



# **Tekstianalyysisovelluksen kehittäminen generatiivisen tekoälyn avulla**

Markus Hautaniemi

Opinnäytetyö, AMK

Joulukuu 2023

Tieto- ja viestintätekniikan tutkinto-ohjelma

**Hautaniemi, Markus**

## **Tekstianalyysisovelluksen kehittäminen generatiivisen tekoälyn avulla**

Jyväskylä: Jyväskylän ammattikorkeakoulu. Joulukuu 2023, 44 sivua.

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

### **Tiivistelmä**

Opinnäytetyössä kehitettiin käyttövalmis tekstianalyysisovellus asiakaspalautteiden analysointiin, johon kuului sekä palvelin- että selainpuolen toteutus. Palautteet luokiteltiin kategorioiden ja sentimentin mukaan, jonka jälkeen ne visualisoitiin graafisessa käyttöliittymässä. Opinnäytetyön toimeksiantaja toimi Jyväskylässä toimivalle Tridea Oy, jonka päätoimiala on suhdetoiminta ja viestintä. Opinnäytetyö oli mukana Kestävää nostetta matkailuyrityksiin: Etelä-Suomi -hankkeessa.

Tehtävänä oli tutkia ja etsiä tekoälyratkaisu suurten tekstimäärien analysointiin, jotta analysoidut tulokset, voidaan esittää loppukäyttäjälle. Tavoitteena oli löytää skaalautuva ratkaisu, jolla saataisiin tehtyä tarkkoja analyyseja, ja tulokset saataisiin esitettyä selkeästi.

Sovelluksen analysointiosuus toteutettiin OpenAi-rajapinnan avulla. Palautteiden luokittelu ja sentimenttianalyysi suoritettiin rajapinnan kautta. Tulosten tallentamiseen käytettiin MySQL-tietokantaa. Palvelinpuoli toteutettiin Node.js ajoympäristöä hyödyntäen. Selainpuoli toteutettiin käyttäen React-kirjastoa ja käyttöliittymässä käytetyissä komponenteissa käytettiin Material-UI-komponenttikirjastoa ja Apexcharts-kaaviokirjastoa.

Opinnäytetyölle asetetut tavoitteet täyttyivät. Tuloksena saatiin käyttöön valmis tekstianalyysisovellus, jossa palautteet analysoidaan kategorian ja sentimentin mukaan. Luokittelu vastasi sille asetettuja tarkkuusvaatimuksia ja analysointiin käytetty OpenAi-rajapinta skaalautuvuusvaatimuksia. Käyttöliittymästä loppukäyttäjä näkee oleelliset tiedot helposti ja pystyy tutkimaan analysoituja tuloksia tarkasti.

### **Avainsanat (asiasanat)**

Koneoppiminen, tekoäly, React, JavaScript

### **Muut tiedot (salassa pidettävät liitteet)**

-

**Hautaniemi, Markus**

**Developing a text analysis application using generative artificial intelligence**

Jyväskylä: JAMK University of Applied Sciences, December 2023, 44 pages.

Degree Programme in Information and Communication Technology. Bachelor's thesis.

Permission for open access publication: Yes

Language of publication: Finnish

**Abstract**

The thesis focused on developing a ready-to-use text analysis application for analyzing customer feedback, encompassing both server-side and client-side implementation. Feedback was categorized based on categories and sentiment, followed by visualization in a graphical user interface. The commissioning party for the thesis was Tridea Ltd., operating in Jyväskylä, primarily involved in public relations and communications. The thesis was part of the Sustainable Boost for Tourism Companies: Southern Finland project.

The task involved researching and finding an AI solution for analyzing large volumes of text to present analyzed results to end-users. The goal was to find a scalable solution for accurate analysis and clear presentation of results.

The analysis part of the application was implemented using the OpenAI API for feedback categorization and sentiment analysis. MySQL database was used for result storage. The server-side was implemented using Node.js runtime environment. The client-side was developed using the React library, employing Material-UI components, and Apexcharts for the graphical interface components.

The objectives set for the thesis were achieved. The outcome was a functional text analysis application where feedback is analyzed based on category and sentiment. The categorization met accuracy requirements, and the OpenAI API fulfilled scalability demands for analysis. The user interface provides essential information conveniently and allows in-depth exploration of analyzed results for end-users.

**Keywords/tags (subjects)**

Machine learning, artificial intelligence, JavaScript, React

**Miscellaneous (Confidential information)**

-

## Sisältö

<b>1</b>	<b>Johdanto .....</b>	<b>4</b>
<b>2</b>	<b>Tutkimusasetelma .....</b>	<b>4</b>
<b>3</b>	<b>Koneoppiminen .....</b>	<b>5</b>
3.1	Koneoppimisen alakategoriat .....	5
3.2	Siirto-oppiminen ja hienosäätäminen .....	6
3.2.1	Siirto-oppimisen hyödyt .....	7
3.2.2	Hienosäädön hyödyt .....	7
3.3	Luonnollisen kielen käsittely .....	8
3.4	Neuroverkot ja syväoppiminen .....	8
3.5	Transformer-arkkitehtuuri .....	9
3.6	Large Language Models .....	9
3.7	LLM Rajapinnat .....	10
3.8	LLM rajapintojen käyttötavat ja haasteet .....	10
<b>4</b>	<b>OpenAi rajapinta .....</b>	<b>11</b>
<b>5</b>	<b>Toteutuksen suunnitelma .....</b>	<b>12</b>
5.1	Käytetyt teknologiat .....	12
5.1.1	MySQL .....	12
5.1.2	Node.js .....	12
5.1.3	React .....	13
5.1.4	Python .....	13
5.1.5	Pytorch .....	14
5.1.6	Hugging Face Transformers .....	15
5.2	Tietokantarakenne .....	16
5.3	Palvelinpuoli .....	17
5.4	Käyttöliittymä .....	17
5.5	Tiedonkulku .....	19
<b>6</b>	<b>Koneoppimismallien hienosäätö .....</b>	<b>20</b>
6.1	Sentimenttianalyysi .....	21
6.1.1	Mallin hienosäätö sentimenttianalyysiin .....	21
6.1.2	Sentimenttianalyysin päätelmä .....	23
6.2	Multi-label-luokittelu .....	24
6.2.1	Multi-label-luokittelun koulutusaineisto .....	24
6.2.2	Mallin hienosäätäminen multi-label-luokitteluun .....	25
6.2.3	Multi-label-luokittelun päätelmä .....	27

<b>7</b>	<b>Back-end toteutus .....</b>	<b>27</b>
7.1	Rajapinnan käyttö .....	28
7.2	Kehotteiden rakentaminen .....	30
7.3	Tulosten tallennus.....	30
<b>8</b>	<b>Front-end toteutus .....</b>	<b>32</b>
8.1	Kirjastot .....	32
8.2	Käyttöliittymän komponentit.....	33
8.2.1	Valikko.....	33
8.2.2	Graafit .....	33
8.2.3	Taulukot .....	35
8.3	Käyttäjävuorovaikutus .....	37
<b>9</b>	<b>Pohdinta.....</b>	<b>38</b>
9.1	Tulokset .....	38
9.2	Jatkokehitys.....	39
	<b>Lähteet .....</b>	<b>40</b>
	<b>Liitteet .....</b>	<b>42</b>
	Liite 1. Sentimenttianalyysin päätelmä .....	42
	Liite 2. Multi-label-luokittelu opetusohjelma .....	43
	Liite 3. Multi-label-luokittelu päätelmä .....	44

## Kuviot

Kuvio 1.	Tietokantarakenne .....	17
Kuvio 2.	Käyttöliittymän malli .....	19
Kuvio 3.	Sovelluksen tiedonkulku .....	20
Kuvio 4.	Sentimenttianalyysin koulutusaineiston rakenne .....	21
Kuvio 5.	Sentimenttianalyysin datan valmistelu ja prosessointi .....	22
Kuvio 6.	Mallin koulutus ja arviointi.....	23
Kuvio 7.	Luokitteluun käytetyn datasetin rakenne .....	24
Kuvio 8.	Multi-label-luokittelun koulutusaineiston valmistelu ja prosessointi .....	26
Kuvio 9.	Pyyntö OpenAi rajapintaan .....	28
Kuvio 10.	Rajapinnan vastaus .....	28
Kuvio 11.	Analysoinnin turvamekanismi.....	29
Kuvio 12.	Tuloksien parsiminen .....	31
Kuvio 13.	Liitostaulun tietojen luonti.....	31

Kuvio 14. Sentimenttiosuuksien jakauman visualisointiin käytetty piirakkakaavio .....	34
Kuvio 15. Sentimenttijakauman visualisointi kategorioittain .....	35
Kuvio 16. Yksityiskohtaisien analyysien esittämiseen käytetty taulukko .....	36
Kuvio 17. Kategorioimattomien palautteiden esittämiseen käytetty taulukko .....	37

# 1 Johdanto

Opinnäytetyön tarkoituksena oli kehittää sovellus suurien tekstimuotoisten datamassojen analysointiin. Työ toteutettiin Jyväskylässä toimivalle Tridea Oy:lle, jonka päätoimiala on suhdetoiminta ja viestintä. Yhtiön palveluita ovat BisLenz-ohjelmisto, jonka tarkoituksena on helpottaa yritysjohtoa hyödyntämään liiketoimintatietoa, sekä ohjelmistoprojektit.

Sovelluksen tarkoitus on luokitella tekoälyn avulla tekstimuotoinen data, kuten asiakaspalautteet, tunneanalyysin ja ennalta määritettyjen kategorioiden perusteella. Analysointi tapahtuu automaattisesti aina, kun uutta dataa on tarjolla. Analysoiduista tuloksista tehdään yhteenveto ja visualisointi, jotta käyttäjän on helppo tarkastella niitä.

Opinnäytetyön tavoitteena oli kehittää käyttöön valmis tuote, jota asiakkaat voivat hyödyntää. Valmiista tuotteesta asiakas näkee oman palvelun kehityskohteet palautteissa esiintyvistä kategorioista ja niiden sentimenteistä. Tulosten perusteella asiakas voi kehittää omaa palveluaan analysoitujen palautteiden pohjalta. Tuotetta on myös mahdollista kehittää tulevaisuudessa mahdollisten asiakkaiden lisätarpeiden mukaan.

# 2 Tutkimusasetelma

Tutkimusongelma oli suurien tekstimäärien automaattinen ja tehokas analysointi. Toimeksiantajan kanssa oli päätetty käyttää koneoppimista ratkaisussa. Omien mallien ja valmiiden rajapintojen suorituskkyä ja tarkkuutta piti tutkia ja vertailla ennen sovelluksen tekoa.

## Tutkimuskysymykset ja -menetelmä

Tutkimusongelman pohjalta hahmotettiin seuraavat kysymykset, joihin työssä etsitään ratkaisuja:

- Saako analyysin tarvittavan tarkaksi?
- Ratkaisun skaalautuvuus?
- Kuinka tulokset esitetään loppukäyttäjälle?
- Jatkokehityksen mahdollisuudet?

Opinnäytetyön tutkimusmenetelmä on tutkimuksellinen kehitystyö, sillä käytännöntoteutus on keskeisessä osassa työtä. Tutkimuksellinen kehittämistyö on prosessi, jossa keskitytään työelämän

kehittämiseen tutkivalla otteella. Tässä lähestymistavassa yhdistyvät konkreettinen kehittämistointi sekä tutkimuksellisten menetelmien soveltaminen ja saadun aineiston analysointi. Lähtökohtana ovat työelämästä nouseva käytännön ongelma ja kysymykset, jotka ohjaavat tiedon tuottamista käytännön toimintaympäristössä.

### **3 Koneoppiminen**

Koneoppiminen on tekoälyn ja tietojenkäsittelytieteen ala, joka keskittyy tietojen ja algoritmien käyttöön jäljittelemään tapaa, jolla ihmiset oppivat, parantaen vähitellen sen tarkkuutta. Muutaman viime vuosikymmenen aikana tallennus- ja prosessointitehon teknologinen kehitys on mahdollistanut joitakin koneoppimiseen perustuvia innovatiivisia tuotteita, kuten Netflixin suositusmoottorin ja itseajavat autot. Koneoppiminen on tärkeä osa kasvavaa datatieteen alaa.

Tilastollisten menetelmien avulla algoritmeja koulutetaan tekemään luokituksia tai ennusteita ja paljastamaan keskeisiä oivalluksia tiedon louhintaprojekteista. Nämä oivallukset ohjaavat myöhemmin päätöksentekoa sovelluksissa ja yrityksissä, mikä vaikuttaa ihanteellisesti tärkeimpiin kasvumittareihin. (What is machine learning? n.d.)

#### **3.1 Koneoppimisen alakategoriat**

##### **Valvottu oppiminen**

Valvottuja koneoppimismalleja koulutetaan käyttäen merkittyjä tietojoukkoja, joiden avulla mallit oppivat ja tarkentuvat ajan myötä. Esimerkiksi algoritmi opetettaisiin koirien ja muiden asioiden kuvilla, jotka ihmiset ovat merkinneet, ja kone oppisi tapoja tunnistaa koirien kuvat. Valvottu koneoppiminen on nykyään yleisin opetustapa. (Brown 2021.)

##### **Valvottoman oppiminen**

Valvomattomassa koneoppimisessa ohjelma etsii kuvioita merkitsemättömästä tiedosta. Valvomattomat koneoppimismallit voivat löytää malleja tai trendejä, joita ihmiset eivät nimenomaisesti etsi. Esimerkiksi valvottoman koneoppimisohjelma voisi tarkastella verkkomyyntitietoja ja tunnistaa erilaisia ostoksia tekeviä asiakkaita. (Brown 2021.)



## Vahvistava oppiminen

Vahvistuskoneoppiminen kouluttaa malleja yrityksen ja erehdyksen kautta toimimaan parhaalla mahdollisella tavalla luomalla palkitsemisjärjestelmän. Vahvistusoppiminen voi kouluttaa malleja pelaamaan pelejä tai itseohjautuvia ajoneuvoja ajamaan, kertomalla koneelle, kun se on tehnyt oikeat päätökset, mikä auttaa sitä oppimaan ajan myötä, miten sen tulisi toimia. (Brown 2021.)

## 3.2 Siirto-oppiminen ja hienosäätäminen

Siirto-oppiminen (engl. transfer learning) on tekniikka koneoppimisessa, jossa yhden tehtävän mallista saatu tieto otetaan käyttöön uuden tehtävän perustana. Koneoppimisen algoritmit käyttävät aiempaa dataa ennusteiden tekemiseen, yleensä keskittyen tiettyihin tehtäviin. Alkuperäinen tehtävä jakaa tietonsa uuden tehtävän kanssa parantaen näin oppimista. Tämä tarkoittaa aiemman tietämyksen hyödyntämistä uuden alueen oppimisen edistämiseksi, jossa alkuperäisen tehtävän ominaisuuksia sovelletaan uuteen tehtävään. Käytettävän mallin kaikki esiopetetut kerrokset jäädytetään ja uusia kerroksia lisätään, jotka opetetaan uudella datalla. Jos tämä prosessi vahingoittaa uuden tehtävän suoritusta, kyseessä on negatiivinen siirto. Tämä tapahtuu, kun alkuperäinen ja uusi tehtävä ovat erilaisia, mikä tekee alkuperäisestä koulutuksesta vähemmän olennaista. Negatiivisen siirron välttäminen on keskeinen haaste siirto-oppimisessa. (Arya 2022.)

Hienosäätö (engl. fine-tuning) on koulutustekniikka, joka sisältää valmiiksi määriteltujen ja esikoulutettujen konvoluutioneuroverkkojen (CNN) arkkitehtuurien uudelleenkäyttämisen. Tämä lähestymistapa hienosäätää tiettyjä verkon kerroksia saavuttaakseen halutut tulokset, säätäen esikoulutetun mallin ominaisuuksia paremmin vastaamaan käsiteltävää ongelmaa. Tämä poistaa tarpeen rakentaa neuroverkon tyhjästä. Hienosäätö on erityisen hyödyllinen, kun tarkkoja ennustemalleja tarvitaan rajallisilla aineistoilla. Sitä sovelletaan yleisesti, kun tarvitaan syväoppimISRatkaisuja, mutta käytettävissä oleva data ei ole riittävä kouluttamaan CNN:ää alusta alkaen. Tässä prosessissa olemassa olevat verkkoparametrit koulutetaan uutta tehtävää varten. Tämä edellyttää mallin rakenteen sopeuttamista ja kouluttamista vaiheiden avulla, kuten kerrosten lisäämistä ja poistamista, vain relevanttien kerrosten jäädyttämistä sekä uusien kerrospainojen päivittämistä uuden datan perusteella. (Torres 2023.) Hienosäätö on vaihe siirto-opetuksessa, jonka tarkoituksena on parantaa mallin suorituskykyä (Arya 2022).

Hienosäätö ja siirto-opetus on tärkeää erottaa, sillä molemmat sisältävät verkkokoulutusta datan ja olemassa olevan tiedon kanssa. Siirto-opetus hyödyntää tietoa yhden tehtävän piirteistä ja soveltaa sitä uuteen tehtävään. Hienosäädössä hienosäädetään prosessia tavoiteltujen tulosten saavuttamiseksi, mukauttaen ja räätälöiden huolellisesti koulutetut malliparametrit parannetun suorituskyvyn saavuttamiseksi. (Arya 2022; Torres 2023.)

### **3.2.1 Siirto-oppimisen hyödyt**

Siirto-oppiminen ratkaisee datan saatavuuden ongelman. Riittävän datan hankkiminen voi olla haastavaa, ja rajallisen datan käyttäminen johtaa usein heikkoon suorituskyvyn. Siirto-oppiminen mahdollistaa pienen harjoitusaineiston käytön hyödyntämällä esikoulutettuja malleja. Koneoppimismallien kouluttaminen on haastavaa ja voi olla aikaa vievää. Uuden monimutkaisen syvä neuroverkoston kouluttaminen vaatii merkittävää aikapanosta. Sen sijaan esikoulutetun mallin käyttö lähtökohtana vähentää huomattavasti uuden mallin rakentamiseen vaadittavaa aikaa. (Torres 2023.)

Esikoulutetun mallin käyttö tarjoaa vahvan perustan. Tämä etu mahdollistaa tehtävien suorittamisen ilman laajaa koulutusta. Aiempi koulutus samankaltaisissa tehtävissä antaa esikoulutetuille malleille parannetun oppimisnopeuden, jonka ansiosta ne voivat sopeutua uusiin tehtäviin nopeammin. Vahvan perustan ja nopeutetun oppimisen yhdistelmä johtaa parannettuun suorituskyvyn. Esikoulutetut mallit saavuttavat korkeamman tarkkuuden tuottaessaan ennusteita. (Torres 2023.)

### **3.2.2 Hienosäädön hyödyt**

Hienosäätö parantaa tehtäväkohtaista suorituskkyä sovittamalla mallin alaan liittyviin tietoihin, mikä johtaa tarkempiin ja kontekstuaalisesti relevantteihin tuloksiin tehtäviä varten. Hienosäätö myös vähentää merkittävästi koulutusaikaa ja laskentaresursseja, jotka tarvitaan haluttujen tulosten saavuttamiseksi, sillä olemassa olevan tiedon päälle voi rakentaa, säästäen aikaa ja kustannuksia. Se mahdollistaa mallien sopeutumisen myös erikoisaloille, kuten lääketieteelliseen tutkimukseen, oikeudelliseen analyysiin tai asiakastukeen. (Quick Concepts: Fine-tuning in Generative AI n.d.)

### 3.3 Luonnollisen kielen käsittely

Luonnollisen kielen käsittely (engl. Natural Language Processing, NLP) on tieteenala, jossa rakennetaan koneita, jotka voivat manipuloida ihmisten kieltä tai ihmiskieltä muistuttavaa dataa tavalla, jolla se kirjoitetaan, puhutaan ja järjestetään. Se kehittyi laskennallisesta lingvistiikasta, joka käyttää tietojenkäsittelytiedettä kielen periaatteiden ymmärtämiseen, mutta teoreettisten puitteiden kehittämisen sijaan NLP on tekniikan tieteenala, joka pyrkii rakentamaan teknologiaa hyödyllisten tehtävien suorittamiseksi. NLP voidaan jakaa kahteen päällekkäiseen alakenttään: luonnollisen kielen ymmärtäminen (engl. Natural Language Understanding, NLU), joka keskittyy semanttiseen analyysiin tai tekstin tarkoitetun merkityksen määrittämiseen, ja luonnollisen kielen generointi (engl. Natural Language Generation, NLG), joka keskittyy koneella tapahtuvaan tekstin luomiseen. NLP on erillinen, mutta usein sen kanssa käytettävästä puheentunnistuksesta, joka pyrkii jäsentämään puhutun kielen sanoiksi, muuttamaan äänen tekstiksi ja päinvastoin. (Natural Language Processing n.d.)

### 3.4 Neuroverkot ja syväoppiminen

Neuroverkko on tekoälytekniikka, joka jäljittelee ihmisaivojen tietojenkäsittelyä. Se on syväoppimisen muoto, jossa käytetään toisiinsa yhdistettyjä solmukohtia, jotka ovat samankaltaisia kuin aivojen neuronit, jotta voidaan luoda oppimisjärjestelmä, joka paranee oppimalla virheistä. Neuroverkot käsittelevät monimutkaisia tehtäviä tarkasti, kuten asiakirjojen yhteenvedon ja kasvojentunnistuksen. Niiden avulla tietokoneet voivat tehdä älykkäitä päätöksiä ilman laajaa ihmisen väliintuloa, koska ne käsittelevät monimutkaisia, epälineaarisia syöte-tulos-suhteita. Neuroverkot löytävät sovelluksia eri aloilla, mukaan lukien lääketieteellinen diagnoosi, kohdennettu markkinointi, talousennusteet, energiaennusteet, laadunvalvonta ja kemiallisten yhdisteiden tunnistaminen. (What Is A Neural Network? n.d.)

Syväoppiminen on koneoppimisen osa-alue, joka käsittelee dataa syväoppimisverkostojen avulla. Toisin kuin perinteiset koneoppimismenetelmät, joissa tarvitaan ihmisen panosta olennaisten ominaisuuksien määrittämiseen, syväoppiminen sisältää raakadatan toimittamisen ohjelmistoon. Syväoppimisverkot poimivat itsenäisesti ominaisuuksia ja oppivat itsenäisesti, mikä mahdollistaa

strukturoidun tietojoukkojen, kuten tekstin, analysoinnin. Tämä lähestymistapa on tehokkaampi ja soveltuu monimutkaisiin tehtäviin. Esimerkiksi kun ohjelmistoa opetetaan tunnistamaan lemmikkikuvia, perinteiset menetelmät sisältävät kuvien manuaalisen merkitsemisen ja tunnistamisominaisuuksien määrittämisen. Sen sijaan syväoppivat neuroverkot oppivat automaattisesti priorisoimaan ominaisuuksia, kuten jalkojen määrän ja kasvojen muodon, mikä tekee prosessista virtaviivaisemman ja tarkemman. (What Is A Neural Network? n.d.)

### 3.5 Transformer-arkkitehtuuri

Transformer-arkkitehtuuri koostuu kooderi- ja dekodieriosista, joista kummallakin on useita huomio-ottolohkoja, jotka määrittävät sanapainot niiden tehtävän merkityksen perusteella. Transformer-arkkitehtuurilla on etuja, kuten tehokas kyky havaita pitkäaikaisia riippuvuuksia, rinnakaistettavuus nopeampaa koulutusta varten sekä esikoulutus kielitieteellisillä tiedoilla laajan kielitaidon ymmärtämisen saavuttamiseksi. (The transformers: a revolution in natural language processing 2023.)

Transformer-arkkitehtuuri on menestynyt NLP-tehtävissä, kuten kääntämisessä, tekstien luomisessa, tunneanalyysissä ja kysymyksiin vastaamisessa. GPT, tunnettu Transformer-malli, erottuu erityisesti tekstien luomisessa. Transformer-arkkitehtuurin vaikutus NLP-malleihin on suuri, ohittaen aiemmat mallit ja tehostaen kielitehtäviä riippuvuuksien tunnistamisen, tehokkaan koulutuksen mahdollistamisen ja kielitaidon laajentamisen avulla. (The transformers: a revolution in natural language processing 2023.)

### 3.6 Large Language Models

Suuri kokoinen transformer-malli, tunnettu nimellä large language model (LLM) on pitkälle kehitetty kielimallityyppi, joka on koulutettu syväoppimisen avulla laajalla tekstiaineistolla ja kykenee tuottamaan ihmisen kaltaista tekstiä ja suorittamaan erilaisia kielitehtäviä. Se antaa todennäköisyyksiä sanajonoille tekstianalyysin perusteella. Nämä mallit voivat vaihdella yksinkertaisista n-grammeista monimutkaisiin neuroverkkoihin, mutta LLM viittaa yleensä syväoppimismalleihin, joissa on miljoonia tai miljardeja parametreja. Ne havaitsevat monimutkaisia kielikuvioita ja tuottavat ihmisen kaltaista tekstiä. (Hore 2023.)

### 3.7 LLM Rajapinnat

LLM rajapinnat, kuten OpenAi:n rajapinta, ovat tuoneet tekoälyt kaikkien saataville. Ennen näitä rajapintoja, tekoälyratkaisuja pystyi hyödyntämään ainoastaan suuret yritykset vaadittavien resurssien vuoksi. Nykyään LLM:t tarjoavat monipuolisia sovelluksia pienille ja keskisuurille yrityksille. Ne voivat tehostaa sisällöntuotantoa, lokalisoitua ja kielimuurien poistamista verkkosivustoilla ja sosiaalisessa mediassa. LLM:t ovat erinomaisia laajojen tekstien tiivistämisessä, päätöksenteon avustamisessa ja toimialatrendien seuraamisessa. Ne myös antavat pienille ja keskisuurille yrityksille mahdollisuuden optimoida sisältöä, luokitella tietoa tehokkaasti ja analysoida asiakkaan mielipidettä automaattisesti sekä parantaa asiakastukea personoidulla keskusteluälyllä ja chatboteilla. (What are Large Language Models (LLMs) and Why They Matter. 2023.)

### 3.8 LLM rajapintojen käyttötavat ja haasteet

Kehotesuunnittelu (engl. prompt engineering), jossa muotoillaan ja hienosäädetään ohjeita chatpohjaisten LLM:en ohjaamiseksi, tarjoaa useita etuja. Se mahdollistaa nopeiden ja suhteellisen helppojen tulosten saamisen käyttämällä muutamaa esimerkkiä oppimiseen, mikä mahdollistaa tekoälyominaisuuksien nopean kehittämisen ja tuomisen markkinoille. Tärkeää on, että se ei vaadi koodaustaitoja; menestyminen riippuu ohjeiden kirjoittamisesta ja testaamisesta. Käsken muokaus kuitenkin sisältää rajoituksia, mukaan lukien vaikeudet reunatapauksien tunnistamisessa ja ratkaisemisessa, mahdolliset ongelmat pitkien ohjeiden kanssa kuten hidastuminen, hintojen nousu ja vaikeudet vianmäärityksessä, kun ohjeiden toteutus on epä johdonmukaista. (Hennings 2023.)

Kehotesuunnittelu sopii LLM-ominaisuuksien validointiin ja ongelmien ratkaisemiseen pienimmässä toimivassa tuotteessa (engl. minimum viable product). Monissa tapauksissa sen tulokset riittävät uusien ominaisuuksien tuomiseen tuotantokäyttöön. Kuitenkin pelkästään käsken muokkaaminen ei välttämättä ole paras lähestymistapa. (Hennings 2023.)

Hienosäätö on vaihtoehtoinen lähestymistapa, joka sisältää olemassa olevan LLM-mallin kouluttamisen lisäohjeiden ja suoritusten sarjalla sen painojen ja vääristymien säätämiseksi. Se mahdollistaa valmiiksi koulutettujen mallien hyödyntämisen ja räätälöinnin tiettyihin tehtäviin. Hienosäätöä voidaan soveltaa laajaan valikoimaan käyttötapauksia, keskustelupohjaisten tekoälyjen, kuten

ChatGPT:n, luomisesta aina vastausten turvallisuuden parantamiseen. Se voi olla iteratiivinen prosessi, joka johtaa huomattaviin parannuksiin jopa alle 20 esimerkillä. Hienosäätö on hyödyllinen odotettujen ohjeiden ja suoritusten muodon muuttamisessa, suurten pyyntömäärien tehokkaassa käsittelyssä ja vähentää injektiohyökkäysten riskiä. Hienosäätö ei ole kuitenkaan erityisen hyvä ratkaisu uuden toimialatiedon opettamiseen mallille. Tämä on paremmin hoidettavissa upotuksilla tai valitsemalla perusmalli, joka jo tuntee toimialatiedon. (Hennings 2023.)

## 4 OpenAi rajapinta

OpenAI:n tarjoama rajapinta on palvelu, jonka avulla eri tekoälymallien kanssa voi olla vuorovaikutuksessa. Sitä voidaan soveltaa käytännössä mihin tahansa tehtävään, joka vaatii luonnollisen kielen ja koodin generoimista tai ymmärtämistä. Rajapintaa voidaan myös käyttää kuvien luomiseen ja muokkaamiseen tai puheen muuttamiseen tekstiksi. (Introduction n.d. a.)

Rajapinnan kanssa voi olla vuorovaikutuksessa HTTP-pyyntöjen avulla käyttäen mitä tahansa kieltä kuten virallisten Python-liitinten, virallisen Node.js-kirjaston tai yhteisön ylläpitämän kirjaston kautta. OpenAI rajapinta käyttää API-avaimia tunnistautumiseen. (Introduction n.d. b.)

Rajapinta mahdollistaa kompleksisenkin ratkaisun nopean käyttöönoton verrattuna perinteisiin koneoppimiseratkaisuihin. Rajapinnan kautta lähetetään ohjeet kehotteessa valitulle generatiiviselle mallille, jonka ansiosta opetusvaihetta ei tarvita. Tämä poistaa datasettien tarpeen, joka on yleensä suurin ongelma pienemmille yrityksille omia koneoppimismalleja tehdessä. Datasettejä ei välttämättä löydy valmiina tarvittavaan tarkoitukseen ja niiden kokoaminen ja luokittelu on kallista ja aikaa vievää.

Rajapinnasta maksetaan käytettyjen tokenien perusteella, yksi tokeni vastaa noin 3–4 merkkiä (Pricing n.d.). Rajapinta maksaa sitä enemmän mitä sitä käytetään, sen sijaan että maksetaan kiinteä hinta oman mallin pilvipalvelimesta. Mahdolliset palvelukatkot tai tuen loppuminen ovat riskeinä. Rajapinnassa on myös käyttörajoitukset, jotka täyttyvät helposti suuren tietomäärän vuoksi. Rajapinnan käyttörajoitusten täytyminen ja aikakatkaisut voivat olla ongelma, sillä jokainen pyyntö on maksullinen.

## 5 Toteutuksen suunnitelma

Ennen teknisen toteutuksen aloitusta luonnollisenkielen tekoälymalleja ja rajapintoja tuli kokeilla ja vertailla. Tärkeimpänä kriteerinä oli tarkkuus suomenkielisen tekstin analysoinnissa. Parhaiten kriteerejä vastanneet ratkaisut olivat Turun yliopiston TurkuNLP ryhmän tuottama FinBERT tekoälymalli ja OpenAin gpt-3.5 ja gpt-4 generatiiviset tekoälymallit. FinBERT on opetettu kokonaan suomenkieliselä opetusdatalla, minkä vuoksi mallin pystyi hienosäätämään suomenkielisten tekstien tunneanalyysiin ja luokitteluun. OpenAin gpt-3.5 ja gpt-4 mallit täyttivät myös suomen kielen kriteerit. Lopulliseksi valinnaksi päätettiin OpenAin mallit niiden skaalautuvuuden vuoksi. Hienosäädetyin mallin opetukseen ja päivittämiseen vaadittavien merkittyjen datasettien kokoaminen ja merkitseminen olisi ollut liian aikaa vievää ja kallista.

### 5.1 Käytetyt teknologiat

Suurin osa työssä käytetyistä teknologioista ovat käytössä yrityksen olemassa olevassa sovelluksessa. Samojen teknologioiden käyttö tekee uuden sovelluksen integroimisesta olemassa olevaan sovellukseen saumattomampaa.

#### 5.1.1 MySQL

MySQL on avoimen lähdekoodin relaatiotietokannan hallintajärjestelmä (engl. relational database management system). Kuten muutkin relaatiotietokannat, MySQL tallentaa dataa taulukkoina, jotka koostuvat riveistä ja sarakkeista. Käyttäjät voivat määritellä, manipuloida, hallita ja hakea dataa käyttäen rakenteellista kyselykieltä, yleisemmin tunnettu nimellä SQL. (Drake 2020.) MySQL on asiakas/palvelinjärjestelmä, joka koostuu monisäikeisestä SQL-palvelimesta, joka tukee erilaisia taustajärjestelmiä, useita erilaisia asiakasohjelmia ja kirjastoja, hallintatyökaluja sekä laajan valikoiman sovellusohjelmointirajapintoja. (What is MySQL n.d.)

#### 5.1.2 Node.js

Node.js on avoimen lähdekoodin ja monialustainen suoritusajaympäristö JavaScript-koodin suorittamiseen palvelinpuolella. Se ei ole kehys eikä ohjelmointikieli. Node.js:ää käytetään rakentamaan taustapalveluita, kuten rajapintoja verkko- tai mobiilisovelluksille. (Node.js introduction n.d.) Node.js on suunniteltu skaalautuvien verkkosovellusten rakentamiseen. Se käyttää asynkronista

tapahtumapohjaista mallia, mikä tarkoittaa, että monta yhteyttä voidaan käsitellä samanaikaisesti. Kun uusi yhteys muodostuu, kutsutaan takaisinkutsua (engl. callback). Mikäli ei ole mitään suoritettavaa, Node.js siirtyy lepotilaan. (About Node.js n.d.)

### 5.1.3 React

React on avoimen lähdekoodin selainpuolen JavaScript-kirjasto, jota ylläpitää Meta sekä yksittäisten kehittäjien yhteisö (Meet the Team n.d). Sitä käytetään muiden kirjastojen kanssa renderöimään tiettyihin ympäristöihin. Esimerkiksi React Nativea voidaan käyttää mobiilisovellusten rakentamiseen. Verkkoon rakentamiseen kehittäjät käyttävät Reactia yhdessä ReactDOMin kanssa. (Getting started with React n.d.)

Reactin ensisijainen tavoite on minimoida virheet, jotka syntyvät, kun kehittäjät rakentavat käyttöliittymiä. Se tekee tämän komponenttien ja itsenäisten, loogisten koodinpätkien avulla, jotka kuvaavat osan käyttöliittymästä. Näitä komponentteja voidaan yhdistellä luomaan kokonainen käyttöliittymän, ja React abstrahoi suuren osan renderöintityöstä. (Getting started with React n.d.)

React ei pakota tiukkoja sääntöjä koodikonventioista tai tiedostojärjestelmästä. Tämä mahdollistaa tiimien asettavan heille parhaiten toimivat konventiot ja ottamaan Reactin käyttöön haluamallaan tavalla. React voi käsitellä yhtä painiketta, muutamia osia käyttöliittymästä tai koko sovelluksen käyttöliittymää. (Getting started with React n.d.)

### 5.1.4 Python

Python on tulkattu, oliopohjainen, korkean tason ohjelmointikieli dynaamisella semantiikalla, jonka kehitti Guido van Rossum ja joka julkaistiin alun perin vuonna 1991. Se tunnetaan aloittelijaystävällisyydestään ja korvasi Javan laajimmin käytettynä aloituskielenä. Pythonia käytetään palvelinpuolen verkkokehityksessä, ohjelmistokehityksessä, matematiikassa ja järjestelmäskriptauksessa. Se on suosittu nopeassa sovelluskehityksessä ja skriptaamisessa tai liitoskielenä olemassa olevien komponenttien välillä. Python vähentää ohjelman ylläpitokustannuksia sen helposti opittavan syntaksin ja luettavuuteen panostamisen ansiosta. Se tukee moduuleja ja paketteja, mikä helpottaa modulaaristen ohjelmien ja koodin uudelleenkäyttöä. (What is Python n.d.)



Python on avoimen lähdekoodin yhteisökieli, johon lukuisat itsenäiset ohjelmoijat kehittävät jatkuvasti kirjastoja ja toiminnallisuuksia. Pythonia käytetään verkkosovellusten luomisessa palvelimelle, työnkulun rakentamisessa, tietokantajärjestelmiin yhdistämisessä, tiedostojen lukemisessa ja muokkaamisessa, monimutkaisessa matematiikassa, suurten tietojoukkojen käsittelyssä, nopeassa prototyypityksessä ja tuotantovalmiin ohjelmiston kehittämisessä. Python on erinomainen palvelinpuolen verkkokehityksessä, tietojen analysoinnissa, tekoälyssä ja tieteellisessä laskennassa. Kehittäjät käyttävät Pythonia myös tuottavuustyökalujen, pelien ja työpöytäsovellusten luomiseen. (What is Python n.d.)

Python on yhteensopiva monien alustojen kanssa, mukaan lukien Windows, Mac, Linux, Raspberry Pi ja muut. Se käyttää yksinkertaista syntaksia, joka on verrattavissa englannin kieleen ja mahdollistaa vähemmän koodirivejä kuin muut ohjelmointikielet. Python toimii tulkijärjestelmällä, joka mahdollistaa koodin välittömän suorittamisen ja nopeuttaa prototyyppien luomista. Sitä voidaan käsitellä proseduraalisella, oliopohjaisella tai funktionaalisella tavalla. Pythonin syntaksi muistuttaa jonkin verran englannin kieltä, matemaattisella vaikutteella ja on suunniteltu luettavaksi. Toisin kuin muut kielet, jotka käyttävät puolipisteitä ja/tai sulkeita kommentojen viimeistelyyn, Python käyttää uusia rivejä saman toiminnon suorittamiseen. Se määrittelee koodin laajuuden (eli silmukat, funktiot, luokat) sisennyksen avulla, käyttäen välilyöntejä, eikä aaltosulkeita. Python on erityisen joustava, poistaen kovat säännöt ominaisuuksien rakentamiselle ja tarjoaa enemmän ongelmanratkaisujoustavuutta erilaisten menetelmien avulla. Se myös mahdollistaa käyttäjien kääntää ja suorittaa ohjelmia aina ongelmakohtaan asti, koska se käyttää suoritusajasta tyyppien tarkistusta eikä käännösaikaista tarkistusta. (What is Python n.d.)

### 5.1.5 Pytorch

PyTorch, avoimen lähdekoodin koneoppimisen kehys, perustuu Pythoniin ja Torch-kirjastoon. Lua-kielellä kirjoitettu Torch on tunnettu avoimen lähdekoodin koneoppimiskirjasto syvien neuroverkkojen luomiseen. PyTorch toimii suosittuna alustana syväoppimisen tutkimuksessa ja on suunniteltu nopeuttamaan siirtymistä tutkimuksen prototyypityksestä käyttöönottoon. PyTorchin ydin mahdollistaa tensoreiden laskennan, verrattavissa NumPy-taulukoihin, mahdollistaen yleisten n-ulotteisten taulukoiden manipuloinnin numeerisissa laskuissa, joita kiihdytetään tehokkaasti GPU-tuella. Näitä tensoreita, niiden moniulotteisia rakenteita, manipuloidaan rajapintojen kautta monipuolisissa numerolaskuissa. (Yasar n.d.)

PyTorchin yksi merkittävä ominaisuus on TorchScript tuotantoympäristö, joka varmistaa saumattoman siirtymisen eri tilojen välillä samalla optimoiden toiminnallisuutta, nopeutta ja joustavuutta. Dynaaminen graafilaskenta erottaa PyTorchin, mahdollistaen käyttäjien muokata verkon käyttäytymistä lennossa, mikä mahdollistaa joustavuuden koodin suorittamisessa. PyTorch loistaa myös automaattisessa differentiaatiossa, olennaisessa tekniikassa neuroverkon koulutuksessa. Tämä toiminto laskee numeerisesti funktioiden derivaatat taaksepäin neuroverkoissa, virtaviivaistaen neuroverkkojen luomista ja koulutusta. (Yasar n.d.)

PyTorch, joka on rakennettu Pythonin päälle, integroituu saumattomasti laajasti käytettyihin kirjastoihin, kuten NumPyhin, SciPyhin, Numbyhin ja Cythoniin, samalla esitellen erikoistoimintoja, kuten muuttujia, jotka kapseloivat tensoreita gradienttien hallintaan laskennallisissa graafeissa. Lisäksi se käyttää Parametreja käsittelemään tensoreita muistuttavia toimintoja, joita ei voida suoraan saavuttaa muuttujilla, ja hyödyntää moduuleja edustaakseen neuroverkkoja, jotka muodostavat tilallisen laskennan ytimen mukauttamalla erilaisia moduuleja ja parametreja sisäänsä. PyTorch hyödyntää myös funktioita muodostaakseen suhteita muuttujien välillä, toimien ilman sisäistä muistia tilaa tai puskureita varten, minkä vuoksi ne ovat ratkaisevassa roolissa muuttujien linkittämisessä sen laskennallisessa kehysessä. (Yasar n.d.)

### 5.1.6 Hugging Face Transformers

Hugging Face Transformers on nykyaikainen koneoppimisen kirjasto, joka tarjoaa laajan valikoiman esikoulutettuja malleja erilaisten modalityettien, kuten teksti, näkö ja ääni, tehtävien suorittamiseen. Näitä malleja voidaan soveltaa luonnollisen kielen tehtäviin, kuten luokitteluun, informaation erotteluun, kysymyksiin vastaamiseen, tiivistämiseen, kääntämiseen ja tekstien luomiseen yli sadalla kielellä. (Transformers n.d. a.)

Alkuperäinen Transformer-malli, joka esiteltiin vuonna 2017, on toiminut innoittajana monille uusille ja kiinnostaville malleille, jotka ulottuvat luonnollisen kielen käsittelyn (NLP) tehtävien ulkopuolelle. Jotkut näistä malleista hyödyntävät vain kooderia tai dekooderia, kun taas toiset käyttävät molempia. Tämä luokitus mahdollistaa erilaisten Transformer-perheen mallien korkean tason erojen tarkastelun. (The Transformers model family n.d.)

Hugging Face Transformers -kirjasto sisältää myös tehokkaita koulutustekniikoita ja menetelmiä koulutukseen, oli se sitten yhdellä tai usealla näytönohjaimella, prosessorilla tai tensorisuorittimella. Lisäksi kirjasto tukee koulutusta TensorFlow-ympäristössä ja erikoistuneella laitteistolla. Kirjasto on suunniteltu avoimen lähdekoodin periaatteiden mukaisesti, edistään ja demokratisoiden tekoälyä. (Transformers n.d. b.)

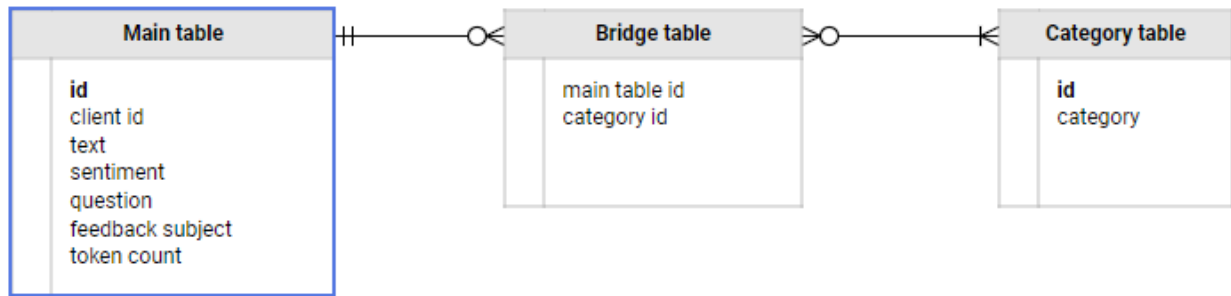
## 5.2 Tietokantarakenne

Analysoitujen tekstien tallentamiseen käytetään MySQL tietokantaa. Sovellukseen käytetään kolmea tietokantataulua:

- Päätaulu, joka sisältää analysoidun tekstin, sentimentin ja muut yleiset kentät, kuten asiakkaaseen liittyvät tiedot.
- Kategoriataulu, joka sisältää kaikki mahdolliset kategoriat ja niiden id:t. Taulu on tarpeellinen, sillä yksi teksti voi sisältää monta kategoriaa, ja kategorioiden päivittäminen ja ylläpito helpottuu.
- Liitostaulu, joka yhdistää pää- ja kategoriataulun. Yksi sarakkeista tallentaa tunnisteet päätaulukosta, luoden yhteyksiä riveihin päätaulukossa. Toinen sarake tallentaa tunnisteet kategoriatietaulusta, luoden yhteyksiä kategorioihin. Jokainen rivi liitostaulussa edustaa suhdetta tietyn päätaulun rivin ja tietyn kategorian välillä kategoriatietaulussa.

Tällainen rakenne mahdollistaa monesta moneen -suhteen päätaulukon ja kategoriatietaulun välillä. Tämä tarkoittaa sitä, että yksi rivi päätaulussa voi olla yhteydessä useisiin kategorioihin ja yksi kategoria voi olla yhteydessä useisiin riveihin päätaulukossa.

Tätä asettelua käytetään yleisesti tietokannoissa käsittelemään tilanteita, joissa useat kohteet liittyvät useisiin muihin kohteisiin, kuten tapauksessa, jossa useat kategoriat voivat olla yhteydessä useisiin riveihin päätaulukossa. Tietokantarakennetta ja taulujen yhteyttä kuvataan kuviossa 1.



Kuvio 1. Tietokantarakenne

### 5.3 Palvelinpuoli

Palvelin hakee tarvittavat tiedot tietokannasta. Näitä tietoja ovat asiakkaiden antamat arvostelut ja niihin liittyvät kategoriat. Selainpuolelta lähetetään asiakkaan tunniste, jota käytetään asiakkaan omien tietojen hakemiseen. Tunnisteen avulla varmistetaan, että asiakas näkee vain omat arvostelunsa ja niihin liittyvät tiedot. Haussa yhdistetään päätaulun ja kategoriataulun tiedot niin, että nämä kaksi tietolähdettä linkitetään toisiinsa. Tämän yhdistämisen tuloksena saadaan koko kuva analysoitujen tekstien sisällöstä ja niihin liittyvistä kategorioista.

Kokonaisuudessaan palvelinpuoli vastaa pyyntöihin asiakkaan antaman tunnisteen perusteella ja yhdistää tietokannasta haetut tiedot siten, että asiakas saa nähtäväkseen vain omat arvostelunsa ja niihin liittyvät tiedot. Tämä mahdollistaa henkilökohtaisemman ja räätälöidymmän näkymän asiakkaan omasta datasta analysoituun sisältöön.

### 5.4 Käyttöliittymä

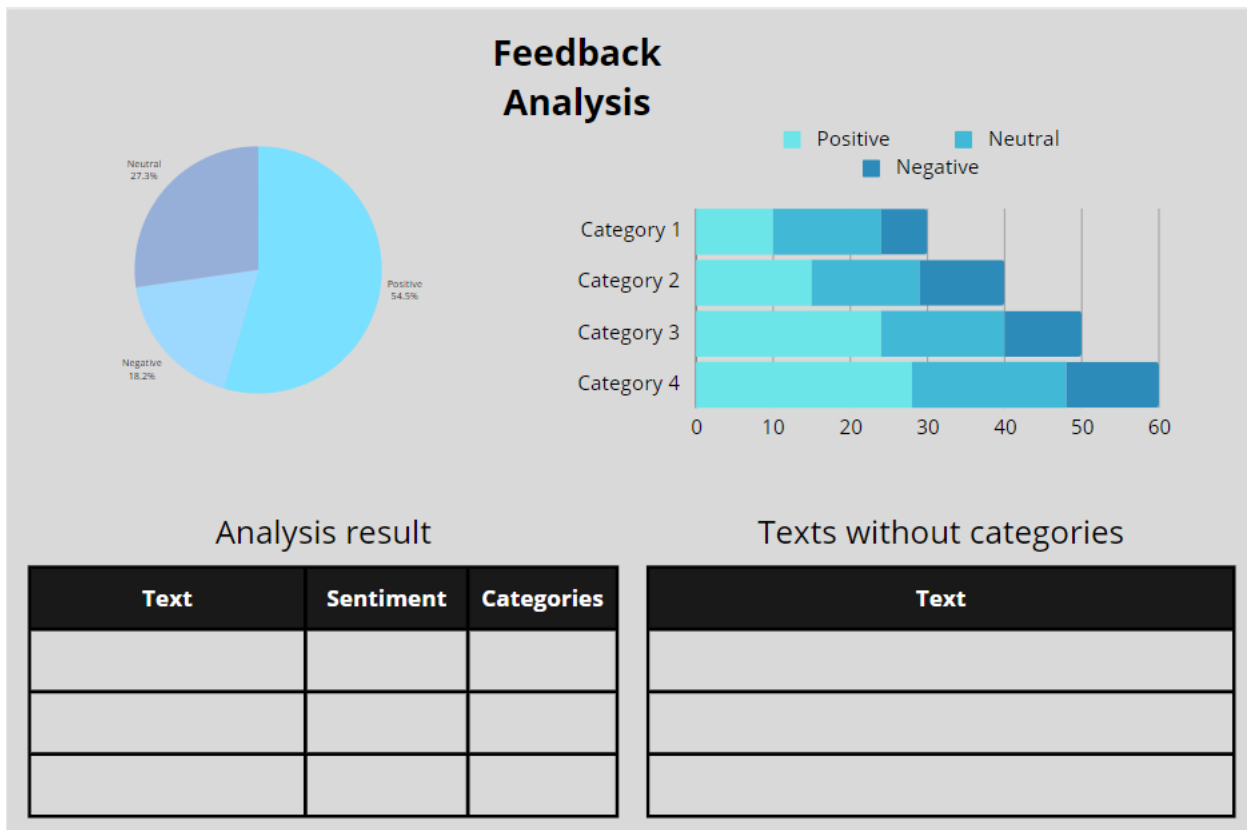
Käyttöliittymä koostuu useasta React-komponentista. Komponenttien käyttämää dataa pystyy suodattamaan kategorioiden ja sentimenttien avulla, jotta käyttäjä voi nähdä halutessaan tarkentaa dataa. Kokonaisuutta kuvataan kuviossa 2.

Ensimmäinen komponentti tarjoaa yleiskuvan arvostelujen tunnetilojen jakautumisesta. Se antaa nopean katsauksen positiivisten, negatiivisten ja neutraalien tunnetilojen suhteelliseen osuuteen koko tietojoukossa.

Toisessa komponentissa esitetään tunnetilojen jakautuminen kussakin kategoriassa. Komponentti syvenyy yksittäisten kategorioiden sisälle ja näyttää, miten tunnetilat jakautuvat niissä. Tämä auttaa tunnistamaan, mitkä tunnetilat hallitsevat tiettyjä kategorioita ja tukee kohdennettua analyysiä.

Kolmas komponentti näyttää yksityiskohtaisen taulukon, jossa on analysoituja arvosteluja niiden vastaavien tunnetilojen ja kategorioiden kanssa. Taulukkomuotoinen esitys tarjoaa kattavan näkymän jokaisesta arvostelusta, mikä helpottaa yksittäisten arvostelujen ja niihin liittyvien tunnetilojen ja kategorioiden tutkimista.

Neljäs on komponentti, joka keskittyy arvosteluihin, joita ei ole onnistuttu kategorisoimaan. Tämä osio tarjoaa mahdollisuuden tarkastella ja ymmärtää, miksi tietyt arvostelut eivät ole saaneet minäänlaista kategoriaa. Komponentti tarjoaa arvokasta tietoa, joka auttaa tunnistamaan ja päättämään syitä sille, miksi tietyt arvostelut ovat jääneet ilman kategoriaa. Se voi auttaa parantamaan luokitteluprosessia ja ymmärtämään paremmin niitä piirteitä tai sisältöjä, jotka ovat vaikeita luokitella. Kategorioimattomien arvostelujen tarkastelu tarjoaa mahdollisuuden hienosäätää luokittelua ja mahdollisesti laajentaa kategorioita, jotta ne vastaisivat paremmin arvostelujen monimuotoisuutta. Tämä osio täydentää kokonais kuvaa ja auttaa parantamaan arvostelujen kattavaa analyysiä.



Kuvio 2. Käyttöliittymän malli

## 5.5 Tiedonkulku

Sovellukseen saapuu uutta tietoa asiakkailta, esimerkiksi arvosteluja tai tekstiä, jonka asiakas syöttää järjestelmään. Saapunut data lähetetään analysoitavaksi rajapintaan. Tässä vaiheessa sovellus käsittelee saadun datan tunnistamalla sanojen tunnetiloja tai liittämällä siihen kategorioita. Analysoitu data tallennetaan tietokantaan, josta sitä voidaan myöhemmin käsitellä ja hakea tarvittaessa. Palvelinpuolella sovellus hakee tallennetun datan tietokannasta ja lähettää sen sitten selainpuolelle. Tässä vaiheessa data siirtyy palvelimen ja selaimen välillä. Selainpuolella saatu data visualisoidaan käyttäjälle. Kuviossa 3 visualisoidaan tiedonkulkua.



Kuvio 3. Sovelluksen tiedonkulku

## 6 Koneoppimismallien hienosäätö

Sovellusta varten luotiin kaksi erillistä ohjelmaa koneoppimismallien hienosäätöä varten. Tehtävät olivat multi-label-luokittelu ja sentimenttianalyysi. Hienosäädössä hyödynnettiin PyTorch -kehystä, sekä Hugging Face Transformers -kirjastoa. Transformers-kirjasto tarjosi valmiiksi opetettuja malleja ja opetusluokkia, jotka olivat hyödyllisiä ja tehokkaita monimutkaisten kielimallien koulutuksessa ja hienosäädössä.

Sovelluksessa hyödynnettiin valmiiksi opetettuja malleja, joissa on valmiiksi opittua tietoa laajasta datamäärästä. Näitä malleja ja niihin liittyviä opetusluokkia käytettiin mallien sovittamiseen ja hienosäätöön. Tämä tarkoitti, että mallit voitiin rakentaa ja muokata vastaamaan haluttuja tehtäviä, useiden kategorioiden luokittelua ja sentimenttianalyysiä.

Käytetyissä malleissa hyödynnettiin merkittyjä datasettejä hienosäätöön. Nämä datasetit tarjosivat tarkasti luokiteltua tietoa, joka toimi oppimisaineistona koneoppimismalleille. Näitä datasettejä käytettiin opettamaan malleille monimutkaisia piirteitä ja yhteyksiä, jotka olivat tarpeen kategorioiden luokittelun ja sentimenttianalyysin suorittamiseen.

Merkityt datasetit olivat keskeisiä mallien koulutuksessa haluttuihin tehtäviin, sillä ne sisälsivät valmiiksi luokiteltua tietoa tarvittavista aiheista. Tämä auttoi malleja sopeutumaan paremmin kyseisten tehtävien erityispiirteisiin ja antoi niille mahdollisuuden oppia tunnistamaan ja ymmärtämään paremmin tietyn tehtävän kontekstia ja ominaisuuksia.

## 6.1 Sentimenttialalyysi

Sentimenttialalyysin tarkoitus on automaattisesti tunnistaa ja arvioida tekstiin ilmaistuja tunteita, mielipiteitä ja asenteita, mikä mahdollistaa tärkeiden tietojen saannin suurista tekstiaineistoista, kuten asiakaspalautteista tai sosiaalisen median viesteistä. Tämä tieto auttaa organisaatioita ymmärtämään paremmin käyttäjien näkemyksiä ja reaktioita sekä tekemään tietoon perustuvia päätöksiä tuotteiden, palveluiden ja markkinoinnin kehittämisessä.

### 6.1.1 Mallin hienosäätö sentimenttialalyysiin

Hienosäädön tavoite oli saada malli tunnistamaan tekstistä sentimentti. Opetukseen käytetty datasetti sisältää teksti ja sentimentti sarakkeet. Sentimentti sarakkeessa 0 edustaa negatiivista sentimenttiä ja 1 positiivista kuvion 4 tavoin. Datasetti sisältää moninaisia virkkeitä ilman mitään erillisiä aiheita tai yhteyksiä joihinkin tiettyihin aloihin, sillä sentimenttien yhtäläisyydet eivät yleensä ole alakohtaisia.

text	sentiment
Aurinkoinen päivä ilahdutti kaikkia puistossa.	1
Uusi kirja tarjosi inspiroivaa ja opettavaista luettavaa.	1
Perheen yhteinen illallishetki loi lämpimiä ja rakkaudentäyteisiä hetkiä.	1
Sateinen sää pilasi suunnitellun ulkoilmapäivän.	0
Hylättyjen suunnitelmien takia pettymys oli selvästi nähtävissä.	0
Huono asiakaspalvelukokemus jätti asiakkaan turhautuneeksi ja tyytymättömäksi.	0

Kuvio 4. Sentimenttialalyysin koulutusaineiston rakenne

Kuvio 5 näyttää opetusohjelman datan valmistelun ja käsittelyn hienosäätämiseksi BERT-pohjaiselle sentimenttialalyysimallille Hugging Face Transformers -kirjastoa käyttäen. Opetusohjelmassa tarkistetaan ensin GPU:n saatavuus tehokkaampaa opetusta varten ja ladataan CSV-muotoisen datasetin Pandas-kirjaston avulla datakehyykseen. Sen jälkeen muunnetaan datasetin sentiment-sarakkeen arvot kokonaisluvuiksi. Hyödyntäen Transformers-kirjastoa, tokenisoidaan datakehys esikoulutetulla BERT-tokenisaattorilla. Tokenisoinnin jälkeen datakehys valmistellaan Transformers-kirjastolle sopivaan muotoon ja jaetaan edelleen koulutus-, testaus- ja validointiaineistoihin käyttäen `train_test_split()` -funktiota.



```

device = "cuda:0" if torch.cuda.is_available() else "cpu"
torch.cuda.empty_cache()

df = pd.read_csv('./fi/train/all_data.csv', sep=';')

df['Sentiment'] = df['Sentiment'].astype(int)
dataset = Dataset.from_pandas(df)

tokenizer = transformers.BertTokenizer.from_pretrained("TurkuNLP/bert-base-finnish-cased-v1")
model = transformers.BertForSequenceClassification.from_pretrained("TurkuNLP/bert-base-finnish-cased-v1",
                                                                    num_labels=2, problem_type="single_label_classification")

def tokenize_function(examples):
    return tokenizer(examples["Text"], truncation=True, padding='max_length')

tokenized_datasets = dataset.map(tokenize_function, batched=True)
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
tokenized_datasets = tokenized_datasets.rename_column('Sentiment', 'label')
tokenized_datasets = tokenized_datasets.remove_columns('Text')
tokenized_datasets = tokenized_datasets.rename_column("label", "labels")

train_testvalid = tokenized_datasets.train_test_split(test_size=0.2)
test_valid = train_testvalid["test"].train_test_split(test_size=0.5)
ds = DatasetDict({
    "train": train_testvalid["train"],
    "test": test_valid["test"],
    "valid": test_valid["train"]})

```

Kuvio 5. Sentimenttialyyisin datan valmistelu ja prosessointi

Kuvio 6 esittää opetusohjelman koulutusvaiheen koodia, jossa määritetään arviointimetriikkafunktio `compute_metrics`, jossa käytetään valittuja metriikoita ja määritellään koulutusparametrit käyttäen `TrainingArguments`-luokkaa. Parametreihin kuuluvat oppimisnopeus, eräkokko ja hakemisto mallin tallentamista varten. Ohjelmassa alustetaan `Trainer`-objekti, joka sisältää mallin, koulutusparametrit, datasetit, tokenisaattorin ja arviointimetriikkafunktion. Malli koulutetaan `trainer.train()`-metodilla ja arvioidaan sen jälkeen validointiaineistolla ennusteiden luomiseksi `trainer.predict()`-metodilla. Lopuksi koulutettu malli tallennetaan määritettyyn hakemistoon `trainer.save_model()`-metodilla.

```

def compute_metrics(eval_pred):
    metric = load_metric("glue", "mrpc")
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)

metric_name = "f1"

training_args = TrainingArguments("sentiment/sentiment_f1",
                                   evaluation_strategy="epoch",
                                   save_strategy = "epoch",
                                   learning_rate=2e-5,
                                   per_device_train_batch_size=4,
                                   num_train_epochs=5,
                                   weight_decay=0.01,
                                   load_best_model_at_end=True,
                                   metric_for_best_model=metric_name,)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=ds["train"],
    eval_dataset=ds["test"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)

trainer.train()
predictions = trainer.predict(ds["valid"])
print(predictions.predictions.shape, predictions.label_ids.shape)
print(predictions)
trainer.save_model()

```

Kuvio 6. Mallin koulutus ja arviointi

### 6.1.2 Sentimenttialyysin päätelmä

Hienosäädetylle mallille luotiin päätelmä (engl. inference), johon syötetään uutta dataa Node.js palvelimelta (ks. liite 1). Päätelmässä käytetään kahta erillistä mallia kielen tunnistuksen perusteella. Jos tunnistettu kieli on englanti, käytetään valmista tunneanalyysimallia, joka on koulutettu englanninkieliselle tekstile. Päätelmässä hyödynnetään transformers.pipeline -metodia englanninkielinen mallin lataamiseen ja tunneanalyysin suorittamiseen syötetekstille. Ennustettu sentimentti johdetaan mallin tuottamista todennäköisyyksistä. Tämä valmiiksi olemassa oleva malli tarjoaa pisteitä sekä positiivisille että negatiivisille tunnetiloille.

Suomen kielelle tunnistetun tekstin tapauksessa päätelmässä vaihdetaan käyttämään hienosäädettyä finBERT-mallia, joka on koulutettu sentimenttianalyysiä varten suomen kielellä. Se tokenisoi syötteen käyttäen suomenkielistä BERT-tokenisaattoria ja siirtää sen hienosäädetyn mallin läpi. Sentimentti määritetään mallin antamien todennäköisyyksien perusteella.

## 6.2 Multi-label-luokittelu

Multi-label-luokittelu (engl. multi-label classification) on koneoppimisen tehtävä, jossa pyritään ennustamaan tai luokittelemaan datapisteitä useiden mahdollisten luokkien tai kategorioiden joukkoon samanaikaisesti. Tämä tarkoittaa sitä, että jokainen datapiste voi kuulua yhteen tai useampaan kuin yhteen luokkaan samanaikaisesti. Tämä eroaa perinteisestä yksittäisen luokan luokittelutehtävästä, jossa jokainen esimerkki voi kuulua vain yhteen luokkaan. Sen tarkoituksena on mahdollistaa monimutkaisempien ja monikerroksisten luokittelujen tekeminen todellisissa tilanteissa, joissa yksi datapiste voi olla useiden erilaisten luokkien osalta relevantti.

### 6.2.1 Multi-label-luokittelun koulutusaineisto

Hienosäätöön käytetty datasetti sisältää text-sarakkeen ja oman sarakkeen jokaiselle kategorialle. Kategoriasarakkeissa 1 edustaa kategorian yhteyttä tekstiin ja 0 sen puuttumista. Esimerkkinä kuvio 7 datasetin rakenteesta.

easy to use and lots of	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
Good clear pictures!	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
The concept of preserving	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
pictures and links	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
the colors animations	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
Good to have information	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
the information was very	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
everything was amazing	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Beautiful photos and	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
The consistent visuals	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
The design. Many things	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Kuvio 7. Luokitteluun käytetyn datasetin rakenne

Multi-label-luokittelu on kompleksinen ongelma, koska se vaatii useiden luokkien samanaikaista ennustamista, mikä edellyttää syvällistä ymmärrystä monista erilaisista konteksteista. Tämän kompleksisuuden käsittelemiseksi koulutusdatan on kohdennettava tietyille teollisuudenaloille. Tämä kohdistettu lähestymistapa mahdollistaa teollisuuskohtaisten hienovaraisuuksien, termino-

logian ja skenaarioiden sisällyttämisen, jotka ovat olennaisia, jotta malli ymmärtäisi kyseisten alojen monimutkaisuudet tehokkaammin. Mukauttamalla koulutusaineistoa tiettyjen teollisuudenalojen perusteella datan laatu ja relevanssi nousevat, varmistaen, että malli kohtaa sen tarkoitettujen sovellusalueiden edustavia skenaarioita oppimisprosessin aikana, mikä tehostaa mallin tarkkuutta ja suorituskkyä näillä aloilla.

### **6.2.2 Mallin hienosäätäminen multi-label-luokitteluun**

Hienosäädön tavoite on saada malli tunnistamaan kaikki yhteen tekstiin liittyvät ennalta määritetyt kategoriat. Kuviossa 8 näytetään kategorisoinnin koulutusaineiston valmistelun koneoppimis-mallin koulutusta varten. Koulutusaineiston valmisteltiin samoilla menetelmillä, kuten luvussa 6.1.1. Luokittelemisen helpottamiseksi tunnisteet tunnistetaan ja kartoitetaan niiden vastaaviin indekseihin arviointi- ja purkuprosesseja varten mallin koulutuksen aikana.

Koulutusaineisto esikäsitellään ja tokenisoidaan käyttäen käytetyn mallin tokenisaattoria. Ohjelmassa funktio `preprocess_data()` koodaa tekstin ja valmistelee siihen liittyvän luokittelutiedon matriisiksi kullekin esimerkille, varmistaen, että data on muotoiltu sopivaksi mallin syötteeksi. Lopuksi käsitelty datasetti muunnetaan muotoon, joka sopii PyTorchiiin, mikä on olennaista integraation kannalta koulutusputkeen ja mahdollistaen tehokkaan koulutuksen luokittelutehtäviin.

```

df = pd.read_csv('../fi/train/multilabel_dataset.csv', sep=";", header=None, names=[
    'Text', 'Matkailu', 'Matkakohde', 'Majoitus',
    'Aktiviteetti', 'Ravintola', 'Nähtävyys',
    'Liikkuminen ja matkustaminen', 'Tapahtuma',
    'Varaus tai tilaus', 'Kestävyys, vastuullisuus ja terveysturvallisuus',
    'Peruutusehdot', 'Verkkokauppa tai OTA-kanava',
    'Tuotetieto ja tuotteen valinta', 'Maksaminen',
    'Toimitus', 'Asiakaspalvelu', 'Markkinointi'])

dataset = Dataset.from_pandas(df)

train_testvalid = dataset.train_test_split(test_size=0.2)
test_valid = train_testvalid["test"].train_test_split(test_size=0.5)
dataset = DatasetDict({
    "train": train_testvalid["train"],
    "test": test_valid["test"],
    "valid": test_valid["train"]})

labels = [label for label in dataset["train"].features.keys() if label not in ['Text']]
id2label = {idx:label for idx, label in enumerate(labels)}
label2id = {label:idx for idx, label in enumerate(labels)}

tokenizer = AutoTokenizer.from_pretrained("TurkuNLP/bert-base-finnish-cased-v1")

def preprocess_data(examples):
    text = examples["Text"]
    encoding = tokenizer(text, padding="max_length", truncation=True, max_length=128)
    labels_batch = {k: examples[k] for k in examples.keys() if k in labels}
    labels_matrix = np.zeros((len(text), len(labels)))
    for idx, label in enumerate(labels):
        labels_matrix[:, idx] = labels_batch[label]

    encoding["labels"] = labels_matrix.tolist()

    return encoding

encoded_dataset = dataset.map(preprocess_data, batched=True, remove_columns=dataset["train"].column_names)

example = encoded_dataset["train"][3]

tokenizer.decode(example['input_ids'])
example['labels']
[id2label[idx] for idx, label in enumerate(example['labels']) if label == 1.0]

encoded_dataset.set_format("torch")

```

Kuvio 8. Multi-label-luokittelun koulutusaineiston valmistelu ja prosessointi

Ohjelman opetusosuus määrittää mallin `AutoModelForSequenceClassification.from_pretrained()`-toiminnolla, jotta se pystyy käsittelemään moniluokittelua (ks. liite 2). Tässä vaiheessa määritellään myös luokkien määrä, luokkien kartoitukset ja otetaan näytönohjain käyttöön laskentaa varten, jos sellainen on saatavilla, muuten käytetään prosessoria. Määritellään koulutuskonfiguraatio, johon kuuluvat oppimismennopeus, eräkokko, koulutuskierrosten määrä ja tarkkuutta mittaavat metriikat.

Opetusohjelma sisältää myös funktiot `multi_label_metrics()` ja `compute_metrics()`, jotka helpottavat metriikoiden, kuten F1-pistemäärän, ROC AUC:n ja tarkkuuden laskemista mallin ennusteiden ja todellisten luokkien perusteella. Trainer-luokka, joka sisältää määritetyn mallin, koulutusparametrit, datasetit koulutusta ja arviointia varten, tokenisaattorin ja metriikoiden laskentafunktiot. Trainer suorittaa mallin koulutusprosessin, arvioinnin määritetyillä metriikoilla ja tallentaa koulutetun mallin.

### 6.2.3 Multi-label-luokittelun päätelmä

Hienosäädetyille malleille luotiin päätelmä, johon syötetään uutta dataa Node.js palvelimelta (ks. liite 3). Jokaisen syötteen kohdalla käytetty malli määräytyy tunnistetun kielen perusteella. Jos kieli tunnistetaan englanniksi, valitaan englanninkielinen malli luokittelua varten, sekä siihen liittyvän tokenisaattori tekstin käsittelyyn. Vastaavasti, jos havaittu kieli on suomi, valitaan suomenkielinen malli, sekä siihen liittyvää tokenisaattori.

Syöte tokenisoidaan ja koodataan valitulla tokenisaattorilla, ja käsitelty syöte syötetään valittuun malliin luokittelua varten. Malli tuottaa logitteja, jotka käyvät läpi sigmoid-funktion saadakseen todennäköisyyspisteet jokaiselle tunnisteelle. Tunnisteet, jotka ylittävät asetetun kynnyksen, katsotaan syötteeseen liittyväksi kategoriaksi. Ennusteet muutetaan vastaaviksi tunnistekategorioiksi ennalta määritettyjen tunnisteiden perusteella. Jos mikään kategoria ei ylitä kynnystä, se palautetaan "No categories". Lopulliset tulokset ja syöte palautetaan JSON-muodossa.

## 7 Back-end toteutus

Back-end toteutus tehtiin käyttäen Node.js ajoympäristöä. OpenAin rajapintaan lähetetään POST-pyyntö osoitteeseen <https://api.openai.com/v1/chat/completions> kuvion 9 näyttämällä tavalla. Pyyntö sisältää kehotteen, jossa on ohjeet halutulle mallille sekä analysoitavan tekstin.

```
const configuration = new Configuration({
  apiKey: process.env.OPENAI_API_KEY,
});
const openai = new OpenAIApi(configuration);

async function getCompletion(prompt, model = "gpt-3.5-turbo") {
  const messages = [{ role: "system", content: prompt }];
  const response = await openai.createChatCompletion({
    model: model,
    messages: messages,
    temperature: 0,
  });
  const result = [response.data.choices[0].message.content, response.status];
  return result;
}
```

Kuvio 9. Pyyntö OpenAi rajapintaan

Rajapinta palauttaa vastauksen JSON muodossa ja itse tulos parsitaan tekstistä. Vastauksessa on myös pyynnön ja vastauksen input ja output tokenien määrä, kuten kuviossa 10 esitetään. Input tokenit tulevat kehoitteen pituudesta ja output tokenit vastauksen pituudesta, molemmilla tokenilla on omat hinnat per 1000 tokenia. Tuloksesta tallennetaan sentimentti, kategoriat ja käytettyjen tokenien määrä, jotta käytön määrää voidaan seurata tarkasti.

```
{
  id: 'chatcmpl-8UFcM1KKF2q5a9W7t17wmVFtA6WQk',
  object: 'chat.completion',
  created: 1702220566,
  model: 'gpt-3.5-turbo-0613',
  choices: [ { index: 0, message: [Object], finish_reason: 'stop' } ],
  usage: { prompt_tokens: 736, completion_tokens: 95, total_tokens: 831 },
  system_fingerprint: null
}
```

Kuvio 10. Rajapinnan vastaus

## 7.1 Rajapinnan käyttö

Sovelluksessa on tarve analysoida suuria määriä dataa, mikä edellyttää lukuisia pyyntöjä OpenAI-rajapintaan lyhyen aikavälin sisällä. Rajapinnassa on käyttörajoituksia, jotka voivat täytyä suuren käytön seurauksena, mikä saattaa aiheuttaa 503-virheitä tai rajapinnan yleisestä suuresta käytöstä

aiheutuvia 502-virheitä. Jokainen pyyntö rajapintaan kuitenkin maksaa, minkä vuoksi ohjelmaan on rakennettu kuviossa 11 esitettävä turvamekanismi estämään sen kaatuminen näiden virheiden vuoksi ennen kuin kaikki tekstit on analysoitu.

Kun virheitä tapahtuu, ohjelma odottaa lyhyen aikavälin ennen kuin se lähettää uuden pyynnön. Jos virheet toistuvat kolme kertaa peräkkäin, ohjelma keskeyttää pyyntöjen lähettämisen. Tämä toiminnallisuus estää ylikuormituksen rajapinnassa. Lisäksi jokaisen uudelleenyrityksen odotusaikaa kasvatetaan asteittain, mikä auttaa välttämään toistuvia virheitä samanlaisissa tilanteissa.

```
const maxRetries = 3;
const delayMs = 2000;
let response;

for (let attempt = 0; attempt < maxRetries; attempt++) {
  try {
    response = await getCompletion(prompt);
    if (response[1] === 200) {
      break;
    }
  } catch (error) {
    console.log(error);
  }
  await new Promise((resolve) => setTimeout(resolve, delayMs * attempt));
}

if (response[0] && response[1] === 200) {
  const analysisResult = response[0];
  resultArray.push({
    id: transformSentence(review),
    client_id: client_id,
    text: review,
    sentiment: sentimentResult(analysisResult),
    categories: categoriesResult(analysisResult),
    question: question,
    token_count: token_count,
  });
} else {
  console.log(`Maximum retries exceeded for review index ${i}. Request failed.`);
}
```

Kuvio 11. Analysoinnin turvamekanismi



## 7.2 Kehotteiden rakentaminen

Rakennettaessa kehoitteita rajapintaan, kehote voidaan jakaa erilaisiin osiin, joilla on roolit "system" tai "user". Rooli "system" liitetään ohjeisiin, jotka lähetetään rajapintaan automaattisesti, kun taas "user"-roolilla käyttäjä voi antaa omia syötteitä. Koska sovelluksessa loppukäyttäjät eivät voi lähettää omia syötteitä, kehote saa roolin "system", joka sisältää koko kehoitteen.

Kehotteiden luominen on osittain kokeilua ja virheiden kautta, koska generatiiviset mallit voivat tuottavaa erilaisia vastauksia pelkästään sanajärjestyksen muuttuessa. Yksinkertaisemmat ohjeet eivät ole yhtä herkkiä tälle vaihtelulle kuin monimutkaisemmat, kuten esimerkiksi pelkkä tunneanalyysi.

Sovelluksessa tehdään sekä tunneanalyysi, että luokittelu samassa kehoitteessa. Kategoriat jaetaan pää- ja alakategorioihin, ja kehoitteessa annetaan konteksti siihen, mitä teksti käsittelee ja mihin kysymykseen vastataan. Koska kategorioiden määrä on suuri, kehoitteeseen on ensin annettava nämä kategoriat toimimaan referenssinä kategorisoinnissa tietokantataulun tavoin, jotta malli osaa luokitella oikein.

Vastausten tulee olla aina samassa muodossa, ja tämäkin on tarkennettava kehoitteessa. Yksinkertaisessa kehoitteessa voi pyytää vastauksen JSON-muodossa, jolloin sen parsiminen on helppoa. Sovelluksessa käytetään kuitenkin monimutkaisempaa kehotetta, ja JSON-muodossa palauttaminen vaikutti luokittelun tarkkuuteen kielteisesti. Ohjeena oli palauttaa kategoriat JavaScript-taulukossa "category"-kentän alle ja tunneanalyysi "sentiment"-kentän alle, jotta vastauksien jäsentäminen olisi yhtenäistä ja tuloksien dynaaminen parsiminen olisi mahdollista.

## 7.3 Tulosten tallennus

Tekstit, joita halutaan analysoida, haetaan tietokantataulusta ja lähetetään sitten OpenAI-rajapintaan kehoitteena. OpenAI-rajapinta palauttaa vastauksen, joka sisältää kaikki analyysin tulokset yhdessä tekstissä. Tämä vastaus käsitellään parsimalla eli erittelemällä tietoa regex-mallien avulla kuviossa 12 esitetyllä tavalla, jotta saadaan eroteltua halutut tiedot laajasta vastaustekstistä.

```

const extractSentiment = (result) => {
  const sentimentRegex = /Sentiment: (\w+)/;
  const sentimentMatch = result.match(sentimentRegex);
  return sentimentMatch ? sentimentMatch[1] : null;
};

const extractCategories = (result) => {
  const categoriesRegex = /Categories: \[(.*?)\]/;
  const categoriesMatch = result.match(categoriesRegex);
  return categoriesMatch ? categoriesMatch[1].split(", ") : [];
};

```

Kuvio 12. Tuloksien parsiminen

Ennalta määritellyt kategoriat haetaan erillisestä tietokantataulusta, jotta niistä saadaan niille määritetyt tunnisteet (id:t). Kuviossa 13 esitetään, kuinka kategorioita verrataan vastauksessa löydettyihin kategorioihin. Kun vastauksista tunnistetaan kategorioita, luodaan JavaScript-objekteja, jotka sisältävät näiden kategorioiden tunnisteet sekä vastauksen tunnisteet. Nämä objektit tallennetaan liitostauluun, joka toimii linkkinä analysoitujen tekstien ja niiden kategorioiden välillä, sisältäen vain analysoitujen tekstien tunnisteet ja vastaavien kategorioiden tunnisteet. Päätauluun tallennetaan itse analysoitu teksti sekä analyysin tuloksena saadut tiedot, kuten sentimentin arvo, käytettyjen tokenien määrä, palautteen aihe, kysymys, johon vastattiin, ja tarvittavat tiedot asiakkaan tunnistamiseen.

```

const categoryLookup = {};
for (const obj of fetchedCategories) {
  categoryLookup[obj.category] = obj.id;
}

const resultArray = [];
for (const obj of result) {
  const categoryIDs = obj.categories.map((category) => categoryLookup[category]);
  for (const categoryID of categoryIDs) {
    if (categoryID !== undefined) {
      resultArray.push({ id: `${obj.id}_${categoryID}`, answer_id: obj.id, categoryID });
    }
  }
}

```

Kuvio 13. Liitostaulun tietojen luonti

Tällä tavalla luodaan linkityksiä analysoitujen tekstien ja niiden kategorioiden välille sekä tallennetaan tarvittava tieto päätauluun, jotta voidaan säilyttää analysoitu teksti ja siihen liittyvät tiedot jatkotarkastelua ja raportointia varten.

## 8 Front-end toteutus

Käyttöliittymä toteutettiin käyttäen JavaScriptiä ja React-kirjastoa. Reactin avulla voitiin hallita komponenttien elinkaarta ja tarpeita, jotka liittyvät komponentin eri vaiheisiin elinkaaren aikana. Tämä mahdollisti toimintojen suorittamisen ja hallinnan kuten komponenttien päivittämisen, mikä oli olennaista käyttöliittymän toiminnan kannalta.

### 8.1 Kirjastot

#### Material-UI

Material-UI on React-komponenttikirjasto, joka sisältää valikoiman valmiita komponentteja, kuten nappeja, ikoneita, lomakkeita, modaaliruutuja, taulukoita ja työkaluvihjeitä, jotka ovat muokattavissa eri käyttötarkoituksiin. Kirjasto noudattaa Material Designin periaatteita ja tuottaa modernin ja visuaalisesti selkeän käyttöliittymän. Komponentit on suunniteltu helposti yhdistettäväksi, jotta saavutettaisiin yhtenäinen ilme. Niiden ominaisuudet, kuten propit ja teemat, mahdollistavat värien, typografian ja ulkoasun säätämisen. Komponentit ovat suunniteltu responsiivisiksi mikä tekee sovellusten toiminnasta sujuvaa eri laitteilla ja näytön koissa. (Material UI – Overview n.d.)

#### ApexCharts

ApexCharts on nykyaikainen kaaviokirjasto, joka on suunniteltu auttamaan kehittäjiä luomaan visuaalisesti houkuttelevia ja vuorovaikutteisia visualisointeja verkkosovelluksille. Se toimii avoimen lähdekoodin projektina MIT-lisenssillä, mikä mahdollistaa ilmaisen käytön kaupallisissa sovelluksissa. Sen tärkeimpiin ominaisuuksiin kuuluvat monipuoliset kaaviotyyppit, jotka mahdollistavat erilaisten kaavioiden yhdistämisen tehokkaaseen datan esittämiseen. Kaaviot ovat täysin responsiivisia ja toimivat niin työpöytäkoneilla, tableteilla kuin mobiililaitteillakin. (Apexcharts.js n.d.)

## 8.2 Käyttöliittymän komponentit

### 8.2.1 Valikko

Käyttöliittymässä toteutettiin valintakomponentti Material-UI:n avulla, joka toimii johdonmukaisena ja uudelleenkäytettävänä osana kaikissa visualisoinneissa. Tämä valintakomponentti toimii avattavana valikkona, tarjoten käyttäjille standardoidun tavan tehdä valintoja tai suodattaa tietoa koko käyttöliittymän laajuudella.

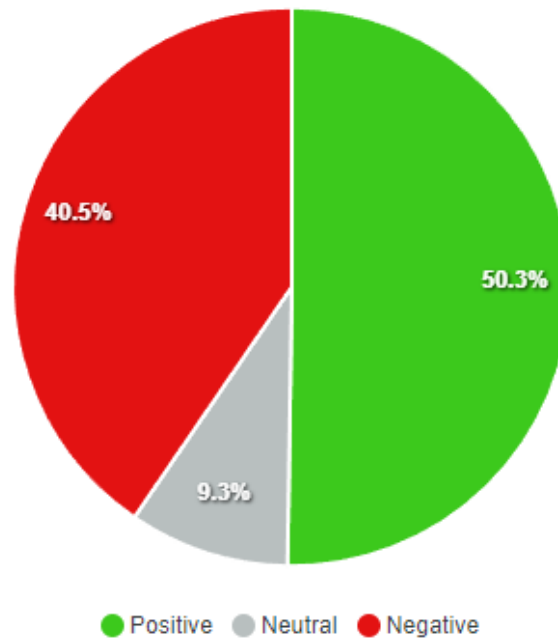
Material-UI:n valintakomponenttia käyttämällä varmistettiin yhtenäinen ja intuitiivinen käyttökokemus. Se mahdollistaa käyttäjille helpon valintojen tekemisen, kuten tunteiden ja kategorioiden suodattamisen, palautteiden aiheen tai tietojen lajittelun, eri osioissa ilman epäjohdonmukaisuuksia toiminnallisuuksissa tai suunnittelussa.

Tämä lähestymistapa ylläpitää visuaalista yhtenäisyyttä, sekä parantaa käytettävyyttä tarjoamalla tutun ja standardoidun käyttöliittymäelementin. Se edistää tehokkuutta ja helppokäyttöisyyttä käyttäjillemme navigoidessaan eri osioihin ja toiminnallisuuksiin käyttöliittymässä.

### 8.2.2 Graafit

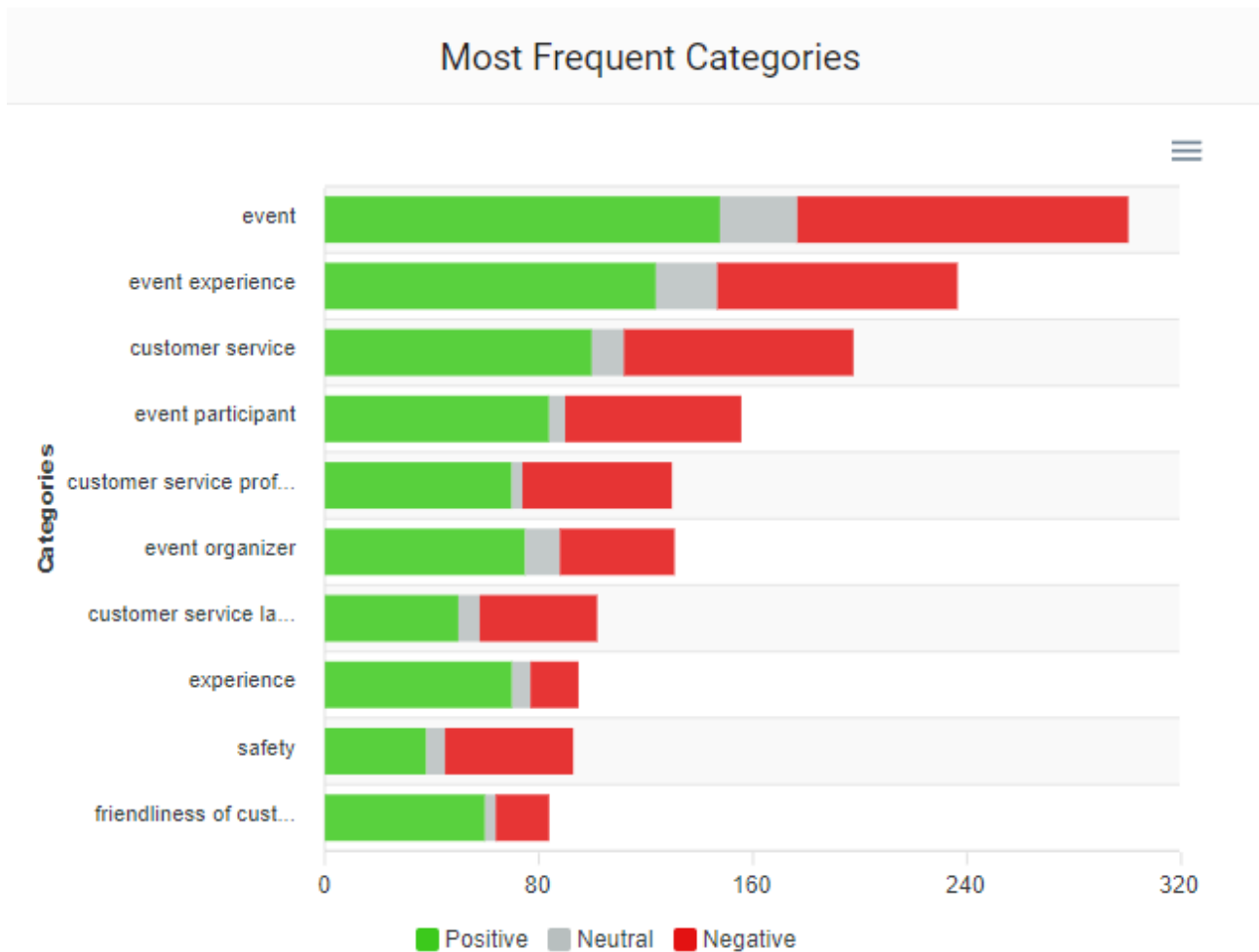
Sentimenttiosuuksien jakautuminen visualisoitiin piirakkakaaviolla, joka tehtiin ApexChartsin avulla kuvaamaan yleistä sentimentti jakaumaa kaikkien arvosteluiden keskuudessa (ks. kuvio 14). Tämä kaavio tarjoaa tiiviin ja intuitiivisen näkymän, näyttäen positiivisten, negatiivisten ja neutraalien tunteiden jakautumisen koko aineistossa. Käyttäjät voivat helposti hahmottaa kunkin tunnekkategorian osuuden yhdellä silmäyksellä, mikä auttaa nopeasti ymmärtämään yleiset tunnetrendit.

### Sentiment Summary



Kuvio 14. Sentimenttiosuuksien jakauman visualisointiin käytetty piirakkakaavio

Sentimenttijakaumaa yksittäisissä kategorioissa visualisoitiin vaakasuuntaisen pinotun pylväskaavioilla, joka tehtiin myös ApexChartsin avulla (ks. kuvio 15). Kaavio tarjoaa yksityiskohtaisen jaotteen, näyttäen tunnevariaatioita kussakin kategoriassa. Käyttäjät voivat visualisoida positiivisten, negatiivisten ja neutraalien tunnetilojen jakauman vaakasuunnassa pinottuna, mikä mahdollistaa vertailevan analyysin eri kategorioiden välillä. Tämä visuaalinen esitys auttaa käyttäjiä tunnistamaan tunnemalleja, jotka ovat erityisiä kullekin kategorialle, helpottaen kohdennettuja näkemyksiä ja analyysiä.



Kuvio 15. Sentimenttijauman visualisointi kategorioittain

### 8.2.3 Taulukot

Käyttöliittymässä käytetään uudelleenkäytettävää taulukko komponenttia, joka on kehitetty Material-UI:n avulla ja oli jo valmiina yrityksen resursseissa. Tämä taulukko komponentti toimii perustana kaikille käyttöliittymässä esiintyville taulukoille. Taulukko on tehty käyttäen Material-UI:n taulukko komponentteja. Tämän hyödyntäminen takaa yhtenäisyyden ja tehokkuuden koko käyttöliittymän laajuudella. Sekä yksityiskohtainen palautteiden analyysitaulukko että kategorioimattomien palautteiden taulukko hyödyntävät tätä standardisoitua taulukko komponenttia.

Feedback		Sentiment	Categories
		All	All
Feedback ↓			
[REDACTED]	positive	event, customer service, friendliness of customer service, customer service professionalism, customer service language, event experience, event organizer, event participant	
[REDACTED]	neutral	event, travel advice, event experience, event organizer, Guidance, Language	
[REDACTED]	positive	event, customer service, customer service professionalism, event experience, event organizer	
		Rows per page: 10	1-10 of 414

Kuvio 16. Yksityiskohtaisten analyysien esittämiseen käytetty taulukko

Hyödyntämällä jo valmiina olevaa ja standardisoitua taulukko komponenttia varmistettiin yhtenäisyys suunnittelussa, toiminnallisuuksissa ja käyttäjäkokemuksessa koko sovelluksen laajuudella. Tämä helpottaa ylläpitoa, edistää yhdenmukaisuutta ulkoasussa ja virtaviivaistaa kehitystoimia, koska tätä komponenttia voidaan käyttää useissa osioissa ilman tarvetta päällekkäiselle tai monistetulle koodille, kuvioissa 16 ja 17 on käytetty samaa taulukko komponenttia eri tietojen esittämiseen.

Not Categorized Feedback		Sentiment
		All
Feedback ↓		Sentiment
[REDACTED]		Negative
[REDACTED]		Negative
[REDACTED]		Negative
[REDACTED]		Negative
[REDACTED]		Positive
Rows per page: 5		121-125 of 163
		< >

Kuvio 17. Kategorioimattomien palautteiden esittämiseen käytetty taulukko

### 8.3 Käyttäjävuorovaikutus

Käyttöliittymä mahdollistaa käyttäjien aktiivisen osallistumisen visualisointeihin, antaen heille mahdollisuuden zoomata graafeja, suodattaa dataa tai keskittyä tiettyihin datapisteisiin, saavuttaen näin yksityiskohtaisemman analyysin. Tämä vuorovaikutteinen toiminnallisuus on integroitu kaikkiin käyttöliittymän komponentteihin tapahtumapohjaisella ohjelmoinnilla.

Tämän toteutuksen avulla käyttäjillä on joustavuus vuoro vaikuttaa dynaamisesti visuaalisten datan esitysten kanssa. He voivat zoomata saadakseen tarkemman näkymän, käyttää suodattimia näytettävän tiedon hienosäätöön tai syventyä tiettyihin aineistoihin yksityiskohtaisemman ymmärryksen saamiseksi. Tämä vuorovaikutustaso parantaa käyttökokemusta, antaen käyttäjille mahdollisuuden tutkia ja tehokkaasti poimia oivalluksia dataesityksistä, joita kojelaudassa on tarjolla.



## 9 Pohdinta

Opinnäytetyön tavoitteena oli kehittää tekstianalyysisovellus asiakaspalautteiden analysointia ja niistä saatujen tulosten visualisointia varten. Asiakaspalautteista haluttiin selvittää erityisesti sentimentti ja palautteisiin liittyvät kategoriat. Työhön kuului eri tekoäly ratkaisujen tutkiminen ja vertailu, ja tuloksien perusteella parhaimman valitseminen. Opinnäytetyön aikana saatiin käyttöönottoon valmis sovellus, joka täyttää kaikki sille asetetut tavoitteet.

### 9.1 Tulokset

Kuten luvussa 2 mainitaan, analyysin tarkkuus ja ratkaisun skaalautuvuus olivat tärkeitä ominaisuuksia lopulliselle ratkaisulle. Alkuperäinen idea oli tehdä omat koneoppimismallit hienosäätämällä esiopetettuja malleja haluttuihin tehtäviin. Tärkeimpiin kriteereihin kuului suomen kielen ymmärrys, sillä ilman sitä ei analyysiä voisi saada tarpeeksi tarkaksi. Sentimenttianalyysin tarkkuus oli suomen kielellä yli 90 % ja koska kieli ei kehity kovin nopeaa, päivittäminen ei olisi ollut kovin suuri ongelma. Luokittelumallin koulutuksessa käytettiin 20 eri luokkaa ja sen tarkkuus oli noin 80 %. Tämä on kohtalaisen hyvä, mutta uusien luokkien lisääminen olisi haastavaa ilman, että mallin tarkkuus kärsisi. Tässä lähestymistavassa ongelmaksi tuli opetusaineistoon käytettävän datan saatavuus ja datasettien merkitseminen. Tästä syystä omien koneoppimismallien päivittäminen olisi tullut erittäin kalliiksi ja työlääksi, joten tämän ratkaisun skaalautuvuus ei olisi ollut kovin hyvä.

Ratkaisujen tutkimisen aikana toiseksi vaihtoehdoksi nousi OpenAin rajapinta. Omien koneoppimismallien tuloksia verrattiin OpenAin gpt-3.5 mallin tuloksiin, ja lopulliseksi ratkaisuksi päätettiin OpenAin rajapinnan käyttö. Kokonaisuudessaan analysointi oli tarkempi OpenAin rajapintaa käyttäen ja sen käyttöönotolla päästiin eroon datan keruusta, datasetin merkitsemisestä, koulutusprosessista ja hienosäädettyjen mallien rajapinnasta. Lopullisessa sovelluksessa on käytössä 99 eri luokkaa ja sentimentteinä positiivinen, negatiivinen ja neutraali. Näin monen luokan käyttö hyvällä tarkkuudella omia koneoppimismalleja käyttäen olisi ollut erittäin haastavaa, kallista ja aikaa vievää. Analyysi saatiin tarpeeksi tarkaksi, ja koska rajapinta poistaa koulutusprosessin tarpeen, ratkaisu on myös hyvin skaalautuva.

Mielestäni tulosten esittäminen ja visualisointi onnistui hyvin. Kojelaudasta saa kaiken relevantin tiedon pelkällä silmäilyllä, ja on mahdollisuus etsiä ja tarkastella yksityiskohtaista tietoa. Käyttöliittymän ulkoasussa voisi silti olla vielä parannettavaa.

## **9.2 Jatkokehitys**

Sovelluksessa on jatkokehitys mahdollisuuksia, OpenAin rajapinnan käyttö mahdollistaa melkein minkä tahansa luonnollisen kielen tehtävän. Visualisointia voidaan myös vielä kehittää, jos sellainen tarve tulee. Sovelluksessa otettiin lopussa käyttöön uusi gpt-4 malli, joka paransi huomattavasti suorituskykyä, mutta sitä ei vielä opinnäytetyön aikana ehditty kokeilemaan kunnolla, joten se on ainakin yksi selkeä jatkokehityskohde. Vaikka sovellukseen ei suoraan tulisi jatkokehitystä, toimii se hyvänä pohjana uusille luonnollisen kielen tehtäviä käyttäville sovelluksille.

## Lähteet

About Node.js. N.d. Nodejs-verkkosivustolla. Viitattu 28.11.2023. <https://nodejs.org/en/about>

Apexcharts.js. N.d. Apexcharts-verkkosivusto. Viitattu 29.11.2023. <https://apexcharts.com/>

Arya, N. 2022. What is Transfer Learning? Artikkelin KD nuggets sivustolla. Julkaistu 5.1.2022. Viitattu 21.8.2023. <https://www.kdnuggets.com/2022/01/transfer-learning.html>

Brown, S. 2021. Machine learning, explained. Artikkelin MIT Sloan sivustolla. Julkaistu 21.4.2021. Viitattu 6.8.2023. <https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>

Drake, M. 2020. What is MySQL? DigitalOcean verkkosivut. Julkaistu 14.12.2020. Viitattu 23.11.2023. <https://www.digitalocean.com/community/tutorials/what-is-mysql>

Getting started with React. N.d. Mdn web docs. Viitattu 23.11.2023. [https://developer.mozilla.org/en-US/docs/Learn/Tools\\_and\\_testing/Client-side\\_JavaScript\\_frameworks/React\\_getting\\_started](https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started)

Hennings, M. 2023. Approach to AI: When to Use Prompt Engineering, Embeddings, or Fine-tuning. Entry Point 24.7.2023. Viitattu 11.9.2023. <https://www.entrypointai.com/blog/approaches-to-ai-prompt-engineering-embeddings-or-fine-tuning/>

Hore, S. 2023. What are Large Language Models (LLMs)? Artikkelin Analytics Vidhya sivustolla. Julkaistu 13.3.2023. Viitattu 8.8.2023. <https://www.analyticsvidhya.com/blog/2023/03/an-introduction-to-large-language-models-llms/>

Introduction. N.d. Dokumentaatio OpenAin api-reference sivustolla. Viitattu 21.8.2023. <https://platform.openai.com/docs/api-reference/introduction>

Introduction. N.d. Dokumentaatio OpenAin dokumentaatio sivustolla. Viitattu 21.8.2023. <https://platform.openai.com/docs/introduction>

Material UI – Overview. N.d. Dokumentaatio mui-verkkosivustolla. Viitattu 29.11.2023. <https://mui.com/material-ui/getting-started/>

Meet the team. N.d. React-verkkosivustolla. Viitattu 23.11.2023. <https://react.dev/community/team>

Natural Language Processing. N.d. DeepLearning.AI verkkosivut. Viitattu 8.8.2023. <https://www.deeplearning.ai/resources/natural-language-processing/>

Node.js introduction. N.d. Artikkelin GeeksforGeeks-verkkosivustolla. Viitattu 28.11.2023. <https://www.geeksforgeeks.org/node-js-introduction/>

Pricing. N.d. OpenAi-verkkosivusto. Viitattu 21.8.2023. <https://openai.com/pricing>

Quick Concepts: Fine-tuning in Generative AI. N.d. Innodata. Viitattu 21.8.2023. <https://innodata.com/quick-concepts-fine-tuning-in-generative-ai/>

The Transformers model family. N.d. Dokumentaatio Hugging Face verkkosivustolla. Viitattu 28.11.2023. [https://huggingface.co/docs/transformers/model\\_summary](https://huggingface.co/docs/transformers/model_summary)

The transformers: a revolution in natural language processing. 2023. Julkaisu StendhalGPT sivustolla. Julkaistu 14.6.2023. Viitattu 8.8.2023. <https://www.stendhalgpt.fr/en/transformers-revolution-natural-language-processing/>

Torres, M. 2023. What is fine-tuning and how does it work in neural networks? Blogi pangeanic sivustolla. Julkaistu 18.4.2023. Viitattu 21.8.2023. <https://blog.pangeanic.com/what-is-fine-tuning>

Transformers. N.d. Dokumentaatio Hugging Face verkkosivustolla. Viitattu 28.11.2023. <https://huggingface.co/docs/transformers/index>

Transformers. N.d. Dokumentaatio Hugging Face github-verkkosivustolla. Viitattu 28.11.2023. <https://github.com/huggingface/transformers>

What are Large Language Models (LLMs) and Why They Matter. HSBC. 5.6.2023. Viitattu 11.9.2023. <https://www.businessgo.hsbc.com/en/article/what-are-large-language-models-llms-and-why-they-matter>

What Is A Neural Network? N.d. aws.amazon verkkosivut. Viitattu 8.8.2023. <https://aws.amazon.com/what-is/neural-network/>

What is MySQL? N.d. Oracle verkkosivut. Viitattu 23.11.2023. <https://www.oracle.com/mysql/what-is-mysql/>

What is machine learning? N.d. IBM verkkosivut. Viitattu 6.8.2023. <https://www.ibm.com/topics/machine-learning>

What is Python? N.d. Teradata verkkosivut. Viitattu 30.11.2023. <https://www.teradata.com/glossary/what-is-python>

Yasar, K. N.d. Pytorch. TechTarget verkkosivustolla. Viitattu 30.11.2023. <https://www.techtarget.com/searchenterpriseai/definition/PyTorch>

# Liitteet

## Liite 1. Sentimenttiansalyysin päätelmä

```
import transformers
import sys, json, os
from langdetect import detect, DetectorFactory
import torch
import numpy as np

if torch.cuda.is_available():
    device = torch.device('cuda')
else:
    device = torch.device('cpu')

torch.cuda.empty_cache()

dir = os.path.dirname(__file__)
modelpath = os.path.join(dir, './test_trainer/sentiment')

model_path = modelpath

tokenizer = transformers.BertTokenizer.from_pretrained('TurkuNLP/bert-base-finnish-cased-v1')

for txt in sys.stdin:
    try:
        DetectorFactory.seed = 0
        language = detect(txt)
        if language == "en":
            model = "distilbert-base-uncased-finetuned-sst-2-english"
            classifier = transformers.pipeline("sentiment-analysis", model=model, device=0)
            predictions = classifier(txt)
            if predictions[0]["label"] == "POSITIVE":
                sentiment = "positive"
            else:
                sentiment = "negative"

            if predictions[0]["label"] == "POSITIVE":
                positive_score = predictions[0]["score"]
                negative_score = 1 - predictions[0]["score"]
            else:
                positive_score = 1 - predictions[0]["score"]
                negative_score = predictions[0]["score"]

            json_output = json.dumps({'text': txt, 'sentiment': sentiment, 'positive_score': positive_score, 'negative_score': negative_score}, ensure_ascii=False).encode('utf-8')
            print(json_output.decode())
        elif language == "fi":
            encoding = tokenizer.encode_plus(txt, add_special_tokens = True, truncation = True, padding = "max_length", return_attention_mask = True, return_tensors = "pt")
            model = transformers.BertForSequenceClassification.from_pretrained(model_path, num_labels=2)
            output = model(**encoding)
            # use softmax function to calculate probabilities
            predictions = output.logits.softmax(dim=-1).detach().cpu().flatten().numpy().tolist()

            if predictions[1] < predictions[0]:
                sentiment = "negative"
            else:
                sentiment = "positive"

            json_output = json.dumps({'text': txt, 'sentiment': sentiment, 'positive_score': predictions[1], 'negative_score': predictions[0]}, ensure_ascii=False).encode('utf-8')
            print(json_output.decode())
        except Exception:
            continue
```

## Liite 2. Multi-label-luokittelu opetusohjelma

```

model = AutoModelForSequenceClassification.from_pretrained("TurkuNLP/bert-base-finnish-cased-v1",
                                                         problem_type="multi_label_classification",
                                                         num_labels=len(labels),
                                                         id2label=id2label,
                                                         label2id=label2id).to("cuda")

device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")

batch_size = 8
metric_name = "f1"

args = TrainingArguments([
    "test_trainer/multilabel_en",
    evaluation_strategy = "epoch",
    save_strategy = "epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=5,
    weight_decay=0.01,
    load_best_model_at_end=True,
    metric_for_best_model=metric_name,
])

def multi_label_metrics(predictions, labels, threshold=0.5):
    sigmoid = torch.nn.Sigmoid()
    probs = sigmoid(torch.Tensor(predictions))
    y_pred = np.zeros(probs.shape)
    y_pred[np.where(probs >= threshold)] = 1
    y_true = labels
    f1_micro_average = f1_score(y_true=y_true, y_pred=y_pred, average='micro')
    roc_auc = roc_auc_score(y_true, y_pred, average = 'micro')
    accuracy = accuracy_score(y_true, y_pred)
    metrics = {'f1': f1_micro_average,
               'roc_auc': roc_auc,
               'accuracy': accuracy}
    return metrics

def compute_metrics(p: EvalPrediction):
    preds = p.predictions[0] if isinstance(p.predictions,
                                           tuple) else p.predictions
    result = multi_label_metrics(
        predictions=preds,
        labels=p.label_ids)
    return result

trainer = Trainer(
    model,
    args,
    train_dataset=encoded_dataset["train"],
    eval_dataset=encoded_dataset["valid"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)

trainer.train()
trainer.save_model()
trainer.evaluate()

```

### Liite 3. Multi-label-luokittelu päätelmä

```

if torch.cuda.is_available():
    device = torch.device('cuda')
else:
    device = torch.device('cpu')

dir = os.path.dirname(__file__)
modelfi = os.path.join(dir, '../test_trainer/multilabel-fi')
modelen = os.path.join(dir, '../test_trainer/multilabel-en')

for txt in sys.stdin:
    try:
        DetectorFactory.seed = 20
        language = detect(txt)
        def check_for_language_support(language):
            if (language == "fi") | (language == "en"):
                return True
            else:
                return False
        if check_for_language_support(language):
            if language == "en":
                model_path = "test_trainer/multilabel-en"
                model = transformers.AutoModelForSequenceClassification.from_pretrained(model_path, num_labels=17)
                model.to(device)
                tokenizer = transformers.AutoTokenizer.from_pretrained('distilbert-base-uncased')
            else:
                model_path = modelfi
                model = transformers.BertForSequenceClassification.from_pretrained(model_path, num_labels=17)
                tokenizer = transformers.BertTokenizer.from_pretrained('TurkuNLP/bert-base-finnish-cased-v1')
                model.to(device)

            encoding = tokenizer(txt, return_tensors="pt", padding=True, truncation=True)
            encoding = {k: v.to(model.device) for k,v in encoding.items()}

            outputs = model(**encoding)

            logits = outputs.logits
            logits.shape

            labels = ['Travel', 'Destination', 'Accommodation', 'Activities', 'Restaurant', 'Sightseeing', 'Traveling', 'Event',
                    'Reservation or order', 'Sustainability, responsibility and health safety', 'Cancellation terms ',
                    'Online store or OTA channel', 'Product information and product selection', 'Payment', 'Delivery',
                    'Customer service', 'Marketing']
            id2label = {idx:label for idx, label in enumerate(labels)}

            sigmoid = torch.nn.Sigmoid()
            probs = sigmoid(logits.squeeze().cpu())
            predictions = np.zeros(probs.shape)
            predictions[np.where(probs >= 0.3)] = 1
            predicted_labels = [id2label[idx] for idx, label in enumerate(predictions) if label == 1.0]
            if np.all(predictions == predictions[0]):
                predicted_labels = ["No categories"]
            json_output = json.dumps({'text': txt, 'labels': predicted_labels}, ensure_ascii=False).encode('utf-8')
            print(json_output.decode())
    except Exception:
        continue

```