön johonkin toiseen alaan ja tietokantaan. Käyttöalasta riippuvainen asia Katen ja muiden (2018) mallissa on sanakirja, jota hyödynnetään sanastollisessa analyysissa ja joka sisältää tietokantaan liittyviä sanoja, kuten taulujen ja sarakkeiden nimiä, jotka siis ovat tietokantakohtaisia. Myös Solanki ja Kumar (2022) ovat kehittäneet järjestelmän, jonka avulla luonnollisesta kielestä voidaan muuntaa SQL-kysely. Solankin ja Kumarin (2022) kehittämässä järjestelmässä hyödynnetään tietokannan käyttöalasta riippuvaisia tietoja, kuten attribuuttien ja relaatioiden nimiä. He kuitenkin toteavat, että heidän kehittämänsä järjestelmä on helposti konfiguroitavissa. Heidän kehittämänsä järjestelmä perustuu kolmiosaiseen arkkitehtuuriin, jossa tietokanta ja siihen liittyvä metatieto, jota muunnosprosessissa hyödynnetään, ovat erillään muista järjestelmän osista. Tällöin ei tarvitse rakentaa kokonaan uutta järjestelmää käytettävän tietokannan vaihtuessa.

Monimerkityksiset sanat aiheuttavat haasteita mappauksessa. Tämä tarkoittaa sitä, että luonnollisen kielen kyselystä jokin sana kuvautuu monelle eri sanakirjan sanalle (Kate ja muut, 2018). He eivät artikkelissaan ottaneet kantaa siihen, miten monimerkityksisten sanojen ongelma tulisi ratkaista. Solanki ja Kumar (2022) eivät artikkelissaan edes maininneet monimerkityksisten sanojen ongelmaa. NaLIR, joka myös muuntaa luonnollista kieltä tietokantakyselyiksi, hyödyntää dialogia monimerkityksisten sanojen kohdalla ja kysyy tarvittaessa käyttäjältä selvennyksen (Affolter ja muut, 2019). Heidän artikkelistaan selviää, että dialogin hyödyntäminen kyselyn tarkentamisessa on paljon käytetty tekniikka. Sitä hyödynnetään muunnosprosesseissa luonnollisesta kielestä myös muihin kyselykieliin, kuten SPARQL ja XQuery.

SQL-kyselyt voivat olla monimutkaisia ja sisältää esimerkiksi erilaisia liitosoperaatioita. Tästä syystä onkin tärkeää tietää, onnistuuko muunnosprosessi luonnollisesta kielestä SQL-kyselyksi, mikäli kysely on monimutkainen. Kate ja muut (2018) toteavat, että heidän kehittämänsä järjestelmä hallitsee myös monimutkaisia

kyselyitä. Kuitenkaan väitettä ei ole sen enempää perusteltu. Artikkelista ei myöskään löydy esimerkkejä monimutkaisen kyselyn muunnosprosessista ja sen onnistumisesta. Solanki ja Kumar (2022) eivät käsittele monimutkaisten kyselyiden muodostamista lainkaan.

2.2 Kyselyn täsmentäminen ja monimutkaiset kyselyt

Li ja Jagadish (2014) ovat kehittäneet ratkaisun monimerkityksisten sanojen ongelmaan. Heidän kehittämässään NaLIR-järjestelmässä käyttäjältä kysytään tarvittaessa täsmennystä, mikäli jollakin sanalla on monta merkitystä. Lisäksi heidän mukaansa NaLIR suoriutuu myös monimutkaisista kyselyistä. Li ja Jagadish (2014) hyödyntävät kyselyn täsmentämisessä *kyselypuita* (query tree). Kyselypuu on s*yntaksipuun* (parse tree) sekä SQL-kyselyn välimuoto ja Lin ja Jagadishin (2014) mukaan käyttäjälle helpommin selitettävä kuin SQL-kysely. NaLIR koostuu kolmesta pääkomponentista, joista ensimmäinen muuntaa luonnollista kieltä olevan kyselyn kyselypuuksi. Sen jälkeen toinen komponentti huolehtii vuorovaikutuksesta käyttäjän kanssa eli esittää käyttäjälle järjestelmän tuottaman kyselypuun ja kysyy tarvittaessa selvennyksiä. Kolmas komponentti huolehtii muunnosprosessin loppuvaiheesta muuntaen kyselypuusta SQL-kyselyn.

Käyttäjän syötteen muuntaminen kyselypuuksi sisältää eri vaiheita. *Parsinta* (parsing) on luonnollisen kielen käsittelyssä paljon käytetty prosessi. Siinä analysoidaan luonnollisen kielen lauseen syntaktista rakennetta: joko sanojen tai fraasien välisiä yhteyksiä (Affolter ja muut, 2019). NaLIRissa parsinnan tuloksena syntynyt syntaksipuu koostuu solmuista, jotka sisältävät sanoja tai fraaseja käyttäjän syötteestä, ja solmujen väliset yhteydet kuvaavat sanojen tai fraasien välisiä yhteyksiä.

Seuraavaksi syntaksipuun solmut mapataan SQL-kielen komponenteille eli luonnollisen kielen sanat korvataan SQL-kielen vastaavilla ilmauksilla tai käytettävään tietokantaan liittyvillä sanoilla. Mikäli mappaus epäonnistuu, käyttäjälle näytetään varoitus. Mikäli jonkin solmun sisällölle on monta eri vaihtoehtoa, järjestelmä valitsee niistä parhaan mutta myös muut vaihtoehdot näytetään käyttäjälle, jotta käyttäjä voi halutessaan muuttaa järjestelmän automaattisesti suorittamaa valintaa. Oletuksena on, että mappauksen sekä käyttäjän mahdollisten täsmennysten jälkeen tuloksena on puu, jonka solmujen sisältö on järjestelmälle ymmärrettävä. Sen jälkeen syntaksipuu muunnetaan kyselypuuksi kahdessa vaiheessa. Ensin järjestetään puun solmut vastaamaan syntaktisesti järjestelmää. Sen jälkeen varmistetaan semanttinen vastaavuus järjestelmän kanssa lisäämällä implisiittistä tietoa uusien solmujen muodossa. Näiden vaiheiden jälkeen kyselypuusta käyttäjälle näytettävä versio on valmis. (Li & Jagadish, 2014.)

Käyttäjän ja järjestelmän välisestä vuorovaikutuksesta huolehtivaa komponenttia hyödynnetään monessa vaiheessa. Mikäli epäselvyyksiä ei ole, järjestelmä suorittaa

vaiheet ilman vuorovaikutusta käyttäjän kanssa. Ensin käyttäjältä kysytään täsmennyksiä korvattaessa syntaksipuun solmujen sisältöä tietokantaan liittyvillä sanoilla. Seuraavaksi käyttäjältä kysytään vahvistusta, kun puun solmut on järjestetty uudelleen. Kolmas vaihe on, kun implisiittistä tietoa lisätään puuhun uusien solmujen muodossa. Kun järjestelmä kysyy käyttäjältä vahvistusta, käyttäjä voi valita sopivan vaihtoehdon. Kun käyttäjä on suorittanut valinnan, NaLIR muokkaa seuraavien vaiheiden lopputuloksia käyttäjän valinnat huomioiden. (Li & Jagadish, 2014.)

Kolmannen komponentin avulla kyselypuusta muunnetaan SQL-kysely. Mikäli kyselyssä ei ole *koostefunktioita* (aggregate function) tai sisäkkäisiä kyselyitä, muunnosprosessi on melko suoraviivainen. SELECT-solmuun liitetty sisältö liitetään kyselyssä SELECT-sanaan. Arvoja ja mahdollisia operaatioita sisältävät solmut lisätään WHERE-sanan yhteyteen. Lopuksi muodostetaan tarvittavat liitosoperaatiot tietokannan skeemasta tehdyn graafin mukaan ja liitosehdot lisätään FROM-sanan yhteyteen. Koostefunktioiden ja sisäkkäisten kyselyiden muunnoksessa funktioon kuuluva kysely tai sisäkkäinen kysely on oma alipuunsa. Jos funktioita tai sisäkkäisiä kyselyitä on monta, muunnosjärjestys alkaa sisimmästä kyselystä. Tällöin pääkyselyä muunnettaessa mahdolliset sisäkkäiset kyselyt ja koostefunktiot on jo muunnettu SQL-muotoon. Pääkyselyllä tarkoitetaan ulointa kyselyä eli koko kyselypuun runkoa.

Näiden vaiheiden jälkeen SQL-kysely on valmis suoritettavaksi tietokantajärjestelmässä. Olennaista kyselyn muodostamisessa on kuitenkin se, että monimerkityksisten sanojen kohdalla käyttäjältä on pyydetty täsmennyksiä. Tämä todettiin vertailemalla NaLIRin suoriutumista niin, että osassa testikäyttötilanteista vuorovaikutusmahdollisuus oli poissa käytöstä. Silloin virheellisten kyselyiden määrä kasvoi. Kyselypuusta SQL-kyselyksi muuntaminen siis onnistuu, mikäli kyselyssä käytettävät sanat ovat sitä, mitä käyttäjä on tarkoittanut. (Li & Jagadish, 2014.)

3 Tekstistä SQL-kyselyksi koneoppimista hyödyntäen

Koneoppimista (machine learning) hyödyntävät lähestymistavat ovat saaneet enemmän jalansijaa lähivuosina. Niiden etuna on se, että koneoppimista hyödyntävät lähestymistavat mahdollistavat suurempaa kielellistä vaihtelua siinä, millaisia kyselyitä luonnollisella kielellä voidaan tehdä (Affolter ja muut, 2019).

3.1 Taustaa

Marslandin (2014, luku 1) mukaan koneoppiminen tarkoittaa toimintaa, jossa ohjelma mukauttaa toimintaansa. Tärkeää toiminnan mukautuksessa on, että toimintojen oikeellisuus ja tarkkuus paranevat. Koneoppiminen on jaettavissa eri lajeihin, joista Marslandin (2014, luku 1) mukaan yleisin koneoppimisen tyyppi on *ohjattu oppiminen* (supervised learning). Seuraavaksi esittelen tutkimusaiheen kannalta tärkeimmät

koneoppimisen tyypit eli ohjatun oppimisen sekä *vahvistusoppimisen* (reinforcement learning).

Ohjatussa oppimisessa on tyypillisesti opetusdataa, joka koostuu syötteistä sekä vastauksista, joita koneoppimisalgoritmin halutaan tuottavan. Tarkoitus siis on, että opetusvaiheen jälkeen algoritmi osaisi tuottaa oikeita vastauksia myös muusta datasyötteestä opittuaan ensin opetusdatan avulla. Regressiossa tavoitteena on tunnettujen arvojen perusteella löytää funktio, jolla saadaan laskettua myös tuntemattomat arvot. Luokittelussa taas tavoitteena on löytää syötteelle sopiva luokka. (Marsland, 2014, luku 1.) Luonnollista kieltä käsiteltäessä luokkia voivat olla esimerkiksi sanaluokat.

Vahvistusoppiminen sijoittuu ohjatun ja ohjaamattoman oppimisen väliin. Ohjatussa oppimisessa opetusdataan sisältyvät oikeat vastaukset, mutta ohjaamattomassa oppimisessa vastauksia ei ole tiedossa, vaan dataa ryhmitellään samankaltaisuuksien perusteella. Vahvistusoppimisessa kokeillaan erilaisia strategioita ja palautteen perusteella valitaan niistä paras. Tavoitteena on löytää parhaiten toimiva strategia kokeilemalla, koska etukäteen ei ole tietoa, mikä toimii ja mikä ei. Vahvistusoppimista voidaan kuvata myös ympäristön ja opittavan asian vuorovaikutuksena. Tällöin ympäristöstä voi saada palautetta algoritmin oppimisen onnistumisesta. (Marsland, 2014, luku 11.)

Luonnollisen kielen käsittelyssä tärkeitä koneoppimisen käsitteitä ovat neuroverkot ja syväoppiminen. Neuroverkoissa hyödynnetään tietoa ihmisten ja eläinten aivojen toiminnasta, koska myös aivotoiminta tapahtuu yksittäisten hermosolujen, neuronien yhteistyöllä (Marsland, 2014, luku 3). Marslandin (2014, luku 3) mukaan neuroverkoissa neuronit saavat syötteen, minkä lisäksi niillä on painokertoimet ja porrasfunktiot. Neuronien saamia syötteitä ei voi muuttaa, mutta sen sijaan painokertoimiin ja porrasfunktioihin voidaan tehdä muutoksia. Näihin vaikuttamalla voidaan vaikuttaa siihen, minkä tulosteen yksittäinen neuroni lähettää eteenpäin. Neuroverkoissa syöte menee ensin yhdelle neuronikerrokselle, joka painokertoimien ja porrasfunktion mukaan lähettää tulosteen seuraavalle neuronikerrokselle, jossa toistuvat samat vaiheet. Tätä jatketaan, kunnes viimeisestä kerroksesta saadaan lopullinen tulos. Tärkeää neuroverkkojen oppimisessa yksittäisten neuronien yhteistyön lisäksi ovat painokertoimet ja porrasfunktiot. Tärkeässä osassa eivät siis ole yksittäiset neuronit vaan se, millä porrasfunktioilla ja painokertoimilla saadaan useimmin oikea tulos.

Syväoppimisessa (deep learning) neuroverkkojen ideaa on jatkokehitetty. Marslandin (2014, luku 17) mukaan syväoppimisen avulla koneoppimisalgoritmit voivat olla monimutkaisia ja siten koneoppimista voidaan hyödyntää laajemmin. Syväoppimisen ydinajatuksena on, että neuroverkossa on enemmän kerroksellisuutta. Tällöin ongelma voidaan jakaa neuroverkon eri kerroksille niin, että eri kerroksilla ratkaistaan

kokonaisongelman osaongelmia. Kun neuroverkon yhdellä kerroksella ratkaistaan osaongelma, kyseisen kerroksen tulos siirtyy syötteenä seuraavalle kerrokselle. Näiden osaongelmien yhdistelmä on kokonaisratkaisu.

3.2 Esikuva-analyysi

Koneoppimiseen perustuvissa lähestymistavoissa tärkeässä osassa on *esikuva-analyysi* (benchmarking). Esikuva-analyysi on pohjana, kun opetetaan syväoppimiseen perustuvia järjestelmiä (Katsogianni-Meimarkis & Koutrika, 2019). Heidän mukaansa tärkeää on opetusdatajoukon suuruus ja saatavuus, jotta syväoppimiseen perustuvia, luonnollista kieltä SQL-kyselyksi muuntavia järjestelmiä voidaan kehittää. Katsogianni-Meimarkisin ja Koutrikan (2019) mukaan tunnetuimpia datajoukkoja ovat WikiSQL sekä Spider. Koska esikuva-analyysejä ja niihin pohjautuvia järjestelmiä on kehitetty paljon, keskityn enimmäkseen näihin kahteen datajoukkoon ja järjestelmiin, joissa hyödynnetään näitä datajoukkoja. Seuraavaksi esittelen tarkemmin WikiSQL:n ja Spiderin. Niiden jälkeen esittelen kolme erilaista metodia, joissa hyödynnetään WikiSQL:ää tai Spideriä. Seq2SQL sekä SQLNet hyödyntävät WikiSQL:ää ja IRNet Spideriä (Kim ja muut, 2020).

WikiSQL on yksi tunnetuimmista esikuva-analyysissä käytettävistä datajoukoista. Se on koottu *joukkoistamalla* (crowdsourcing). (Katsogiannis-Meimarakis & Koutrika, 2021.) Siihen sisältyy 21 531 Wikipedian HTML-taulukoista muodostettua taulua sekä 80 654 kysymysparia, joissa on sekä luonnollisen kielen kysymys että sitä vastaava SQL-kysely (Kim ja muut, 2020). WikiSQL:n rajoitteena on, että kysymykset kohdistuvat vain yhteen tauluun kerrallaan eikä se siis tue useamman taulun välisiä relaatioita (Katsogiannis-Meimarakis & Koutrika, 2021). Lisäksi WikiSQL mahdollistaa vain yksinkertaisia kyselyitä, jotka eivät sisällä liitosehtoja, sisäkkäisiä kyselyitä, ryhmittelyä tai hakutulosten järjestämistä (Kim ja muut, 2020). Katsogiannis-Meimarakis ja Koutrika (2021) huomauttavat myös, että WikiSQL sisältää monia epäselvyyksiä ja virheitä, jotka heikentävät WikiSQL:ää hyödyntävien järjestelmien suorituskykyä.

Spider on *laaja monialainen* (large-scale cross-domain) muunnosprosessissa luonnollisen kielen tekstistä SQL-kyselyksi hyödynnettävä datajoukko (Katsogiannis-Meimarakis & Koutrika, 2021). Spider sisältää 200 tietokantaa, jotka sisältävät tietoa eri aloilta. Lisäksi siihen kuuluu 10 121 kysymysparia, jotka sisältävät Wiki-SQL:n tapaan kysymyksen sekä luonnollisella kielellä että SQL-kyselynä. (Kim ja muut, 2020.) Merkittävää Spiderissä on, että siihen sisältyvissä kysymyspareissa mukana on myös monimutkaisia kyselyitä ja kyselyiden monimutkaisuus vaihtelee yksinkertaisesta monimutkaiseen. Kysymyspareihin sisältyvät SQL-kielen peruselementit, joiden lisäksi kysymyspareista löytyy myös sisäkkäisiä kyselyitä. (Katsogiannis-Meimarakis & Koutrika, 2021.)

3.3 SEQ2SQL

Zhongin ja muiden (2017) kehittämä Seq2SQL lähestyy ongelmaa vahvistusoppimista sekä neuroverkkoja hyödyntäen. Seq2SQL:n opetusdatana on WikiSQL, joka julkaistiinkin Seq2SQL:n yhteydessä. Seq2SQL on *lause lauseeksi metodi* (Sequence-to-Sequence) lyhyemmin ilmaistuna Seq2Seq. Seq2Seq-mallit muuntavat lähdemuodosta kohdemuotoon lause kerrallaan. Tässä yhteydessä lähdemuotona on luonnollinen kieli ja kohdemuotona SQL-kysely. Seq2Seq-mallit koostuvat *enkooderista ja dekooderista* (encoder, decoder), joissa molemmissa hyödynnetään neuroverkkoja. Enkooderi saa syötteenä muunnettavan lauseen ja muuntaa sen kontekstivektoriksi, jossa jokainen sana esitetään reaalilukuna. Vektori puolestaan menee syötteenä dekooderille, joka hoitaa lopun käännöstyöstä. Dekooderissa hyödynnetään *huomiomekanismia* (attention). Huomiomekanismin ansiosta neuroverkon aiempien kerrosten tulosteet otetaan myös mukaan dekooderin saamaan syötteeseen. Lisäksi huomiomekanismi helpottaa muunnosprosessia määrittelemällä muunnosprosessin kannalta merkittävimmät syötteen osat ja keskittymällä niihin. (Kim ja muut, 2020.)

Seq2SQL mahdollistaa perusmuotoiset kyselyt sisältäen SELECT-, FROM- sekä WHERE-sanat. Muunnosprosessia varten Seq2SQL tarvitsee syötteeksi kysymyksen luonnollisella kielellä sekä sarakkeiden nimet taulusta, josta tietoja halutaan saada. Lisäksi koostekyselyt, kuten COUNT, ovat mahdollisia. Seq2SQL:n perustana on ajatus SQL-kyselyn kolmesta komponentista: koostefunktio, sarakkeiden valinta eli SELECT sekä ehto eli kyselyn WHERE-osa. Myös Seq2SQL koostuu kolmesta komponentista, joista jokainen muuntaa oman osansa SQL-kyselystä. Koska muunnettava asia on erilainen, myös mallin opetustapa ja rakentuminen vaihtelee komponenteittain. (Zhong ja muut, 2017.) Seuraavissa kappaleissa tarkastelen näiden kolmen komponentin toimintaa tarkemmin.

Ensimmäinen komponentti huolehtii kyselyn koostefunktion muunnoksesta. Syötteenä olevasta lauseesta muunnetaan enkooderilla vektori, jossa jokaiselle kysymyksen sanalle on numeerinen esitystapa. Tämän jälkeen hyödynnetään monta kerrosta sisältävää neuroverkkoa, monikerroksista perseptronia, jotta vektorissa mahdollisesti oleva koostefunktio voidaan muuntaa. Kyseessä on luokitteluongelma, jossa jokainen koostefunktio on oma luokkansa. Lisäksi on yksi NULL-luokka, joka on muuntamisen tuloksena silloin, kun kyselyssä ei ole koostefunktiota lainkaan. (Zhong ja muut, 2017.)

SELECT-osan sarakkeiden tunnistaminen on *vastaavuusongelma* (matching problem). SELECT-osan sarakkeet ovat sarakkeena myös tietokantataulussa, josta kyselyitä tehdään. Käyttäjän luonnollisella kielellä esittämästä kysymyksestä on löydettävä Seq2SQL:n syötteenä saaman taulun sarakkeita vastaavat sarakkeet. Enkooderi saa syötteenä sekä taulun sarakkeet että käyttäjän esittämän kysymyksen.

Enkooderi tuottaa syötteestä kaksi eri esitystapaa. Nämä kaksi esitystapaa menevät syötteenä eteenpäin monikerroksiselle perseptronille, joka käsittelee syötettä edelleen pisteyttäen vaihtoehdot ja normalisoiden pisteet. Muodostuneen jakauman avulla saadaan selville kyselyn SELECT-osan sarake tai sarakkeet. (Zhong ja muut, 2017.)

WHERE-osion opettaminen sekä toteutus eroaa aiemmista osioista. -Ero johtuu siitä, että myös ongelma on erilainen: kyselyn lopputulos voi olla täysin sama erilaisilla WHERE-osan vaihtoehdoilla. Esimerkiksi ehtojen järjestys voi vaihdella, mutta lopputulos on silti sama. Tästä syystä WHERE-osioon ei toimi samanlainen ratkaisu kuin vastaavuusongelmaan. Sen sijaan, että muunnosprosessin joka askeleen oikeellisuus vahvistettaisiin, vahvistus oikeellisuudesta saadaan vasta, kun muunnettu kysely on valmis ja tämä tuloksena syntynyt SQL-kysely voidaan toteuttaa eli sen avulla voidaan saada tietoja taulusta. Tyypillistä vahvistusoppimisessa on juuri se, ettei tieto suorituksen oikeellisuudesta tule heti. WHERE-osasta huolehtivan komponentin tapauksessa onkin löytää toimintatapa, jonka avulla saadaan suoraan muunnettua oikeanlaisia kyselyitä ilman välivaiheiden vahvistusta. Vahvistusoppimisen hyödyntäminen paransi Seq2SQL:n muuntamien kyselyiden oikeellisuutta verrattuna versioon ilman vahvistusoppimista. (Zhong ja muut, 2017.)

3.4 SQLNet

Luonnokseen perustuva (sketch-based) lähestymistapa yksinkertaistaa muunnosprosessia. Lähestymistavan perustana on SQL-kyselyn luonnos, johon on jätetty tyhjiksi kohdiksi sarakkeiden nimet SELECT-osan jälkeen sekä WHERE-osan jälkeen tulevat ehdot. Sen sijaan, että tavoitteena olisi tuottaa koko kysely alusta alkaen, pyritäänkin koneoppimisen avulla tunnistamaan SELECT-osaan tulevat sarakkeet ja mahdolliset koostefunktiot sekä WHERE-osan ehdot. Tällöin kyselyluonnoksen tyhjien aukkojen täyttäminen on luokitteluongelma, jossa lasketaan todennäköisyyksiä sille, mitkä taulun sarakkeista todennäköisimmin kuuluvat haluttuun SQL-kyselyyn. (Katsogianni-Meimarkis & Koutrika, 2019.) Seuraavaksi esittelen luonnokseen perustuvaa lähestymistapaa hyödyntävän SQLNetin.

Xun ja muiden (2017) kehittämä SQLNet on luonnokseen perustuva lähestymistapa. Sen kehittämisessä on hyödynnetty WikiSQL:ää, joten se suoriutuu vain yhteen tauluun kohdistuvista kyselyistä. SQLNetissä käytettävä luonnoskyselypohja on sellainen, joka toimii vain perusmuotoiseen SELECT-kyselyyn, jossa voi olla koostefunktioita tai WHERE-osa. Lähtökohta SQLNetissä on, että SELECT- ja WHERE-osiin tulee sarakkeiden nimiä ja lisäksi WHERE-osaan tulee luonnollisen kielen kysymyksen osamerkkijonoja arvoiksi, jotka ovat osa WHERE-osan ehtoja. WHERE-osan tyypillinen ehto koostuu siis sarakkeen nimestä, vertailuoperaattorista kuten <-merkistä sekä arvosta, johon taulun sarakkeen arvoa verrataan. Luonnokseen perustuvassa lähestymistavassa WHERE-osan ehtojen järjestyksellä ei ole merkitystä. Tämä auttaa siinä, ettei kahta

muuten samanlaista kyselyä tulkita eri kyselyiksi vain siksi, että WHERE-osan ehdot ovat eri järjestyksessä. (Xu ja muut, 2017.)

SQLNetissä keskeinen osa muunnosprosessia ovat neuroverkot. Ensin käsitellään syöte eli kysymys luonnollisella kielellä sekä tarvittavan tietokantataulun sarakkeet siten, että jokainen yksittäinen sana on oma vektorinsa, jossa sanaa kuvataan numerolla 1. Sen sijaan, että pyrittäisiin löytämään SELECT- ja WHERE-osissa esiintyvät sarakkeet tietyssä järjestyksessä, niitä ajatellaan osajoukkona. Kaikki taulun sarakkeet muodostavat yhden joukon ja kyselyssä tarvittavat sarakkeet ovat sen osajoukko. Taulun sarakkeet päätyvät osajoukkoon sen perusteella, kuinka todennäköisesti ne esiintyvät syötteenä saadussa, luonnollista kieltä olevassa kyselyssä. Lisäksi sarakkeiden valinnasta suoriutumista parantaa myös jo aiemmin käsitelty huomiomekanismi. (Xu ja muut, 2017.)

Xun ja muiden (2017) mukaan WHERE-osio on haastavin osa muunnosprosessia. Ennen sarakkeiden valintaa on ennustettava se, kuinka monta ehtoa WHERE-osassa on. Mikäli ehtoja on kaksi, WHERE-osaan päätyvät kaksi saraketta, joiden todennäkyisyydet ovat korkeimmat. Operaattorin valinta on luokitteluongelma, jossa luokkia ovat operaattorit (<, > tai =), joista valitaan oikea luokka. WHERE-osan arvoiksi valitaan syötteenä saadun merkkijonon osat. (Xu ja muut, 2017.)

Xu ja muut (2017) vertailivat kehittämäänsä SQLNetiä sekä Seq2SQL:ää. Heidän artikkelissaan todetaan Se2SQL:n suoriutuvan muunnosprosessista paremmin verrattuna siihen, mitä Zhong ja muut (2017) ilmoittivat Seq2SQL:n muunnostarkkuudeksi. Syytä tähän Xu ja muut (2017) eivät tiedä, koska Seq2SQL:n lähdekoodi ei ollut saatavilla. Vaikka Seq2SQL:n muunnostarkkuus oli Xun ja muiden (2017) mukaan parempi heidän käyttäessään sitä, se ei kuitenkaan yltänyt SQLNetin tasolle. Kummankin opetusdatana on käytetty samaa WikiSQL:ää. Merkittävä ero on kuitenkin käytettyjen lähestymistapojen erilaisuus.

3.5 IRNet

Seq2Seq-metodien ongelmana on riski syntaktisesti virheellisiin SQL-kyselyihin. Tämä johtuu Katsogianni-Meimarkisin ja Koutrikan (2019) mukaan siitä, ettei tarkkoja kielioppisääntöjä oteta huomioon kyselyä luotaessa. Heidän mukaansa *kielioppiin perustuvat* (grammar-based) lähestymistavat tuottavat tulosteena yksittäisistä sanoista koostuvan lauseen sijaan joukon kielioppisääntöjä, joiden pohjalta SQL-kysely voidaan luoda. Lisäksi he toteavat kielioppiin perustuvissa lähestymistavoissa syntaktisesti virheellisten kyselyiden määrän laskeneen. Seuraavaksi esittelen yhden kielioppiin perustuvista lähestymistavoista, IRNetin.

Guo ja kollegat (2019) ovat kehittäneet IRNetin. Siinä hyödynnetään väliesitystapaa, joka sijoittuu muunnosprosessissa luonnollisen kielen ja SQL:n väliin. Väliesityksen muotona on SemQL, joka on *sovelluskohtainen kieli* (domain-specific language). Kun luonnollista kieltä oleva kysymys muunnetaan SemQL-muotoon, kyselystä muodostuu

puurakenne. Puurakenteeseen ei kuitenkaan sisällytetä kaikkia SQL-kyselyyn tulevia osia sellaisenaan, ja FROM-osa eliminoidaankin SemQL:stä kokonaan. Sen sijaan, että WHERE- ja HAVING-osat olisivat mukana puurakenteessa, niistä ovat mukana vain ehdot, joiden avulla kyselyssä rivejä suodatetaan. Puurakenteesta eliminoidut osat ovat seuraavassa muunnosvaiheessa kuitenkin pääteltävissä muiden tietojen avulla. Lisäksi SemQL:ssä on määritelty taulujen nimet siten, että ne voidaan yhdistää oikeisiin sarakkeisiin. Taulujen ja sarakkeiden yhdistämisen avulla voidaan erottaa toisistaan eri tauluihin kuuluvat samannimiset sarakkeet.

Toinen tärkeä osa muunnosprosessia on *skeeman yhdistäminen* (schema linking). Siinä tarkoituksena on yhdistää kysymyksessä olevat entiteetit tietokannan vastaaviin entiteetteihin hyödyntäen merkkijonojen vastaavuutta. Entiteettejä tässä yhteydessä ovat taulujen ja sarakkeiden nimet sekä sarakkeiden arvot. Mikäli syötteenä saadun kysymyksen sana vastaa jonkin sarakkeen nimeä kokonaan tai osittain, se määritellään sarakkeeksi. Mikäli kysymyksen sana vastaa taulun nimeä kokonaan tai osittain, se tunnistetaan taulun nimeksi. Tilanteissa, joissa sana vastaa sekä sarakkeen että taulun nimeä, se määritellään sarakkeeksi. Sanat, jotka alkavat tai loppuvat ”-merkillä, määritellään sarakkeiden arvoiksi. Kun lauseen yksittäinen osa on identifioitu, sitä muistuttavat vastaavat sanat poistetaan syötelauseesta. Näin saadaan yhdistettyä syöte sekä käytettävä skeema ilman sanoja, joilla on päällekkäisiä merkityksiä. (Guo ja muut, 2019.)

Muunnos luonnollisesta kielestä SemQL-muotoon tapahtuu useiden eri komponenttien avulla. Komponentit ovat erilaisia enkoodereita ja dekoodereita, joissa hyödynnetään erilaisia neuroverkkoja. Luonnollisen kielen enkooderi yhdistää ja prosessoi käyttäjän esittämää kysymystä sekä skeeman yhdistämisen tuloksia siten, että niille saadaan numeerinen esitystapa. Skeemaenkooderi käsittelee erikseen skeeman yhdistämisen tuloksia muuttaen ne numeeriseen muotoon. Näiden vaiheiden jälkeen dekooderit muuntavat syötteestä SemQL-kyselyn. Käytössä on kielioppiin perustuva dekooderi, jolla on tiedossa myös SemQL:n puurakenne. Ensimmäisenä vaiheena valitaan tuotantosääntö, jonka perusteella kysely luodaan. Sen jälkeen dekooderi ennustaa käytettävät sarakkeet. Kun tarvittavat sarakkeet on ennustettu, tarvittavat taulut ennustetaan sen perusteella, missä tauluissa sarakkeet esiintyvät. Näin saadaan karsittua tarpeettomat taulut. Ensimmäisessä vaiheessa ensimmäinen dekooderi luo karkean SemQL-version. Sen jälkeen yksityiskohtaisempi dekooderi täydentää SemQL-kyselyä täydentäen puuttuvat yksityiskohdat. (Guo ja muut, 2019.)

Lopuksi SemQL-kysely muunnetaan SQL-kyselyksi. Ensin SemQL:n puurakenne käydään läpi. Sen jälkeen solmut mapataan niitä vastaavien SQL-kielen elementtien kanssa. Kyselyn muodostamisessa hyödynnetään tuotantosääntöä, joka kertoo, minkälainen SQL-kysely on kyseessä. Tuotantosääntö kertoo esimerkiksi, onko

kyselyssä joukko-operaatioita, WHERE-, GROUP By -osia tai koostefunktioita. FROM-osaan tulevat taulut taas päätellään hyödyntäen tauluista tehtyä suuntaamatonta graafia, jossa solmuina ovat kaikki SemQL-kyselyssä esiintyvät taulut. FROM-osa muodostuu etsimällä lyhyin reitti graafin solmusta toiseen. (Guo ja muut, 2019.)

IRNetin lisäksi muissakin koneoppimista hyödyntävissä järjestelmissä käytetään välivaiheena puurakennetta. Yksi näistä on SyntaxSQLNet (Kim ja muut, 2020). Merkittävä ero järjestelmissä on se, ettei SyntaxSQLNetissä käytetä kielioppiin perustuvaa dekooderia kuten IRNetissä (Kim ja muut, 2020). Guo ja muut (2020) vertailivat IRNetin ja SyntaxSQLNetin muunnosprosessien tarkkuutta, koska kumpikin hyödynsi esikuva-analyysissä Spideriä. Mittarina oli muunnetun kyselyn sekä testidatan SQL-kyselyn täydellinen vastaavuus. Kävi ilmi, että IRNetin muunnostarkkuus SyntaxSQLNetiin verrattuna oli 27 prosenttiyksikköä suurempi. Lisäksi muunnostarkkuutta vertailtiin siten, että testidata jaettiin vaikeusasteiltaan erilaisiin kyselyihin. Silloinkin IRNetin muunnostarkkuus oli korkeampi. Molemmissa vertailun kohteena olevissa järjestelmissä muunnostarkkuus kuitenkin laski kyselyiden vaikeutuessa. (Guo ja muut, 2019.)

4 Puheesta SQL-kyselyksi

Luonnollista kieltä olevan puheen muuntaminen SQL-kyselyksi hyödyttää suurempaa ihmisryhmää verrattuna tekstin muuntamiseen SQL-kyselyksi. Mahdollisuus tehdä tietokantakyselyitä hyödyttää kummassakin tilanteessa heitä, joilla ei ole tietokantaosaamista. Etuna puhekäyttöliittymällä on, että kysyjän kädet ovat vapaana, kun kyselyä ei tarvitse kirjoittaa näppäimistöllä (Lyons, Tran, Binnig, Cetintemel & Kraska, 2016). Lisäksi he toteavat puhekäyttöliittymien hyödyttävän eri tavoin vammaisia henkilöitä, jotka eivät voi käyttää esimerkiksi näppäimistöä. Puhekäyttöliittymät parantavat siis tietokantojen saavutettavuutta suuremmalle joukolle.

4.1 VoiceQuerySystem

Muunnosprosessia puheesta SQL-kyselyksi on lähestytty eri tavoilla. Song, Wong, Zhao & Jiang (2022) ovat kehittäneet VoiceQuerySystemin, jossa muunnosprosessia puheesta SQL-kyselyksi lähestyttiin kahdella eri tavalla. *Peräkkäisten tapahtumien lähestymistapa* (cascaded approach) on ratkaisu, jossa yhdistyy kaksi komponenttia: puheentunnistus sekä moduuli, joka muuntaa puheentunnistuksen avulla saadun tekstin SQL-kyselyksi. *Päästä päähän lähestymistavassa* (end-to-end approach) muunnetaan luonnollisen kielen puhesignaalit suoraan SQL-kyselyksi. Seuraavissa kappaleissa tarkastelen edellä mainittuja lähestymistapoja tarkemmin.

Ennen lähestymistapojen tarkempaa tarkastelua käyn läpi VoiceQuerySystemin arkkitehtuuria. Songin ja muiden (2022) kehittämä VoiceQuerySystem koostuu kolmesta osasta. Rajapinnan kautta käyttäjä pääsee hyödyntämään VoiceQuerySystemiä ja

esittämään puheen avulla kysymyksiä luonnollisella kielellä. Lisäksi rajapinnan kautta käyttäjä pääsee valitsemaan käytettävän tietokannan, tarkastamaan muodostetun SQL-kyselyn sekä tarkastelemaan kyselyn tuloksia. *Malli* (model) sisältää teknologiat, joiden avulla puhe muunnetaan SQL-kyselyksi. Malli sisältää peräkkäisten tapahtumien lähestymistapaa varten puheentunnistusmoduulin sekä teknologiaa, jonka avulla muunnetaan tekstistä SQL-kysely. Päästä päähän -lähestymistapaa varten malli sisältää teknologiaa, jonka avulla puhesignaalit muunnetaan suoraan SQL-kyselyksi koneoppimista hyödyntäen. Dataosuus sisältää WikiSQL- sekä Spider-datajoukot, joita hyödynnetään muunnettaessa tekstistä SQL-kyselyitä. Lisäksi dataosaan kuuluu päästä päähän -lähestymistavassa hyödynnettävä SpeechQL. Dataosaan kuuluu myös tietokanta, josta kyselyitä halutaan tehdä.

Peräkkäisten tapahtumien lähestymistavassa Songin ja muiden (2022) mukaan ensimmäisenä vaiheena on puheen muuntaminen tekstiksi. VoiceQuerySystemissä on käytössä Songin ja muiden itse kehittämä automaattinen puheentunnistusmoduuli. Seuraavaksi puheentunnistuksen avulla aikaan saatu luonnollista kieltä oleva teksti pitää muuntaa SQL-kyselyksi. Vaihtoehdoiksi tähän Song ja muut esittävät Seq2SQL:n, EditSQL:n sekä IRNetin. VoiceQuerySystemissä he kuitenkin päätyivät IRNetiin sen hyvän suorituskyvyn takia. Song ja muut kuitenkin toteavat, että IRNetin tilalla voisi olla mikä tahansa tekstiä SQL-kyselyksi muuntava teknologia.

Songin ja muiden (2022) VoiceQuerySystemissä päästä päähän -lähestymistavassa puhesignaalit muunnetaan suoraan tietokantakyselyksi ilman tekstiksi muuntamisen välivaihetta. He hyödyntävät tässä itse kehittämäänsä SpeechSQL-mallia, joka koostuu neljästä komponentista. Nämä kaikki komponentit ovat erilaisia enkoodereita sekä dekoodereita. Puhe-enkooderissa hyödynnetään konvoluutioneuroverkkoa puhesignaalin muuntamiseen. Samaan aikaan skeemaenkooderilla kyselyyn vaikuttava skeema muunnetaan graafiksi. Näiden kummankin enkooderin tulostukset siirtyvät syötteenä relaatiotietoiselle puhe-skeemaenkooderille, joka yhdistää saamansa syötteet sekä tunnistaa viittaukset tietokantataulujen ja taulujen sarakkeiden nimiin luonnollisessa kielessä. Viimeisenä vaiheena on SQL-tietoinen dekooderi, joka muuntaa saamansa syötteen SQL-muotoon.

Song ja muut (2022) vertailivat neljää erilaista puheen SQL-kieleksi muuntavaa metodia. Jokainen metodi oli opetettu samalla datajoukolla, ja suoriutumisen mittarina oli odotetun lopputuloksen sekä muunnetun SQL-kyselyn tarkka vastaavuus. Kaksi metodeista perustui peräkkäisten tapahtumien lähestymistapaan ja toiset kaksi päästä päähän -lähestymistapaan. Kävi ilmi, että peräkkäisten tapahtumien lähestymistapojen muunnostarkkuus oli heikompi. Merkittävä syy on virheiden kasaantuminen, kun pieni virhe puheentunnistusvaiheessa johtaa suurempiin virheisiin muunnettaessa tekstistä SQL-kyselyä. Lisäksi Songin ja muiden (2022) kehittämää päästä päähän -

lähestymistapaa verrattiin toiseen päästä päähän -lähestymistapaan, jossa hyödynnettiin aiemmin käsittelemääni lause lauseeksi -metodia. Song ja muut (2022) toteavat heidän kehittämänsä päästä päähän -ratkaisun suoriutuvan muunnosprosessista paremmin. Heidän mukaansa tämä todistaa sen, että heidän kehittämässään mallissa skeeman tunnistamiseen liittyvä *esikoulutus* (pre-training) on hyödyllistä. Lisäksi Songin ja muiden (2022) mukaan heidän kehittämässään mallissa käytetyt neuroverkkoratkaisut ovat toinen syy järjestelmän parempiin muunnostuloksiin.

4.2 Keskustelua tietokannan kanssa

Aina käyttäjän puhuma kysely ei sisällä kaikkia kyselyn muuntamiseen tarvittavia tietoja. Tästä syystä on hyödyllistä, mikäli järjestelmä kysyy käyttäjältä täsmennyksiä tarvittavien yksityiskohtien jäädessä vajaiksi. Tällöin voi syntyä ikään kuin keskustelu käyttäjän ja tietokannan välille. Lyonsin ja muiden (2016) kehittämä EchoQuery mahdollistaa keskustelun tietokannan kanssa.

EchoQuery on *väliohjelma* (middleware) puhekäyttöliittymän ja tietokannan välillä. Puhekäyttöliittymänä toimii Amazonin Echo, joka on langaton äänikomennoilla toimiva älykaiutin. Lisäksi käyttöliittymässä hyödynnetään Amazonin Alexaa, joka välittää viestit Echon ja Amazonin palvelimen välillä. EchoQueryssä Amazonin palvelin tekee puheentunnistuksen ja välittää käyttäjän syötteen EchoQuerylle. Sen jälkeen EchoQuery käsittelee syötettä. Syötteen tyypistä riippuu, mikä EchoQueryn osa sitä käsittelee. (Lyons ja muut, 2016.)

Keskustelunaloituksesta tietokannan kanssa huolehtii kyselyihin tarkoitettu osa. Se ottaa vastaan kyselyitä, joissa voi olla sarakkeiden valintaa, koostefunktioita, WHERE-osio ehtoineen sekä tulosten ryhmittelyä GROUP BY -osion avulla. Tulokseen haluttavia sarakkeita voi olla monia ja myös GROUP By -osiossa voi olla monia sarakkeita, joiden perusteella tulos ryhmitellään. Myös WHERE-osiossa voi olla useita attribuutteja. EchoQuery ei tue kaikkea luonnollista kieltä, vaan mahdollisia kyselyitä on rajoitettu. Mahdollisia kysymysfraaseja ovat esimerkiksi ”how many” tai ”what is the number of”. Mahdollisia ilmauksia on rajoitettu verrattuna mihin tahansa luonnollisen kielen syötteisiin. EchoQueryn vastaanottama syöte on siis rajoitettu osa luonnollisesta kielestä, mutta helpompi puhua ja oppia verrattuna SQL-kieleen. Liitosoperaatioita käyttäjän ei tarvitse puhua ääneen. Ne päätellään skeeman taulujen viiteavaimista. (Lyons ja muut, 2016.)

Kyselyn paranteluun tarkoitettu osio aktivoituu käyttäjän tehdessä muutoksia viimeisimpään kyselyynsä. Muutokset voivat kohdistua haluttuihin tulossarakkeisiin eli SELECT-osaan: sarakkeita voidaan lisätä tai poistaa. Samat toimenpiteet ovat mahdollisia myös koostefunktioille. Lisäksi on mahdollista tehdä muutoksia myös mahdollisiin WHERE- ja GROUP BY -osioihin. Mikäli tulosrivejä on paljon, EchoQuery tiivistää hakutuloksia ilmoittaen vain hakua vastaavien tulosrivien määrän. Tämä on

Lyonsin ja muiden (2016) mukaan käyttäjälle helpommin ymmärrettävä tulosten esitystapa verrattuna yksityiskohtaisempaan tulosten esittämiseen, kun käyttäjä saa tiedot vain kuulonvaraisesti. Kyselyn kehittämiseen tarkoitettu osa aktivoituu käyttäjä kysyessä tarkempia tietoja hakutuloksista. (Lyons ja muut, 2016.)

Kolmas osa huolehtii kyselyn tarkentamisesta. Kyselyn tarkentamisella tarkoitetaan tilanteita, joissa EchoQuery tarvitsee lisätietoja kyselyn muodostamiseksi. Tämä tilanne voi esiintyä esimerkiksi silloin, mikäli skeeman monessa eri taulussa on samanniminen sarake. Tällöin EchoQuery kysyy käyttäjältä täsmennystä siitä, minkä taulun saraketta hän tarkoittaa. Mikäli tällaista ominaisuutta ei olisi, käyttäjälle ilmoitettaisiin virheestä ja hänen olisi aloitettava kysely alusta. EchoQuery tarkentaa kyselyä niin monta kertaa kuin on tarpeen epäselvien osien selvittämiseksi. (Lyons ja muut, 2016.)

EchoQuery poikkeaa muista tässä kirjallisuuskatsauksessa esitetyistä muunnosprosesseista. Erona on, ettei EchoQuery kata muunnosprosessia käyttöliittymästä tietokantaan asti, vaan toimii väliohjelmana Amazonin Echon ja käytettävän tietokannan välillä. Koska käytössä on Amazonin Echo ja Alexa, käyttäjän syötteet eli tietokantakyselyt menevät Amazonin palvelimelle (Lyons ja muut, 2016). Tällöin käyttäjän syötteiden mukana Amazonin palvelimelle päätyy myös mahdollisesti tietoja tietokannasta. Siltikään Lyons ja muut (2016) eivät ottaneet kantaa tietosuoja- ja yksityisyyskysymyksiin.

5 Pohdintaa

On monta tapaa muuntaa luonnollisesta kielestä SQL-kysely. Ensinnäkin eroja on syötteissä, kun osa tekee muunnosprosessin puheesta ja osa taas tekstistä. Toinen tärkeä käyttäjälle näkyvä eroavaisuus on muunnosprosessin lopputulos ja muunnosprosessin tarjoamat mahdollisuudet. Asia, joka ei käyttäjälle suoraan näy, on teknologia, jota muunnosprosessissa hyödynnetään. Käytettävä teknologia näkyy käyttäjälle kuitenkin muunnosprosessin onnistumisessa sekä erilaisten kyselyvaihtoehtojen määrässä.

Osa järjestelmistä mahdollistaa vain yksinkertaiset kyselyt yhdestä taulusta kerrallaan. Tällaisia järjestelmiä ovat esimerkiksi WikiSQL:ään pohjautuvat järjestelmät (Kim ja muut, 2020). Järjestelmässä hyödynnetty datajoukko vaikuttaa suoraan siihen, kuinka monimutkaisista kyselyistä järjestelmä suoriutuu. ATISia ja GeoQueryä hyödyntävät järjestelmät suoriutuvat laajemmasta määrästä erilaisia kyselyitä, mutta niiden heikkoutena on, että niitä voi hyödyntää vain tietyllä alalla: GeoQueryn taulut sisältävät dataa maantieteestä ja ATISin taulut taas lentovarauksista (Kim ja muut, 2020). Sen sijaan Spider sisältää myös moneen tauluun kohdistuvia ja monimutkaisempia kyselyitä (Kim ja muut, 2020).

Yksinkertaisista kyselyistä suoriutuu moni järjestelmä. Sen sijaan monimutkaisemmista kyselyistä suoriutuvien järjestelmien määrä on pienempi. Lisäksi kyselyn monimutkaisuus on myös muunnostarkkuutta laskeva tekijä (Guo ja muut, 2019).

Zhong ja muut (2017) toteavat SQL-kieltä osaamattomien hyötyvän mahdollisuudesta tehdä tietokantakyselyitä luonnollisella kielellä. Esimerkiksi heidän kehittämässään Seq2SQL:ssä hyöty jää melko pieneksi, koska kyselyt kohdistuvat vain yhteen tietokantatauluun kerrallaan, ja monimutkaisemmat kyselyt eivät onnistu. Tällöin SQL-kieltä osaamaton käyttäjä menettää paljon sekä tietokantojen että SQL-kielen mahdollisuuksista. Monimutkaisista kyselyistä suoriutuminen vastaisi kuitenkin parhaiten myös SQL-kieltä osaamattomien käyttäjien tarpeisiin. Esimerkiksi myös Solanki ja Kumar (2022) toteavat käyttäjien hyötyvän mahdollisuudesta tehdä tietokantakyselyitä luonnollisella kielellä. Vaikka käyttäjien kokemaa hyötyä käytetään perusteluna, käyttäjäkokemuksia ei monistakaan tutkimuksista löydy. Tähän kirjallisuuskatsaukseen päätyneistä tutkimuksista vain Lin ja Jagadishin (2014) tutkimuksessa oli tutkittu myös käyttäjien kokemuksia.

Eri muunnosprosessien haasteet eroavat toisistaan. Luonnollista kieltä olevaa puhetta SQL-kyselyksi muunnettaessa haasteena on puheentunnistuksen onnistuminen. Mikäli puheentunnistuksessa tapahtuu virhe, se voi johtaa suurempiin virheisiin muunnettaessa tunnistettua tekstiä SQL-kyselyksi (Song ja muut, 2017). Songin ja muiden (2017) mukaan haasteeseen yksi ratkaisu on muuntaa puhesignaalit SQL-kyselyksi ilman tekstiksi muuntamista. Tästä tarvitaan kuitenkin lisää tutkimuksia.

Muunnosprosessi tekstistä SQL-kyselyksi ei myöskään ole ongelmaton. Sääntöpohjaisten lähestymistapojen haasteena ovat monimerkityksiset sanat sekä monimutkaiset kyselyt (Kate ja muut, 2018). Monimerkityksisten sanojen ongelmaan yksi ratkaisu on dialogi käyttäjän kanssa, jolloin voidaan varmistaa käyttäjältä, mitä sanaa hän tarkoittaa (Li & Jagadish, 2014). Monimutkaiset kyselyt ovat haasteena koneoppimiseen perustuvissa lähestymistavoissa mutta eri syystä. Luonnokseen perustuvassa lähestymistavoissa syynä on luonnoskyselyn yksinkertaisuus (Xu ja muut, 2017). Lisäksi monissa koneoppimiseen perustuvissa lähestymistavoissa käytetään opetusdatana sellaista datajoukkoa, jossa ei ole monimutkaisia kyselyitä (Kim ja muut, 2020). Lisäksi koneoppimiseen perustuvissa lähestymistavoissa haasteena on harjoitusdatan vähyys ja mahdolliset epätarkkuudet (Katsogianni-Meimarkis & Koutrika, 2019).

Eri järjestelmien ja lähestymistapojen vertailu on hankalaa. Tämä johtuu siitä, että eri lähestymistavoilla on omat hyvät ominaisuutensa sekä haasteensa. Lisäksi eri järjestelmiä on testattu erilaisissa olosuhteissa. Esimerkiksi Seq2SQL:n muunnostarkkuudeksi on ilmoitettu kaksi eri lukemaa (Zhong ja muut, 2017) (Xu ja muut, 2017). Lisäksi Xu ja muut (2017) toteavat, ettei testidatan oikean SQL-kyselyn ja järjestelmän muuntaman

kyselyn tarkka vastaavuus ole pätevä mittari. Tämä johtuu siitä, että kyselyillä saadaan sama tulos, vaikka WHERE-osan ehdot olisivat eri järjestyksessä ja siitä syystä kyselyt eivät olisi kaksi samanlaista merkkijonoa. Kim ja muut (2020) toteavatkin, että on otettava huomioon myös semanttinen yhdenmukaisuus. Silloin muunnosprosessin onnistumisen mittaamisessa verrataan SQL-kyselyiden tulosten samankaltaisuutta.

Luonnollista kieltä SQL-kyselyksi muuntavat järjestelmät sisältävät myös erilaisia oletuksia. Spiderin sisältämän datan oikeellisuus on tarkastettu käsin (Katsogianni-Meimarkis & Koutrika, 2019). Samaa laatua oletetaan myös IRNetissä, joka on opetettu Spiderillä. Muunnos SemQL:stä SQL:ksi perustuu oletukseen, että tietokantaskeema on määritetty tarkkaan. Skeeman tarkka määrittäminen tarkoittaa esimerkiksi, että toisiin tauluihin viitatessa viiteavaimia on käytetty oikein. (Guo ja muut, 2019.) WikiSQL:ssä esimerkkitaulujen sarakkeet ovat kaikki luonnollisen kielen sanoja. WikiSQL:n avulla opetettu SQLNet perustuu oletukseen, että käytettävän tietokannan taulujen sarakkeet on myös nimetty luonnollisen kielen sanoilla. Tällöin syötteeksi riittävät tietokantataulun sarakkeiden nimet sekä kysymys luonnollisella kielellä. (Xu ja muut, 2017.)

Tavoitteena oli tutkia luonnollisen kielen muuntamistani SQL-kyselyksi. Erilaisia tapoja muunnosprosessille löytyi useita, mutta jokaiseen tapaan liittyy merkittäviä haasteita. Koska tutkielman muotona oli kirjallisuuskatsaus, vertailu eri järjestelmien välillä oli hankalaa. Vertailu olisi voinut olla hankalaa myös siinä tapauksessa, vaikka tutkimukseen olisi sisältynyt empiirinen osuus, koska kaikki järjestelmät lähdekoodeineen eivät ole julkisesti saatavilla. Lähdekirjallisuudesta selvisi, että muunnosprosessi luonnollisesta kielestä SQL-kyselyksi onnistuu. Kehitystyötä tarvitaan, jotta muunnoksen tarkkuus ja suoriutuminen monimutkaisista kyselyistä paranevat. Lisäksi viimeisimpien vuosien aikana julkaistuissa tutkimuksissa koneoppimisella on merkittävä rooli. Kehittyneimmissä muunnosprosesseissa koneoppiminen onkin vahvasti mukana.

6 Yhteenveto

Muunnosprosessi luonnollisesta kielestä SQL-kyselyksi onnistuu monilla eri tavoilla. Varsinkin viimeisinä vuosina koneoppiminen on noussut keskiöön aihetta tutkittaessa.

Mahdollisuus tehdä tietokantakyselyitä luonnollisella kielellä lisää tietokantojen saavutettavuutta. Tekstikäyttöliittymä hyödyttää heitä, joilla ei ole SQL-osaamista. Heitä hyödyttävät myös puhekäyttöliittymät, minkä lisäksi puhekäyttöliittymistä hyötyvät motorisista vaikeuksista kärsivät (Lyons ja muut, 2016). Näistä syistä luonnollisen kielen muuntaminen SQL-kyselyksi on tärkeä tutkimuskohde.

Tämä kirjallisuuskatsaus osoittaa sen, että tutkimusta aiheesta on jo tehty. Lisäksi lähdekirjallisuudesta on selvinnyt, että riippumatta siitä, miten ongelmaa on lähestytty,

haasteita ilmenee. Tutkimukset myös osoittavat, että monia haasteita myös ratkotaan ja on ratkaistu.

7 Lähdeluettelo

Affolter, K., Stockinger, K. & Bernstein, A. (2019). A comparative survey of recent natural language interfaces for databases. The VLDB Journal, 28, 793–819. https://doi.org/10.1007/s00778-019-00567-8

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J. G., Liu, T. & Zhang, D. (2019). Towards complex text-to-sql in cross-domain database with intermediate representation. arXiv. *https://doi.org/10.48550/arXiv.1905.08205*

Kate, A., Kamble, S., Bodkhe, A. & Joshi, M. (2018, March). Conversion of natural language query to SQL query. Second International Conference on Electronics, Communication and Aerospace Technology (ICECA) (s. 488–491). IEEE. https://doi.org/10.1109/ICECA.2018.8474639

Katsogiannis-Meimarakis, G. & Koutrika, G. (2021, June). A deep dive into deep learning approaches for text-to-sql systems. Proceedings of the 2021 International Conference on Management of Data (s. 2846–2851). https://doi.org/10.1145/3448016.3457543

Kim, H., So, B. H., Han, W. S. & Lee, H. (2020). Natural language to SQL: Where are we today?. Proceedings of the VLDB Endowment, 13(10), 1737–1750. https://doi.org/10.14778/3401960.3401970

Li, F. & Jagadish, H. V. (2014). Constructing an interactive natural language interface for relational databases. Proceedings of the VLDB Endowment, 8(1), 73–84. https://doi.org/10.14778/2735461.2735468

Marsland, S., (2014). Machine Learning: An Algorithmic Perspective, CRC Press. https://doi.org/ 10.1201/b17476

Lyons, G., Tran, V., Binnig, C., Cetintemel, U. & Kraska, T. (2016, June). Making the case for query-by-voice with EchoQuery. In Proceedings of the 2016 International Conference on Management of Data (s. 2129–2132). https://doi.org/10.1145/2882903.2899394

Solanki, A. & Kumar, A. (2022). A system to transform natural language queries into SQL queries. International Journal of Information Technology. 14, 437–446. https://doi.org/10.1007/s41870-018-0095-2

Song, y., Wong, r. C., Zhao, X. & Jiang, D., (2022). VoiceQuerySystem: A Voice-driven Database Querying System Using Natural Language Questions. Proceedings of the 2022 International Conference on Management of Data (s. 2385–2388). https://doi.org/10.1145/3514221.3520158

TEPA-termipankki. julkaisuaika tuntematon. haettu 26.02.2023. https://termipankki.fi/tepa/fi/

Xu, X., Liu, C. & Song, D. (2017). SQLNet: Generating structured queries from natural language without reinforcement learning. arXiv https://doi.org/10.48550/arXiv.1711.04436

Zhong, V., Xiong, C. & Socher, R. (2017). Seq2SQL: Generating structured queries from natural language using reinforcement learning. arXiv. *https://doi.org/10.48550/arXiv.1709.00103*