
Ilmar Heinonen

# Tekoälyn hyödyntäminen yrityksen vastuullisuusraportin noutamiseen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

4.4.2024

# Tiivistelmä


| | |
|---|---|
| Tekijä: | Ilmar Heinonen |
| Otsikko: | Tekoälyn hyödyntäminen yrityksen vastuullisuusraportin noutamiseen |
| Sivumäärä: | 32 sivua |
| Aika: | 4.4.2024 |
| Tutkinto: | Insinööri (AMK) |
| Tutkinto-ohjelma: | Tieto- ja viestintätekniikka |
| Ammatillinen pääaine: | Mobiilisovellukset |
| Ohjaajat: | Tutkijaopettaja Hannu Markkanen<br>TKI-päällikkö Timo Nykopp |


---


Opinnäytetyössä keskitytään toimeksiantajan Vastrap verkkosivuston jatkokehittämiseen. Vastrapin tehtävänä on hakea yhtiöiden vastuullisuusraportteja tekoälyä hyödyntäen. Työn tärkeimpänä tavoitteena on parantaa tekoälyn kykyä tunnistaa vastuullisuusraportteja eri lähteistä. Lisäksi palvelimen tehokkuutta sekä käyttöliittymää jatkokehitetään.

Tekstin luokittelua varten luodaan uusi tekoälymalli suurella määrällä koulutusmateriaalia. Uutta ja vanhaa tekoälymallia verrattaessa huomattiin, että uudempi suoriutuu paremmin. Vastrapin vanha palvelin kaapi verkkosivuja. Kaapiminen teki palvelimesta vikaherkän ja tehottomamman. Kaapiminen nosti myös eettisiä ongelmia verkkosivujen kaapimisen kannalta. Palvelin kehitettiin hyödyntämään hakukonetta ja käyttöliittymää parannettiin mukautuvaksi tehostaen käyttäjäkokemusta.

Opinnäytetyön tuloksena jatkokehitetty Vastrap toimii huomattavasti paremmin, mutta parannettavaa myös on. Tekoälymallin parantamiseksi voitaisiin sitä kouluttaa suuremmalla määrällä koulutusmateriaalia. Tämän lisäksi useita tekoälymalleja voitaisiin kouluttaa, joista kukin erikoistuu erilaisiin raporttimuotoihin, kuten linkkeihin tai tiedostoihin.



Avainsanat: tekoäly, vastuullisuusraportti, sovelluskehitys, Google Cloud Platform


---

# Abstract


| | |
|---|---|
| Author: | Ilmar Heinonen |
| Title: | Leveraging artificial intelligence for corporate responsibility report retrieval |
| Number of Pages: | 32 pages |
| Date: | 4 April 2024 |
| Degree: | Bachelor of Engineering |
| Degree Programme: | Information and Communication Technology |
| Professional Major: | Mobile Solutions |
| Supervisors: | Hannu Markkanen, Researching lecturer<br>Timo Nykopp, RDI-manager |


---


This thesis focuses on the further development of the commissioner's website. The task of the website is to retrieve corporate responsibility reports using artificial intelligence. The main objective of the thesis is to improve the AI's ability to identify sustainability reports from different sources. In addition, the efficiency of the server and the user interface will be further developed.

A new AI model will be created for text classification, with a larger amount of training material. A comparison of the new and old AI models shows that the newer one performs better. The software's old server scraped web pages, which turned out to be error-prone, as well as less efficient. This also raised ethical concerns about website scraping. The server was upgraded to make use of a search engine. In addition, the user interface was made more adaptive, improving the user experience.

As a result of the thesis, the newer software works much better, but there is room for improvement. The AI model could be trained with more training material, or several AI models could be trained, each specialised in a different report format.



Keywords: artificial intelligence, sustainability report, software development, google cloud platform

# Sisällys

# 1 Johdanto

Vastuullisuusraportointi on tärkeä osa yritysten läpinäkyvyyttä ja yhteiskuntavastuun toteuttamista. Näiden raporttien kerääminen ja analysointi on kuitenkin usein työlästä ja aikaa vievää. Tämän opinnäytetyön toimeksiantaja on Puhtaat ja kestävät ratkaisut, joka on Metropolia Ammattikorkeakoulun innovaatiokeskittymä. Puhtaat ja kestävät ratkaisut tuottaa kestävää kehitystä ja yritystoimintaa tukevia kehityshankkeita, joissa kehitetään uusia ratkaisuja, tekniikoita ja toimintatapoja ympäristölle, ihmisille ja yrityksille.

Toimeksiantaja on kehittänyt verkkosivun, joka hyödyntää tekoälyä vastuullisuusraporttien hakemiseen. Verkkosivustoa kutsutaan nimellä Vastrap. Tämän opinnäytetyön arvo toimeksiantajalle on kehittää Vastrapia siten, että se vastaa paremmin kasvavaan tarpeeseen vastuullisuusraporttien hakemisessa. Vastuullisuusraportoinnin muuttuessa pakolliseksi vuoteen 2025 mennessä tehokkaiden työkalujen merkitys raporttien keräämiseen korostuu entisestään. Aihe valittiin juuri tästä syystä, sillä se antaa toimeksiantajalle kilpailuetua tarjoamalla uusia innovaatioita ja palveluja ajankohtaiseen tarpeeseen.

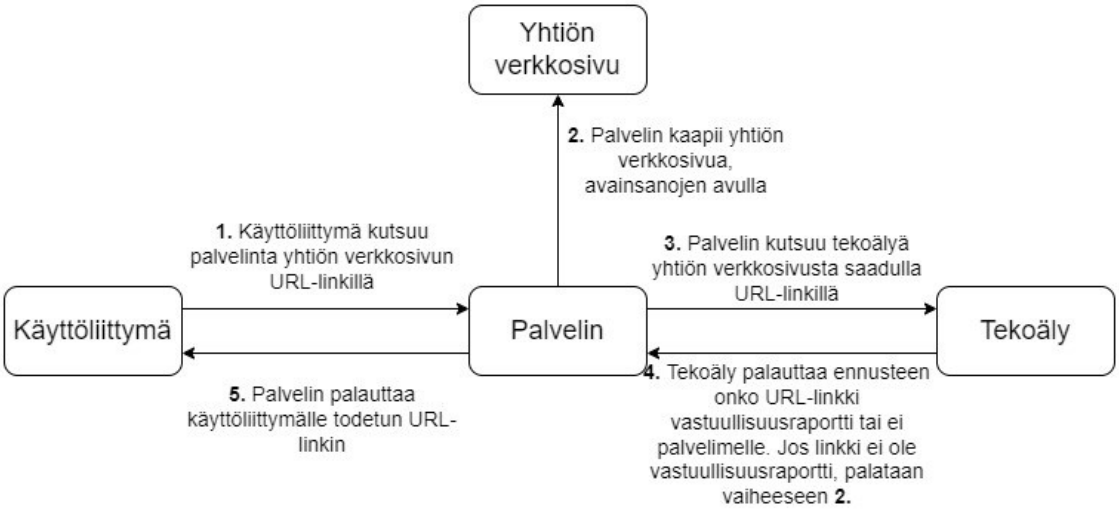
Vastrapin tekoälyn suorituskykyä pyritään parantamaan luomalla uusi tekstinluokittelumalli, joka on koulutettu suuremmalla määrällä koulutusmateriaalia kuin aiempi malli. Palvelimen osalta tavoitteena on siirtyä pois verkkosivujen kaapimisesta. jonka sijaan palvelin integroidaan hyödyntämään hakukonetta tehokkaamman raporttien haun mahdollistamiseksi. Käyttöliittymää myös parannetaan kunnostuksella sekä tekemällä siitä mukautuva.

# 2 Vastrapin ohjelmiston nykytila

Vastrapin tehtävänä on hakea yrityksen vastuullisuusraportti käyttäjän puolesta. Käyttöliittymälle syötetään linkki yrityksen kotisivuille, jonka jälkeen palvelin pyrkii hakemaan vastuullisuusraportin ja palauttamaan sen käyttäjälle. Tämän työn tavoitteena on jatkokehittää Vastrapin ohjelmistoa. Ohjelmistossa nousee esille eri ongelmia, ja ottamatta huomioon tekoälyyn liittyviä puutteita näihin kuuluvat

- loputtomat palvelinsilmukat
- eettiset ja lailliset ongelmat verkkosivun kaapimisen kannalta
- käyttäjäliittymän parantaminen käyttäjäkokemuksen vuoksi.

Vastrapin prosessi koostuu viidestä vaiheesta. Ensin yhtiön verkkosivun URL-osoite syötetään käyttöliittymään, josta tulee kutsu palvelimeen. Toiseksi palvelin kaapii aiemmin syötettyä verkkosivua avainsanojen avulla. Kolmanneksi palvelin kutsuu tekoälyä verkkosivun URL-osoitteella. Neljänneksi tekoäly arvioi, sisältääkö kyseinen linkki vastuullisuusraportin. Viidenneksi, arvion ollessa myönteinen, linkki palautetaan käyttäjälle. Jos arvio on kielteinen, palvelin jatkaa kaapimista verkkosivulla avainsanojen avulla ja palauttaa uuden URL-osoitteen tekoälylle. Tämä jatkuu, kunnes palvelin on tehnyt joko liian monta kutsua, jäänyt loputtomaan silmukkaan koodivirheiden takia tai palauttaa oikean vastauksen. Kuvassa 1 kuvataan ohjelmiston prosessi vastuullisuusraporttia hakiessa.




Kuva 1. Ohjelmiston prosessi vastuullisuusraporttia hakiessa.

Vastrapin tekoäly

Tekoäly on vastuullisuusraportin hakemisen tärkein osa, minkä takia verkkosivuston jatkokehityksen tärkein tehtävä on optimoida ja parantaa sitä. Tekoäly on rakennettu Vertex AI:n AutoML-tekstinluokituksen avulla. Vertex AI - ja AutoML käsitteet määritellään tarkemmin luvussa 4. Vastuullisuusraportin arvioimiseen voitaisiin tehdä monia eri koneoppimismalleja, kuten kuvanluokittelumalleja. Kyseinen malli vaatisi palvelimen, jonka avulla tekoälymalli saisi kuvankaappauksia verkkosivuista. Tekstin luokittelu on kuitenkin näppärin ja tehokkain ratkaisu tähän ongelmaan. Kuvien luokittelu on kalliimpaa ja hitaampaa kuin tekstin sekä hankalampi toteuttaa palvelimessa. Vuonna 2024 kuvanluokittelumallin tuntihinta on 1,375 $, ja tekstinluokittelumallin käytössä tuntihinta on 0,05 $ [1].

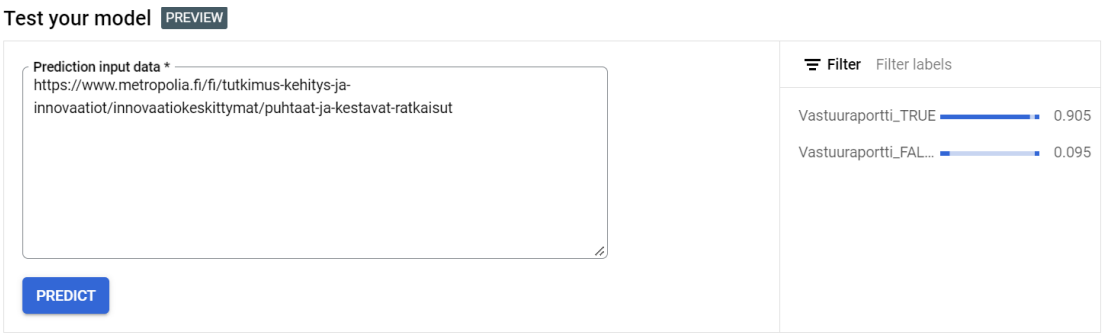
Kuvan luokittelulla toimiakseen Vastrap vaatii uudenlaisen ohjelmiston. Ohjelmiston tulisi toimittaa vastuullisuusraportin hakemiseen tarvittavia verkkosivuston näyttökuvia tekoälylle. Tekstin luokitteluun tekoäly tarvitsee vain verkkosivun URL-osoitteen tai verkkosivun tekstitiedot. Verkkosivujen tekstitietojen kaapimisessa tulee esiin sekä laillisia että eettisiä kysymyksiä, joita on tärkeää ottaa huomioon jo palvelimen suunnitteluvaiheessa. Esimerkiksi kaapiminen voi

mahdollisesti rikkoa tekijänoikeuslakeja, mikäli verkkosivun materiaali on tekijänoikeudella suojattua. Tämän ja tehokkuuden vuoksi hyödynnetään URL-osoitteita.

Vastrapin nykyisessä tekoälymallissa on puutteita. Malli on koulutettu 141 linkillä, joista noin 70 % on vastuullisuusraportteja ja 30 % ei. Parhaimman tuloksen saamiseksi koulutusmateriaalin pitäisi olla tasapainossa oikeiden ja väärien vastauksien suhteen. Lisäksi Googlen dokumentaatio painottaa, että laadukkaamman datan käyttö mahdollistaa tarkemman tekoälymallin saavuttamisen. [2.] Tekoälymallin tarkkuus on 85,7 %. Kuvissa 2 ja 3 näkyy, miten malli on usein väärässä, erityisesti pitkissä linkeissä. Kuvassa 2 tekoälylle on syötetty satunnainen pitkä URL-linkki. Vaikka kyseinen linkki ei ole vastuullisuusraportti, tekoäly on 84-prosenttisen varma, että linkki on vastuullisuusraportti.


Kuva 2. Mallin testaaminen satunnaisella linkillä [3].

Kuvassa 3 esitetään sama ongelma, mutta pidemmällä linkillä. Kuvassa korostuu ongelma enemmän, ja tekoäly on 90-prosenttisen varma, että satunnainen linkki olisi vastuullisuusraportti.

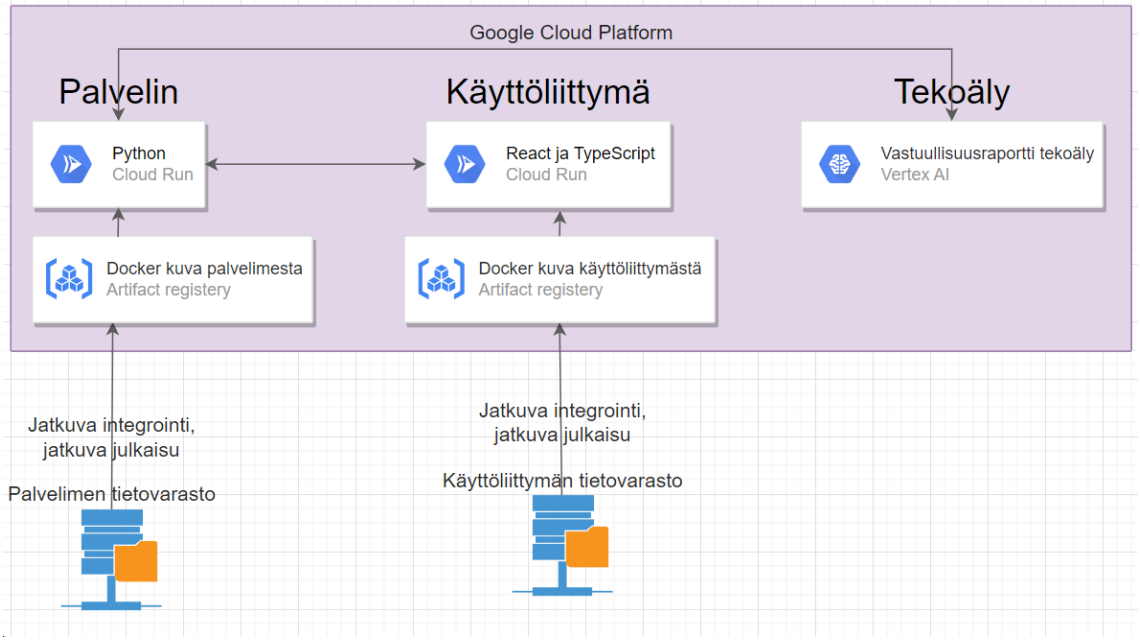



Kuva 3. Mallin testaaminen satunnaisella linkillä [3].

Koska tekoälylle syötetään vain linkkejä, jotka liittyvät vastuullisuusraportteihin, väärät positiiviset ennustukset eivät ole suuri ongelma. Käyttäjän kannalta ei ole kovin tärkeää, mikäli tekoäly antaa väärän positiivisen tuloksen, sillä tekoälyn käsittelemät URL-linkit liittyvät aina jollain tavalla vastuullisuusraportteihin. Kuitenkin on tärkeää kouluttaa tekoälyä olemaan oikeassa suurimman osan ajasta ja minimoimaan virheelliset tulokset. Tämä luo käyttäjälle luottamusta ohjelmistoon ja parantaa käyttäjäkokemusta.

Vastrapin ohjelmisto koostuu käyttöliittymästä, palvelimesta ja tekoälystä. Palvelimen sekä käyttöliittymän lähdekoodit ovat omissa tietovarastoissa (engl. repository). Varastoissa on omat jatkuvat integrointi- ja julkaisuputket, jotka automatisoivat koodin julkaisun. Käyttöliittymä on ohjelmoitu käyttäen Reactia ja TypeScriptiä. Palvelin on ohjelmoitu käyttäen Pythonia, tekoäly on koulutettu hyödyntäen Vertex AI:n AutoML-prosessia ja ohjelmisto on julkaistu GCP:n (Google Cloud Platform) avulla. Käyttöliittymä ja palvelin on säiliöity (engl. containerization) Dockerin avulla. Kuvassa 4 näkyy koko verkkosivuston ohjelmiston

arkkitehtuuri.




Kuva 4. Verkkosivuston ohjelmiston arkkitehtuuri.

Parannukset

Vastrapin parannuksen kohteet ovat palvelin, käyttöliittymä ja tekoäly. Tekoälymallia tehostetaan kouluttamalla sitä laajemmalla aineistolla kuin aikaisempaa mallia. Uuden aineiston käyttäminen parantaa tekoälymallin tunnistuskykyä. Palvelimen koko nykyinen funktionaalisuus muutetaan. Nykyinen toimintamalli kaapii syötetyn verkkosivun tiedot ja palauttaa linkin tekoälylle. Uusi palvelin tulee hyödyntämään hakukonetta ja avainsanaa. Näiden ominaisuuksien avulla palvelin etsii hakukoneesta tulokset, jotka palautetaan tekoälylle. Käyttöliittymän uudessa versiossa käyttäjälle palautetaan hakutuloksia viidestä olennaisimmasta osumasta. Lisäksi osumiin lisätään tekoälyn ennustus siitä, mikä linkki on todennäköisimmin vastuullisuusraportti. Käyttöliittymään lisätään myös mukautuvuus eri laitteille ja käyttäjäkokemusta parannetaan kokonaisvaltaisesti visuaalisilla muutoksilla.

# 3 Koneoppiminen

Koneoppiminen on tekoälyn osa-alue. Tässä kappaleessa keskitytään vain koneoppimisen osiin, jotka ovat tarpeellisia ymmärtää Vastrapin tekoälyn kannalta.

## 3.1 Valvottu koneoppiminen

Valvotussa koneoppimisessa (engl. supervised learning) syötetään merkittyä harjoitusaineistoa algoritmille. Merkitty harjoitusaineisto on luokiteltua dataa. Esimerkiksi jos tehtäisiin koneoppimismallia, jonka tehtävä on luokitella kissa- ja koirakuvat erikseen, mallille syötettäisiin kuvia kissoista ja koirista, jotka on merkitty oikeilla vastauksilla. Tällaisen koulutuksen jälkeen malli pyrkisi erottamaan uudet kuvat kissojen ja koirien väliltä. [4, s. 1–3.]

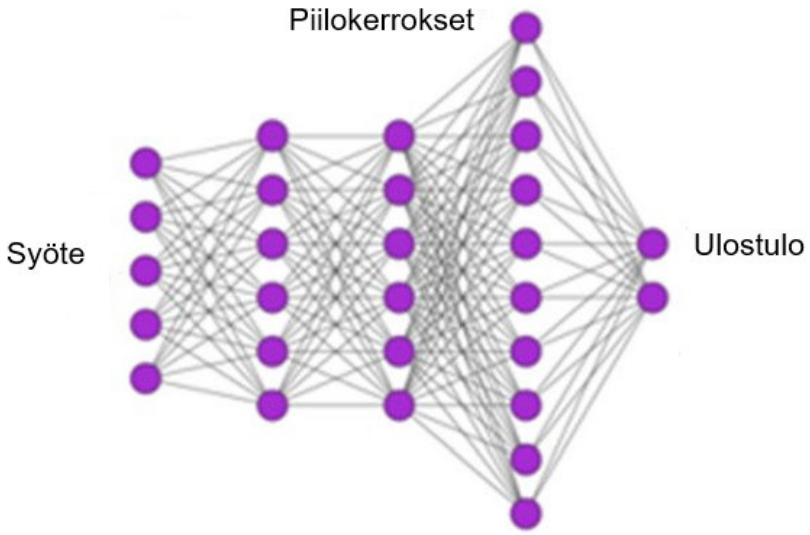
Valvotussa koneoppimisessa on erilaisia algoritmeja vaihtoehtoina, mutta suosituimmat ovat regressio ja luokitus. Regressioalgoritmit ennustavat jatkuvaa arvoa. [4, s. 137.] Vastrapin tekoälyssä ei ole tälle tarvetta, joten siinä hyödynnetään vain binääristä eli kahden ryhmän luokitusalgoritmia. Luokitusalgoritmeja käytetään datan ryhmittelyyn. Vastrapissa tämä tarkoittaa sitä, että data luokitellaan sen perusteella, onko se vastuullisuusraportti vai ei. [5.]

## 3.2 Neuroverkot

Neuroverkkojen (engl. neural network) terminologia ja implementaatio ovat saaneet vaikutteita biologisista järjestelmistä. Neuroverkot jäljittelevät biologisten neuronien välistä kommunikaatiomekanismia. [4, s. 226.]

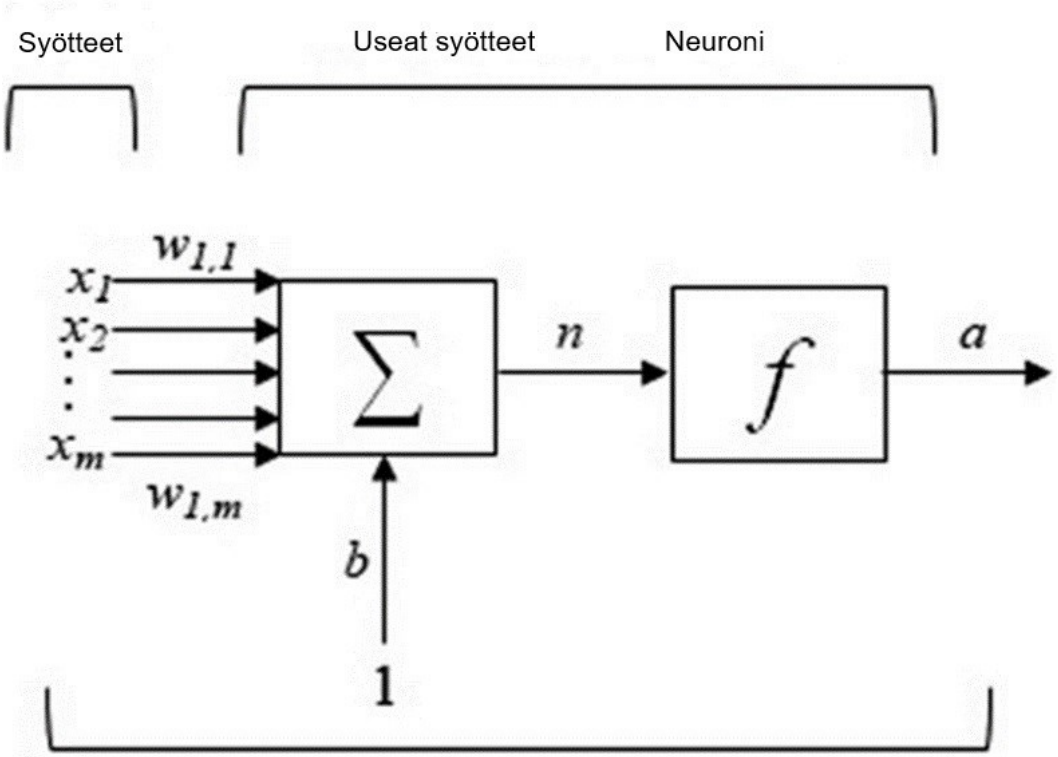
Neuroverkot koostuvat monesta tasosta, ja tasot koostuvat keinotekoisista neuroneista. Neuroverkon tasot koostuvat syötekerroksesta (engl. input layer), piilokerroksista (engl. hidden layer) ja ulostulokerroksesta (engl. output layer). Neuroverkko, jossa on enemmän kuin kolme tasoa, voidaan luokitella

syväoppimisalgoritmiksi. Kuvassa 5 visualisoidaan neuroverkon arkkitehtuuri.




Kuva 5. Syväoppimisneuroverkon arkkitehtuuri [6].

Neuronin vastaanottaessa signaaleja se laskee signaaleiden painotetut arvot yhteen, summaa lukuun harhan ja välittää tuloksen aktivointifunktiolle. Painotetut arvot koostuvat painoarvosta (engl. weight), jonka tehtävä on kertoa neuronille, kuinka tärkeä syötetty tieto on. Neuroni vastaanottaa tietoa muilta neuroneilta joihin se on yhteydessä. Jokaisella näillä yhteyksillä on oma painoarvonsa. Mitä suurempi painoarvo neuronin syötteessä on, sitä suurempi vaikutus sillä on neuronin laskettuun ulostuloon. [6; 7.] Kuvassa 6 visualisoidaan neuronin rakenne.




Kuva 6. Neuronin rakenne [8, s. 230].

Kuvassa 6 sisääntulosyötteet voidaan määritellä vektoreina $X_1$, $X_2$, …, $X_m$ ja painot matriiseina $W_{1,1}$, …, $W_{1,m}$. Esimerkiksi $X_1$-sisääntulosyöte painotetaan vastaavalla $W_{1,1}$-painolla. Painon ensimmäinen indeksi osoittaa, mikä on kyseisen painon seuraava neuroni ja toinen indeksi edustaa vastaanottavan neuronin signaalilähdettä. Neuronilla on lisäksi harha $b$, joka summautuu painotettuihin syötteisiin. Aktivointifunktiota kuvataan symbolilla $f$. [8, s. 225–250.]

Harha

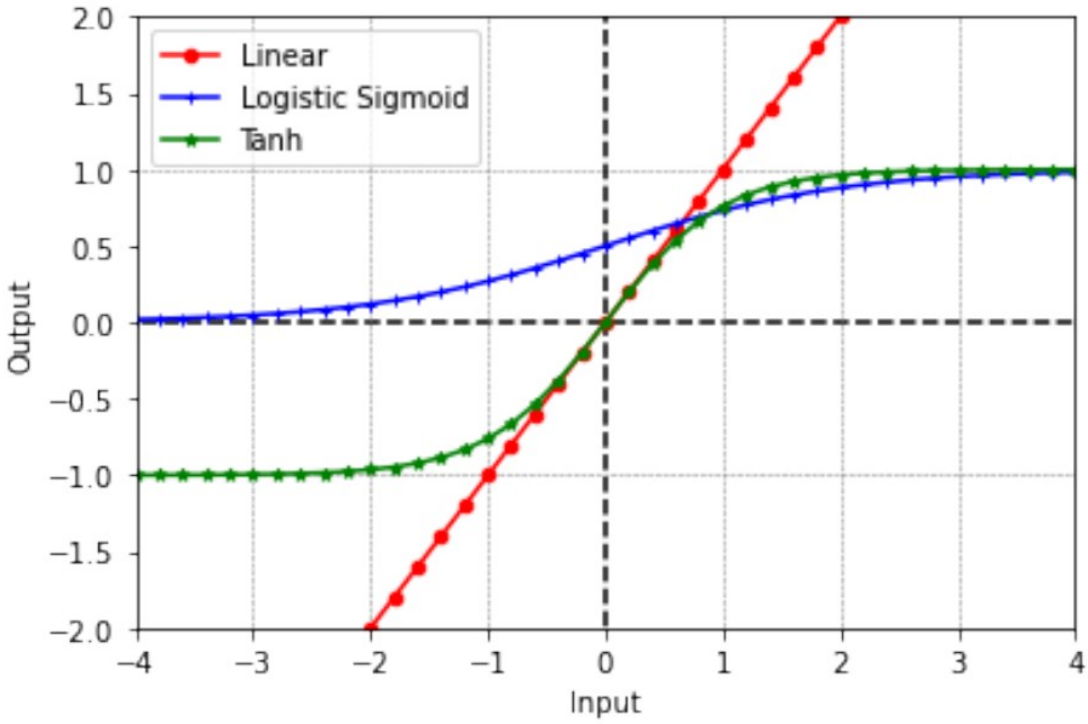
Harha on vakioluku, joka lisätään neuronin painotettujen syötteiden summaan. Yhtälössä 1 esitetään, kuinka harha voidaan lisätä painoarvoon. Yhtälössä $b$ esittää harhaa. [8, s. 225–250.]

$$n = w_{1,1}x_1 + w_{1,2}x_2 + \cdots w_{1,m}x_m + b \quad (1)$$

Harhan tehtävä on antaa joustavuutta neuroverkon toimintaan. Se tekee tämän muuttamalla neuronin ulostuloarvon vahvuutta. Kun positiivinen harha lisätään painotettuun arvoon, neuronin ulostuloarvo on vahvempi. Jos harha on negatiivinen, neuronin ulostuloarvo on heikompi. Harhan arvot päivitetään ja optimoidaan koulutusprosessin aikana. Ilman harhaa neuroverkko voisi tehdä hyviä ennusteita vain niille syötteille, jotka ovat lähellä nollaa. Harhan käyttö parantaa neuroverkon kykyä oppia ja tehdä tarkkoja ennusteita laajalle joukolle syötteitä. [6; 7.]

## 3.3 Aktivointifunktio

Aktivointifunktion tarkoituksena on ottaa painotettu summa ja laskea siitä ulostuloarvo. Aktivointifunktioita on monia, ja niitä käytetään eri tarkoituksiin. Sigmoidifunktio rajoittaa ulostuloarvon 0–1 välille, mikä on hyödyllistä luokitteluongelmissa. Lineaarinen aktivointifunktio ei muuta ulostuloarvoa, minkä takia sitä ei usein hyödynnetä kompleksissa neuroverkoissa. Tanh rajaa ulostuloarvon välille -1–1, mikä mahdollistaa laajemman eron gradienttimenetelmän laskennassa. Kuvassa 7 visualisoidaan mainitut aktivointifunktiot. Kuvassa x-akseli kuvaa aktivointifunktion sisäänsyöttöä ja y-akseli ulossyöttöä. [9.]



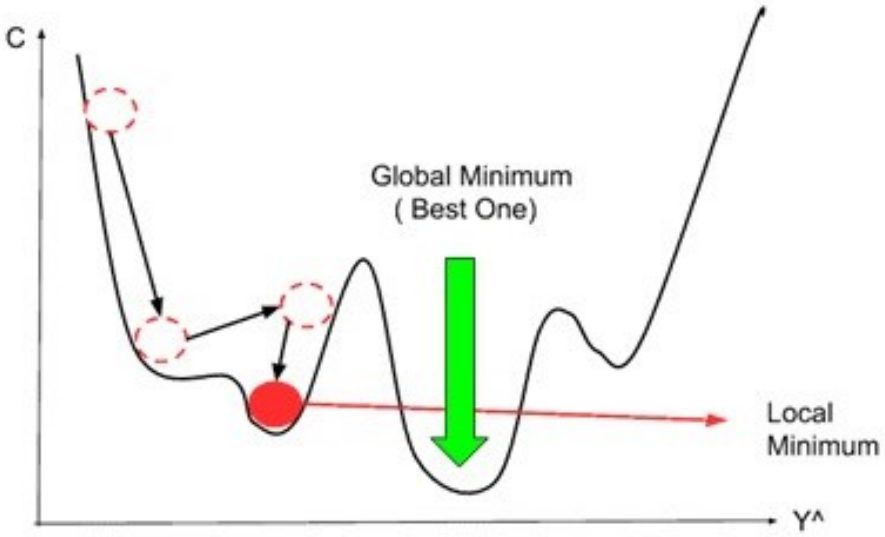

Kuva 7. Lineaarinen aktivointifunktio, sigmoidi-aktivointifunktio ja tanh-aktivointifunktio visualisoituna [9].

## 3.4 Kustannusfunktio ja gradienttimenetelmä

Tekoälymallin koulutuksen aikana malli vertaa ennustettua vastausta oikeaan vastaukseen. Kustannusfunktio laskee eron näiden kahden arvon välillä. Esimerkki kustannusfunktion toiminnasta, on kun malli antaa ennustuksen jossa tietty teksti luokitellaan vastuullisuusraportiksi 60 prosentin todennäköisyydellä. Seuraavaksi kustannusfunktio arvioi mallin tarkkuutta vertaamalla ennustettua arvoa oikeaan vastaukseen. [10; 4, s. 41.]

Gradienttimenetelmä on algoritmi, jonka avulla voidaan saada mahdollisimman pieni tulos kustannusfunktiosta. Gradienttimenetelmä toimii iteratiivisesti. Algoritmi valitsee satunnaisen arvon josta aloittaa, ja laskee annetun pisteen negatiivisen kaltevuuden. Negatiivinen kaltevuus osoittaa, missä kustannusfunktion arvo laskee, eli mallin ennustaminen parantuu. Iterointien jälkeen algoritmi on löytänyt joko paikallisen minimin (engl. local minima) tai globaalin minimin (engl.

global minima). [11.] Kuvassa 8 visualisoidaan paikallisen minimin ja globaalin minimin ero ja mitä ongelmia gradienttimenetelmämetodi voi tuoda esiin.




Kuva 8. Gradienttimenetelmä piirrettynä, jossa näkyy paikallisen minimin sekä globaalin minimin ero [12].

Kuten kuvasta 8 voidaan huomata, gradienttimenetelmässä voi tulla ennustuksia, jotka eivät ole parhaimpia mahdollisia. Gradienttimenetelmän iteroinnin aikana alfa-arvo tai oppimisnopeus on arvo, jonka avulla algoritmi päättää, kuinka suuria askelia se ottaa laskennan aikana. Korkeat arvot nopeuttavat laskentaa, mutta lisäävät riskiä minimin ohi menemiseen. Pienet arvot saattavat hidastaa laskentaa tai jäädä jumiin paikalliseen minimiin. Gradienttimenetelmistä on eri versioita, joilla pyritään korjaamaan nämä virheet, kuten alajoukkogradienttimenetelmä (engl. mini-batch gradient descent). Alajoukkogradienttimenetelmässä erotellaan kouluttamisaineisto pienempiin joukkoihin, joihin erikseen suoritetaan gradienttimenetelmä. Näin gradienttimenetelmä löytää globaalin minimin isommalla todennäköisyydellä. [10; 7.]

## 3.5 Mallin arviointimittarit

### Tarkkuus

Tarkkuus (engl. precision) on mitta, joilla pyritään arvioimaan, kuinka tarkkoja klassifiointitekoälymallit ovat. Tarkkuuden tehtävänä on osoittaa, kuinka tarkasti malli ennustaa positiivisia arvoja. Korkea tarkkuus mallissa tarkoittaa, että malli ennustaa vähemmän virheellisiä positiivisia tuloksia. Tarkkuus lasketaan jakamalla oikein ennustetut positiiviset oikeiden positiivisien ja virheellisien positiivisien summalla. [13.]

$$Tarkkuus = \frac{OP}{OP+VP} \quad (2)$$

### Herkkyys

Herkkyys (engl. recall) on toinen tärkeä suorituskyvyn mittari koneoppimismalleissa. Herkkyyden avulla voidaan kuvata, kuinka hyvin malli pystyy tunnistamaan positiiviset tapaukset oikein. Korkea herkkyys mallissa kertoo, että malli ennustaa vähemmän virheellisiä negatiivisia tai vähemmän ennustuksia jää tekemättä. Herkkyys lasketaan jakamalla oikein ennustetut positiiviset oikeiden positiivisien ja virheellisien negatiivisien summalla. [13.]

$$Herkkyys = \frac{OP}{OP+VN} \quad (3)$$

### F-arvo

F-arvossa (engl. F-measure) hyödynnetään tarkkuutta sekä herkkyyttä. F-arvo on yleisesti käytetty mittari mallin arvioinnissa. F-arvo antaa tasapainon tarkkuuden ja herkkyyden välillä, mikä on hyödyllinen, jos virheelliset positiiviset sekä virheelliset negatiiviset pitää ottaa huomioon. F-arvo lasketaan hyödyntämällä tarkkuuden ja herkkyyden painotettua harmonista keskiarvoa, kuten on kuvattu yhtälössä 4.

$$F_\beta = (1 + \beta^2) \cdot \frac{tarkkuus \cdot herkkyys}{(\beta^2 \cdot tarkkuus) + herkkyys} \quad (4)$$

Yhtälössä 4 kuvatun $\beta$:n yleisin arvo on 1, joka tunnetaan F1-mittana. [13.]

# 4 Vertex AI

Vertex AI on koneoppimisalusta GCP:ssä eli Google Cloud Platformissa. Vertex AI helpottaa tekoälyn luomista ja käyttöönottoa. GCP taas mahdollistaa tekoälyn ja käyttöliittymän käyttöönoton verkossa.

Vertex AI tarjoaa kahta eri vaihtoehtoa tekoälyteknologia mallien kouluttamiseen, AutoML ja custom training. AutoML-algoritmit ja koulutusmenetelmät valikoituvat neuroarkkitehtuurihaun (engl. neural architecture search) avulla. AutoML tarjoaa tavan kouluttaa neuroverkkoja ilman koodia. Tämän takia AutoML:n käytössä on rajoituksia, esimerkiksi mallien on oltava suunnattuja ennalta määriteltyihin tavoitteisiin. Custom training tarjoaa käyttäjälle mahdollisuuden hyödyntää omia koneoppimiskirjastoja ja ohjelmistokehyksiä. Tämän seurauksena käyttäjällä on enemmän vapautta kouluttaa tekoälymallia ja räätälöidä se yksilölliseksi. Custom training kuitenkin vaatii merkittävää vaivannäköä ja aikaa. [14; 15.] Taulukossa 1 näkyy AutoML:n sekä custom trainingin ero Vertex AI:ta hyödynnettäessä.

Taulukko 1. AutoML:n ja custom trainingin ero.

| | **AutoML** | **Custom training** |
|---|---|---|
| Datatieteen asiantuntemus | Ei vaadita. | Vaaditaan koulutussovelluksen kehittämiseen ja myös |

| | | |
|---|---|---|
| | | joidenkin tietojen valmisteluun, kuten ominaisuuksien suunnitteluun. |
| Ohjelmointikyky | Ei tarvita, AutoML on kooditon. | Tarvitaan koulutussovelluksen kehittämiseen. |
| Koulutettuun malliin kuluva aika | Säästää aikaa. Datan valmistelu on vähäisempää, eikä ohjelmistokehitystyötä tarvita. | Vie enemmän aikaa kuin AutoML. Datan käsittelyä on lisättävä ja ohjelmistokehitystyötä vaaditaan. |
| Koneoppimisen tavoitteita koskevat rajoitukset | Rajoitettu. On tähdättävä johonkin AutoML:n ennalta määritettyyn tavoitteeseen. | Ei rajoitteita. |
| Mallin suorituskyvyn optimointi hyperparametreillä | Ei voida optimoida manuaalisesti hyperparametrien viritysellä. AutoML tekee jonkin verran automaattista hyperparametrien viritystä, mutta käytettyjä arvoja ei voida muuttaa. | Voidaan optimoida manuaalisesti hyperparametrien viritysellä. Mallia voidaan virittää jokaisen harjoitusajon aikana kokeiluja ja vertailua varten. |

| | | |
|---|---|---|
| Koulutusympäristön osa-alueet | Rajoitettu. Kuva- ja taulukkomuotoisten tietokokonaisuuksien osalta voidaan määrittää koulutettavien solmutuntien lukumäärä. Lisäksi voidaan päättää, sallitaanko koulutuksen aikainen lopettaminen. | Ei rajoitteita. Voidaan määrittää ympäristön osatekijöitä, kuten Compute Engine -koneen tyyppi, levyn koko, koneoppimiskehys ja solmujen määrä. |
| Tietojen kokorajoitukset | Rajoitettu. AutoML käyttää hallittuja tietokokonaisuuksia; tietokokorajoitukset vaihtelevat tietokokonaisuuden tyypin mukaan. | Hallitsemattomien tietokokonaisuuksien osalta ei rajoitteita. Hallituilla tietokokonaisuuksilla on samat rajoitukset kuin Vertex AI:ssa luoduilla tietokokonaisuusobjekteilla, joita käytetään AutoML mallien kouluttamiseen. |

AutoML hyödyntää algoritmeja ja työkaluja, jotka piilottavat tekoälymallin yksityiskohdat käyttäjältä. Tämän vuoksi yksityiskohtia mallista ei voida tietää, kuten mitä kustannusfunktiota käytetään. Tästä huolimatta AutoML:n rajoitukset ovat minimaaliset Vastrapin tekoälyn kontekstissa, ja tekoäly voidaan rakentaa AutoML:ää hyödyntäen. AutoML:n käyttö nopeuttaa tekoälyn kehitys- ja suunnitteluvaiheita. [15.]

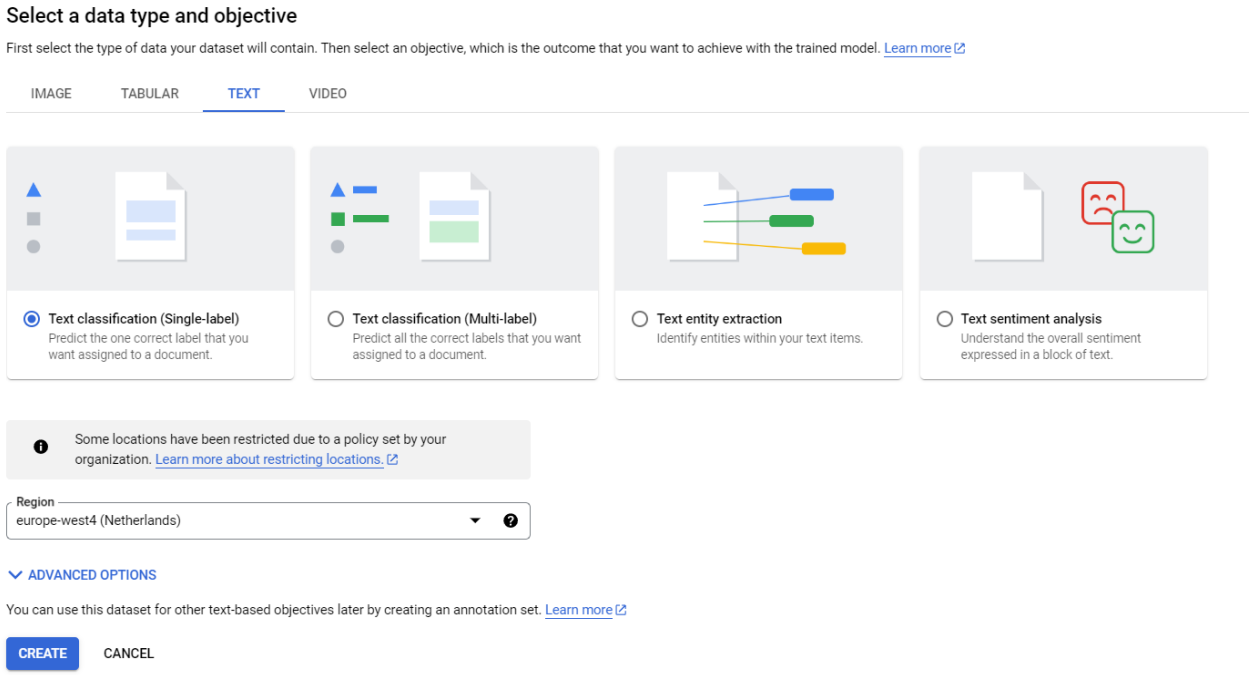
AutoML tukee taulukko-, kuva-, video- ja tekstidataa. Kuten luvussa 2 on käsitelty, verkkosivuston tekoäly tulee käsittelemään vain tekstiä. Mallin voi kouluttaa luokittelemaan tekstidataa kategorioihin, poimimaan tietoa tekstistä tai tunnistamaan tekstin sisältämiä tunteita. Tekstidatan luokitus sopii täydellisesti verkkosivustolle luokittelemaan onko tekstidata vastuullisuusraportti vai ei.

Vertex AI:n avulla voidaan saada tekstipohjaisista malleista verkkoennusteita ja eräennusteita. Verkkoennusteet ovat synkronisia pyyntöjä, jotka tehdään mallin ohjelmointirajapinnan osoitteeseen reaaliajassa. Niitä käytetään tilanteissa, joissa tarvitaan välittömiä vastauksia. Eräennusteet ovat asynkronisia pyyntöjä, jotka tehdään suoraan malliresurssista ilman, että malli tarvitsee ohjelmointirajapinnan osoitetta. Eräennusteet ovat hyödyllisiä, kun ei tarvita välitöntä vastausta ja halutaan käsitellä suuria määriä dataa yhdellä pyynnöllä. [14.]

# 5 Ohjelmistokehitysprosessi

## 5.1 Vertex AI AutoML -mallin tekeminen

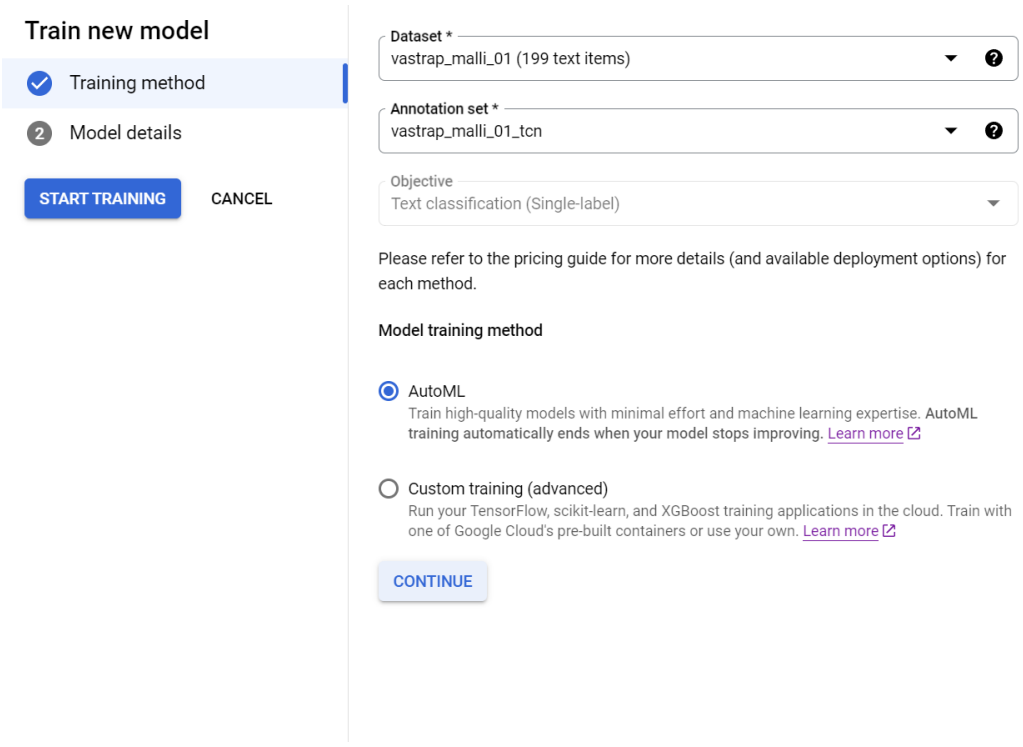
AutoML -mallin tekeminen on suoraviivaista GCP:ssä. Ennen mallin luomista, malli tarvitsee tietokokonaisuuden (engl. dataset). Tietokokonaisuuden luomisessa valitaan, halutaanko kuva-, tabulaari-, video- vai tekstitietoa. Jokaisessa kategoriassa on omat alakategoriat, jotka ovat AutoML:n tavoitteita koskevat rajoitukset. Kuvassa 9 näkyy tekstitietokokonaisuuden alakategoriat ja muut asetukset, joita voidaan asettaa tietokokonaisuuden luomisessa.




Kuva 9. Tekstitietokokonaisuuden alakategoriat [3].

Tietokokonaisuuden luomisen jälkeen kokonaisuus pitää täydentää tiedoilla ja luokitella tämä tieto. Uuden tekoälymallin kouluttamiseen on kerätty 199 linkkiä: 100 oikeaa, ja 99 väärää. Vanhassa mallissa käytettiin 100 oikeaa, ja 50 väärää linkkiä. Kaikki käytetyt linkit ovat erilaisten yhtiöiden verkkosivuilta.

Tietokokonaisuuden luomisen jälkeen voidaan kouluttaa tekoälymalli. Kuvassa 9 tekstitietokokonaisuuden alakategorioiden lisäksi näkyy tekstiin liittyvät tekoälymallikategoriat. Tähän kuuluu tietokokonaisuus, tietokokonaisuuden luokittelu ja mallin kouluttamisen menetelmä. Kuvassa 10 näkyy eri vaihtoehdot tekoälymallin luomisikkunasta. Luomisikkunassa valitaan aikaisemmin tehty tietokokonaisuus ja merkinnät siitä, mitkä linkit ovat vastuullisuusraportteja ja mitkä eivät.



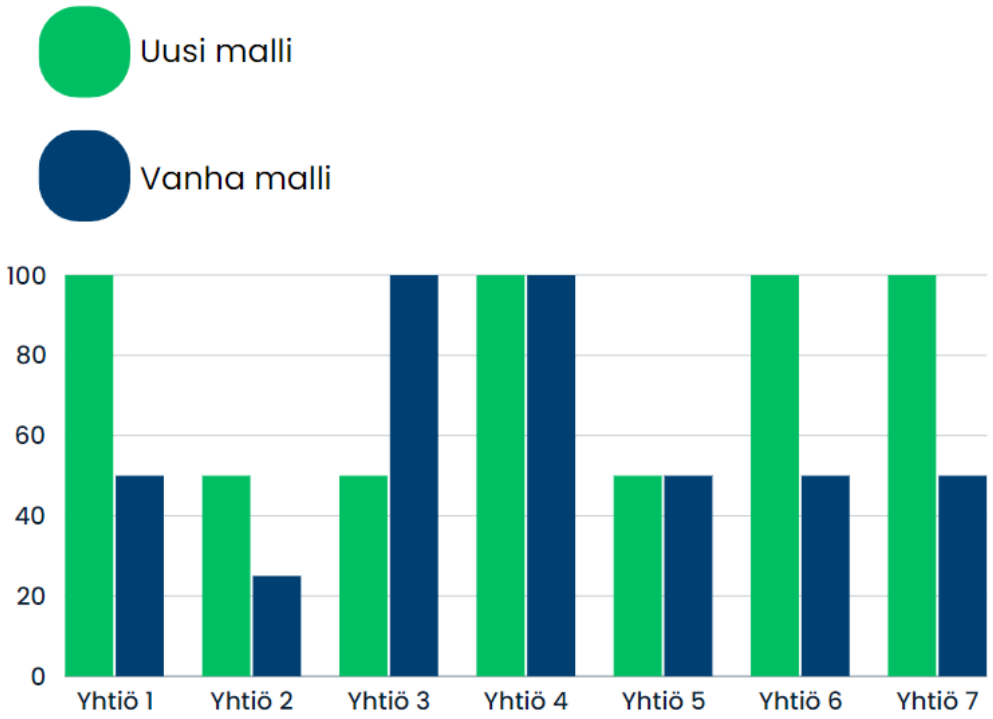

Kuva 10. Tekoälymallin luomisikkuna [3].

Mallin kouluttamiseen menee usein monia tunteja. Kun malli on koulutettu, sen arviointimittareita, tarkkuutta tai herkkyyttä, voidaan tarkastella. Eräennustuksia voidaan myös tehdä, mutta synkronisia tai yksittäisiä ennustuksia ei, sillä mallilla ei ole ohjelmointirajapinnan osoitetta. Ohjelmointirajapinnan osoite luodaan

mallin Deploy & Test -osiosta. Osiossa voidaan luoda ohjelmointirajapinnan osoite tai julkaista malli valmiiseen osoitteeseen. Lisäksi voidaan määritellä ohjelmointirajapinnan yksityisyys. Rajapinnan luomisen jälkeen mallia voidaan kutsua synkronisilla tai yksittäisillä kutsuilla.

## 5.2 Mallien vertailu

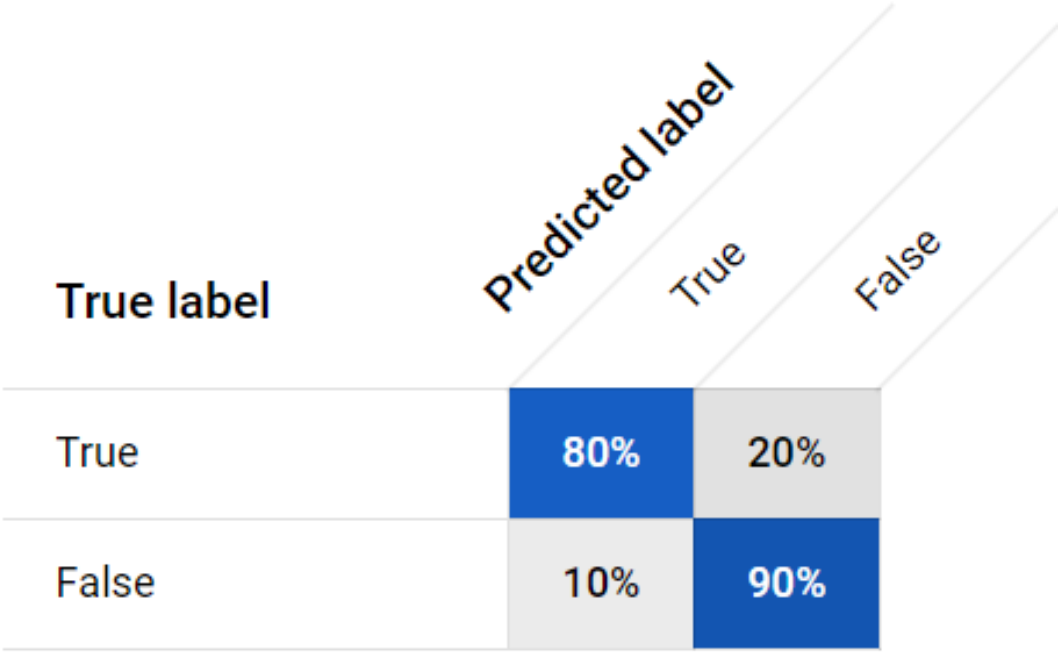
Uusi tekoälymalli on koulutettu suuremmalla ja merkityksellisemmällä materiaalilla kuin vanha tekoälymalli. Vanhempi malli kärsi huonoista koulutusmateriaaleista, minkä takia se suoriutui huonosti. Kuvassa 11 esitetään uuden ja vanhan tekoälymallin tehokkuus, kun molemmille syötetään viisi olennaisinta hakutulosta. Kuvassa jäljitellään uuden palvelimen toimintaa, joka käyttää hakukonetta palauttamaan viisi oleellisinta hakutulosta tekoälylle. Hakutulos koostuu yhtiön nimestä ja vastuullisuusraportti avainsanasta. Kuvassa mallit on pisteytetty sen perusteella, kuinka relevantti mallin vastaus on verrattuna oikeaan vastaukseen.




Kuva 11. Kaavio mallien vertailusta.

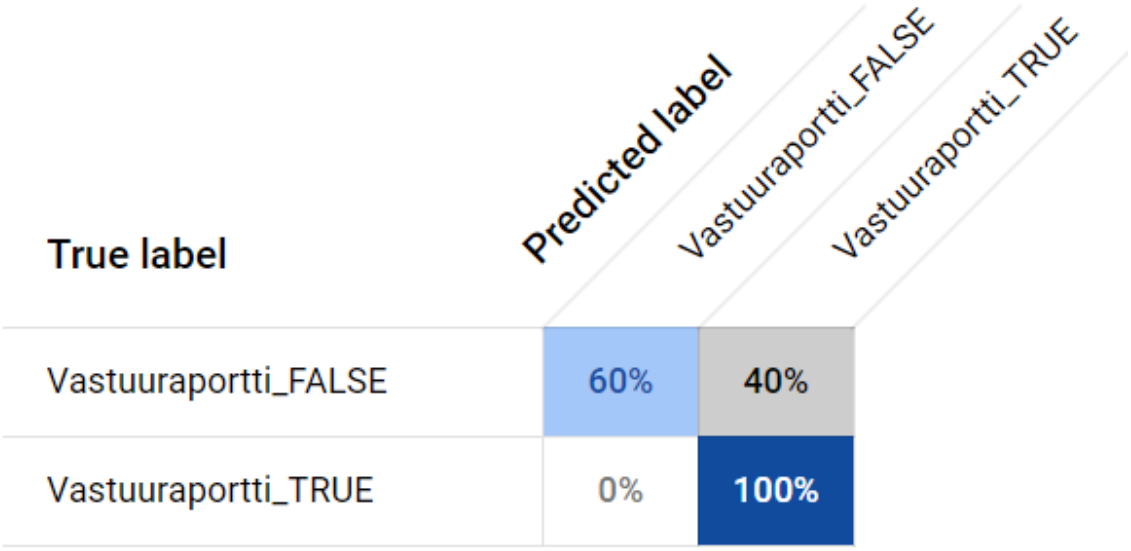
Kuvan 11 pylväsdiagrammi esittää mallien suoriutumisen satunnaisesti valituissa yhtiöissä. Pisteet 0–100 kertovat, kuinka hyvin malli suoriutui. 100 pistettä tarkoittaa, että malli onnistui täydellisesti. 50 pistettä tarkoittaa, että malli ottaa toiseksi parhaimman tuloksen, esimerkiksi vanhemman vuoden vastuullisuusraportin. 25 pistettä tarkoittaa, että malli ottaa kolmanneksi parhaimman tuloksen, esimerkiksi blogikirjoituksen, jossa kerrotaan yhtiön julkaisseen vastuullisuusraportin.

Vastrapin tekoälymallit ovat seuranneet yleisiä koulutusmateriaalin jakauksia. Koulutusmateriaali jaetaan 80 prosenttia koulutukseen, 10 prosenttia validointiin ja 10 prosenttia testaamiseen. Googlen konsolista voi tarkastella testiaineistosta luotua sekaannusmatriisia (engl. confusion matrix). Sekaannusmatriisi on työkalu, joka vertaa todellisia arvoja mallin ennustettuihin arvoihin. Esimerkiksi malli voi ennustaa oikean vastuullisuusraportin vääräksi, mikä ilmenee sekaannusmatriisissa, helposti visualisoituna. Kuvissa 12 ja 13 esitetään vasemmalla puolella oikeiden vastauksien tosi ja epätosi. Kuvien yläosassa esitetään tekoälymallin ennustetut arvot. Uudempi malli suoriutuu paremmin testiaineistossa, kuten kuvien 12 ja 13 sekaannusmatriiseista ilmenee. Kuvassa 12 uusi tekoälymalli ennustaa oikeista vastuullisuusraporteista 80 prosenttia oikeaksi ja 20 prosenttia vääräksi. Vääristä vastuullisuusraporteista se ennustaa 10 prosenttia oikeaksi ja 90 prosenttia vääräksi.




Kuva 12. Uuden tekoälymallin sekaannusmatriisi [3].

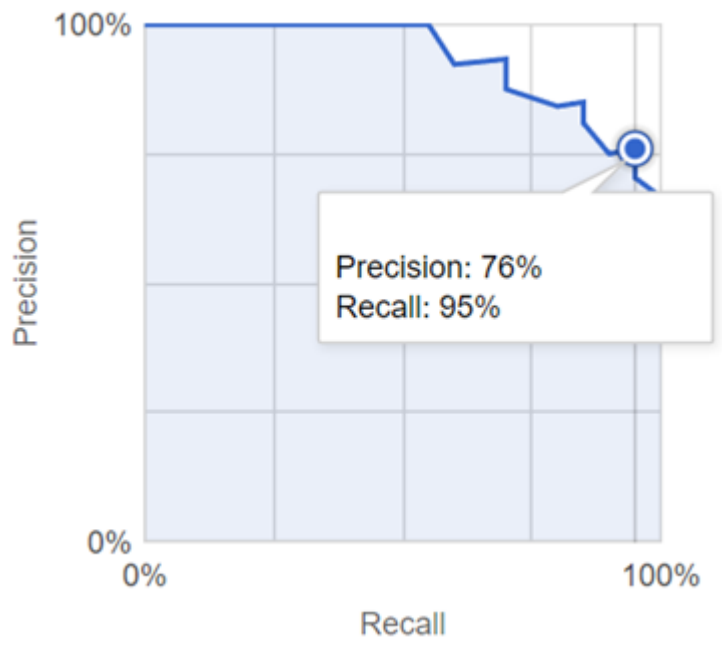
Kuvassa 13 vanha tekoälymalli ennustaa oikeista vastuullisuusraporteista 60 prosenttia oikeaksi ja 40 prosenttia vääräksi. Vääristä vastuullisuusraporteista se ennustaa 0 prosenttia oikeaksi ja 100 prosenttia vääräksi. Vanha tekoälymalli suoriutuu uutta tekoälymallia paremmin väärien vastuullisuusraporttien seulonnasta. Tulokset voivat kuitenkin olla yksittäisiä havaintoja, koska vanhan mallin testidata on pieni.




Kuva 13. Vanhan tekoälymallin sekaannusmatriisi [3].

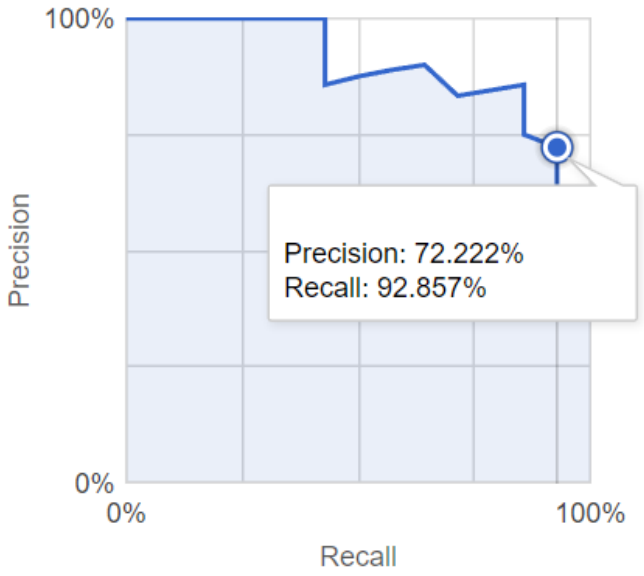
Sekaannusmatriisissa siniset solut osoittavat oikean ennusteen. Tässä yhteydessä sekaannusmatriisi ei anna kokonaista kuvaa mallista käytännössä. Tämä johtuu siitä, että vastauksen ei tarvitse olla täysin oikea, jotta se voitaisiin palauttaa käyttäjälle. Voidaan kuitenkin todeta, että uudempi malli, joka on koulutettu laajemmalla ja merkityksellisemmällä koulutusaineistolla, suoriutuu yleisesti paremmin.

Uuden verkkosivuston palvelimen avulla käyttäjälle palautetaan viisi erilaista linkkiä. Linkeistä käyttäjä voi itse tarkistaa, onko kyseessä vastuullisuusraportti, minkä takia on tärkeää, että mallilla on korkeampi herkkyys kuin tarkkuus. Korkeampi herkkyys tarkoittaa, että tekoälymalli ennustaa useammin linkin oikeaksi. Googlen konsolin avulla arviointimittareita voidaan muuttaa helposti. Mittareita muuttamalla voidaan muokata herkkyyttä ja tarkkuutta. Kuvassa 14 on kaavio uuden tekoälymallin tarkkuudesta ja herkkyydestä. Kaavioissa y-akseli kertoo tarkkuuden ja x-akseli herkkyyden. Tekoälymallin mittareita muutettaessa saadaan hyvä tasapaino herkkyyden ja tarkkuuden välillä, missä herkkyys on 95 prosenttia, tarkkuus 76 prosenttia ja F1-mitta on 84 prosenttia.




Kuva 14. Kaavio uuden tekoälymallin arviointimittareista [3].

Vanhempi tekoälymalli ei suoriudu yhtä hyvin kuin uusi. Kuvan 15 kaavio esittää vanhan tekoälymallin tarkkuuden ja herkkyyden. Mallille hyvä tasapaino herkkyyden ja tarkkuuden välillä olisi kohta, jossa herkkyys on 93 prosenttia, tarkkuus 72 prosenttia ja F1-mitta 81 prosenttia.




Kuva 15. Kaavio vanhan tekoälymallin arviointimittareista [3].

Näiden arviointimittareiden perusteella voidaan todeta, että uusi tekoälymalli suoriutuu paremmin vastuullisuusraporttien kategorioimisessa.

## 5.3 Tekoälyn ja ohjelmiston integrointi

Tekoälyn integrointi palvelimeen on keskeinen osa verkkosivuston kehitystä, sillä se mahdollistaa tehokkaan ja älykkään tiedonkäsittelyn.

Google Cloudin rajapinnat ja palvelukohtaiset avaimet mahdollistavat vuorovaikutuksen Vastrapin palvelimen ja tekoälyn välillä. Koodiesimerkki 1 esittää funktion, jolla tekoälyä kutsutaan palvelimesta.

```
def bot_call(site_url):
     ...
    client = aiplatform.gapic.PredictionServiceClient(client_op-
tions=client_options)
    instance = predict.instance.TextClassificationPredictionInstance(
        content=site_url,
    ).to_value()
    instances = [instance]
    parameters_dict = {}
    parameters = ParseDict(parameters_dict, Value())
    endpoint = client.endpoint_path(
        project=project, location=location, endpoint=endpoint_id
    )
    response = client.predict(
        endpoint=endpoint, instances=instances, parameters=parameters)
```

Koodiesimerkki 1. Funktio, jolla tekoälyä kutsutaan palvelimesta.

Kun tekoälyn tarjoamat vastaukset on saatu palvelimelta, ne välitetään takaisin käyttöliittymälle käsiteltäviksi ja esitettäviksi käyttäjälle. Tämä tapahtuu HTTP-pyyntöjen avulla, joissa palvelin lähettää vastauksena tekoälyn antaman tiedon ja hakukoneen palauttamat linkit. Käyttöliittymä voi seuraavaksi käsitellä nämä tiedot ja näyttää ne käyttäjälle. Koodiesimerkissä 2 näytetään yksinkertaisesti, kuinka käyttöliittymä vastaanottaa palvelimen tarjoamat tiedot ja käsittelee niitä.

```
try {
    const response = await axios.post("url/search", data, {headers});
    const sortedResults = response.data.results.sort((a: { confidence:
number; }, b: {
        confidence: number;
    }) => b.confidence - a.confidence);
    setResults(sortedResults);
    setIsLoading(false)
} catch (err) {
    setError(true);
}
```

Koodiesimerkki 2. Osa koodista, jolla käyttöliittymä ottaa yhteyden palvelimeen, ja vastaanottaa vastauksen.

## 5.4 Ohjelmiston ongelmat ja korjaukset

Verkkosivuston kehityksen keskeiset haasteet liittyvät käyttöliittymän heikkouteen, mukautumisen puutteeseen sekä palvelimen toimintavikoihin.

Palvelimen korjaukset

Yksi merkittävä ongelma palvelimessa on sen kyvyttömyys löytää vastuullisuusraportteja tehokkaasti, mikä johtuu kaapimisprosessin tehottomuudesta. Tämä puute heikentää verkkosivuston käyttökelpoisuutta ja saattaa aiheuttaa käyttäjäpettymyksiä. Aiemmin verkkosivusto ei hyödyntänyt hakukonetta, mutta nyt hakukone on integroitu Googlen koodattavalla hakukoneella.

Hakukoneen käyttöönotto mahdollistaa tehokkaamman ja tarkemman hakuprosessin, mikä parantaa käyttäjäkokemusta ja verkkosivuston käyttökelpoisuutta. Haku koostuu haetusta yhtiöstä ja avainsanasta vastuullisuusraportti, jotta hakutulokset kohdistuvat todennäköisemmin oikeaan tietoon. Avainsanan lisäys varmistaa, että sivusto pystyy löytämään vastuullisuusraportit nopeasti ja tarkasti. Esimerkkikoodissa 3 näkyy funktio, jonka avulla voidaan hyödyntää hakukonetta koodissa.

```
def google_search(search_term):
  service = build("customsearch", "v1", developerKey=API_KEY)
  res = service.cse().list(q=search_term, cx=CSE_ID).execute()
  return res['items'][:5]  # Return the top  5 results
```

Esimerkkikoodi 3. Ote koodista, joka kutsuu Googlen hakukonetta ja palauttaa viisi parasta vastausta.

Palvelimella ei aiemmin noudatettu eettisiä tai laillisia ohjeita robots.txt-tiedoston suhteen, joka saattoi aiheuttaa konflikteja ja esteitä tiedonhankinnalle. Eettisten ohjeiden noudattaminen tarkoittaa esimerkiksi tietosuojan ja käyttäjien oikeuksien kunnioittamista. Robots.txt on verkkosivuston juurikansioon sijoitettava tekstitiedosto, joka antaa ohjeita hakukoneille ja verkkokaapijoille siitä, mitkä sivuston polut niiden tulisi tarkastella ja mitkä jättää huomioimatta. Vanha kaapimisprosessi johti monimutkaiseen ja huonolaatuiseen koodiin, joka lisäsi sivuston toimintahäiriöiden riskiä. Esimerkkikoodissa 4 esitetään osa vanhasta koodista, jolla kaavittiin sivuja vastuullisuusraportin etsimiseksi.

```
                for url in parent_urls:
                    reqs = requests.get(url)
                    soup = BeautifulSoup(reqs.text, 'html.parser')
                    for element in soup.find_all('a'):
                        if 'href' in element.attrs:
                            href = element.get('href')
                            if href.startswith('/'):
                                full_url = urljoin(site_url, href)
                                # Ignores different localization links
                                if "/fi/" in full_url or "/sv/" in
full_url or "/en/" in full_url or "/de/" in full_url:
                                    continue
                                # If the same url is already on each
of the lists, don't add it
                                if full_url not in parent_urls and
full_url not in child_urls:
                                    child_urls.append(full_url)
```

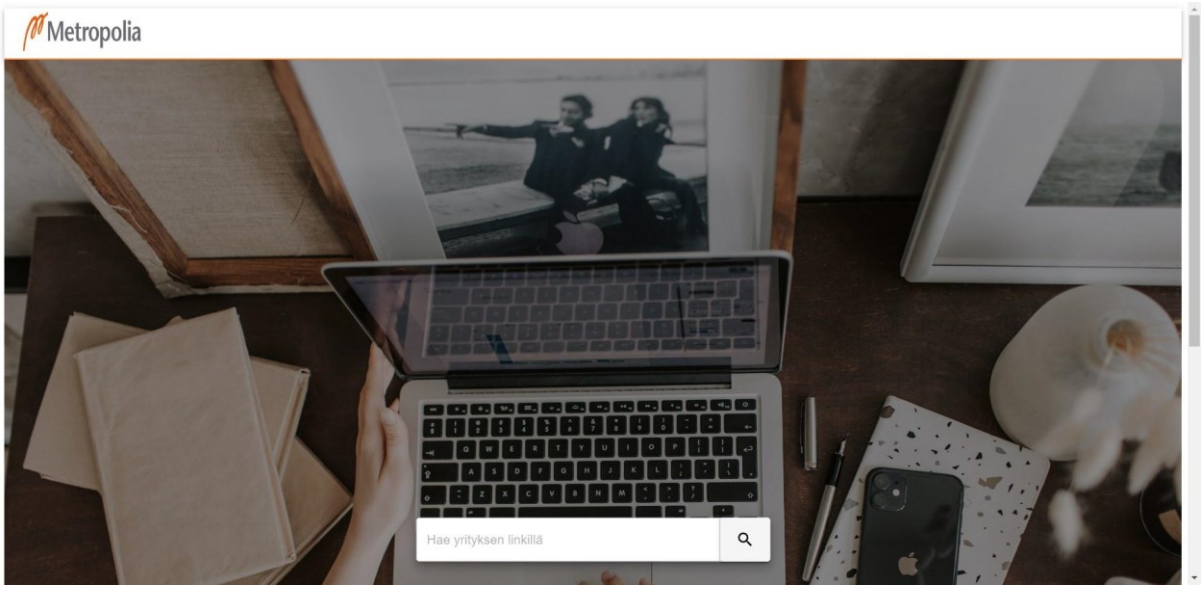
Esimerkkikoodi 4. Osa vanhaa koodia kaapimisfunktiossa.

Hakukoneen avulla voidaan noudattaa eettisiä ja laillisia periaatteita varmistamalla, että sivusto toimii yhteensopivasti robots.txt-tiedoston kanssa. Periaatteiden noudattaminen vähentää mahdollisia konflikteja ja esteitä tiedonhankinnassa. Lisäksi palvelimella toteutetaan muita parannuksia, kuten tarpeettoman ja ylikompleksin koodin poistaminen, mikä parantaa sivuston suorituskykyä ja

vähentää mahdollisten toimintahäiriöiden riskiä. Tämä yksinkertaistettu lähestymistapa parantaa merkittävästi koodin laatua ja käytettävyyttä.

Käyttöliittymän ongelmat ja korjaukset

Vanha käyttöliittymä on hankala ja mukauttamaton, mikä vaikeuttaa käyttäjien navigointia ja heikentää käyttökokemusta. Lisäksi sivun ylä- ja alatunnisteet ovat puutteelliset, mikä vaikeuttaa sivuston käyttöä eri näyttökoilla. Kuvassa 16 esitellään verkkosivuston vanha käyttöliittymä.

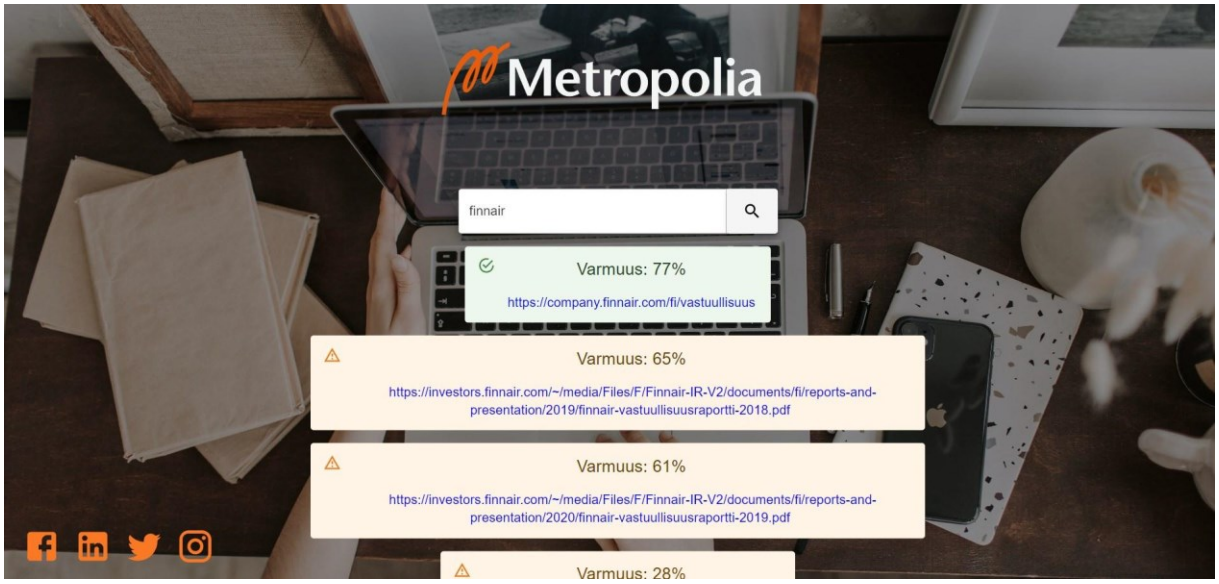



Kuva 16. Kuvankaappaus Vastrapin vanhasta käyttöliittymästä.

Ratkaisuna edellä mainittuihin haasteisiin on toteutettu useita korjauksia ja parannuksia. Käyttöliittymän parantamiseksi verkkosivulle on lisätty latausikoni, joka antaa visuaalisen vihjeen toiminnasta. Kuvassa 17 näkyy uusi käyttöliittymä sekä lisätty latausikoni.


Kuva 17. Kuvankaappaus Vastrapin uudesta käyttöliittymästä ja latausikonista.

Uusi käyttöliittymä esittää hakutulokset käyttäjälle viitenä linkkinä, joista kukin on merkitty tekoälyn varmuudella siitä, onko linkki vastuullisuusraportti vai ei. Näin käyttäjä voi tehdä päätöksen siitä, minkä linkin valita. Kuvassa 18 näkyy kuvankaappaus uudesta käyttöliittymästä hakukoneen tuloksilla. Käyttöliittymää voi selata alaspäin, jotta jokainen tulos näkyy.




Kuva 18. Kuvankaappaus uuden käyttöliittymän palauttamista hakukoneen tuloksista.

# 6 Haasteet ja jatkokehittäminen

Haasteet

Merkittävä haaste tekoälyn kouluttamisessa ja parantamisessa liittyy datan keräämiseen ja sen laatuun. Tekoälyä koulutetaan tunnistamaan vastuullisuusraportit keräämällä linkkejä eri yritysten raporteista. Kuitenkin tämän tiedon kerääminen voi olla vaikeaa, sillä yritykset voivat julkaista raporttinsa esimerkiksi eri nimillä, mikä saattaa sekoittaa tekoälyn tunnistamiskykyä. Joidenkin yhtiöiden julkaisuissa käytetään nimitystä luotettavuusraportti tai yhteiskuntavastuuraportti. Lisäksi vastuullisuusraportit voivat olla saatavilla eri muodoissa, kuten PDF-tiedostoina tai suoraan yritysten verkkosivustoilta.

Toinen haaste liittyy hakukoneen palauttamiin linkkeihin. Vaikka hakukone pyrkii tarjoamaan relevantteja tuloksia, joskus linkit voivat olla epäselviä tekoälylle. Linkit voivat olla esimerkiksi suoria linkkejä vastuullisuusraporttien PDF:iin. Usein PDF-linkkien URL-osoite on dynaamisesti generoitu, minkä takia se on epäselvä. Tämä voi johtaa siihen, että tekoälyn käsittelemät linkit eivät aina ole optimaalisia.

Jatkokehittäminen

Seuraava huomattava parannus on tarjota tekoälylle enemmän ja laadukkaampaa dataa. Tämä voi sisältää laajemman valikoiman vastuullisuusraportteja eri yrityksistä ja mahdollisesti erilaisia datalähteitä, jotka tukevat tekoälyn koulutusta.

Käyttöliittymään voisi lisätä asetuksen, jossa käyttäjä voi valita haluamansa tavan saada vastuullisuusraportit, joko suorana linkkinä PDF-tiedostoon tai yrityksen verkkosivuille. Tämä voisi auttaa vähentämään tekoälyn sekaannusta erilaisten tiedonlähteiden kanssa ja tarjota käyttäjille paremman käyttökokemuksen. Käyttämällä useampia tekoälymalleja, joista kukin on erikoistunut tiettyyn raporttimuotoon, voitaisiin parantaa tekoälyn suorituskykyä ja tarkkuutta. Näin

tekoälyn ei tarvitse yrittää kattaa kaikkia eri raporttimuotoja yhdellä algoritmilla, mikä voi johtaa virheisiin.

# 7 Yhteenveto

Tämän opinnäytetyön tavoitteena oli jatkokehittää palvelua, joka hyödyntää tekoälyä vastuullisuusraporttien etsimiseen ja luokittelemiseen eri lähteistä. Työssä keskityttiin parantamaan tekoälyn kykyä tunnistaa vastuullisuusraportteja, tehostamaan palvelimen toimintaa sekä parantamaan käyttöliittymää.

Tekoälyn suorituskykyä parannettiin luomalla uusi tekstinluokittelumalli, joka koulutettiin suuremmalla määrällä koulutusmateriaalia verrattuna aiempaan malliin. Palvelimen osalta siirryttiin pois verkkosivujen kaapimisesta, joka todettiin vikaherkäksi ja epäeettiseksi menetelmäksi. Kaapimisen sijaan palvelin integroitiin hyödyntämään hakukonetta tehokkaamman ja eettisemmän raporttien haun mahdollistamiseksi. Käyttöliittymä kehitettiin mukautuvaksi, mikä paransi käyttökokemusta eri laitteilla.

Merkittävä haaste tekoälyn kouluttamisessa ja parantamisessa liittyi datan keräämiseen ja sen laatuun. Tekoälyä koulutettiin tunnistamaan vastuullisuusraportit keräämällä linkkejä eri yritysten vastuullisuusraportteihin. Linkit voivat kuitenkin olla sekavia tekoälylle, sillä yritykset saattavat julkaista vastuullisuusraporttinsa eri tavoin ja eri nimillä. Toinen haaste liittyi hakukoneen palauttamiin linkkeihin, jotka ovat joissain tapauksissa epäselviä tekoälymallille.

Jatkokehitysmahdollisuuksina tunnistettiin tekoälyn koulutusdatan laadun ja määrän parantaminen, mahdollisuus valita raporttien hakutapa sekä useampien tekoälymallien käyttö eri raporttimuotojen tunnistamiseen. Nämä muutokset voisivat parantaa tekoälyn suorituskykyä ja käyttökokemusta entisestään.

## Lähteet


1 Vertex AI pricing. Verkkoaineisto. Google. <https://cloud.google.com/vertex-ai/pricing>. Luettu 31.1.2024.

2 AutoML beginner's guide. Verkkoaineisto. Google. <https://cloud.google.com/vertex-ai/docs/beginner/beginners-guide>. Luettu 07.02.2024.

3 Google Cloud Console. Versio 3.13. 2024. Google.

4 Bishop, Christopher. 2006. Pattern Recognition and Machine Learning. E-kirja. Springer.

5 What is Supervised Learning? Verkkoaineisto. Google. <https://cloud.google.com/discover/what-is-supervised-learning>. Luettu 22.1.2024.

6 Khoei, Talaei; Slimane, Hadjar & Kaabouch, Naima. 2023. Deep learning: systematic review, models, challenges, and research directions. s. 1–6.

7 Ciaburro, Giuseppe & Venkateswaran, Balaji. Neural Networks with R. E-kirja. Packt Publishing.

8 Zarei, Soheila; Bozorg-Haddad, Omid & Nikoo, Reza Mohammad. 2022. Computational Intelligence for Water and Environmental Sciences. E-kirja. Springer.

9 Dubey, Ram Shiv; Singh, Kumar Satish & Chaudhuri, Baran Bidyut. 2022. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. s. 1–3.

10 What is gradient descent? Verkkoaineisto. IBM. <https://www.ibm.com/topics/gradient-descent>. Luettu 26.1.2024.

11 Gradient descent. Verkkoaineisto. Khan Academy. <https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent>. Luettu 26.1.2024.

12 Stochastic Gradient Descent- A Super Easy Complete Guide! Verkkoaineisto. MLTUT. <https://www.mltut.com/stochastic-gradient-descent-a-super-easy-complete-guide/>. Luettu 26.1.2024.

13 Kulkarni, Ajay; Chong, Deri & Batarseh, Feras. 2021. Foundations of data imbalance and solutions for a data democracy. s. 4–5.

14 Train and use your own models. Verkkoaineisto. Google. <https://cloud.google.com/vertex-ai/docs/training-overview>. Luettu 5.2.2024.

15 Choose a training method. Verkkoaineisto. Google. <https://cloud.google.com/vertex-ai/docs/start/training-methods>. Luettu 5.2.2024.