IT-Helpdesk Chatbot

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus

Kevät 2025

Mikael Rokkanen

HAMK
Hämeen ammatti-
korkeakoulu

Tiivistelmä

Koulutus Tietojenkäsittelyn koulutus
Tekijä Mikael Rokkanen
Työn nimi IT-Helpdesk Chatbot
Ohjaaja Elina Kulmala

Vuosi 2025

Tämän opinnäytetyön tavoitteena oli kehittää toimiva IT-Helpdesk chatbot, joka kykenee vastaamaan käyttäjien yleisiin kysymyksiin. Chatbot hyödyntää oikean vastauksen löytämiseen tekstin vektorointia sekä kosinin samankaltaisuutta. Kehityksessä hyödynnettiin Hugging Facen Transformers-kirjastoa ja Gradio käyttöliittymää. Työllä ei ollut toimeksiantajaa, vaan se perustui tekijän omiin havaintoihin suoritetussa harjoittelussa IT-Helpdesk työssä.

Teoreettinen osa käsittelee chatbot teknologian keskeisiä käsitteitä, sekä koneoppimiseen perustuvan kielimallin toimintaa. Opinnäytetyö on toiminnallinen, ja chatbotin toiminnallisuus arvioitiin käytännön testauksilla manuaalisesti kehittäjän toimesta.

Tulokset osoittivat, että chatbot kykenee vastaamaan IT-Helpdeskin yleisiin kysymyksiin tehokkaasti, ja se tunnistaa tietyssä määrin muokattuja kysymyksiä, jolloin käyttäjän ei tarvitse kysyä kysymystä täsmällisesti oikeassa muodossa. Kuitenkaan täysin uusiin kysymyksiin, jotka eivät sisälly datasettiin, ei pystytty tuottamaan vastausta ilman, että kysymykset päivitetään datasettiin manuaalisesti.

Kuitenkin tulokset ovat hyviä ja jatkokehitysmahdollisuudet ovat laajat.

Avainsanat

Chatbotti, BERT, Hugging Face, Gradio, Tekoäly

Sivut 23 sivua ja 1 liite

HAMK
Hämeen ammatti-
korkeakoulu

Abstract

DP Degree Programme in Business Information Technology
Author Mikael Rokkanen
Year 2025
Subject IT-Helpdesk Chatbot
Supervisors Elina Kulmala

The aim of this thesis was to develop a functional IT-Helpdesk chatbot, that can answer common questions. The chatbot utilizes vectorization and cosine similarity to find a solution. The development utilized Hugging Face's Transformers Library and Gradio user interface. The work did not have a client but was based on the author's own observations during an internship in IT-Helpdesk job.

The theoretical part discusses the key concepts of chatbot technology, as well as the operation of a language model based on machine learning. The thesis is functional, and the functionality of the chatbot was evaluated manually by the developer through practical testing.

The results showed that the chatbot is able to answer common IT-Helpdesk questions effectively, and it recognizes modified questions to a certain extent, so that the user does not have to ask the question in the exact correct form.

However, it was not possible to produce an answer for completely new questions that are not included in the dataset without manually updating the questions in the dataset.

However, the results were good and there are a lot of possibilities for further development.

Keywords

Chatbot, BERT, Hugging Face, Gradio, Artificial Intelligence

Pages 23 pages and 1 appendix

Sisällys

Sanasto

Hugging Face	Koneoppimisen ja luonnollisen kielen prosessoinnin kirjastoiden alusta
BERT	Syväoppimiseen perustuva kielimalli
Gradio	Työkalu käyttöliittymän rakentamiseen
Python	Ohjelmointikieli
Transformers	Kirjasto, joka tarjoaa valmiita kielimalleja
Dataset	Aineisto, joka sisältää kysymykset, synonyymit ja vastaukset
AI	Tekoäly
NLP	Natural Language Processing / Luonnollisen kielen käsittely

Kuvat

Ohjelmakoodit

Liitteet – esimerkki

1 Johdanto

Tekoäly on jatkuvasti kasvavassa määrin läsnä jokapäiväisessä elämässämme, ja ihmisillä on siitä ristiriitaisia tuntemuksia. Toiset pelkäävät tekoälyn yleistymistä, ja toiset näkevät paljon uusia mahdollisuuksia sen hyödyntämisessä. Kielimallit kuten ChatGPT ovat yleistyessään saaneet suuren määrän huomiota ja herättäneet keskustelua. ChatGPT:n yleistyessä ihmiset ovat myös ensikertaa itse päässeet suoraan keskustelemaan tekoälypohjaisen chatbotin kanssa.

Yritykset ja organisaatiot pyrkivät löytämään tapoja hyödyntää tekoälyn mahdollisuuksia sen yleistyessä. Varsinkin tukipalvelut saavat usein toistuvia pyyntöjä yksinkertaisista ongelmista, jotka vievät työntekijöiltä paljon aikaa, joka voitaisiin käyttää myös ”oikeiden” ongelmien ratkaisemiseen. Chatbotit parantavat asiakaspalvelua automatisoimalla vastauksia yksinkertaisiin usein saatuihin kysymyksiin, jolloin resurssit voidaan kohdentaa tehokkaammin monimutkaisten ongelmien ratkaisemiseen.

Opinnäytetyöni tavoite on vastata tähän tarpeeseen kehittämällä toimiva, helppokäyttöinen chatbot, joka hyödyntää BERT-kielimallia Hugging Facesta. Hugging Face on alusta, joka tarjoaa valmiita malleja erilaisiin käyttötarkoituksiin, kuten chatbotteihin, tekstinkääntämiseen ja kuvantunnistukseen. Chatbot on suunnattu nimenomaan IT-Helpdesk-ympäristöön, jossa hoidetaan perustason IT-tukea, kuten ohjelmistojen käyttöön liittyviä ongelmia ja laitteiden perushuoltoa. Pienellä muokkaamisella sen toimintaa voidaan myös laajentaa muihin vastaavanlaisiin käyttötarkoituksiin. Opinnäytetyö toteutettiin toiminnallisena kehitysprojektina. Toteutus eteni vaihettain ja muistutti ketteriä menetelmiä, jossa suunnittelu, toteutus ja testaus toistuivat kehitysprosessin aikana.

Opinnäytetyössä pyritään vastaamaan seuraaviin kysymyksiin:

- Kuinka hyvin chatbot pystyy käsittelemään IT-tuen yleisimpiä kysymyksiä suomen kielellä?
- Miten Hugging Face mallien ja Gradio käyttöliittymän hyödyntäminen helpottaa chatbotin kehitysprosessia?
- Kuinka hyvin chatbot pystyy mukautumaan uusiin kysymyksiin ja kuinka helppoa sen ylläpitäminen on?

2 Tekoäly ja kielimallit

Mitä oikeastaan tarkoittaa nykyisin varmasti kaikille tutuksi tullut termi tekoäly? ”Tekoäly (AI) on tietokoneen tai tietokoneohjatun robotin kyky suorittaa tehtäviä, jotka yleisesti yhdistetään älykkäisiin olentoihin. Termiä käytetään usein hankkeista, joissa kehitetään järjestelmiä, joilla on ihmisen älyllisiä prosesseja muistuttavia kykyjä, kuten päättely, merkityksen löytäminen, yleistämiskyky ja oppiminen aiemmista kokemuksista." (B.J. Copeland, 2025)

Nykypäivänä sana tekoäly on jo kaikkien huulilla, syynä tähän on tietomäärän kasvu, algoritmien kehittyminen, sekä laskentatehon ja tallennuskapasiteetin parantuminen. Ensimmäisen kerran tekoäly mainittiin kuitenkin jo vuonna 1956, jolloin tekoälytutkimus keskittyi ongelmanratkaisun ja symbolisten menetelmien tyylisiin aiheisiin. Nykyisin tekoälyä voidaan hyödyntää esimerkiksi verkkokauppaostosten tekemisen tehostamiseen tarjoamalla henkilökohtaisia suosituksia, ja esittelemällä eri tuotevaihtoehtoja. Tekoälyteknologialla voidaan myös tunnistaa mahdollisia vilpillisiä maksutapahtumia, sekä automatisoida paljon manuaalista työtä vaativia tiedonhallintatehtäviä. (SAS Institute, n.d.-b)

Kielimalleista erityisesti tutuiksi ovat tulleet Transformers-pohjaiset kielimallit kuten BERT ja GPT. Näistä kahdesta mallista GPT on paljon suurempi, ja sitä käytetäänkin ChatGPT:n kielimallina, kielimallin suuri koko on mahdollistanut huiman edistyksen luonnollisessa kielen käsittelyssä, mutta mallin kouluttaminen vaatii todella suuren määrän laskentatehoa, mallin kouluttaminen on toteutettu 175 miljardilla parametrillä, se on 10 kertainen määrä verrattuna mihinkään muuhun aiemmin kehitettyyn non-sparse kielimalliin. GPT:n suorituskyky onkin todella vahva esimerkiksi kääntämis-, kysymyksiin vastaamis- ja muissa nopeaa päättelykykyä edellyttävissä tehtävissä, kuten sanojen purkamisessa. (Brown et al., 2020)

Tekoälyn tarjoamat ratkaisut organisaatioiden sisällä ovat kasvaneet huimasti, ja ne näkyvät esimerkiksi automatisoiduissa vastauksissa asiakkaille, sekä paremmin kohdennetussa palveluiden tarjonnassa, sillä tekoäly kykenee analysoimaan erilaisia mielipiteitä. Voidaankin todeta, että tekoäly on tullut arkeemme jäädäkseen.

Tekoäly on yhä keskeisemmässä roolissa arjessamme, sillä se mahdollistaa älykkäiden ohjelmistojen ja laitteiden kehittämisen, sekä analysoinnin. Yksi tunnetuimmista tekoälypohjaisista sovelluksista on chatbot, joka jäljittelee ihmisen kanssa käytävää keskustelua tekstin tai äänen avulla. Ymmärtääkseen käyttäjän syötteitä ja tuottaakseen sopivia vastauksia chatbotit hyödyntävät luonnollisen kielen käsittelyä (NLP). (Adamopoulou & Moussiades, 2020)

Chatbotit yhdistetään usein viihdekäyttöön, mutta niillä on myös paljon käyttöä myös koulutuksessa, tiedonhaussa, liiketoiminnassa ja verkkokaupoissa. Ne ovatkin saavuttaneet suuren suosion. Chatbotit ovat usein myös todella helppokäyttöisiä, sillä ne ovat monesti alustariippumattomia, eivätkä ne vaadi erillistä asentamista, usein ne ovat myös saatavilla ilman erillistä kirjautumista. (Adamopoulou & Moussiades, 2020)

2.1 BERT

Bidirectional encoder repsentations from transformers (BERT) on Googlen vuonna 2018 kehittämä luonnollisen kielen käsittelyn kaksisuuntainen malli, joka tarkoittaa sitä, että malli kykenee tarkastelemaan tekstiä samanaikaisesti vasemmalta oikealle, sekä oikealta vasemmalle. (Devlin et al., 2019)

Jotta voidaan kouluttaa syvällinen kaksisuuntainen esitys, tietty osa syötteen tokeneista peitetään, jonka jälkeen mallia opetetaan ennustamaan peitetyt tokenit. Kyseistä menetelmää kutsutaan Masked Language Modelingiksi (MLM), usein sitä kutsutaan myös Cloze menetelmäksi. Kyseisessä menetelmässä peitettyjä sanoja vastaavat piilovektorit syötetään softmax funktioon, joka ennustaa puuttuvat sanat sanaston perusteella kuten perinteisessä kielimallissa. (Devlin et al., 2019)

2.2 Gradio

Koneoppimisen (ML) parissa käytettävyys on suuri haaste, tyypillisesti ML-mallit vaativat erikoislaitteistoa, sekä kokemusta ML:n parissa, tämä tuottaa ongelmia varsinkin käyttäjien päässä, joka puolestaan vaikeuttaa esimerkiksi palautteen antamista mallista, joka on mallin kehityksen kannalta todella tärkeää.

Päätin käyttää Gradio:ta käyttöliittymän luomiseen, sillä se on tehty todella helpoksi, sekä myös jakaminen on hyvin yksinkertaista. Näin säästyy aikaa enemmän itse mallin suoriutumisen parantamiseen. Näin (Abid et al., 2019), kertoo Gradiosta artikkelissaan, ”parantaaksemme saavutettavuutta ja helpottaaksemme yhteistyötä, kehitimme avoimen lähdenköödin Python paketin, jonka avulla tutkijat voivat luoda nopeasti visuaalisen käyttöliittymän ML-malleilleen. Gradio tekee pääsystä mihin tahansa ML-malliin yhtä helppoa kuin URL-osoitteen jakaminen.”

2.3 Koneoppiminen

Koneoppiminen on yksi tekoälyn haaroista, joka perustaa ideaan, jossa järjestelmät voivat oppia datasta, tunnistaa kaavoja/malleja, sekä tehdä päätöksiä minimaalisella ihmisen väliintulolla. Koneoppiminen syntyi kuviontunnistuksen ja koneiden itsenäisen oppimisen teorian perusteella, jonka mukaan koneet voivat oppia ilman, että niitä on ohjelmoitu suorittamaan tiettyjä tehtäviä. (SAS Institute, n.d.-a)

Kiinnostus koneoppimista kohtaan on kasvanut todella paljon, syitä siihen ovat vaihteleva saatavilla oleva data, laskentatehon määrä, joka on tehokasta ja halpaa, sekä edullinen tietojen tallennus. Näiden yhdistelmä mahdollistaa nopean ja automatisoidun tavan tuottaa malleja, jotka kykenevät analysoimaan monimutkaista dataa, sekä tuottamaan nopeasti tarkkoja tuloksia, jopa erittäin suuressa mittakaavassa. Koneoppimista hyödyntävät useimmat suuria tietomääriä käsittelevät teollisuudenalat, kuten rahoituspalvelut, terveydenhuolto, vakuutus, biotieteet, julkinen sektori, sekä vähittäiskauppa ja kulutustavarat. Koneoppimisteknologioita hyödyntämällä organisaatiot saavat etulyöntiaseman kilpailijoihin nähden. (SAS Institute, n.d.-a)

Ohjattu oppiminen (supervised learning) ja ohjaamaton oppiminen (unsupervised learning), ovat kaksi yleisimmin käytettyä koneoppimismenetelmää. (SAS Institute, n.d.-a) artikkelissa ohjatusta oppimisesta kerrotaan seuraavasti ”Ohjattua oppimista käytetään yleisesti sovelluksissa, joissa historialliset tiedot ennustavat todennäköisiä tulevia tapahtumia. Se voi esimerkiksi ennakoida, milloin luottokorttitapahtumat ovat todennäköisesti vilpillisiä tai mikä vakuutusasiakas todennäköisesti tekee korvausvaatimuksen.” Kun taas ohjaamattomasta oppimisesta he kirjoittavat näin ”Ohjaamaton oppiminen toimii hyvin tapahtumatiedoissa. Se voi esimerkiksi tunnistaa asiakassegmenttejä, joilla on samanlaiset

ominaisuudet ja joita voidaan kohdella samalla tavalla markkinointikampanjoissa. Tai se voi löytää tärkeimmät attribuutit, jotka erottavat asiakassegmentit toisistaan. Suosittuja tekniikoita ovat itseorganisoituvat kartat, lähin naapurikartoitus, k-keskiarvojen klusterointi ja singulaariarvojen hajottaminen. Näitä algoritmeja käytetään myös tekstiaiheiden segmentointiin, kohteiden suosittelemiseen ja poikkeavien tietojen tunnistamiseen.”

2.4 Luonnollisen kielen käsittely

Luonnollisen kielen käsittely (NLP) on myös yksi tekoälyn haara, joka auttaa tietokoneita ymmärtämään ihmisten kieltä. Tietokoneen ”äidinkieli” tunnetaan konekielenä, joka koostuu vain suuresta määrästä nollia, sekä ykkösiä, jotka tuottavat loogisia toimia. (SAS Institute, n.d.-c)

NLP tekee siis mahdolliseksi sen, että koneet ymmärtävät tekstiä ja puhetta, pystyvät tulkitsemaan niitä, sekä mittaamaan tunteita, ja määrittämään mitkä osiot ovat tärkeitä. NLP myös auttaa ratkaisemaan kielen epäselvyyksiä ja lisää dataan hyödyllistä numeerista rakennetta, tämä on hyödyllistä etenkin puheentunnistuksessa tai tekstianalyysissä.

Tekstidata ja äänipohjainen data vaihtelevat suuresti, kuten myös niiden käytännön sovellukset, joten tarvitaan laaja valikoima erilaisia lähestymistapoja. NLP sisältääkin monia erilaisia tekniikoita ihmisen kielen tulkitsemiseen, ja sen perustehtäviä ovat muun muassa tokenisointi, jäsentäminen, lemmatisointi, kielen havaitseminen ja semanttisten suhteiden tunnistaminen. Yksinkertaistettuna NLP jakaa kielen lyhyemmiksi elementeiksi, yrittää ymmärtää kappaleiden välisiä suhteita ja tutkii, kuinka palaset toimivat yhdessä merkityksen luomiseksi. (SAS Institute, n.d.-c)

3 IT-tuki

Yritykset ovat vahvasti riippuvaisia teknologiasta ja jatkuvasti toimivista yhteyksistä. Laitteisto- tai ohjelmistohäiriöt saattavat aiheuttaa esimerkiksi suurta liiketoiminnallista tappiota. Näitä ongelmia on monesti ensimmäisenä hoitamassa IT-tuki, joka tarjoaa teknistä tukea loppukäyttäjille, asiakkaille, sekä työntekijöille. IT-tiimin piiriin yleisesti kuuluvia tehtäviä ovat muun muassa: Laitteisto- ja ohjelmistovianetsintä, palvelinhuolto, pääsyn ja salasanan palauttaminen, tietojen varmuuskopiot, säännölliset tarkastukset ja jatkuva valvonta. (GetGuru, 2025)

IT-tiimi huolehtii jatkuvasta ylläpidosta ja tuesta, näin varmistetaan järjestelmien toimivuus ja pidetään käyttäjät tyytyväisenä. Toimiva tukipalvelu on elintärkeä, sillä se ei ainoastaan ratkaise ongelmia, vaan myös ehkäisee niitä. Sen ansiosta yritykset saavat keskittyä liiketoiminnan kasvattamiseen ilman ylimääräisiä häiriöitä.(Topteam, 2024)

4 Chatbot

Chatbotit ovat tekoälypohjaisia ratkaisuja, jotka tarjoavat tukea, sekä ratkaisuja käyttäjien ongelmiin. Ne käyttävät luonnollista kielen tunnistusta (NLP), ja vastaavat usein kysyttyihin kysymyksiin nopeasti ja tehokkaasti. Kun chatbotti vastaanottaa kysymyksen, se analysoi sitä ja tunnistaa niin sanotut avainsanat, jonka jälkeen hakee esimerkiksi tietokannasta sopivimman vastauksen. Varsinkin yksinkertaisiin tehtäviin tarkoitettujen chatbottien kanssa ongelmiin saatetaan törmätä, mikäli käyttäjän kysymys on monitulkintainen tai chatbottia ei ole koulutettu vastaamaan kyseiseen kysymykseen. Tämä on havaittu aiemassa tutkimuksessa, jossa chatbottien haasteiksi on todettu monitulkintaiset käyttäjäsyötteet, sekä monimutkaiset toimintopyynnöt (Dahiya, 2024).

Chatbotin toimivuuden kannalta käyttäjien kysymysten tarkkuus ja selkeys ovat keskeisiä tekijöitä sen toimivuuden kannalta. (Rospigliosi, 2023) korostaa artikkelissaan ChatGPT:stä, että ChatGPT:n tarkoitus on olla vuorovaikutuksessa keskustelun muodossa, joka muodostuu käyttäjän kysymyksistä, sekä jatkokysymyksistä, jotta käyttäjä saa haluamansa vastauksen. Tässä huomataan chatbottien ja hakukoneiden välinen ero, sillä perinteiset hakukoneet, kuten google palauttavat hakutuloksia ilman keskustelun jatkuvuutta.

4.1 Chatbottien historia

Yksi aikaisin ja tunnetuin chatbot on Eliza, jonka loi Joseph Weizenbaum vuonna 1966. Eliza käytti ennalta määriteltyjä skriptejä, jotka jäljittelivät psykoterapeuttia. Eliza ei kuitenkaan ymmärtänyt keskustelua, vaan etsi vain oikeita vastauksia kuvion tunnistuksen avulla. (Shum et al., 2018)

Parry on toinen jo vuonna 1975 Kenneth Colbyn kehittämä chatbotti ja sen tarkoitus oli käyttäytyä vainoharhaisen henkilön tavoin. Myös Parry on Elizan tavoin sääntöpohjainen ja sen rakenne on samantyylinen, mutta Parryn kielen ymmärryskyky on parempi, sekä se osaa paremmin simuloida botin tunteita. Esimerkiksi se vastaa vihamielisesti, mikäli viha taso on korkea. (Shum et al., 2018)

Ensimmäiset chatbotit syntyivät Turingin testin innoittamina. Testin perusajatuksena on arvioida, että pystyykö kone jäljittelemään ihmisen kaltaista älykkyyttä niin hyvin, että keskustelukumppani ei erota sitä oikeasta ihmisestä. (Shum et al., 2018).

4.2 Chatbottien rooli nyt ja tulevaisuudessa

Nykyisin chatbotit ovat muuttuneet yksinkertaisista sääntöpohjaisista järjestelmistä kehittyneiksi keskusteluboteiksi, jotka ymmärtävät käyttäjäsyötteitä paljon paremmin, sekä kykenevät tarjoamaan käyttäjälle merkityksellisempiä vuorovaikutuskokemuksia. (Saksman, 2024)

Asioit sitten missä vain nykypäivänä, on hyvin todennäköistä, että sinulle tarjotaan mahdollisuutta olla vuorovaikutuksessa chatbotin kanssa. Verkkosivustoilla ja mobiilisovelluksissa käytetään paljon chatbotteja asiakaspalvelun hoitamiseen. Terveydenhuollossa chatbotteja käytetään arvioimaan oireita ja ohjaamaan mahdolliseen hoitoon. Myös virtuaaliavustajat, kuten Siri, Cortana, Google Assistant ja Alexa ovat tulleet hyvin tunnetuiksi. (Yan & Alterovitz, 2024)

On sanomattakin selvää, että tulevaisuudessa tekoäly ja chatbotit tulevat olemaan entistä keskeisempi osa arkeamme, kuten myös liiketoimintaa. Chatbotit tulevat jatkuvasti oppimaan ja mukautumaan, ja ne tulevat olemaan enemmän personoituja älykkäitä digitaalisia avustajia, sekä virtuaaliavustajia.

5 Chatbotin suunnittelu ja toteutus

Chatbotin toteutus eteni toiminnallisena kehitysprojektina, jossa suunnittelu, toteutus ja testaus muodostavat selkeän kokonaisuuden. Työ jakautui vaiheisiin, jotka tukivat toisiaan ja edistivät chatbotin toiminnallisuuden rakentumista.

Luonnollinen kielen käsittely (NLP) on chatbotin ydin, sen avulla botti ymmärtää käyttäjien syötteitä, sekä tuottaa niihin vastauksia. Mallina toimii BERT multilingual base model (cased). Mallin kouluttamiseen käytetään Google Colab ympäristöä, joka tarjoaa T4-näytönohjaimen ja mahdollistaa täten nopeamman mallin koulutuksen.

Chatbotin käyttöliittymä tulee olla selkeä ja käyttäjäystävällinen, jotta sitä voidaan käyttää tehokkaasti. Toteutuksessa käytetään Gradiota, joka mahdollistaa käyttöliittymän rakentamisen, sekä chatbotin testaamisen suoraan selaimessa.

BERT-malli toimii tässä keskeisenä komponenttina Google Colabin kanssa. Mallin kouluttaminen ja optimointi voidaan suorittaa Colabia hyödyntäen ilman paikallisia resurssirajoitteita.

Chatbot tarvitsee tietokannan, josta se hakee kysymyksiin vastauksia. Tämä on toteutettu yksinkertaisella JSON-tiedostolla, joka sisältää tallennetut vastaukset kysymyksiin, sekä niiden synonyymeihin. JSON tiedoston kysymyksien, synonyymien ja vastauksien teettämiseen on hyödynnetty tekoälyä (ChatGPT).

Chatbotin kehitysprosessi koostuu useammista vaiheista, joilla kaikilla on oma roolinsa kokonaisuuden rakentamisessa. Seuraavaksi esitellään vaiheet tarkemmin.

Aluksi chatbotille täytyy kerätä dataa, jota botti käyttää vastauksissaan. Tässä vaiheessa luodaan lista usein toistuvista kysymyksistä, synonyymeista ja oikeista vastauksista JSON-tiedostoon

Seuraavaksi tulee kouluttaa BERT-malli, tämä aloitetaan lataamalla se Google Colabin ympäristöön, jossa se opetetaan ymmärtämään käyttäjän syötteitä. Ladataan myös koulutusdata, jotta malli oppii tunnistamaan kysymyksiä ja vastauksia niihin.

Chatbot tarvitsee myös käyttöliittymän, jonka avulla toimintaa voidaan testata. Tätä varten ladataan Google Colabiin Gradio-kirjasto, sekä luodaan chatbotille yksinkertainen käyttöliittymä.

Viimeisenä vuorossa on chatbotin testaus, sekä optimointi. Testaus suoritetaan syöttämälle erilaisia syötteitä, ja varmistetaan, että chatbot pystyy käsittelemään ne tehokkaasti. Täytyy myös olettaa, että chatbot ei osaa vastata kaikkiin kysymyksiin, joten myös virheenkäsittely tulee ottaa huomioon.

6 Tekninen toteutus

Tässä luvussa kuvataan chatbotin teknistä toteutusta, sekä eri kehitysvaiheita. Chatbotista luotiin kaksi prototyyppimallia, joista ensimmäinen keskittyi perustoiminnallisuuteen ja täsmällisten kysymysten vastausten löytämiseen datasetistä. Toinen prototyyppi laajensi toiminnallisuutta, sekä paransi vastausten tarkkuutta. Alaluvuissa esitellään chatbotin kehitystä sekä keskeisempiä yksityiskohtia.

6.1 Prototyyppi 1

Ensimmäinen prototyyppi on yksinkertainen chatbot käyttöliittymä, joka hyödyntää valmista kysymys vastaus JSON-tiedostoa. Botti palauttaa käyttäjälle oikean vastauksen, mikäli se löytyy datasetistä, jos oikeaa vastausta ei löydy, palautetaan ilmoitus ”en löytänyt vastausta kysymykseesi”.

Ensimmäiseksi otetaan tarvittavat kirjastot käyttöön (Ohjelmakoodi 1). Tämän jälkeen alustetaan BERT-malli (Ohjelmakoodi 2) ja ladataan datasetti (Ohjelmakoodi 3).

Kun käyttäjä syöttää kysymyksen, hyödynnetään oikean vastauksen hakulogiikkaa (Ohjelmakoodi 4). Kysymyksen käsittely, sekä keskusteluhistorian hallinta tapahtuvat (Ohjelmakoodi 5) mukaisesti. Viimeisenä rakennetaan käyttöliittymä Gradiolla, sekä käynnistetään se (Ohjelmakoodi 6).

Ohjelmakoodi 1. Prototyyppi 1 Import-lauseet tarvittaville kirjastoille

```
# Tarvittavat importit
from transformers import AutoTokenizer, AutoModelForQuestionAnswering, pipeline # Transformers-kirjasto NLP-mallia varten
import gradio as gr # Gradio käyttöliittymän luomiseen
import json # JSON-tiedoston käsittelyyn
```

Ohjelmakoodi 2. Prototyyppi 1 BERT-mallin lataaminen ja alustaminen

```
# Luodaan tarvittavat muuttujat ja ladataan BERT-malli
checkpoint = "bert-base-multilingual-cased"
device = "cuda"  # "cuda" tai "cpu"
tokenizer = AutoTokenizer.from_pretrained(checkpoint) # Syötteen käsittelyyn
model = AutoModelForQuestionAnswering.from_pretrained(checkpoint).to(device) # Mallin lataus
qa_pipeline = pipeline("question-answering", model=model, tokenizer=tokenizer) # Luodaan pipeline
```

Ohjelmakoodi 3. Prototyyppi 1 Datasetin lataaminen ja virheenkäsittely

```
# Datasetin polku
dataset_path = "/content/drive/MyDrive/Colab Notebooks/dataset_fixed.json"

# Datasetin lataus ja virheenkäsittely
def load_dataset(file_path):
    try:
        with open(file_path, "r", encoding="utf-8") as file:
            dataset = json.load(file)
        if isinstance(dataset, list) and len(dataset) > 0:
            print(f"Kysymysten määrä: {len(dataset)}")
            return dataset
        else:
            print("Datasetin rakenne ei vastaa odotuksia.")
            return []
    except Exception as e:
        print(f"Datasetin lataus epäonnistui: {e}")
        return []

# Ladataan datasetti
dataset = load_dataset(dataset_path)
```

Ohjelmakoodi 4. Prototyyppi 1 Oikean vastauksen hakulogiikka

```
# Oikean vastauksen löytämisen logiikka
def find_answer(question):
    for item in dataset:
        all_questions = [item["question"]] + item.get("synonyms", []) # Luodaan lista, joka sisältää pääkysymyken ja synonyymit
        if question.lower() in [q.lower() for q in all_questions]: # Muutetaan kysymykset pieniksi kirjaimiksi
            return item["answer"] # Oikean vastauksen palautus
    return None
```

Ohjelmakoodi 5. Prototyyppi 1 Kysymysten käsittely ja historian päivitys

```
# Kysymysten käsittely
def predict(message, history):
    if history is None:
        history = [] # Mikäli historiaa ei ole, luodaan tyhjä lista
    history.append([message, ""])  # Lisätään käyttäjän kysymys historiaan

    dataset_answer = find_answer(message) # Etsitään oikeaa vastausta datasetistä

    if dataset_answer:
        response = dataset_answer # Oikea vastaus löytyy
    else:
        response = "En löytänyt vastausta kysymykseesi." # Oletusvastaus mikäli oikeaa vastausta ei löydy

    history[-1] = [message, response]  # Päivitetään vastaus historiaan
    return history # Palautetaan päivitetty historia
```

Ohjelmakoodi 6. Prototyyppi 1 Gradio käyttöliittymän luominen ja käynnistäminen

```
# Luodaan gradio käyttöliittymä
with gr.Blocks() as demo:
    gr.Markdown("# Chatbot - Keskustele AI:n kanssa 🤖") # Käyttöliittymän otsikko
    chatbot = gr.Chatbot(label="Chatbot")
    prompt = gr.Textbox(max_lines=1, label="Kirjoita viesti") # Käyttäjän syötteen teksikenttä

   # Syötetään käyttäjän syöte botille
    prompt.submit(predict, inputs=[prompt, chatbot], outputs=chatbot)
    # Lähettämisen jälkeen tyhjennetään käyttäjän tekstikenttä
    prompt.submit(lambda: "", None, [prompt])

# Käynnistetään chatbotti
demo.launch()
```

6.2 Prototyyppi 2

Prototyyppi 2 on edellistä kehittyneempi chatbot-käyttöliittymä (Kuva 1), joka ei enään perustu pelkästään tarkkoihin vastaavuuksiin. Tämä malli muuntaa tekstin vektorimuotoon ja hyödyntää kosinin samankaltaisuutta, jolloin chatbot tarkistaa ensin, löytyykö täsmällinen osuma datasetistä. Mikäli vastaava kysymys löytyy, chatbot palauttaa sen. Jos täsmällistä osumaa ei löydy, chatbot vertaa sitä valmiiksi laskettuihin kysymysten vektoreihin käyttämällä kosinin samankaltaisuus menetelmää, jonka avulla voidaan mitata kahden lauseen samankaltaisuutta, sitä käytetäänkin usein monissa koneoppimis ja data-analyysisovelluksissa. Mikäli vastaavaa osumaa ei löydy asetetun raja-arvon mukaan, opastetaan käyttäjää olemaan yhteydessä asiantuntijaan.

Uusi malli parantaa chatbotin tehokkuutta merkittävästi, sillä se tunnistaa samankaltaisia kysymyksiä, eikä ole riippuvainen täysin samasta lausemuodosta. Tämä tekee siitä joustavamman ja tehokkaamman verrattuna ensimmäiseen malliin. Lisäksi käyttössä on CUDA (Compute Unified

Device Architecture), tämä optimoi BERT-mallin laskentaa erityisesti jos käytössä on suurempia datasettejä. CUDA mahdollistaa GPU:n käytön rinnakkaislaskentaan, joka nopeuttaa laskentaa huomattavan paljon, sillä GPU sisältää moninkertaisen määrän ytimiä verrattuna CPU:hun.

Viimeisenä lisäyksenä käyttäjää varten on lisätty dropdown valikko, josta käyttäjä voi etsiä valmiita kysymyksiä ja valita niistä haluamansa. Tämä malli on joustavampi, nopeampi ja käyttäjäystävällisempi kuin edeltäjänsä, ja sen ansiosta se soveltuu huomattavasti paremmin IT-tuen avuksi.

Ohjelmakoodi 7 käsittelee tarvittavien kirjastojen tuonnin. BERT- mallin lataus ja käytettävän laitteen valinta on toteutettu (Ohjelmakoodi 8) mukaisesti. Datasetin kysymykset muunnetaan vektorimuotoon (Ohjelmakoodi 9). Käyttäjän esittämän kysymyksen esikäsittely ja vastauksen etsiminen käsitellään (Ohjelmakoodi 10) ja (Ohjelmakoodi 11) mukaisesti. Viimeisenä Gradio käyttöliittymän luonti, sekä käynnistäminen (Ohjelmakoodi 12)

Ohjelmakoodi 7. Prototyyppi 2 tarvittavat importit

```
# Tarvittavat importit
import json
import torch
import numpy as np
from transformers import AutoTokenizer, AutoModel
from sklearn.metrics.pairwise import cosine_similarity
import gradio as gr
import re
```

Ohjelmakoodi 8. Prototyyppi 2 laitteiston valinta ja mallin lataus

```
# Valitaan joko GPU tai CPU (oletuksena GPU)
device = "cuda" if torch.cuda.is_available() else "cpu"

# Mallin lataus
print("Ladataan BERT-malli...")
tokenizer = AutoTokenizer.from_pretrained("bert-base-multilingual-cased")
model = AutoModel.from_pretrained("bert-base-multilingual-cased").to(device)

# Datasetin lataus (varmistetaan, että tiedosto on olemassa)
dataset_path = "/content/drive/MyDrive/Colab Notebooks/dataset_updated.json"

# Ladataan JSON-tiedosto
with open(dataset_path, "r", encoding="utf-8") as file:
    dataset = json.load(file)
```

Ohjelmakoodi 9. Prototyyppi 2 kysymysten vektorointi

```
# Luodaan sanakirja kysymysten vektorimuodoille
question_embeddings = {}

# Muunnetaan teksti numeeriseen muotoon
def embed_text(text):
    tokens = tokenizer(text, return_tensors="pt", padding=True, truncation=True).to(device)
    with torch.no_grad():
        output = model(**tokens)
    embedding = output.last_hidden_state[:, 0, :].cpu().numpy()
    return embedding

# Käydään kysymykset läpi
for item in dataset:
    all_questions = [item["question"]] + item.get("synonyms", [])
    question_embeddings[item["question"]] = {
        "embeddings": [embed_text(q) for q in all_questions],
        "answer": item["answer"]
    }
```

Ohjelmakoodi 10. Prototyyppi 2 kysymysten esikäsittely ja haku

```
# Kysymyksen esikäsittely
def preprocess_question(question):
    clean_question = re.sub(r'[^a-zA-ZäöåÄÖÅ0-9\s\?\!]', '', question).strip() # Poistetaan erikoismerkit
    return clean_question.lower() if clean_question else None # Muutetaan pieniksi kirjaimiksi

# Parhaan mahdollisen vastauksen hakeminen
def find_best_answer(question):
    question = preprocess_question(question)
    if not question or len(question) < 10:
        return "En ymmärtänyt kysymystä. Voitko muotoilla sen tarkemmin. Esim. 'Kuinka voin vaihtaa salasanan?'"

    # Tarkistetaan, löytyykö suora osuma datasetistä
    for item in dataset:
        if question.lower() == item["question"].lower():
            return item["answer"]

    # Jos suoraa osumaa ei ole, käytetään vektorimallia etsimään lähin vastaus
    question_embedding = embed_text(question).reshape(1, -1)
    best_match = None
    highest_similarity = -1
```

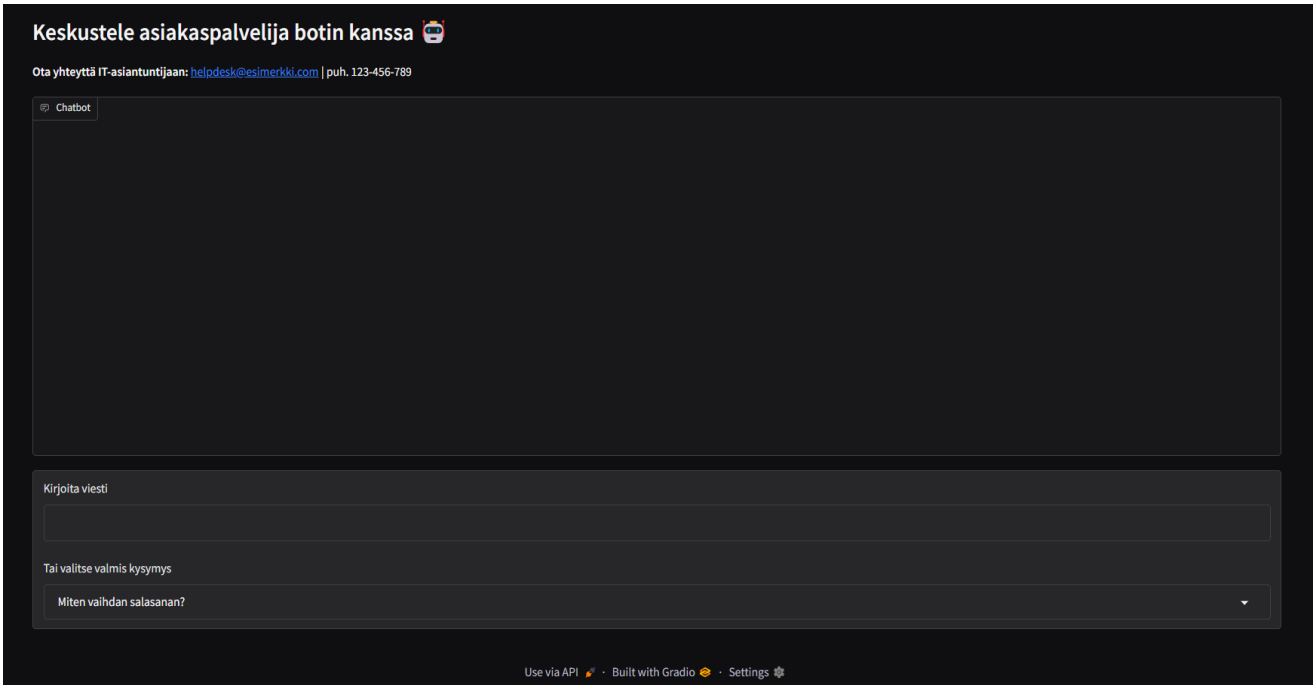
Ohjelmakoodi 11. Prototyyppi 2 semanttinen haku ja vastauslogiikka

```
# Käydään läpi tallennetut kysymykset ja lasketaan samankaltaisuus käyttäjän kysymykseen
for stored_question, data in question_embeddings.items():
    similarities = [cosine_similarity(question_embedding, emb.reshape(1, -1))[0][0] for emb in data["embeddings"]]
    max_similarity = max(similarities)
    if max_similarity > highest_similarity:
        highest_similarity = max_similarity
        best_match = data["answer"]


# Ohjataan käyttäjä ottamaan yhteyttä asiantuntijaan, mikäli sopiva vastaus ei ylitä raja-arvoa
if highest_similarity < 0.85: # Säädettävä raja-arvo
    return "En löytänyt tarkkaa vastausta. Voit myös ottaa yhteyttä IT-tukeen: helpdesk@esimerkki.com tai puh. 123-456-789."
return best_match
```

Ohjelmakoodi 12. Prototyyppi 2 chatbotin käyttöliittymä Gradio-alustalla

```
# Keskusteluhistorian ylläpito
def simple_response(message, history):
    response = find_best_answer(message)
    history.append([message, response])
    return history

# Gradio-käyttöliittymän luominen chatbotille
with gr.Blocks() as demo:
    gr.Markdown("# Keskustele asiakaspalvelija botin kanssa 🤖")
    gr.Markdown("**Ota yhteyttä IT-asiantuntijaan:** helpdesk@esimerkki.com | puh. 123-456-789")
    chatbot = gr.Chatbot(label="Chatbot")
    prompt = gr.Textbox(max_lines=1, label="Kirjoita viesti")

    # Lisätään dropdown-valikko, josta käyttäjä voi suoraan etsiä yleisiä kysymyksiä
    suggestions = gr.Dropdown(
        [item["question"] for item in dataset],
        label="Tai valitse valmis kysymys",
    )

    # Käsitellään käyttäjän viesti ja tyhjennetään syötekenttä yhdellä funktiolla
    def handle_input(user_input, history):
        return simple_response(user_input, history), ""

    # Lähetetään syöte chatbotille
    prompt.submit(handle_input, inputs=[prompt, chatbot], outputs=[chatbot, prompt])
    suggestions.change(handle_input, inputs=[suggestions, chatbot], outputs=[chatbot, prompt])

# Käynnistetään chatbot käyttöliittymä
print("Käynnistetään chatbot...")
demo.launch()
```

Kuva 1. Prototyyppi 2 valmis käyttöliittymä



6.3 Testaus ja toimivuus

Chatbotin testaus toteutettiin manuaalisesti syöttämällä erilaisia IT-tukeen liittyviä kysymyksiä. Testien tarkoituksena oli arvioida botin kykyä vastata, sekä suoraan datasetistä löytyviin kysymyksiin että muunneltuihin kysymyksiin. Testituloksien mukaan chatbottia päivitettiin havaittujen puutteiden ja uusien toiminnallisuuksien mahdollisuuksien mukaisesti.

Täsmällisiin kysymyksiin vastattiin onnistuneesti lähes kaikissa tapauksissa. Kuitenkin muunneltujen kysymysten kohdalla havaittiin rajoitteita, kun synonyymit poikkesivat merkittävästi alkuperäisestä kysymyksestä. Tulosten perusteella chatbot on kuitenkin tehokas vastaamaan IT-helpdeskin yleisiin kysymyksiin, vaikka toistaiseksi muotoiltuihin kysymyksiin chatbot osaa vastata rajallisesti.

Synonyymien käsittelyssä ja vektorihaussa on parannettavaa, jotta muunneltuja kysymyksiä voidaan käsitellä paremmin. Chatbotin nopeus on hyvä, eikä käyttäjällä ole merkittävää viivettä keskustelussa. Vastauksen saa noin 1–2 sekunnissa.

Mikäli botti saa epäselvän kysymyksen, palautetaan käyttäjälle ”En löytänyt tarkkaa vastausta. Voit myös ottaa yhteyttä IT-tukeen: helpdesk@esimerkki.com tai puh. 123-456-789.” Virheenkäsittely ei kuitenkaan toimi täysin moitteettomasti, sillä satunnaisesti asiaankuulumatonta kysymystä kysyttäessä, palautetaan väärä vastaus, eikä suunniteltua fall-back vastausta.

7 Tulokset ja johtopäätökset

Opinnäytetyön tuloksena syntyi toimiva chatbot käyttöliittymä, joka pystyy vastaamaan IT-tuen yleisiin kysymyksiin. Chatbot hyödyntää BERT-mallia ja kosinin samankaltaisuus analyysia, joka auttaa bottia löytämään vastauksia, vaikka kysymys ei ole täysin vastaavanlainen kuin datasetissä.

Vaikka malli kykenee tunnistamaan muunneltuja kysymyksiä, se ei kuitenkaan osaa vastata uusiin datasettiin kuulumattomiin kysymyksiin, joten vielä toistaiseksi täysin uudet kysymykset tulee lisätä datasettiin manuaalisesti. Kehitetty chatbot kuitenkin tarjoaa jo joustavan ratkaisun IT-Helpdeskin yleisten kysymysten käsittelyyn.

Tutkimuskysymys 1: Kuinka hyvin chatbot pystyy käsittelemään IT-tuen yleisimpiä kysymyksiä suomen kielellä?

Yleisimpiin kysymyksiin vastaaminen onnistuu tehokkaasti ja sujuvasti, erityisesti kun kysymykset ovat selkeästi muotoiltuja. Mitä lähempänä kysymys on alkuperäistä, sitä paremmin botti toimii.

Tutkimuskysymys 2: Miten Hugging Face-mallien ja Gradio käyttöliittymän hyödyntäminen helpottaa chatbotin kehitysprosessia?

Valmiin kielimallin käyttö mahdollisti mallin rakentamisen ilman omaa mallin koulutusta alusta alkaen, joka nopeutti kehitysprosessia merkittävän paljon. Gradio tarjosi helpon ja nopean tavan rakentaa, sekä testata käyttöliittymää suoraan selaimessa, tämä teki testauksesta vaivatonta.

Tutkimuskysymys 3: Kuinka hyvin chatbot pystyy mukautumaan uusiin kysymyksiin ja kuinka helppoa sen ylläpitäminen on?

Chatbot ei kykene vastaamaan täysin uusiin kysymyksiin ilman manuaalista päivitystä datasettiin. Uusien kysymysten lisääminen datasettiin on kuitenkin melko vaivaton prosessi, sillä toistaiseksi käytössä on JSON-tiedosto.

8 Yhteenveto

Tutkimuskysymyksiin vastaaminen onnistui hyvin. Chatbot pystyy käsittelemään IT-Helpdeskin yleisimpiä kysymyksiä suomen kielellä, sekä ymmärtää kysymysten erilaisia muotoja kosinin samankaltaisuuden avulla.

Seuraava kehitysvaihe olisi luoda chatbotille eri kielivaihtoehdot, jotta se olisi monikielinen. Lisäksi tulisi vielä parantaa chatbotin kykyä ymmärtää vajaita tai erityylisesti muotoiltuja kysymyksiä vielä tarkemmin ja näihin oikean vastauksen tarjoamista paremmalla todennäköisyydellä.

Hugging Facen BERT-mallin ja Gradio käyttöliittymän käyttö helpotti, sekä nopeutti chatbotin kehitysprosessia huomattavan paljon. Näiden työkalujen käyttö on melko yksinkertaista pienen opiskelun myötä. Molemmista aiheista löytyy verkkosivuilta hyvät dokumentaatiot. Gradion käyttöliittymää hyödyntämällä malleja on melko vaivatonta testata. Myös UI:n muotoiluun löytyisi vaihtoehtoja, mutta tässä opinnäytetyössä tavoite oli enemmänkin chatbotin toiminnallisuus eikä niinkään UI puoli.

Chatbotin kehittämistä voisi jatkaa suuremmilla kielimalleilla, ja myös lisätoiminnallisuuksia kuten puheen välityksellä tapahtuva keskustelu olisi hyvä lisätä. Nämä kuitenkin vaativat jo enemmän laskentatehoa. Myös verkkosivu- ja tietokantaintegraatio olisivat jatko askelia, jotka tekisivät chatbotista monipuolisemman ja tehokkaamman.

Jatkokehityksen kannalta myös käyttäjätestaus olisi hyödyllistä, sillä chatbotin testauksen suoritin täysin itsenäisesti. Ulkopuolisilta käyttäjiltä saisi mahdollisesti hyviä kehitysideoita, joita ei itse välttämättä huomaa kehitysprosessin aikana.

Lähteet

Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A., & Zou, J. (2019). *Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild* (arXiv:1906.02569). arXiv. https://doi.org/10.48550/arXiv.1906.02569

Adamopoulou, E., & Moussiades, L. (2020). An Overview of Chatbot Technology. *Artificial Intelligence Applications and Innovations*, *584*, 373–383. https://doi.org/10.1007/978-3-030-49186-4_31

B.J. Copeland. (2025, February 24). *Artificial intelligence (AI) | Definition, Examples, Types, Applications, Companies, & Facts | Britannica*. https://www.britannica.com/technology/artificial-intelligence

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). *Language Models are Few-Shot Learners* (arXiv:2005.14165). arXiv. https://doi.org/10.48550/arXiv.2005.14165

Dahiya. (2024). (PDF) A Tool of Conversation: Chatbot. *ResearchGate*. https://www.researchgate.net/publication/321864990_A_Tool_of_Conversation_Chatbot

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (arXiv:1810.04805). arXiv. https://doi.org/10.48550/arXiv.1810.04805

GetGuru. (2025). *Mikä on IT-tuki? Toiminnot, tyypit, työkalut ja paljon muuta [2025]*. https://www.getguru.com/fi/reference/it-support

Rospigliosi, P. 'asher.' (2023). Artificial intelligence in teaching and learning: What questions should we ask of ChatGPT? *Interactive Learning Environments*, *31*(1), 1–3. https://doi.org/10.1080/10494820.2023.2180191

Saksman, M. (2024, August 5). Chatbotit ja tekoäly—Asiakaspalvelun uusi aikakausi. *Sovellin*. https://www.sovellin.com/chatbotit-ja-tekoaly-asiakaspalvelun-uusi-aikakausi/

SAS Institute. (n.d.-a). *Machine Learning: What it is and why it matters*. Retrieved February 25, 2025, from https://www.sas.com/fi_fi/insights/analytics/machine-learning.html

SAS Institute. (n.d.-b). *Mitä on tekoäly (AI) ja miksi se on tärkeää?* Retrieved February 25, 2025, from https://www.sas.com/fi_fi/insights/analytics/what-is-artificial-intelligence.html

SAS Institute. (n.d.-c). *Natural Language Processing (NLP): What it is and why it matters*. Retrieved February 24, 2025, from https://www.sas.com/fi_fi/insights/analytics/what-is-natural-language-processing-nlp.html

Shum, H.-Y., He, X., & Li, D. (2018). *From Eliza to XiaoIce: Challenges and Opportunities with Social Chatbots* (arXiv:1801.01957). arXiv. https://doi.org/10.48550/arXiv.1801.01957

Topteam. (2024, October 28). Kattavat IT-palvelut: Ylläpito ja tuki turvanasi. *Topteam IT Oy*. https://topteam.fi/kattavat-it-palvelut-yllapito-ja-tuki-turvanasi/

Yan, N., & Alterovitz, G. (2024). *A General-purpose AI Avatar in Healthcare* (arXiv:2401.12981). arXiv. https://doi.org/10.48550/arXiv.2401.12981

Liite 1: Aineistonhallintasuunnitelma

Opinnäytetyön aineiston kuvaus

Projektin aikana tuotettu materiaali, kuten analyysit, sekä koodit ja tulosteet

Missä muodossa analysoitava aineisto on: tekstitiedoistoina, mittaustuloksia, koodina

Aineiston tallennus ja säilytys

Omalla salasanalla suojatulla tietokoneella

Varmuuskopio muistitikulle, sekä pilveen

Aineistoa käsittelee ainoastaan opinnäytetyön tekijä, sekä opinnäytetyön ohjaaja

Opinnäytetyössä ei käsitellä arkaluontoista dataa, eikä henkilötietoja

Ainesto perustuu projektin tuloksiin ja julkisiin kaikkien saatavilla oleviin lähteisiin

Aineiston omistajuus

Opinnäytetyön aineiston ja tulosten omistajuus kuuluu opinnäytetyön tekijälle

Aineiston jatkokäyttö työn valmistumisen jälkeen

1) Tutkimusainestoa ei anneta jatkokäyttöön.

Opinnäytetyö säilytetään tietoturvallisesti vuoden ajan, jonka jälkeen aineisto hävitetään tietoturvallisesti