Katri Tolonen

RAG-järjestelmän tiedonhaun parantaminen retriever-komponentin fine-tuningilla

Insinööri (AMK)
Tieto- ja viestintätekniikka
Kevät 2026

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Tolonen Katri

Työn nimi: RAG-järjestelmän tiedonhaun parantaminen retriever-komponentin fine-tuningilla

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: Retrieval-Augmented Generation (RAG), mallin hienosäätö, kielimallit

Opinnäytetyössä tarkasteltiin Retrieval-Augmented Generation (RAG) -arkkitehtuuria keskittyen erityisesti tiedonhaun eli retriever-komponentin kehittämiseen. RAG-järjestelmissä kyky löytää käyttäjän kysymykseen relevantti lähdeteksti on keskeinen tekijä lopullisen vastauksen laadun kannalta. Työn päätavoitteena oli selvittää, voidaanko retriever-komponentin toimialakohtaisella fine-tuningilla parantaa tiedonhaun osumatarkkuutta. Lisäksi tavoitteena oli havainnollistaa, miten mahdollinen hakutarkkuuden paraneminen heijastuu generointivaiheeseen eli tuotettuun vastaukseen. Työ rajattiin hakukomponentin kehittämiseen eikä koko RAG-putkea optimoitu.

Fine-tuning toteutettiin eläkevakuuttamisen toimialueen aineistolla, joka koostui verkkosivusisällöistä sekä PDF-muotoisista ohjeista. Aineisto esikäsiteltiin, paloiteltiin semanttisesti sopiviksi tekstikappaleiksi ja indeksoitiin FAISS-vektori-indeksiin. Retriever-mallina käytettiin monikielistä intfloat/multilingual-e5-small -embedding-mallia, jota hienosäädettiin valvotusti Sentence Transformers -kirjastolla. Fine-tuningin onnistumista arvioitiin vertaamalla hienosäädetyn ja alkuperäisen mallin hakutuloksia samalla testiaineistolla käyttäen mittareina Recall@K- ja MRR-mittareita.

Tulokset osoittivat, että tiiviillä koulutusaineistolla tiedonhaun tarkkuus parani selvästi: Recall@1 nousi 26 %, Recall@5 22 % ja MRR@5 32 %. Teemakohtaisesti suurimmat parannukset nähtiin työkyvyttömyyseläkemaksun ja työkyvyttömyyseläkkeen kysymyksissä. Laajemmalla, heterogeenisemmalla aineistolla kokonaisparannusta ei saavutettu, mikä viittaa aineiston kohdentamisen ja tasapainon merkitykseen retriever-komponentin hienosäädössä. Generointivaiheen esimerkit osoittivat, että käytetyn monikielisen generointimallin (microsoft/Phi-4-mini-instruct) tuottamien suomenkielisten vastausten laatu ei ole sellaisenaan riittävä asiantuntijatason käyttötarkoituksiin, vaikka joissakin tapauksissa parantunut tiedonhaku heijastuikin myös vastauksen sisältöön.

Työn johtopäätöksenä on, että toimialakohtainen fine-tuning voi parantaa tiedonhaun tarkkuutta jo kohtuullisen pienellä, mutta huolellisesti laaditulla aineistolla. Vaikutus ei kuitenkaan jakaudu tasaisesti kaikkiin teemoihin. Tulokset korostavat datan laadun ja tasapainoisen teema-edustuksen merkitystä RAG-pohjaisten tiedonhakumenetelmien fine-tuningissa. Tulokset osoittavat, että retriever-komponentin parantaminen yksinään ei välttämättä riitä asiantuntijatason RAG-järjestelmän toteuttamiseen, vaan jatkokehitys edellyttää koko RAG-putken tarkastelua.

KAMK • University of Applied Sciences

Abstract

Author: Tolonen Katri

Title of the Publication: Improving Retrieval in a RAG System through Retriever Fine-Tuning

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: Retrieval-Augmented Generation (RAG), fine-tuning, large language models

This thesis examined the Retrieval-Augmented Generation (RAG) architecture, with a particular focus on the development of the retrieval component. In RAG systems, the retriever's ability to find a source passage relevant to the user's query is a key factor in the quality of the final answer. The main objective of the thesis was to investigate whether domain-specific fine-tuning of the retriever component can improve retrieval accuracy. An additional goal was to illustrate how potential improvements in retrieval accuracy are reflected in the generation phase, that is, in the produced answer. The work was limited to the development of the retrieval component, and the entire RAG pipeline was not optimized.

Fine-tuning was performed using domain data from the field of pension insurance, consisting of website content and PDF-based guidelines. The data was preprocessed, split into semantically appropriate text passages, and indexed into a FAISS vector index. The retriever model used was the multilingual intfloat/multilingual-e5-small embedding model, which was fine-tuned in a supervised manner using the Sentence Transformers library. The success of the fine-tuning was evaluated by comparing the retrieval results of the fine-tuned model with those of the original model using the same test dataset. The evaluation metrics used were Recall@K and MRR.

The results showed that with a more compact training dataset, the retriever's performance improved clearly: Recall@1 increased by 26%, Recall@5 by 22%, and MRR@5 by 32%. The largest improvements by topic were observed in questions related to disability pension contributions and disability pensions. With a broader, more heterogeneous dataset, no overall improvement was achieved, indicating the importance of data targeting and balance in retriever fine-tuning. Examples from the generation phase showed that the quality of the Finnish-language responses produced by the multilingual generation model (microsoft/Phi-4-mini-instruct) was not sufficient as such for expert-level use cases, although in some cases improved retrieval was also reflected in the content of the answer.

The main conclusion of the thesis is that domain-specific fine-tuning can improve the accuracy of a retriever even with a relatively small but carefully constructed dataset. However, the effect is not evenly distributed across all topics. The results highlight the importance of data quality and balanced topic representation in the fine-tuning of RAG-based retrieval systems. The results also indicate that improving the retriever component alone may not be sufficient for building an expert-level RAG system, and further development requires consideration of the entire RAG pipeline.

Sisällys

1 Johdanto

Tämän opinnäytetyön aihe valikoitui kiinnostuksesta selvittää, miten asiantuntijatason asiakasneuvontatyötä voidaan kehittää tekoälyn, erityisesti suurten kielimallien avulla. Yritysten asiakaspalvelussa keskeistä on yleensä yrityksen oman, eri lähteissä olevan aineiston hyödyntäminen asiakastyön tukena. Tässä työssä tekoälyn hyödyntämistä tarkastellaan RAG-arkkitehtuurin (Retrieval Augmented Generation) kautta. RAG on tekoälyn ratkaisu, joka yhdistää tiedonhaun ja tekstin tuottamisen: siinä hyödynnetään ulkoisia tai organisaation omia tietolähteitä vastauksen generointia varten, mikä voi parantaa tiedon ajantasaisuutta ja vähentää hallusinaatioiden riskiä [1].

RAG-järjestelmä koostuu kolmesta pääkomponentista: Retriever (tiedonhaku), Augmentation (haetun tiedon liittäminen kontekstiin) ja Generation (tekstin tuottaminen). Järjestelmässä on useita muuttujia, jotka vaikuttavat sen toimivuuteen [1]. Esimerkiksi dokumenttien haun tarkkuus, kontekstin muotoilu ja tekstin paloittelun (chunking) strategia voivat kaikki parantaa tai heikentää lopputulosta.

Oikean tiedonhaun osumatarkkuus on keskeinen tekijä RAG-järjestelmän tuottaman vastauksen laadussa. Jos lähdeaineistosta ei löydetä kysymykseen parhaiten vastaavaa tietoa, ei myöskään generointivaiheessa synny laadukkainta mahdollisinta vastausta. On siis mahdollista, että järjestelmä ei tavoita relevanttia osumaa tai että oikea osuma jää hakutuloksissa alemmas – mikä heikentää järjestelmän hyödyllisyyttä.

Tässä opinnäytetyössä selvitettiin keinoja erityisesti tiedonhaun toiminnan parantamiseen. Päätavoitteena oli selvittää, voidaanko retriever-komponentin fine-tuningilla parantaa mallin kykyä löytää lähdeaineistosta kysymykseen parhaat tekstiosumat. Fine-tuningissa malli koulutettiin valitun toimialueen, eläkevakuuttamisen, aineistolla. Fine-tuningin onnistumisen arviointi toteutettiin vertaamalla jatkokoulutetun mallin tuottamia tuloksia alkuperäisen mallin tuloksiin samalla testiaineistolla.

Työ keskittyi hakukomponentin kehittämiseen, ei koko RAG-putken optimointiin. Vastauksen generointiin käytettyä mallia ei jatkokoulutettu, mutta lisätavoitteena oli havainnollistaa, miten mahdollinen hakutarkkuuden paraneminen heijastuu myös generointivaiheeseen eli tuotettuun vastaukseen. Työssä käytettiin avoimen lähdekoodin malleja ja suomenkielistä aineistoa, mikä rajasi käytettävissä olevien mallien valikoimaa.

2 Tekoäly ja kielimallit tiedonhaussa

Tekoäly (AI) ja erityisesti luonnollisen kielen käsittelyn (Natural Language Processing, NLP) kehittyminen ovat muuttaneet tapaa, jolla tekstipohjaista tietoa voidaan hakea ja jäsentää. Perinteiset tiedonhakumenetelmät, jotka perustuvat sanojen täsmälliseen vastaavuuteen, eivät kykene huomioimaan kielen semanttisia merkityksiä tai käsitteiden välisiä suhteita. Kielimallit mahdollistavat sen sijaan semanttisen tiedonhaun, jossa tavoitteena on ymmärtää kysymyksen merkitys ja löytää sitä sisällöllisesti vastaavaa tietoa riippumatta yksittäisten sanojen muodosta. [2.]

Nykyiset kielimallit perustuvat pääosin transformer-arkkitehtuuriin. Transformer-mallien keskeinen ominaisuus on self-attention-mekanismi, jonka avulla se tunnistaa, mitkä sanat ovat merkityksellisiä toisiinsa nähden ja miten ne vaikuttavat lauseen kokonaismerkitykseen. Self-attention-mekanismi mahdollistaa myös kaukana toisistaan olevien sanojen välisten riippuvuuksien mallintamisen ja kielen syvällisemmän ymmärtämisen. [3.] Tämä on olennaista erityisesti tiedonhaussa, jossa kysymyksen ja dokumentin välinen merkitysyhteys ei välttämättä perustu suoraan samoihin sanoihin, vaan käsitteelliseen vastaavuuteen.

Self-attention-mekanismin ympärille rakentuva transformer-arkkitehtuuri koostuu tyypillisesti kahdesta pääkomponentista: encoderista ja decoderista. Encoderin tehtävänä on lukea ja muuntaa syöte (esimerkiksi teksti) numeeriseen vektoriesitykseen, joka säilyttää sisällön merkityksen. Decoder puolestaan hyödyntää näitä vektoriesityksiä tuottaakseen vastauksia, ennusteita tai jatkaakseen tekstiä. [4, s. 2–3.]

Tiedonhaun näkökulmasta erityisen merkittäviä ovat encoder-pohjaiset kielimallit, jotka tuottavat tekstistä sanojen ja lauseiden merkityssisältöä kuvaavia numeerisia vektoriesityksiä eli embeddingejä. Näissä malleissa sekä kysymykset että dokumentit muunnetaan samaan vektoriavaruuteen. Embeddingien avulla voidaan vertailla kysymyksen ja dokumentin samankaltaisuutta vektoritasolla. [4, s. 4–6.]

Embedding-pohjainen haku mahdollistaa semanttisen tiedonhaun, jossa käyttäjän kysymys voidaan yhdistää oikeaan sisältöön, vaikka sanamuodot tai terminologia poikkeaisivat toisistaan. Semanttinen haku on erityisen tärkeää toimialakohtaisissa aineistoissa, joissa samaan ilmiöön voidaan viitata useilla eri käsitteillä tai joissa termien merkitys vaihtelee kontekstin mukaan. Tällaisissa tapauksissa pelkkään avainsanahakuun perustuvat menetelmät voivat jäädä puutteellisiksi.

Kielimallien käyttö tiedonhaussa ei kuitenkaan tarkoita sitä, että malli sisältäisi kaiken tarvittavan tiedon itsessään. Suurten kielimallien tieto on sidottu esikoulutusdataan ja ajankohtaan, eikä sitä voida helposti päivittää ilman uudelleenkoulutusta. Tiedonhaussa hyödynnetäänkin yhä useammin arkkitehtuureja, joissa kielimalli toimii yhdessä ulkoisen tietolähteen kanssa. Näissä ratkaisuissa kielimallin tehtävänä on ymmärtää kysymys ja tuottaa merkityksellisiä vektoriesityksiä, kun taas varsinainen tieto haetaan erillisestä dokumenttikokoelmasta. [1, s. 1–2.] Lähestymistapa muodostaa perustan Retrieval-Augmented Generation (RAG) -arkkitehtuurille, jossa tiedonhaku ja tekstin generointi yhdistetään.

3 RAG (Retrieval-Augmented Generation)

Retrieval-Augmented Generation (RAG) on menetelmä, joka yhdistää kaksi tiedonlähteen muotoa: kielimalliin tallennetun parametrisen tiedon sekä ulkoisista dokumenteista haettavan ei-parametrisen tiedon. RAG-mallit voivat olla ratkaisu tilanteissa, joissa tarvitaan ajankohtaista ja faktoihin perustuvaa vastausta, mutta koko tietosisältöä ei voida tallentaa mallin painoihin. [1, s. 1.]

RAG mahdollistaa mallin tietosisällön päivittämisen ilman uudelleenkoulutusta, mikä tekee siitä kustannustehokkaan menetelmän. Menetelmä voi parantaa vastausten luotettavuutta ja tarjota räätälöityjä, toimialakohtaisia sekä ajankohtaisia vastauksia. RAG sopii erityisesti avoimen domainin kysymys-vastaustehtäviin, dokumenttien hakuun ja tekstin generointiin faktoihin perustuen. [5, s. 11–12.]

3.1 Retrieval-Augmented Generation (RAG) -arkkitehtuuri

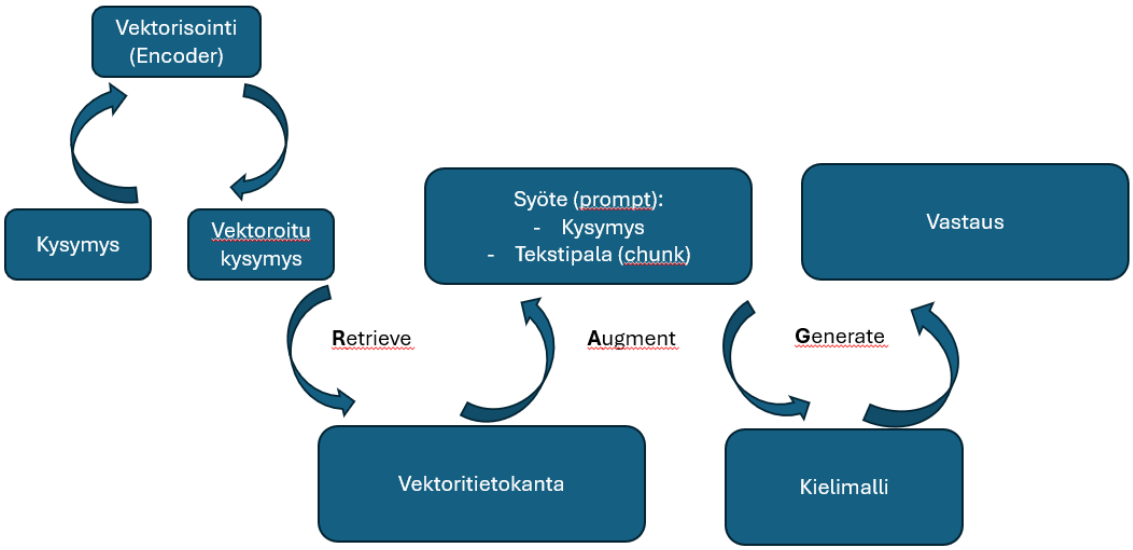
RAG-menetelmä perustuu kolmeen pääkomponenttiin:

1. **Haku (Retrieval)**: Retriever etsii käyttäjän kysymyksen perusteella osuvimmat dokumenttipalat (chunkit) ulkoisesta tietolähteestä, esimerkiksi vektoripohjaisesta tietorakenteesta (indeksi). Indeksi rakennetaan etukäteen esikäsittelemällä ja pilkkomalla dokumentit pienempiin osiin, jotka muunnetaan vektoreiksi embedding-mallilla. Nämä vektorit tallennetaan hakuindeksiin, jota retriever käy läpi vertaamalla kysymyksen vektoria dokumenttien vektoreihin.

2. **Laajennus (Augmentation)**: Haetut dokumenttipalat yhdistetään alkuperäiseen kysymykseen laajennetuksi syötteeksi (prompt). Tästä muodostuu generointimallille asiayhteys, joka sisältää käyttäjän kysymyksen sekä haetut tiedot.

3. **Generointi (Generation)**: Generointimalli tuottaa vastauksen laajennetun syötteen perusteella. Vastaus perustuu sekä mallin sisäiseen kielelliseen osaamiseen että retrieverin tarjoamaan ulkoiseen tietoon. [5, s. 10–12.]

Kuvassa 1 esitetään RAG-menetelmän toiminta. Kysymys muunnetaan vektorimuotoon ja haetaan siihen liittyvää tietoa tietopohjasta. Näin saatu tieto yhdistetään kysymykseen, ja laajennettu syöte annetaan kielimallille vastauksen generointia varten.



Kuva 1. Retriever-Augmented Generation (RAG) -menetelmän toiminta.

Lewis ym. (2020) esittelivät ensimmäisenä varsinaisen RAG-arkkitehtuurin, jossa yhdistettiin Dense Passage Retriever (DPR) ja generatiivinen BART-malli yhdeksi järjestelmäksi. Järjestelmä saavutti korkeampaa tarkkuutta verrattuna pelkkiin generatiivisiin malleihin. Tutkimus korostaa erityisesti retriever-mallin laadun merkitystä koko järjestelmän onnistumiseen sekä retrieverin roolia hakutulosten relevanssin parantamisessa. [1, s. 2–5.]

RAG-menetelmää on sovellettu menestyksekkäästi myös esimerkiksi juridisten asiakirjojen tiivistämiseen ja tiedonhakuun. Mukund ja Easwarakumar kehittivät RAG-järjestelmän, jossa käytettiin BM25-hakualgoritmia retrieverinä ja strategiaa, jossa valittiin kolme parasta osumaa vastauksen pohjaksi. Järjestelmässä hyödynnettiin myös domain-spesifiä entiteettien tunnistusta (NER), jonka avulla hakua kohdennettiin juridisesti merkityksellisiin osiin lähdeaineistoa. RAG-menetelmä osoitti parempaa luotettavuutta verrattuna pelkkään kielimalliin. [6.] Tutkimus tukee käsitystä, että RAG-menetelmän suorituskyky riippuu oleellisesti tiedonhaun tehokkuudesta ja siitä, miten hyvin haettu tieto vastaa käyttäjän kysymystä. Tämä on keskeinen lähtökohta myös tämän opinnäytetyön toteutuksessa.

3.2 Retriever-menetelmät

Dense Passage Retrieval (DPR) on RAG-malleissa yleisesti käytetty hakumenetelmä. Se perustuu bi-encoder-arkkitehtuuriin, jossa kysymykselle ja dokumentille lasketaan erikseen niiden vektoriesitykset. Dokumenttien ja kysymyksen samankaltaisuus lasketaan matemaattisesti ja malli valitsee jatkokäsittelyyn ne dokumentit, jotka ovat sisällöllisesti lähimpänä kysymystä. DPR-menetelmä mahdollistaa tehokkaan haun suurista tietokannoista. DPR:n avulla rakennetut indeksit muodostavat RAG-mallin ei-parametrisen muistin, joka voidaan päivittää ilman mallin uudelleenkoulutusta. [1, s. 2–3.]

RAG-malleissa retriever voidaan kouluttaa yhdessä generointimallin kanssa, jolloin ne optimoituvat yhteistyössä parhaan vastaustarkkuuden saavuttamiseksi. Vaihtoehtoisesti retriever voidaan pitää kiinteänä (pre-trained), mikä nopeuttaa kehitystä. Lisäksi retriever voidaan toteuttaa myös perinteisillä sparse-hakumenetelmillä ilman vektoriesityksiä, kuten BM25-algoritmilla, joka perustuu sanojen esiintymistaajuuksiin ja niiden painotuksiin dokumenteissa. Ne voivat toimia hyvin tilanteissa, joissa kysymyksen ja vastauksen avainsanat vastaavat toisiaan täsmällisesti. Tutkimuksissa on kuitenkin havaittu, että DPR-pohjainen haku voi parantaa tuloksia erityisesti avoimen tiedonhaun tehtävissä verrattuna BM25-menetelmään. [1, s. 1–5.]

Sparse- ja dense-menetelmien lisäksi käytössä on myös hybrid retrieval, jossa molemmat lähestymistavat yhdistetään. Hybridiratkaisut hyödyntävät sparse-menetelmien tarkkuutta ja dense-menetelmien semanttista ymmärrystä. Tämä voi olla toimiva vaihtoehto laajoissa tai heterogeenisissa dokumenttikokoelmissa, joissa yksittäinen hakumalli ei kata kaikkia kysymystyyppejä. [7.]

3.3 Retriever-komponentin parantamisen keinot

RAG-järjestelmässä retriever-komponentin kyky löytää oikea lähdeteksti on olennainen osa koko järjestelmän laatua. Erilaisin hakuprosessin optimoinnin keinoin voidaan pyrkiä vaikuttamaan haun laatuun sekä tehokkuuteen, ennen mahdollista komponentin fine-tuningia.

Koska käytännössä embedding-mallien käyttö edellyttää lähdeaineistojen pilkkomista tekstinpätkiksi, merkityksellistä on se, miten pilkkominen onnistuu eli miten hyvin tekstit saadaan jaettua merkityksellisiin palasiin ilman, että esimerkiksi asiayhteys katkeaa tekstinpätkien välillä [8]. Aineistoon sopivalla pilkkomisstrategialla ja sen hyvällä toteutuksella voitaneenkin saada retrieverin osumatarkkuutta parannettua ja näin vaikutettua tiedonhaun laatuun.

Tekstin pilkkomisen tulokseen ei kuitenkaan yksistään vaikuta pelkästään valittu metodi vaan myös se, kuinka hyvin lähdeaineisto on esikäsitelty ennen pilkkomista. Laadukas ja rakenteensa säilyttävä tekstin poiminta eri lähteistä mahdollistaa sen, että myöhemmissä vaiheissa muodostuvat tekstipätkät ovat informatiivisia sekä tietyssä kontekstissa pysyviä. Näin ne ovat myös paremmin tiedonhakumallin haettavissa. Tässä työssä käytettiin menetelmiä, joiden avulla pyrittiin säilyttämään dokumenttien looginen rakenne. Käytetyt kirjastot Trafilatura (HTML-sisältöjen siistittyyn poimintaan) sekä PyMuPDF4LLM (PDF-dokumenttien muuntamiseen rakenteiseksi Markdowniksi) esitellään tarkemmin kappaleessa 6.1.

Hakuprosessin optimointiin voidaan lisäksi vaikuttaa itse hakumenetelmien tasolla. RAG-järjestelmän hakutulosten laatua voidaan pyrkiä parantamaan esimerkiksi kaksivaiheisella haulla (two-stage retrieval), jossa dokumentteja käsitellään usean peräkkäisen menetelmän kautta. Ensimmäisessä vaiheessa suoritetaan karkeampi esihaku, jonka tarkoituksena on rajata suuri dokumenttijoukko hallittavamman kokoiseksi joukoksi mahdollisesti relevantteja osumia. Toisessa vaiheessa hyödynnetään uudelleenjärjestämistä (re-ranking), jossa esihakuvaiheen palauttamat dokumentit pisteytetään uudelleen tarkemmalla mallilla. Re-rankereina käytetään yleensä transformer-malleja, jotka ottavat kysymyksen ja dokumentin yhteisesti syötteenä ja arvioivat niiden todellisen semanttisen vastaavuuden. Tämä menetelmä mahdollistaa täsmällisemmän relevanssiarvion kuin pelkkä embedding-haku. [9.]

4 Fine-tuning

Fine-tuning eli hienosäätö tai jatkokoulutus tarkoittaa esikoulutetun kielimallin lisäkouluttamista esimerkiksi toimialakohtaisella datalla. Mallin aikaisemmin oppimaa tietoa hyödynnetään ja täydennetään uuteen käyttökontekstiin sopivaksi. Hienosäädön tavoitteena on parantaa mallin kykyä suoriutua rajatusta käyttötarkoituksesta, esimerkiksi vastata tietyn alan kysymyksiin tai hakea relevanttia tietoa dokumenteista. Hienosäätö mahdollistaa korkean suorituskyvyn saavuttamisen pienemmällä datamäärällä ja vähäisemmällä laskentateholla verrattuna mallin kouluttamiseen alusta lähtien. [5, s. 8–10.]

Tehtäväkohtainen fine-tuning mukauttaa suuria kielimalleja tiettyihin jatkokäyttötehtäviin hyödyntämällä tehtävään sopivasti muotoiltua ja puhdistettua dataa. Esimerkkejä tehtävistä ovat tekstin tiivistäminen tai luokittelu. Toimialakohtainen fine-tuning keskittyy mallin mukauttamiseen siten, että se ymmärtää ja tuottaa tekstiä, joka liittyy tiettyyn toimialaan tai alaan. Kouluttamalla mallia kohdealueelta peräisin olevalla aineistolla parannetaan sen kontekstuaalista ymmärrystä ja asiantuntemusta toimialakohtaisissa tehtävissä. [5, s. 35–36.]

4.1 Retriever-komponentin fine-tuningin rooli

Hienosäädön roolista ja merkityksestä dense-retrievereiden parantamisessa ei löydy selkeää yhtä näkemystä. Osa tutkimuksista esittää, että suurin osa semanttiseen hakuun liittyvästä tiedosta opitaan jo esikoulutusvaiheessa, eikä tätä tietoa voida merkittävästi laajentaa pelkällä hienosäädöllä. Esimerkiksi Yao ym. (2025) havaitsivat, että dense retrieverien suorituskyky perustuu suurelta osin esikoulutusvaiheessa opittuun semanttiseen tietoon ja että hienosäätö vaikuttaa pääasiassa siihen, miten mallin sisäisiä vektoriesityksiä hyödynnetään [10]. Toisaalta tutkimuksissa on havaittu, että hienosäätö voi edelleen merkittävästi parantaa hakutulosten relevanssia, kun retrieveriä mukautetaan domain-kohtaiseen aineistoon [11].

Tässä työssä käytetty intfloat/multilingual-e5-small-malli on jo valmiiksi esikoulutettu retrieval-tehtäviin, joten sen voidaan olettaa sisältävän yleisen tiedon semanttista tiedonhakua varten. Eläkevakuuttamisen toimialalla on kuitenkin omanlaisensa termistö. Toimialalla myös esiintyy keskenään samankaltaisia käsitteitä, joiden merkitys vaihtelee kontekstin mukaan (esimerkiksi työkyvyttömyyseläke viittaa henkilölle myönnettävään etuuteen, kun taas työkyvyttömyyseläkemaksu työnantajan maksuvelvoitteeseen).

Tässä työssä retriever-komponentin hienosäädön tavoitteena oli parantaa mallin kykyä erotella tällaisia semanttisesti läheisiä mutta sisällöllisesti erilaisia käsitteitä oikeissa käyttöyhteyksissään. Näin pyrittiin parantamaan hakutulosten tarkkuutta ja relevanssia.

4.2 Menetelmät retriever-malleissa

Esikoulutetun kielimallin hienosäätöä (fine-tuning) voidaan toteuttaa eri menetelmillä, mutta kaikki lähestymistavat eivät sovellu yhtä hyvin semanttiseen tiedonhakuun ja retriever-komponenttien kehittämiseen. Retriever-malleissa hienosäädön tavoitteena on parantaa kysymysten ja dokumenttien välistä suhteellista semanttista vastaavuutta vektoriavaruudessa siten, että relevantti lähdeteksti sijoittuu lähemmäs kysymystä kuin epäolennaiset vaihtoehdot [12].

Täysi fine-tuning (full fine-tuning) on menetelmä, jossa kaikki mallin parametrit päivitetään uutta dataa käyttäen. Tämä mahdollistaa mallin kokonaisvaltaisen mukautumisen uuteen tehtävään. Täysi fine-tuning vaatii kuitenkin paljon laskentatehoa, suuremman määrän koulutusdataa ja enemmän muistiresursseja verrattuna kevyempiin menetelmiin. [5, s. 15.]

Vaihtoehtona täydelle fine-tuningille voidaan käyttää parametrisesti tehokkaita fine-tuning-menetelmiä (Parameter-Efficient Fine-Tuning, PEFT), joissa vain osa mallin parametreista päivitetään. Näihin kuuluvat esimerkiksi LoRA-pohjaiset menetelmät. LoRA-menetelmässä painomatriisit jaetaan matalan ulottuvuuden matriiseiksi, joita koulutetaan siten, että ne approksimoivat alkuperäistä painopäivitysmatriisia. Nämä kaksi pienempää matriisia sisältävät vain murto-osan koulutettavista parametreista, mikä pienentää muistitarvetta ja nopeuttaa koulutusta. Alkuperäiset mallin painot pidetään muuttumattomina ja vain LoRA-matriiseja koulutetaan. [5, s. 38–39.]

Retriever-mallien fine-tuning on toteutettu useimmiten täytenä fine-tuningina [12] ja LoRA-pohjaisista menetelmistä ei retriever-kontekstissa löydy yhtä paljon tutkimuksia.

Fine-tuning voidaan toteuttaa myös eri tavoin koulutusdatan perusteella. Molemmat lähestymistavat voidaan yhdistää eri fine-tuning-tekniikoihin, kuten täyteen fine-tuningiin tai PEFT-menetelmiin.

- **Valvottu fine-tuning (Supervised Fine-Tuning)** tarkoittaa, että mallia koulutetaan nimikoidulla datalla, jossa jokaisella syötteellä on tunnettu oikea vastaus tai luokka. Tämä lähestymistapa on yleinen esimerkiksi tekstiluokittelussa ja kysymys–vastaustehtävissä, mutta edellyttää usein suuria määriä huolellisesti laadittua aineistoa. [5, s. 9.]

- **Valvomaton fine-tuning (Unsupervised Fine-Tuning)** sen sijaan ei vaadi nimikoitua dataa. Malli koulutetaan kohdealueelta kerätyllä raakatekstillä, jonka avulla se syventää kielentuntemustaan tietyssä kontekstissa. Tätä menetelmää voidaan hyödyntää esimerkiksi silloin, kun nimikoitua aineistoa ei ole saatavilla, mutta halutaan mukauttaa malli toimialakohtaisiin termeihin ja rakenteisiin. [5, s. 9.]

Retriever-mallien kehittämisessä valvottu fine-tuning on vallitseva lähestymistapa. Malli koulutetaan nimikoidulla kysymys–dokumentti-aineistolla erottamaan relevantit ja epäolennaiset dokumentit toisistaan. Tässä opinnäytetyössä menetelmää sovellettiin eläkevakuuttamisen aihealueen aineistoon, ja tavoitteena oli parantaa retriever-komponentin kykyä löytää käyttäjän kysymykseen relevantti lähdeteksti. Mallin jatkokoulutus toteutettiin täytenä fine-tuningina, jossa kaikki mallin painot päivitettiin.

5 Mallien arviointimenetelmät

Laajan kielimallin arviointia voi lähestyä eri näkökulmista. Yleisesti hyödyllisiä mittareita ovat esimerkiksi vastauksen merkityksellisyys (relevance), oikeellisuus sekä hallusinaatio (onko sitä). RAG-järjestelmässä kontekstuaalinen merkitys on olennaista: sen mittaamisella voidaan arvioida sitä, pystyykö järjestelmä löytämään olennaisimman tiedon kyseisessä kontekstissa. [13.]

Tässä kappaleessa esitetään arviointimenetelmiä, jotka liittyvät RAG-järjestelmän mittaamiseen. Retrieval-augmented generation (RAG) -järjestelmä koostuu erillisistä, säädettävistä komponenteista, kuten esimerkiksi kappaleessa 3.1 on esitetty. Jokainen komponenteista voidaan arvioida erikseen ja toisaalta heikkous missä tahansa komponentissa voi yksinään heikentää koko järjestelmän suorituskykyä. Tiedonhaun vaiheessa tärkein arviointi koskee dokumenttien haun tarkkuutta, mutta sen onnistumiselle on merkityksellistä myös esikäsittelyvaiheiden, kuten sopivan tekstin paloittelustrategian, valinta. [14.]

Jos arvioidaan RAG-järjestelmää kokonaisuutena, tarvitaan mittareita, jotka kuvaavat retriever-generator -yhdistelmän toimivuutta. Tällaisia arviointikriteerejä ovat esimerkiksi:

- Faithfulness (uskollisuus) — Mittaa, onko kielimallin tuottama tieto faktuaalisesti yhdenmukaista haetun kontekstin kanssa

- Answer Relevancy (vastausrelevanssi) —Arvioi, kuinka ytimekkäitä ja kysymykseen liittyviä vastauksia RAG-generaattori tuottaa. [13.]

Kokonaisjärjestelmän arviointikriteerien lisäksi RAG:n retriever-komponenttia voidaan arvioida erikseen sen tuottamien hakutulosten laadun perusteella. Retriever-komponentin suorituskykyä voidaan arvioida useilla mittareilla, joista osa kuvaa itse hakutulosten laatua ja osa retrieverin sisäisen embedding-mallin kykyä kuvata tekstien semanttista samankaltaisuutta. Retriever-komponentin arvioinnissa keskeistä on mitata nimenomaan hakutulosten järjestyksen ja kattavuuden laatua, sillä nämä määrittävät, millaista kontekstia generatiivinen kielimalli saa käyttöönsä.

Tiedonhaun vaiheeseen liittyviä mittareita ovat:

- Recall@K — Osuus kysymyksistä, joissa vähintään yksi relevantti dokumentti löytyy K:n parhaan hakutuloksen joukosta. Esimerkiksi Recall@5 = 0,9 tarkoittaa, että 90 prosentissa kysymyksistä relevantti dokumentti löytyy viiden ensimmäisen tuloksen joukosta [15].

- Precision@K — Osuus kärkituloksista, jotka ovat relevantteja. Esimerkiksi Precision@10 = 0,9 tarkoittaa, että 9/10 kärkituloksesta on oikeita osumia [14].

- Retrieval accuracy — Binäärinen mittari, joka ilmaisee, löytyykö ensimmäinen relevantti dokumentti tavoiterankingin (esim. top 1 tai top 3) sisällä [14].

- Semanttinen samankaltaisuus (Semantic similarity score) — Mittaa, kuinka lähellä kaksi tekstivektoria ovat toisiaan vektoriavaruudessa, esimerkiksi kosinietäisyyden avulla [14].

- MRR (Mean Reciprocal Rank) — Ensimmäisen relevantin dokumentin sijoituksen käänteisarvojen keskiarvo eri kyselyiden yli. Korkeampi MRR tarkoittaa, että relevantit tulokset löytyvät aikaisemmin hakutuloksissa [15].

- nDCG (normalized discounted cumulative gain) — Arvioi rankingin laatua painottamalla enemmän oikein järjestettyjä, erittäin relevantteja dokumentteja [14].

Esimerkiksi jos mallin retriever-komponentilla on korkea recall mutta matala precision, pakottaa se kielimallin käsittelemään paljon epäolennaista sisältöä, mikä lisää hallusinaatioiden riskiä. Toisaalta korkea precision mutta matala recall voi johtaa siihen, että olennaista tietoa jää pois. Embedding-mallin osalta taas huonosti sovitettu malli voi vääristää samankaltaisuuspisteitä siten, että semanttisesti epäolennaiset kohdat sijoittuvat relevanttien edelle. Tämä heikentää haun laatua riippumatta siitä, kuinka hyvä myöhempi ranking- tai kyselymalli on. [14.]

6 Fine-tuningin toteutus

Lähtökohtana työn tekemiselle olivat käytettävissä olevat, avoimen lähdekoodin mallit, niin tiedonhaun (retriever-komponentin) osalta kuin havainnollistavassa generoinnin osuudessa. Ennen retriever-mallin hienosäätöä pyrittiin muodostamaan mahdollisimman vahva lähtötilanne optimoimalla aineiston esikäsittely, paloittelu ja vektorihakua tukevat ratkaisut.

6.1 Lähdeaineistot ja niiden käsittely

Retriever-mallin hienosäädön vaikutusta tiedonhaun tarkkuuteen testattiin eläkevakuuttamisen toimialueen aineistolla. Pääpaino oli julkisten alojen eläkevakuuttamisessa, erityisesti Kevan tuottamissa verkkosivusisällöissä ja PDF-dokumenteissa, joita täydennettiin Eläketurvakeskuksen työeläkelakipalvelun ohjeistuksilla. Vaikka aineisto rajautuu tiettyyn toimialaan, se kattaa sisällöllisesti laajan kokonaisuuden sisältäen useita teemoja, käsitteitä ja asiakasryhmiä.

Ennen aineiston hyödyntämistä tiedonhaussa ja mallin jatkokoulutuksessa lähdetekstit esikäsiteltiin ja jaettiin pienempiin tekstikappaleisiin (chunkkeihin). Tavoitteena oli muodostaa sellaisia tekstikokonaisuuksia, jotka säilyttävät merkityksellisen asiayhteyden, mutta ovat riittävän lyhyitä embedding-mallin kontekstirajoihin nähden. Aineisto käsiteltiin yhtenäisellä periaatteella riippumatta alkuperäisestä tiedostomuodosta.

Sekä verkkosivusisällöt että PDF-dokumentit muunnettiin ensin rakenteiseksi Markdown-muodoksi. **Verkkosivuaineisto** haettiin Trafilatura-kirjaston avulla, joka suodattaa HTML-sisällöstä epäolennaiset elementit ja tuottaa rakenteisen tekstiesityksen. Trafilatura on kehittynyt Python-työkalu, joka on suunniteltu verkkotekstin keräämiseen ja raakamuotoisen HTML:n muuttamiseen rakenteiseksi ja merkitykselliseksi dataksi. Se sisältää kaikki tarvittavat tiedonhaku- ja tekstinkäsittelykomponentit, joilla voidaan suorittaa verkon indeksointia, latauksia, tietojen keruuta (scraping) sekä päätekstien, metadatan ja kommenttien poimintaa. [16.]

PDF-dokumentit muunnettiin vastaavasti Markdown-muotoon PyMuPDF4LLM-kirjaston avulla, joka hyödyntää dokumenttien visuaalisia vihjeitä, kuten otsikkorakennetta ja sivuasettelua. Näin muunnettu teksti säilyttää otsikot, listat ja taulukot mahdollisimman alkuperäisessä muodossaan. PyMuPDF4LLM on PyMuPDF-kirjastoon pohjautuva laajennus, joka on suunniteltu erityisesti suurten kielimallien ja RAG-sovellusten tarpeisiin. [17.]

Markdown-muotoon muunnettu teksti jaettiin ensin otsikkotasojen perusteella loogisiin kokonaisuuksiin, minkä jälkeen tekstit paloiteltiin pienemmiksi tekstikappaleiksi Recursive splitter -menetelmällä. Jokaiselle tekstikappaleelle lisättiin metatiedot sekä dokumentin otsikkorakenteeseen perustuva kontekstimerkintä, jotta retriever-malli pystyy huomioimaan tekstin asiayhteyden hakutilanteessa.

Koko lähdeaineistosta muodostettiin kaksi erillistä kokonaisuutta: laajempi aineisto, joka sisälsi 1208 tekstikappaletta, sekä tiiviimpi aineisto, joka sisälsi 332 tekstikappaletta. Lähdeaineistot on listattu liitteessä 1.

6.2 Faiss-indeksin muodostus

Esikäsitellyistä ja tekstikappaleisiin jaetuista lähdeaineistoista muodostettiin FAISS-vektori-indeksi semanttista tiedonhakua varten. Tekstikappaleille laskettiin embedding-vektorit intfloat/multilingual-e5-small -mallilla, joka tukee monikielistä semanttista hakua. FAISS (Facebook AI Similarity Search) on avoimen lähdekoodin kirjasto, joka on tarkoitettu tiheiden vektorien tehokkaaseen samankaltaisuushakuun ja lähimmän naapurin etsimisen (nearest neighbor search) [18].

FAISS-indeksi muodostettiin LangChain-kirjaston FAISS-integraation oletusasetuksilla käyttäen flat-tyyppistä indeksiä. Embeddingit normalisoitiin ennen indeksointia, jolloin samankaltaisuusvertailu vastaa kosinipohjaista semanttista samankaltaisuutta. Indeksin yhteyteen tallennettiin dokumenttien tekstit ja metatiedot LangChainin docstore-rakenteeseen, jotta hakutulokset voidaan palauttaa alkuperäisinä tekstikatkelmina lähdetietoineen. Dokumentit indeksoitiin E5-mallin suosittelemaa “passage:”-prefiksiä käyttäen.

6.3 Koulutus- ja testiaineiston muodostaminen

Retriever-mallin hienosäätöä varten muodostettiin erilliset koulutus-, validointi- ja testiaineistot, jotka koostuivat kysymys–tekstipareista. Testiaineistoa käytettiin alkuperäisen mallin ja hienosäädetyn mallin suorituskyvyn vertailuun, eikä se ollut käytettävissä fine-tuning-prosessin aikana.

Laajempi aineisto:

- Koulutusaineisto 1050 riviä sisältäen 315 positiivista ja 735 negatiivista kysymys-tekstiparia
- Validointiaineisto 65 kpl kysymyksiä, joihin yhteensä 93 kpl luokiteltuja, relevantteja tekstinpätkiä
- Testiaineisto mallien tulosten vertailuun: 69 kpl kysymyksiä, joihin yhteensä 81 kpl luokiteltuja, relevantteja tekstinpätkiä

Tiiviimpi aineisto:

- Koulutusaineisto: 194 kpl positiivisia kysymys-tekstipareja
- Validointiaineisto: 33 kpl kysymyksiä, joihin yhteensä 41 kpl luokiteltuja, relevantteja tekstinpätkiä
- Testiaineisto mallien tulosten vertailuun: 57 kpl kysymyksiä, joihin yhteensä 78 kpl luokiteltuja, relevantteja tekstinpätkiä

6.3.1 Kysymysten muodostaminen

Kysymykset muodostettiin lähdeaineistojen pohjalta hyödyntäen ChatGPT5.2-kielimallia, jota ohjattiin tuottamaan kysymyksiä annettujen esimerkkien perusteella. Mallin tuottamat kysymykset tarkistettiin ja muokattiin manuaalisesti, jotta ne olivat sisällöllisesti täsmällisiä ja vastasivat todellisia tiedontarpeita.

Kysymysten muodostamisessa kiinnitettiin erityistä huomiota semanttisesti läheisiin käsitteisiin, joissa oikea käyttökonteksti on olennainen, kuten työkyvyttömyyseläke ja työkyvyttömyyseläkemaksu. Kysymykset jaettiin karkeasti kahteen kokonaisuuteen: työnantajan eläkevakuuttamiseen (noin 80 % kysymyksistä) ja henkilön eläketurvaan (noin 20 %). Lisäksi oli jaoteltavissa 15 eri teemaan.

Kysymyksistä muodostettiin kaksi erillistä settiä. Tiiviimpi kysymyssetti keskittyi Kevan verkkosivusisältöihin ja PDF-dokumentteihin, ja siinä panostettiin monipuoliseen termistöön rajatun aineiston sisällä. Laajempi kysymyssetti sisälsi lisäksi Eläketurvakeskuksen aineistoihin perustuvia

kysymyksiä ja kattoi laajemman teemakokonaisuuden. Kysymykset jaettiin koulutus-, validointi- ja testisetteihin siten, että eri teemojen painoarvo säilyi kaikissa aineistoissa.

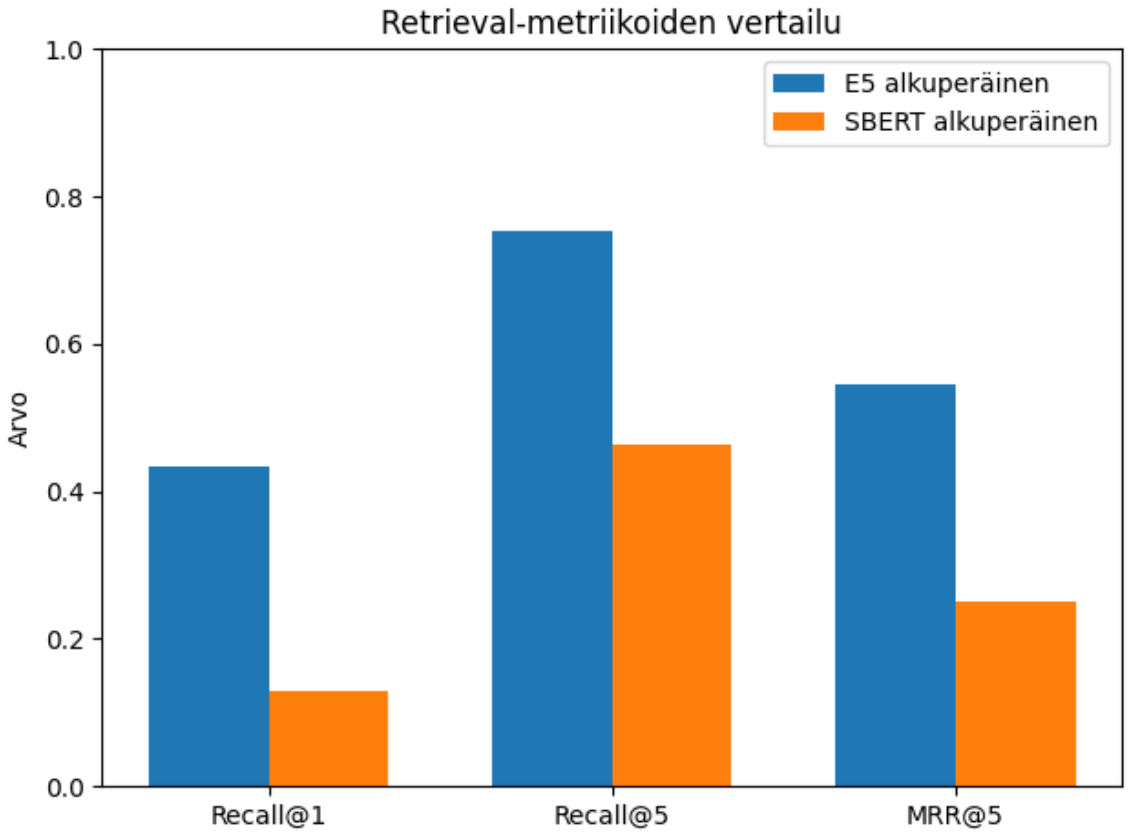
6.3.2 Kysymys–tekstiparien muodostaminen

Koulutusaineiston kysymys–tekstipareissa tekstikappale oli joko relevantti tai ei-relevantti suhteessa kysymykseen. Laajemman aineiston osalta koulutusdataa täydennettiin niin sanotuilla hard negative -näytteillä. Näissä hyödynnettiin alkuperäisen retriever-mallin tuottamia top-5-hakutuloksia, joista kysymykseen nähden epäolennaiset mutta semanttisesti läheiset tekstikappaleet valittiin manuaalisesti negatiivisiksi esimerkeiksi. Tutkimuskirjallisuudessa on osoitettu, että tällaiset niin sanotut hard negative -näytteet tehostavat dense retriever -mallien oppimista. Niitä käytetään opettamaan mallia erottamaan relevantit ja ei-relevantit tekstikatkelmat toisistaan. [19, s. 1–2.]

Negatiivisia näytteitä hyödynnettiin fine-tuning -prosessin testauksessa, kun käytössä oli negatiivisia näytteitä tarvitseva loss-funktio. Vastaavat validointi- ja testiaineistot muodostettiin siten, että niihin sisällytettiin vain relevantit tekstikappaleet. Validointiaineistoa käytettiin fine-tuning-prosessin aikana mallin suorituskyvyn arviointiin jokaisen koulutuskierroksen jälkeen, kun taas testiaineistoa hyödynnettiin ainoastaan alkuperäisen ja hienosäädetyn mallin vertailussa.

6.4 Mallin valinta

Mallin valintaan vaikutti edellytys sen toimimiseen suomen kielellä. Valinta tehtiin suomen kielellä koulutetun TurkuNLP/sbert-uncased-finnish-paraphrase -mallin ja monikielisen intfloat/multilingual-e5-small -mallin välillä. Testikysymykset ajettiin molemmilla malleilla ja valittiin paremmin lähtötilanteessa selvinnyt e5-malli. Mallien vertailussa käytettiin laajemman aineiston testikysymyksiä. Kuvassa 2 sekä taulukossa 1 esitetään mallien suoriutuminen mittareiden valossa, testikysymysten määrän ollessa 69 kpl.



Kuva 2. Mallien lähtötilanteen vertailu eri metriikoilla.

Taulukko 1. Mallien lähtötilanteen vertailu.

Mittari	e5	SBERT
Recall@1	0.435	0.130
Recall@5	0.754	0.464
MRR@5	0.545	0.251
Kosini keskiarvo	0.886	0.658

Työhön valikoitunut intfloat/multilingual-e5-small perustuu vuonna 2023 julkaistuun multilingual-E5-embedding-malliperheeseen, jonka kolmesta mallista tässä työssä käytetty on pienin. Mallin koulutus on toteutettu kaksivaiheisella menetelmällä: ensin laajamittaisella heikosti valvotulla kontrastiivisella esikoulutuksella (weakly-supervised contrastive pre-training), jossa malli oppii erottamaan semanttisesti samankaltaiset ja erisisältöiset tekstiparit toisistaan, ja sen jälkeen valvotulla hienosäädöllä (supervised fine-tuning) monikielisillä tietojoukoilla. Malli tukee yli sataa kieltä ja soveltuu erityisesti tiedonhaun tehtäviin. [20, s. 1.]

6.5 Fine-tuning -prosessin kuvaus ja arviointi

Mallin fine-tuning toteutettiin paikallisessa koulutusympäristössä käyttäen henkilökohtaista työasemaa. Koulutus suoritettiin CPU-pohjaisesti, mikä oli riittävä kevyelle mallille ja datamäärälle, erityisesti kun koulutusta tehtiin muun raportoinnin rinnalla. Retriever-malli hienosäädettiin käyttäen Sentence Transformers -kirjastoa ja valmiiksi koulutettua monikielistä mallia intfloat/multilingual-e5-small. Mallin valintaa on kuvattu tarkemmin kappaleessa 6.4. Mallin suorituskykyä arvioitiin fine-tuningin aikana InformationRetrievalEvaluator-arvioijalla jokaisen koulutuskierroksen (epochin) jälkeen. Tämä evaluointityökalu on osa Sentence Transformers -kirjaston arviointikomponenttia, ja se laskee retriever-mallin hakutulosten laatuun liittyviä mittareita, kuten Recall@K ja MRR [21].

Hienosäädetty malli tallennettiin ja alkuperäisistä tekstikappaleista muodostettiin uusi FAISS-indeksi käyttäen hienosäädettyä mallia embedding-vektoreiden muodostuksessa. Tämän jälkeen testidata ajettiin sekä alkuperäisellä että hienosäädetyllä mallilla tulosten vertailua varten.

6.5.1 Loss-funktion testaus ja valinta

Fine-tuning-prosessissa loss-funktio vaikuttaa ratkaisevasti mallin suorituskykyyn. Loss-funktiot kuvaavat, miten hyvin malli suoriutuu annetusta datasta, ja ohjaavat optimointialgoritmia päivittämään mallin painoja siten, että virhe pienenee. Tämä muodostaa mallin koulutusprosessin perustan. [22.]

Retriever-mallin hienosäädössä keskeistä on ohjata mallia oppimaan sellainen upotusavaruus (embedding space), jossa semanttisesti relevantit kysely–dokumentti-parit sijoittuvat lähelle toisiaan ja ei-relevantit kauemmas [12]. Tätä varten hyödynnetään kontrastiiviseen oppimiseen perustuvia loss-funktioita, jotka vertaavat positiivisia ja negatiivisia esimerkkejä toisiinsa ja pakottavat relevantit dokumentit saamaan korkeamman samankaltaisuuspisteen kuin ei-relevantit [23].

Fine-tuning -prosessissa testattiin kolmea kontrastiiviseen oppimiseen perustuvaa loss-funktiota: Multiple Negatives Ranking Loss, Contrastive Loss ja Triplet Loss.

Multiple Negatives Ranking Loss on suunniteltu erityisesti dense retrieval -tehtäviin. Menetelmässä jokaiselle kyselylle sen oikea dokumentti käsitellään positiivisena esimerkkinä, ja kaikki

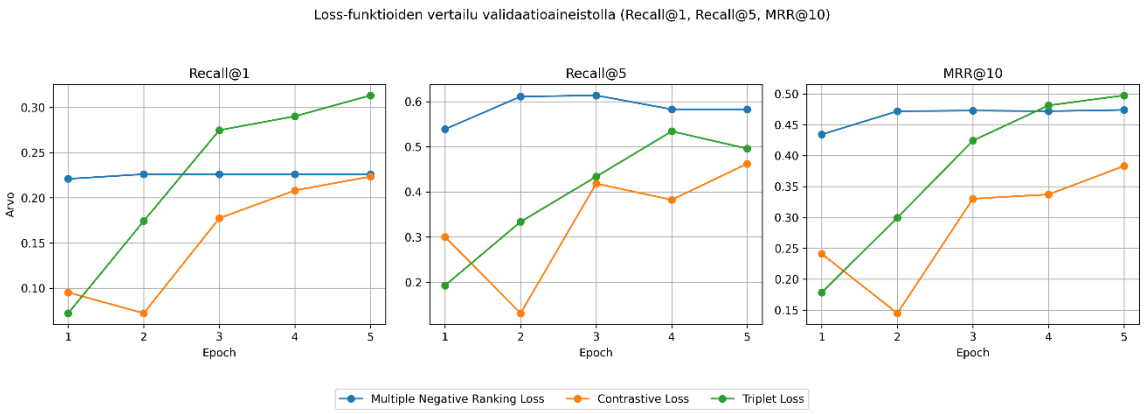
muut saman minibatchin dokumentit toimivat negatiivisina. Menetelmä ei edellytä erillistä negatiivista koulutusdataa, koska negatiiviset esimerkit muodostuvat automaattisesti minibatchin sisällä. Loss-funktion tavoitteena on kasvattaa relevanssia kyselyn ja positiivisen dokumentin välillä sekä pienentää relevanssia kyselyn ja negatiivisten dokumenttien välillä. [23.]

Triplet Loss soveltuu käytettäväksi silloin, kun koulutusdata sisältää hard negative -esimerkkejä. Se hyödyntää kolmikoita: kysely (anchor), relevantti dokumentti (positive) ja ei-relevantti dokumentti (negative). Sen tavoitteena on varmistaa, että kysely on lähempänä relevanttia dokumenttia kuin ei-relevanttia dokumenttia vähintään ennalta määritellyn marginaalin verran. [24.]

Myös **Contrastive Loss** tarvitsee negatiivisia esimerkkejä. Se käsittelee, toisin kuin Triplet Loss, kysely–dokumentti-pareja toisistaan riippumattomina ja ohjaa mallia pienentämään positiivisten parien välistä etäisyyttä sekä kasvattamaan negatiivisten parien etäisyyttä. [25.]

Loss-funktioiden testaus suoritettiin laajemmalla datasetillä, joka sisälsi yhteensä 1050 esimerkkiä (315 positiivista ja 735 negatiivista). MultipleNegativeLoss hyödynsi vain positiivisia rivejä. Triplet Lossia varten muodostettiin 965 triplettiä (kysymys, oikea teksti, väärä teksti). Validointiaineisto sisälsi 65 kysymystä.

Funktioiden testitulokset on esitetty kuvassa 3. Testaus suoritettiin batch-koolla 8, mallin oletusoppimisnopeudella sekä warmupsteps-arvolla 20.



Kuva 3. Loss-funktioiden tulokset koulutuskierroksittain. Batch-koko 8.

Parametrien jatkotestaukseen valittiin Multiple Negatives Ranking Loss, jolla saavutettiin paras Recall@5-arvo ja tasainen suoritus muilla mittareilla. Triplet Loss kehittyi hitaammin, mutta saavutti lopulta hyviä tuloksia ja olisi siten potentiaalinen vaihtoehto jatkotyöhön. Contrastive Loss osoittautui tässä aineistossa epävakaimmaksi.

6.5.2 Validointi eri parametreillä

Fine-tuningin onnistumiseen vaikuttavat merkittävästi käytetyt hyperparametrit, jotka ohjaavat oppimisprosessin kulkua, nopeutta ja vakautta. Tässä työssä keskityttiin seuraaviin parametreihin.

- **Epoch** kuvaa sitä, kuinka monta kertaa koko koulutusaineisto käydään läpi fine-tuningin aikana. Pieni epoch-määrä voi johtaa alisovittamiseen (underfitting), kun taas liian suuri määrä kasvattaa ylisovittamisen (overfitting) riskiä, erityisesti silloin, kun käytössä on rajallinen määrä koulutusaineistoa. [26.]

- **Batch-koko (Batch size)** määrittelee, kuinka monta opetusparia käsitellään samanaikaisesti yhden painopäivityksen aikana [26].

- **Learning rate (Oppimisnopeus)** määrittelee, kuinka suuria painopäivityksiä malli tekee kunkin optimointiaskeleen aikana. Liian suuri oppimisnopeus voi aiheuttaa oppimisprosessin epävakautta, kun taas liian pieni arvo hidastaa oppimista merkittävästi. [27.]

- **Warmup steps.** Warmup steps määrittelee alkuvaiheen optimointiaskeleiden määrän, joiden aikana oppimisnopeus kasvatetaan lineaarisesti määriteltyyn maksimiarvoon [28].

- **Weight decay** on neuroverkkojen regularisointimenetelmä, jonka tarkoituksena on estää mallin ylisovittamista. Menetelmä toimii pienentämällä mallin painoarvoja hieman jokaisessa oppimisvaiheessa, jolloin malli pysyy yksinkertaisempana ja yleistyy paremmin uuteen dataan. [29.]

Parhaiden parametrien etsiminen toteutettiin kahdessa vaiheessa:

1. Testattiin eri batch-kokoja ja eri warmupsteps-arvoja viiden koulutuskierroksen (epoch) ajan, muiden parametrien ollessa oletusasetuksilla

2. Valitulla batch-koolla ja warmupsteps-arvolla testattiin eri learning rate- ja weight decay -arvoja.

Taulukossa 2 esitetään käytetyt parametrit sekä testivaiheessa testatut parametrivälit.

Taulukko 2. Valitut parametrien arvot.

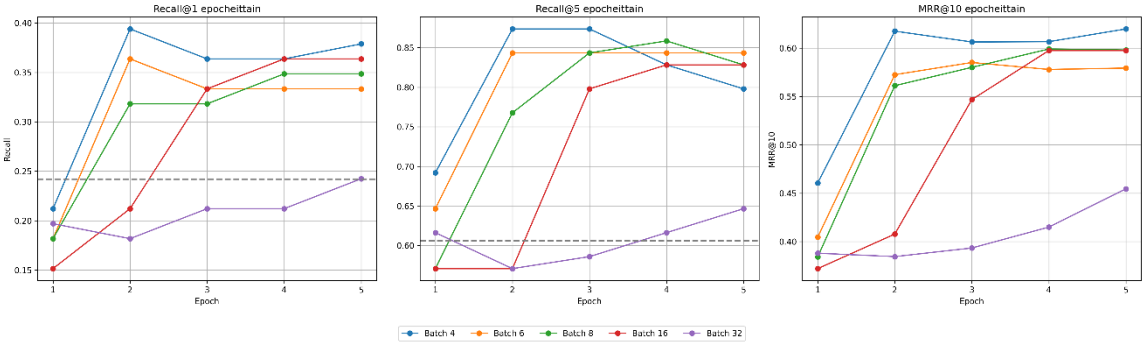
Parametri	Valittu arvo	Testattujen arvojen väli
Epoch	3	1…5
Batch-koko	4	4…32
Warmup steps	30	0…40
Learning rate	0.00003	0.00001…0.0001
Weight decay	0.00 (default)	0.00…0.1

Fine-tuning -prosessi eri parametreilla toteutettiin tiiviimmällä aineistolla, jossa oli 194 positiivista treeniparia sekä 33 validointiparia. Aineisto rajattiin teemoihin, joiden osalta tiedonhaun parantumisella on suurin merkitys työnantajien eläkevakuuttamisen asiakaspalvelun näkökulmasta.

Evaluointitulosten perusteella pienet batch-koot tuottavat tässä aineistossa parempia tuloksia, vaikka yleisesti on arvioitu Multiple Negatives Ranking Loss -funktion hyötyvän suurista batcheistä. Lisäksi havaittiin, että mallin suorituskyky parani voimakkaimmin kahden ensimmäisen epochin aikana. Toisen epochin jälkeen validointitulokset tasaantuivat tai heikkenivät, ja kolmannen epochin jälkeen saavutetut parannukset olivat pääosin marginaalisia, lukuun ottamatta suurinta testattua batch-kokoa.

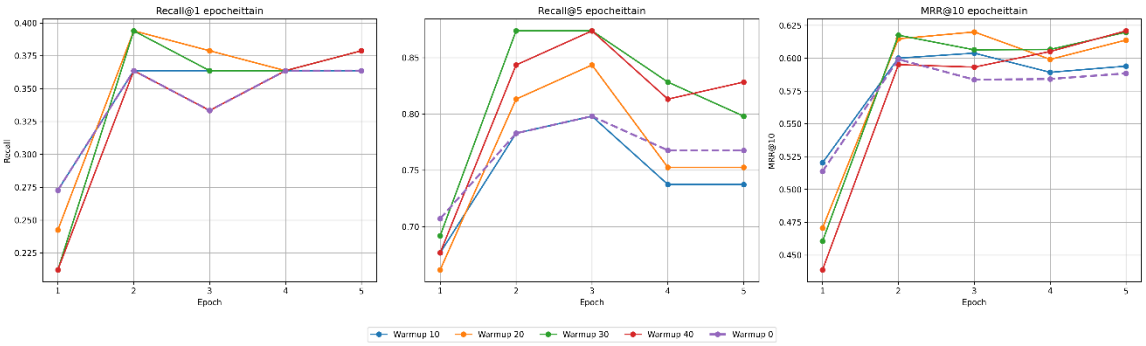
Kuvassa 4 esitetään fine-tuning -prosessin tulokset vertaillen eri batch-kokoja. Warmupstes-parametrin arvona on käytetty 30, joka osoittautui kaikilla batch-kokovaihtoehdoilla toimivaksi. Muut parametrit pidettiin mallin oletusasetuksissa. Erityisesti batch-koko 4 saavutti korkeimmat Recall@1- ja MRR@10-arvot kaikissa tarkastelluissa epocheissa, mikä viittaa parempaan tarkkuuteen ja dokumenttien järjestykseen.

Alkuperäisellä e5-mallilla validointiaineistossa osumatarkkuus (recall@1) on 0,242 ja viiden ensimmäisen osuman tarkastelussa (recall@5) 0,606. Huomionarvoista on, että mitä suurempi batch-koko on, sitä enemmän tulos heikkenee ensimmäisen epochin jälkeen ennen kuin fine-tuning-prosessi alkaa tuottamaan parempaa tulosta.



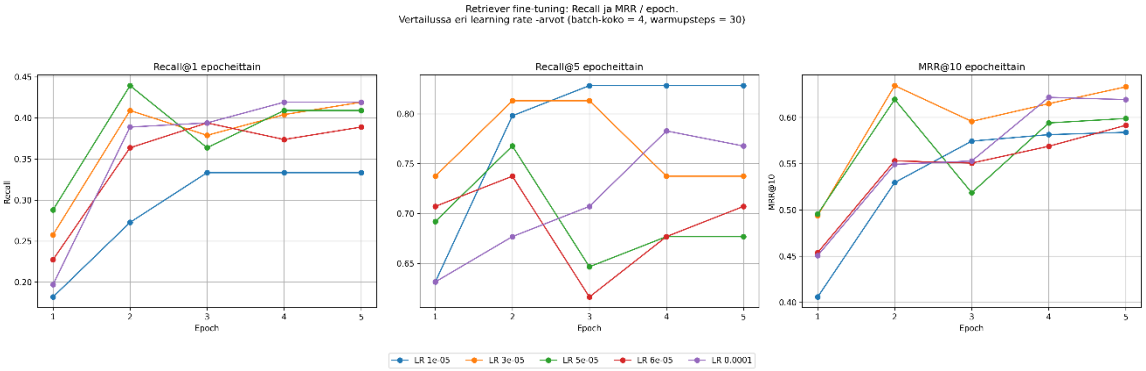
Kuva 4. Recall@1-, Recall@5- ja MRR@10-kehitys epocheittain, batch-koot 4,6,8,16 sekä 32.

Batch-koolla 4 testattiin tarkemmin warmupsteps -parametrin vaikutusta validoinnin tuloksiin. Kuva 5 esittää tuloksia, joista nähdään, että arvo 30 tuotti kokonaisuutena parhaat tulokset.



Kuva 5. Recall@1-, Recall@5- ja MRR@10-arvojen kehitys epocheittain eri warmupsteps-parametrin arvoilla. Batch-koko 4.

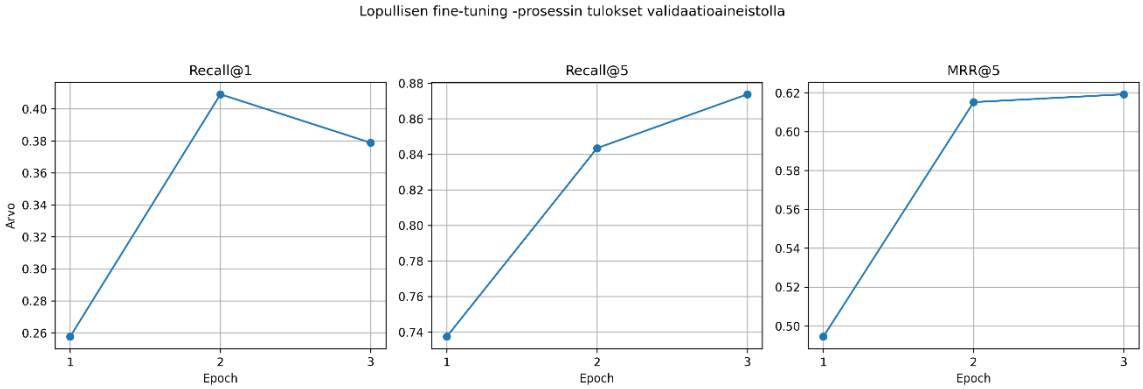
Valitulla batch-koolla ja warmupsteps-arvolla testattiin eri oppimisnopeuksia. Kuvasta 6 nähdään, että kaikilla testatuilla oppimisnopeuksilla malli oppii mutta eri asioita. Pienempi learning rate parantaa Recall@5-arvoa. Suurempi learning rate taas terävöittää Recall@1:tä mutta heikentää kattavuutta.



Kuva 6. Recall@1-, Recall@5- ja MRR@10-arvojen kehitys epocheittain eri learning rate-parametrin arvoilla. Batch-koko 4, warmupsteps 30.

Weight decay -parametrin arvoja testattiin valitulla oppimisnopeudella 0.00003, mikä oli kompromissi hyvän ensimmäisen osuman (Recall@1) ja kattavuuden (Recall@5) suhteen. Weight Decay ei yleensä vaikuta suoraan Recall tai MRR -mittareihin vaan epäsuorasti parantamalla mallin yleistymiskykyä ja vähentämällä ylisovittamista. Tästä syystä parametria arvioitiin tarkastelemalla koulutuksen loss-arvoja epocheittain. Kokeissa weight decay -parametria kasvatettiin merkittävästi, mutta koulutuksen aikana mitatut loss-arvot pysyivät käytännössä identtisinä verrattuna oletusasetuksiin. Tämä viittaa siihen, että käytetyssä fine-tuning-asetelmassa – lyhyt koulutus, pieni batch-koko ja kontrastiivinen loss-funktio – weight decay -parametrin säätämisellä ei ollut merkittävää vaikutusta mallin oppimiseen. Tämän vuoksi parametri säilytettiin oletusarvossaan (0.00).

Kuvassa 7 esitetään lopullisen fine-tuning -prosessin tulokset validointiaineistolla Recall-mittareiden sekä MRR@5-mittarin valossa. Vaikka tulokset poikkeavat hieman yksittäisistä testausvaiheista epoch-määrän pienentämisen vuoksi, ne ovat linjassa aiempien havaintojen kanssa. Lopulliset tulokset osoittavat, että malli vakautuu nopeasti ja saavuttaa parhaan suorituskyvyn kahden ensimmäisen epochin aikana. Recall@1- ja MRR@5-mittarit paranevat merkittävästi jo toisen epochin jälkeen, mikä viittaa huomattavasti tarkempaan dokumenttien järjestykseen verrattuna perusmalliin. Recall@5-mittari jatkaa kasvuaan myös kolmannessa epochissa.



Kuva 7. Lopulliset tulokset validointiaineistolla Recall- ja MRR-mittareiden valossa.

Taulukossa 3 esitetään vertailu alkuperäisen mallin sekä hienosäädetyn mallin suorituskyvystä samalla validointiaineistolla. Myös tämä viittaa mallin parempaan sovittumiseen haluttuun aineistoon.

Taulukko 3. Alkuperäisen ja hienosäädetyn mallin tulokset koulutuksen validointidatalla.

Mittari	Alkuperäinen malli	Hienosäädetty malli	Muutos %
Recall@1	0.24	0.38	+ 58 %
Recall@5	0.61	0.87	+ 43 %
MRR@5	0.36	0.62	+ 72 %

7 Fine-tuningin tulokset

Tässä luvussa tarkastellaan retriever-komponentin fine-tuningin vaikutuksia mallin suorituskykyyn. Analyysi keskittyy siihen, miten fine-tuning vaikuttaa relevanttien tekstikappaleiden löydettävyyteen, hakutulosten järjestykseen sekä embedding-avaruuden rakenteeseen. Tuloksia tarkastellaan sekä tiiviimmällä, rajatun teema-alueen testiaineistolla, että laajemmalla aineistolla.

7.1 Käytetyt arviointimenetelmät

Luvussa 5 on kuvattu RAG-järjestelmän arvioinnissa yleisesti käytettäviä mittareita. Tässä työssä arviointi kohdistuu nimenomaisesti retriever-komponentin suorituskykyyn ja siihen, miten fine-tuning vaikuttaa oikeiden tekstikappaleiden löytymiseen. Mittareiden valinnassa huomioitiin, että ne kuvaavat tiedonhaun laatua ja ovat toteutettavissa kohtuullisella työmäärällä käytettävissä olevalla aineistolla.

Arviointi perustui erikseen laadittuun testiaineistoon, joka on kuvattu kappaleessa 6.3. Kullekin testiaineiston kysymykselle oli manuaalisesti määritetty yksi tai useampi vastaava oikea tekstikappale. Testauksessa hyödynnettiin koko dokumenttisisällön sisältävää CSV-tiedostoa, joka sisälsi kaikkien mahdollisten tekstipätkien tunnisteet ja sisällöt.

Retrieverin suorituskykyä arvioitiin FAISS-pohjaisen haun avulla vertaamalla palautuneita hakutuloksia ennalta määriteltyihin oikeisiin dokumenttipaloihin. Arvioinnissa tarkasteltiin sekä sitä, löytyykö oikea dokumentti hakutulosten joukosta, että sitä, kuinka korkealle se sijoittuu hakutulosten rankingissa. Lisäksi embedding-mallin käyttäytymistä analysoitiin laskemalla kysymysten ja niihin liittyvien oikeiden tekstipätkien välinen kosinisamankaltaisuus.

Tässä työssä käytetyt arviointimittarit olivat:

- Recall@K, joka toimi työn keskeisimpänä mittarina ja kuvaa, kuinka usein relevantti dokumentti löytyy hakutulosten K parhaan joukosta.
- MRR (Mean Reciprocal Rank), joka arvioi hakutulosten järjestyksen laatua mittaamalla ensimmäisen relevantin dokumentin sijoitusta. Mitä suurempi MRR-arvo, sitä aikaisemmin oikea dokumentti keskimäärin sijoittuu hakutulosten listalla.

- Kosinisamankaltaisuus (Cosine similarity), jota käytettiin tukimittarina embedding-mallin laadun ja fine-tuningin vaikutusten analysointiin.

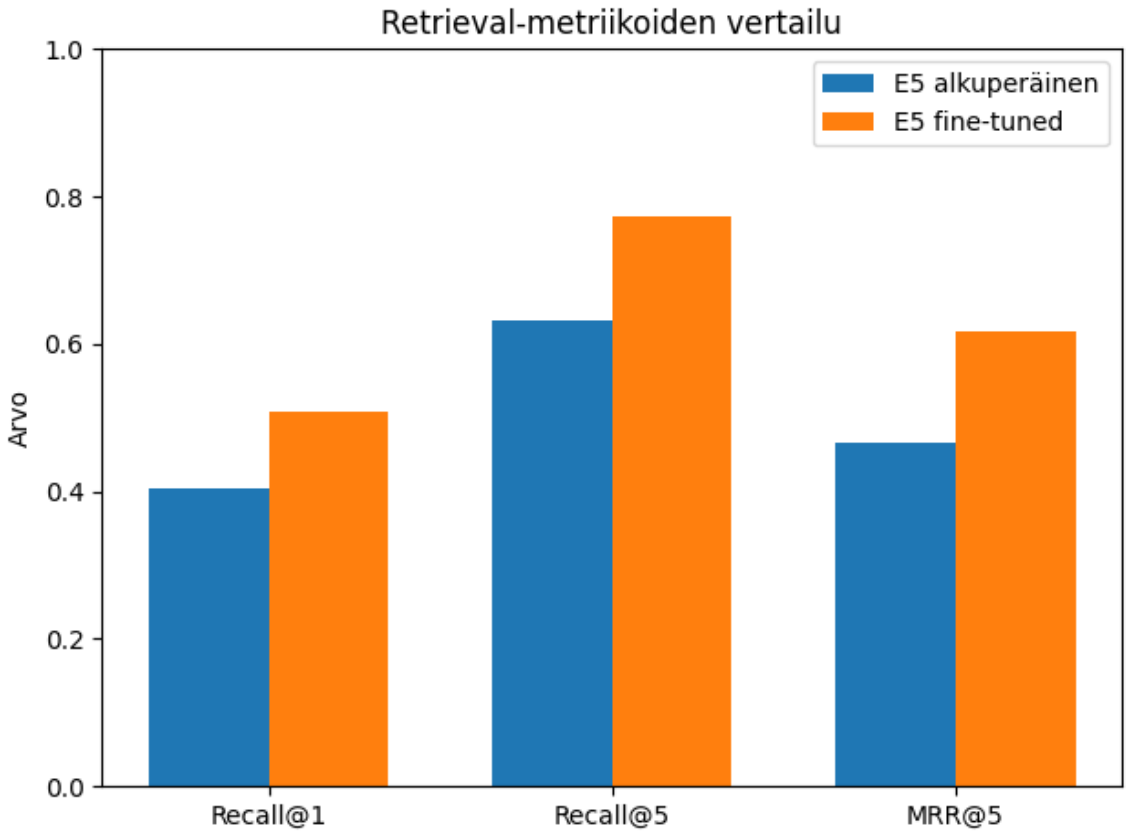
Recall@K ja MRR mittaavat suoraan retriever-komponentin kykyä löytää ja järjestää relevantit dokumentit, kun taas kosinisamankaltaisuus kertoo embedding-avaruuden muutoksista ennen ja jälkeen mallin fine-tuningin.

7.2 Tulokset tiiviimmällä koulutusaineistolla

Tässä osiossa esitetään alkuperäisen retriever-mallin sekä jatkokoulutetun mallin tulokset tiiviimmällä koulutusaineistolla. Retriever-mallina toimi intfloat/multilingual-e5-small, jonka lähtötasotestin tulokset olivat selkeästi vertailumallia TurkuNLP/sbert-uncased-finnish-paraphrase paremmat. Mallien vertailu on esitelty kappaleessa 6.4 ja mallin fine-tuning -prosessi kappaleessa 6.5.

Testiaineisto sisälsi 57 kysymystä, joille oli määritetty yksi tai useampi oikea tekstikappale. Testiaineisto oli samassa suhteessa eri teemoja kuin koulutusaineistossa. Aineiston muodostaminen on kuvattu tarkemmin kappaleessa 6.3. Tuloksia tarkastellaan mittareilla Recall@1, Recall@5 sekä MRR@5.

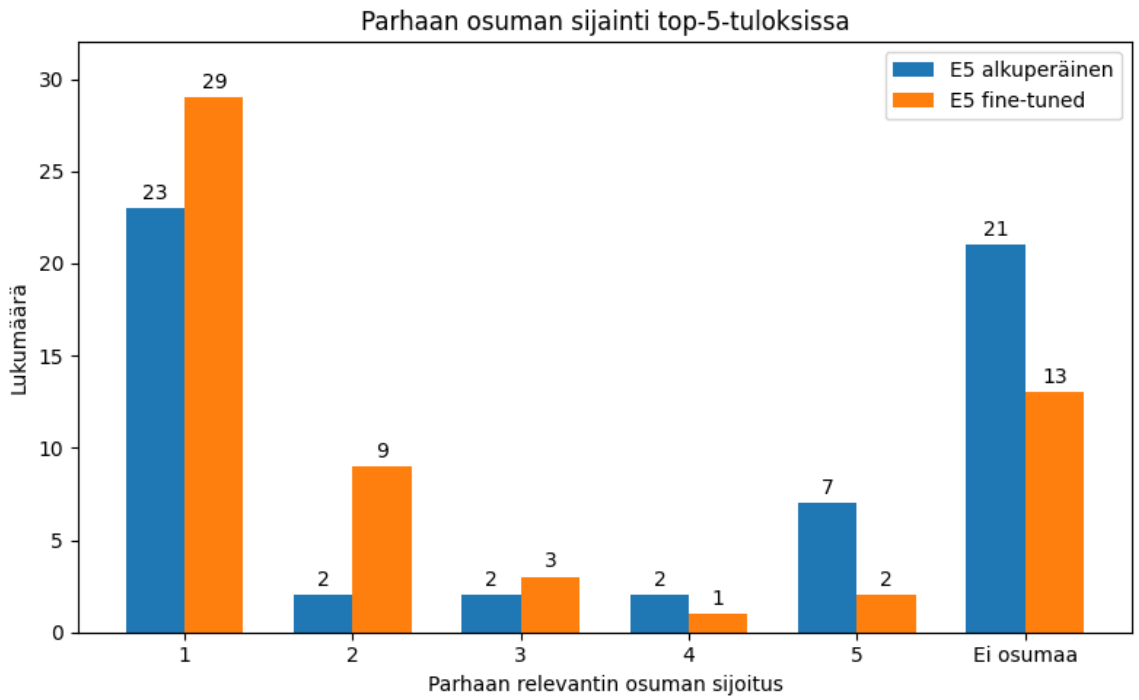
Kuvassa 8 esitetään tulokset alkuperäisellä sekä hienosäädetyllä mallilla. Hienosäädetty malli löytää oikean dokumentin useammin heti ensimmäisenä (Recall@1), mutta myös löydettävyys viiden parhaan osuman joukosta (Recall@5) on parantunut. Lisäksi MRR@5-arvon kasvu osoittaa, että hakutulosten järjestys on hienosäädetyllä mallilla keskimäärin parempi.



Kuva 8. Tulokset alkuperäisellä sekä hienosäädetyllä mallilla.

Alkuperäisellä mallilla Recall@1 oli 40,4 %. Hienosäädön jälkeen osumatarkkuus nousi 50,9 prosenttiin, mikä vastaa 26 prosentin suhteellista parannusta. Recall@5-arvo parani 63,2 prosentista 77,2 prosenttiin, eli oikea dokumentti löytyi viiden parhaan osuman joukosta 22 % useammin. MRR@5-arvo nousi vastaavasti 0,466:sta 0,617:ään, mikä on 32 prosentin suhteellinen parannus ja viittaa merkittävään parannukseen dokumenttien keskimääräisessä sijoituksessa.

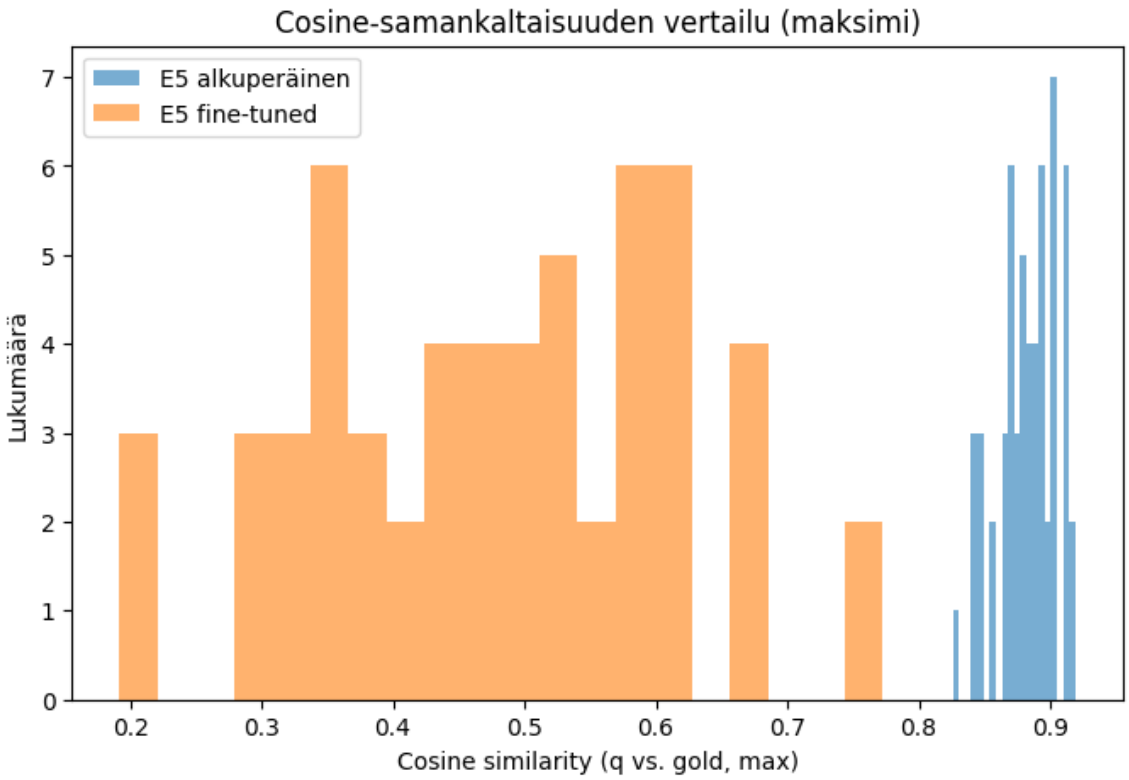
Hakutulosten sijoitusten jakaumaa tarkasteltaessa havaitaan, että alkuperäisellä mallilla huomattava osa kysymyksistä jäi ilman relevanttia osumaa viiden parhaan tuloksen joukossa. Hienosäädetyllä mallilla tällaisten tapausten määrä väheni 21:stä 13:een (-38 %), ja samalla osumien sijoittuminen erityisesti toiselle sijalle lisääntyi. Tämä näkyy kuvassa 9 ja selittää osaltaan Recall@5- ja MRR@5-mittareiden paranemista.



Kuva 9. Parhaan osuman sijainti top5-tulosten joukossa sekä ilman osumaa alkuperäisellä sekä hienosäädetyllä e5-mallilla.

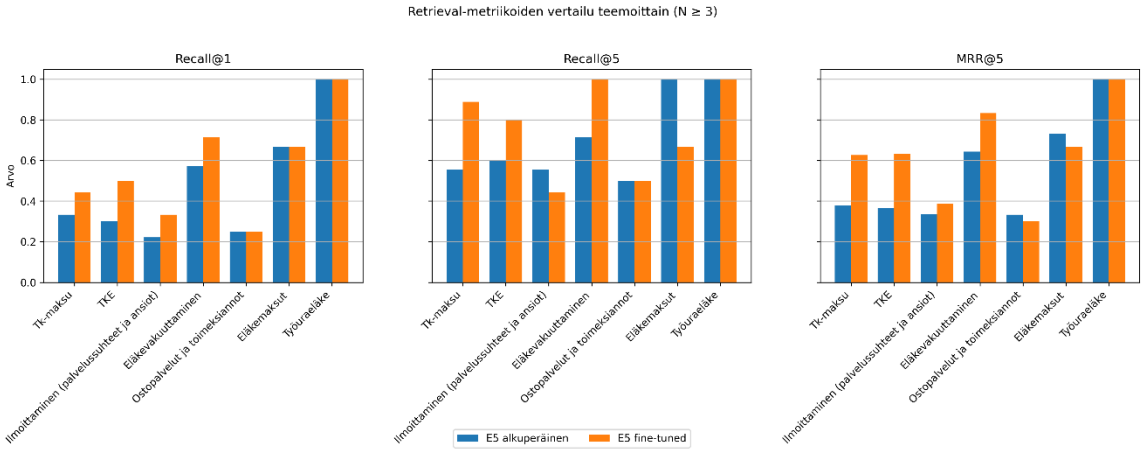
Kun verrataan fine-tuning -prosessin validointiaineistoa sekä testiaineistoa mittareiden valossa, huomataan, että validointiaineistolla perusmallin lähtötaso oli selvästi heikompi kuin testiaineistolla. Tämä viittaa siihen, että validointiaineiston kysymykset olivat mallille haastavampia. Tästä huolimatta hienosäädetty malli paransi suorituskykyä molemmissa aineistoissa verrattuna perusmalliin. Tämä tukee tulkintaa siitä, että fine-tuning on opettanut tehtävään liittyviä yleisiä piirteitä eikä ylisovittanut validointiaineistoon.

Kosinisamankaltaisuuden (Cosine Similarity) mittarilla tarkasteluna fine-tuning muutti embedding-avaruuden rakennetta. Kuvasta 10 nähdään, että hienosäädetyn mallin tuottamat kosinisamankaltaisuudet ovat selvästi hajautuneempia verrattuna alkuperäiseen E5-malliin, jonka samankaltaisuusarvot keskittyvät kapealle ja korkealle alueelle. Yksistään tämän tuloksen perusteella ei voida päätellä mallin suorituskyvyn muutosta. Mutta vaikka kyselyn ja oikean dokumentin välinen kosinisamankaltaisuus keskimäärin pienenee, dokumenttien järjestys paranee, mikä nähdään Recall- ja MRR-mittareiden kasvuna.



Kuva 10. Kosinisamankaltaisuus alkuperäisellä sekä hienosäädetyllä mallilla.

Entä miten kävi eri teemojen tuloksille? Yhtenä erityisenä kiinnostuksen kohteena oli selvittää, voidaanko malli opettaa erottamaan toimialan semanttisesti läheiset, mutta sisällöllisesti eri merkityksiä kantavat käsitteet, erityisesti työkyvyttömyyseläkemaksu ja työkyvyttömyyseläke. Tämän tavoitteen onnistumista voidaan tarkastella kuvasta 11, joka esittää testiaineiston tulokset teemoittain.



Kuva 11. Alkuperäisen mallin ja hienosäädetyn mallin tulokset teemakohtaisesti.

Hienosäädetty malli suoriutui paremmin lähes kaikissa teemoissa, vaikka parannusten suuruus vaihteli. Erityisesti haun tarkkuus näyttää parantuneen työkyvyttömyyseläkemaksussa (Tk-maksu), työkyvyttömyyseläkkeessä (TKE) sekä eläkevakuuttamisen teeman kysymyksissä. Sen sijaan jopa laskua on nähtävissä ilmoittamisen ja eläkemaksujen teemoissa.

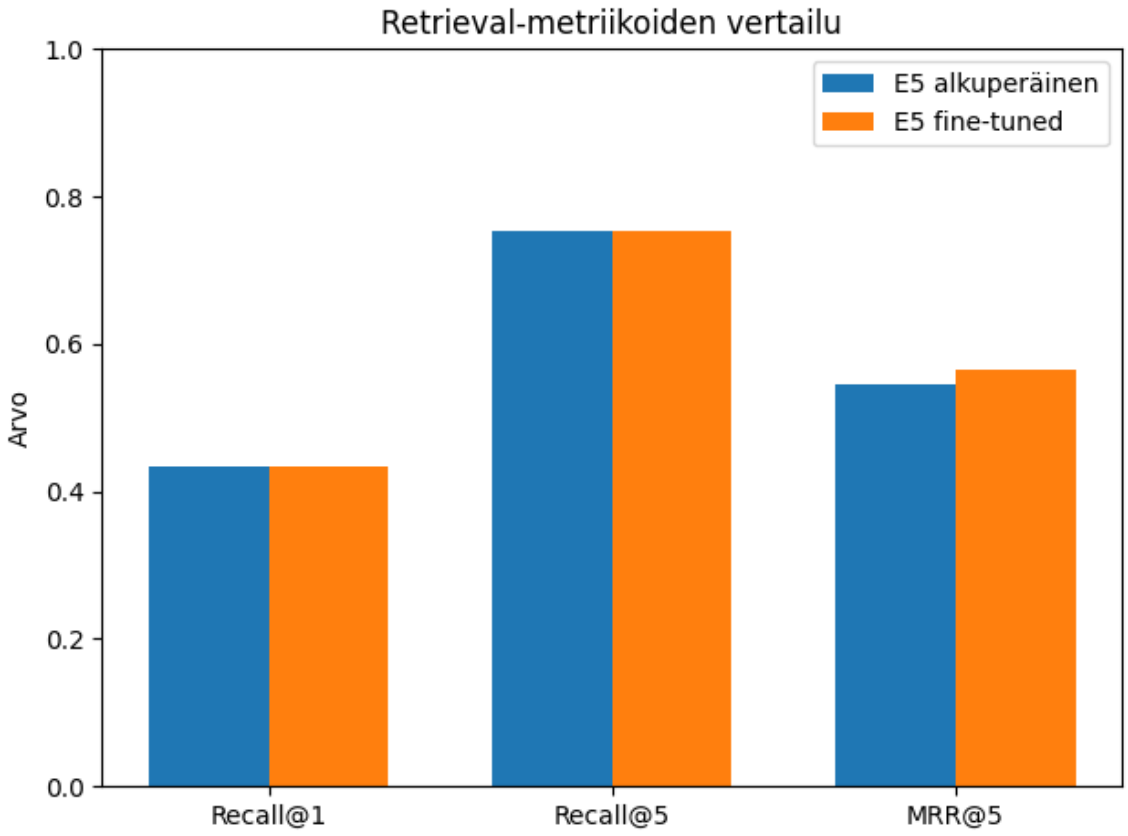
Taulukosta 4 nähdään, että juuri mielenkiinnonkohteena olevissa teemoissa, tk-maksu ja TKE, parannus on ollut suurin. Näyttäisi siis siltä, että parannusta on saatu siihen, mitä tavoiteltiin.

Taulukko 4. Alkuperäisen ja hienosäädetyn mallin suorituskyvyn muutos teemakohtaisesti.

Teema	Kysymysten lukumäärä	Recall@1 muutos	Recall@5 muutos	MRR@5 muutos
Tk-maksu	18	+ 33 %	+ 60 %	+ 65 %
TKE	10	+ 67 %	+ 30 %	+ 74 %
Ilmoittaminen	9	+ 50 %	- 20 %	+ 15 %
Eläkevakuuttaminen	7	+ 25 %	+ 40 %	+ 30 %
Toimeksiannot	4	0 %	0 %	- 10 %
Eläkemaksut	3	0 %	- 30 %	- 9 %
Työuraeläke	3	0 %	0 %	0 %
Painotettu keskiarvo		+ 29 %	+ 23 %	+ 35 %

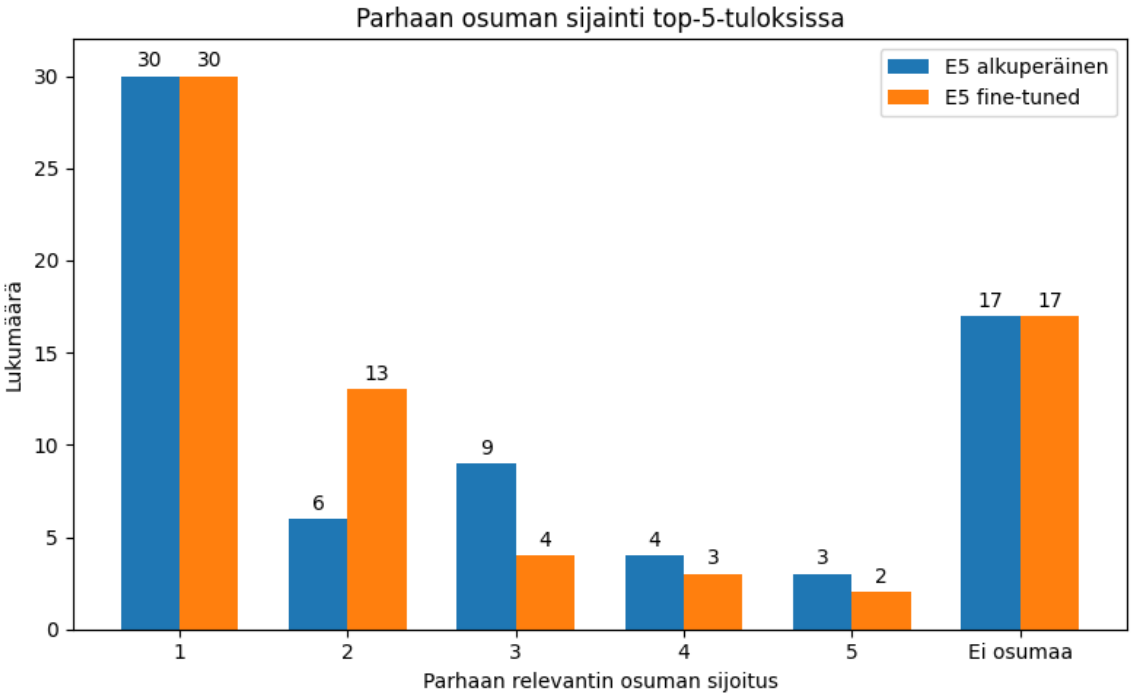
7.3 Tulokset laajemmalla koulutusaineistolla

Vertailun vuoksi toteutettiin fine-tuning myös koko laajalle koulutusdatalle, jossa teemoja ei ollut rajattu. Testiaineisto sisälsi 69 kysymystä. Oletus oli, että tulokset jäävät heikommiksi, koska dataaineisto ei ollut aihealueen laajuuteen nähden yhtä kattava kuin tiiviimpi aineisto. Kuvasta 12 nähdään, että fine-tuning ei parantanut kokonaisuutena osumatarkkuutta. Ainoastaan aivan pientä parannusta tulosten sijoituksiin nähdään MRR@5-mittarin muutoksesta.



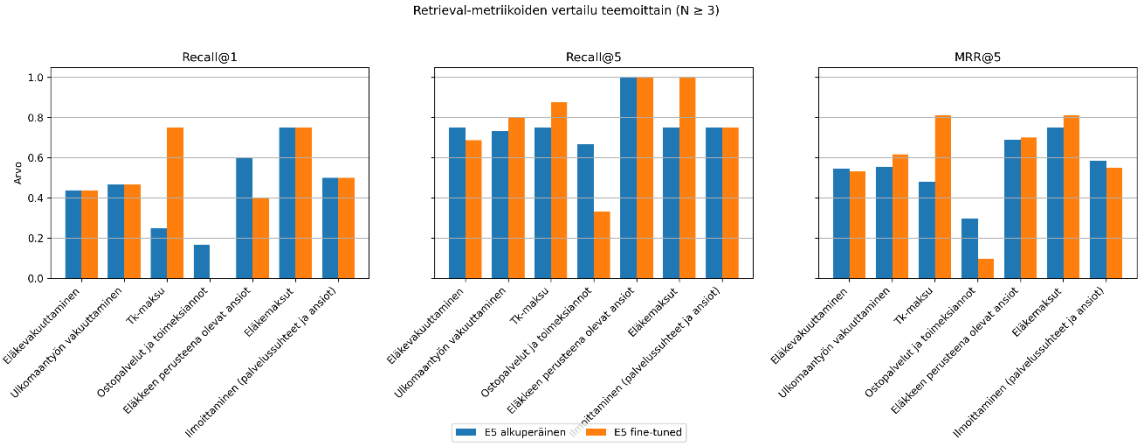
Kuva 12. Tulokset alkuperäisellä sekä hienosäädetyllä mallilla.

Kuvasta 13 nähdään, että ainoa kehitys koskee Recall@2-arvon muutosta eli alkuperäisen mallin hieman alemmilta sijoilta on noussut joitain kysymyksiä toiseen osumaan. Kuitenkaan esimerkiksi sijojen 1 – 5 ulkopuolelle jäävien osuus ei ole lainkaan pienentynyt.



Kuva 13. Parhaan osuman sijainti top5-tulosten joukossa sekä ilman osumaa alkuperäisellä sekä hienosäädetyllä e5-mallilla.

Kuva 14 näyttää erot teemakohtaisesti niissä teemoissa, jossa testikysymyksiä oli vähintään kolme kappaletta. Teemojen kehityksessä on yllättävänkin suuria eroja, vaikka kokonaisuutena kehitystä ei tapahtunut kumpaankaan suuntaan.



Kuva 14. Alkuperäisen mallin ja hienosäädetyn mallin tulokset teemakohtaisesti.

Taulukossa 5 on esitetty teemakohtaiset kysymysmäärät sekä tulosten muutokset fine-tuningin jälkeen. Huomionarvoista on, että tulokset eivät seuraa N-lukua. Eli esimerkiksi suurimmasta eläkevakuuttamisen teemasta oli eniten kysymyksiä, mutta tulokset jopa huonontuivat. Suuri vaihtelu osoittanee, että koulutusdata oli liian heterogeenistä sen määrään nähden.

Taulukko 5. Alkuperäisen ja hienosäädetyn mallin suorituskyvyn muutos teemakohtaisesti.

Teema	Kysymysten lukumäärä	Recall@1 muutos	Recall@5 muutos	MRR@5 muutos
Eläkevakuuttaminen	16	0 %	- 8 %	- 2 %
Ulkomaantyön vakuuttaminen	15	0 %	+ 9 %	+ 12 %
Tk-maksu	8	+ 200 %	+ 17 %	+ 70 %
Toimeksiannot	6	- 100 %	- 50 %	- 67 %
Eläkkeen perusteena olevat ansiot	5	- 33 %	0 %	+ 1 %
Eläkemaksut	4	0 %	+ 33 %	+ 8 %
Ilmoittaminen	4	0 %	0 %	- 6 %

8 Vastauksen generointi

Tiedonhaun mallin fine-tuningin vaikutusta generoidun vastauksen laatuun oli tavoitteena havainnollistaa esimerkein. Generointiin käytettiin microsoft/Phi-4-mini-instruct -mallia, joka on chat-tyyppinen tekstin generointimalli. Kyseessä on kevyt, avoin malli, joka on rakennettu synteettisen datan sekä suodatettujen, julkisesti saatavilla olevien verkkosivustojen pohjalta. [30.]

Tässä työssä fokus oli retriever-komponentin tarkastelussa ja raportoinnissa. Generoinnin tuloksia ei systemaattisesti tarkasteltu siten, että kokonaislaadusta voitaisiin raportoida tuloksia. Yleisenä huomiona voidaan kuitenkin todeta, että malli ei ainakaan tässä kontekstissa kykene tuottamaan tasaisesti laadukkaita vastauksia, vaikka haettu tekstikappale olisikin oikea. Huomattiin myös, että pelkästään esimerkiksi viiden parhaan tekstikappaleen yhdistäminen tietolähteeksi ei tuota hyvää tulosta. Tällöin tietolähde sisältää myös turhaa ja väärää tietoa ja malli tuottaa selkeästi virheellisempää vastausta, kuin käyttämällä pelkästään parasta, oikeaa tekstikappaletta. Näin retriever-mallin hieman päälle 40 % osumatarkkuus (recall@1) ei lähtökohtaisesti ole riittävä tuottamaan oikeaa tietoa vastausten pohjaksi.

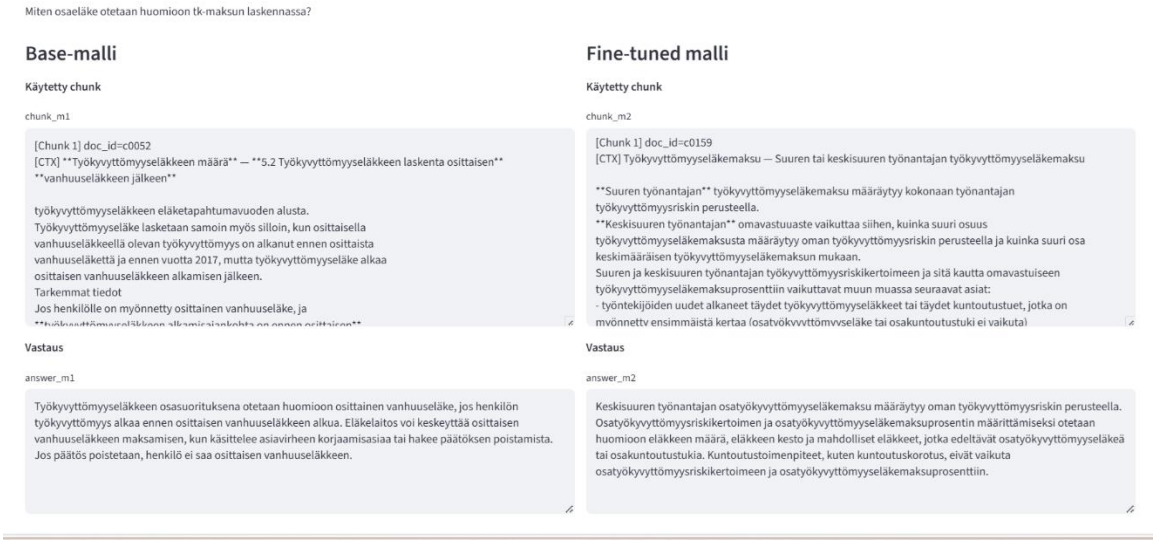
Kuvassa 15 esitetään kehotteen (prompt) muodostus generointimallia varten. Testiaineiston kysymykset ja parhaat osumat otettiin talteen retriever-mallien testausvaiheesta, josta tuotiin promptiin kontekstit.

```
def make_user_prompt(question: str, context: str) -> str:
    return f"""Tehtävä: Vastaa kysymykseen käyttäen alla olevaa KONTEKSTIA.

Kysymys:
{question}

KONTEKSTI:
{context}
"""

def build_model_input(question: str, context: str) -> str:
    user_msg = make_user_prompt(question, context)

    messages = [
        {
            "role": "system",
            "content": (
                "Olet asiantuntija. Vastaa kysymykseen tiiviisti ja selkeästi. "
                "Vastauksesi tulee perustua annettuun kontekstiin eikä ulkopuolisiin tietoihin."
            ),
        },
        {"role": "user", "content": user_msg},
    ]

    # Rakennetaan prompt tokenizerin chat-templatea käyttäen
    prompt = tokenizer.apply_chat_template(
        messages,
        tokenize=False,
        add_generation_prompt=True
    )
    return prompt
```

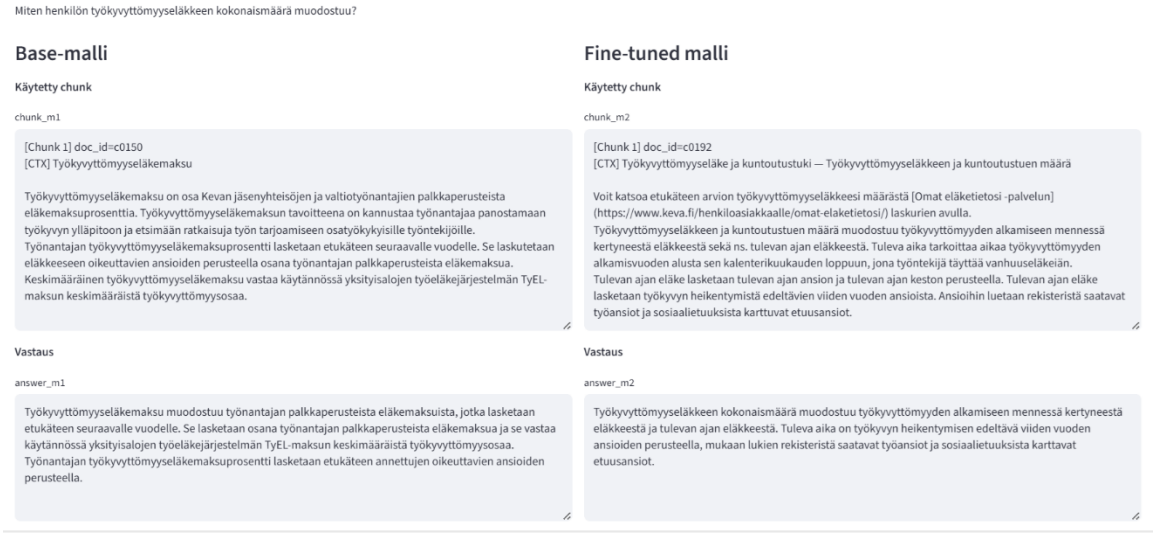
Kuva 15. Promptin muodostus.

Seuraavaksi havainnollistetaan esimerkkeinä kysymys–haettu top1 tekstikappale–vastaus -kombinaatioita, niin alkuperäisellä kuin jatkokoulutetulla retriever-mallilla.

Kuvissa 16 ja 17 on esimerkit tilanteista, jossa retriever-mallin fine-tuning on parantanut mallin kykyä löytää tekstikappale oikeaan teemaan, kun kyse on työnantajan työkyvyttömyyseläkemaksusta vs henkilön työkyvyttömyyseläkkeestä. Kuvan 16 tapauksessa generointimalli ei kuitenkaan osaa tästä (oikeasta) tekstikappaleesta koostaa oikeaa vastausta. Sen sijaan kuvassa 17 vastaus on myös parantunut.



Kuva 16. Esimerkki tilanteesta, jossa fine-tuning parantanut oikean tiedon hakua, mutta generointimalli ei tuota oikeaa vastausta.



Kuva 17. Esimerkki tilanteesta, jossa fine-tuning on parantanut tiedon hakua. Generointimalli tuottaa nyt oikeaa tietoa vastaukseksi.

9 Johtopäätökset

Työn tavoitteena oli selvittää, voidaanko retriever-komponentin toimialakohtaisella fine-tuningilla parantaa mallin kykyä löytää lähdeaineistosta kysymykseen parhaiten vastaavat tekstiosumat. Työssä tarkasteltiin useita keinoja retriever-komponentin suorituskyvyn parantamiseksi ja testattiin valitun mallin fine-tuningia eri hyperparametri-asetuksilla. Tulokset osoittavat, että fine-tuning parantaa osumatarkkuutta useilla keskeisillä mittareilla. Näin ollen työn keskeinen tavoite saavutettiin ja työtä voidaan pitää onnistuneena.

Huomionarvoista on, että vaikka käytetty koulutusaineisto oli kooltaan kohtuullisen pieni, se riitti parantamaan mallin suorituskykyä. Tämä viittaa siihen, että toimialakohtainen ja huolellisesti laadittu aineisto voi olla tehokasta retriever-mallin jatkokoulutuksessa, vaikka aineiston määrä ei olisi suuri. Samalla havaittiin, että koulutusaineiston teemajakauma erityisesti tiiviimmässä koulutusaineistossa vaikutti tuloksiin: teemat, joihin oli saatavilla enemmän koulutusdataa, hyötyivät fine-tuningista selkeämmin, kun taas vähäisemmällä aineistolla edustetut teemat eivät parantuneet yhtä johdonmukaisesti ja joissakin tapauksissa suorituskyky jopa heikkeni. Tämä korostaa tasapainoisen ja edustavan koulutusaineiston merkitystä.

Datan laatu ja kattavuus osoittautuivat kokonaisuudessaan keskeisiksi tekijöiksi. Laajemmalla lähdeaineistolla tulokset eivät olleet yhtä hyviä kuin rajatummassa aineistossa. Tämä viittaa siihen, että suurempi dokumenttikokoelma asettaa retriever-mallille korkeampia vaatimuksia. Tällaisessa asetelmassa hyvien tulosten saavuttaminen edellyttäisi todennäköisesti huomattavasti enemmän ja monipuolisempia kysymyksiä eri teemoista, jotta malli kykenee oppimaan relevantteja erotteluja.

Fine-tuning -prosessissa loss-funktiolla on merkittävä vaikutus. Tässä työssä lähtökohdaksi valittiin Multiple Negative Ranking Loss -funktio. Jatkossa olisi kuitenkin perusteltua tutkia Triplet Loss -funktiota tarkemmin. Mikäli aineistossa pystyttäisiin huomioimaan kattavasti negatiiviset (hard negatives) esimerkit, voisi Triplet Loss mahdollisesti tuottaa tällaisessa kohtuullisen pienessä aineistossa hyvää tulosta, koska jo nyt lähtötulokset ilman tarkempien parametrien valintaa olivat kohtuullisia.

Tässä työssä keskityttiin hakuvaiheen parantamiseen. Luontevana jatkokehityssuuntana olisi tarkastella koko RAG-putken optimointia, mukaan lukien generatiivisen komponentin vaikutus lo-

pulliseen vastausten laatuun. Tässä työssä testattu generointimalli ei esimerkiksi tuottanut riittävän laadukkaita vastauksia, joten generointimallien tarkempi selvitys ja parannusmahdollisuudet olisivat paikallaan. On myös syytä huomioida, että tässä työssä testattiin malleja suhteellisen selkeästi muotoilluilla ja lyhyillä kysymyksenasetteluilla. Todellisuudessa asiakkaiden kysymykset voivat olla eritasoisia. Tämä kannattaisi jatkossa ottaa myös huomioon koulutusaineiston laadinnassa.

Lähteet

1. Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. [internet]. 2020. Saatavilla: https://doi.org/10.48550/arXiv.2005.11401

2. Traditional Search vs Semantic Search: A Deep Dive into Smarter Information Retrieval. [internet]. [Viitattu 6.1.2026]. 2025. Saatavilla: https://blog.expertrec.com/traditional-search-vs-semantic-search-a-deep-dive-into-smarter-information-retrieval/

3. Merritt R. What Is a Transformer Model? [Internet]. [Viitattu 14.10.2025]. Saatavilla: https://blogs.nvidia.com/blog/what-is-a-transformer-model/

4. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, Kaiser L, Polusukhin I. Attention is all you need. [internet]. 2017. Saatavilla: https://doi.org/10.48550/arXiv.1706.03762

5. Parthasarathy VB, Zafar A, Khan A, Shahid A. The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs:An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities. CeADAR; 2024. Saatavilla: https://doi.org/10.48550/arXiv.2408.13296

6. Mukund SA, Easwarakumar KS. Optimizing legal text summarization through dynamic retrieval-augmented generation and domain-specific adaptation. Symmetry [Internet]. 2025;17,633. Saatavilla: https://doi.org/10.3390/sym17050633

7. Mandikal P, Mooney R. Sparse Meets Dense: A Hybrid Approach to Enhance Scientific Docu-ment Retrieval. [internet]. 2024. Saatavilla: https://doi.org/10.48550/arXiv.2401.04055

8. Schwaber-Cohen R, Patel A. Chunking Strategies for LLM Applications [internet]. [Viitattu 18.10.2025]. Saatavilla: https://www.pinecone.io/learn/chunking-strategies

9. Pinecone. Rerankers and Two-Stage Retrieval [Internet]. [Viitattu 2.12.2025]. Saatavilla: https://www.pinecone.io/learn/series/rag/rerankers/

10. Yao Z, Wang S, Zuccon G. Pre-training vs. Fine-tuning: A Reproducibility Study on Dense Retrieval Knowledge Acquisition. [internet]. 2025. Saatavilla: https://doi.org/10.48550/arXiv.2505.07166

11. Sultania D, Lu Z, Naik T, Dernoncourt F, Yoon S, Sharma S, Bui T, Gupta A, Vatsa T, Suresha S, Verma I, Belavadi V, Chen C, Friedrich M. Domain-specific Question Answering with Hybrid Search. [internet]. 2024. Saatavilla: https://doi.org/10.48550/arXiv.2412.03736

12. Karpukhin V, Oguz B, Min S, Lewis P, Wu L, Edunov S, Chen D, Yih W. Dense Passage Retrieval for Open-Domain Question Answering. [internet]. 2020. Saatavilla: https://doi.org/10.48550/arXiv.2004.04906

13. Ip J. LLM Evaluation Metrics: The Ultimate LLM Evaluation Guide [Internet]. 10.10.2025 [viitattu 14.10.2025]. Saatavilla: https://www.confident-ai.com/blog/llm-evaluation-metrics-everything-you-need-for-llm-evaluation

14. RAG evaluation: a technical guide to measuring retrieval-augmented generation [Internet]. 15.8.2025 [viitattu 7.11.2025]. Saatavilla https://toloka.ai/blog/rag-evaluation-a-technical-guide-to-measuring-retrieval-augmented-generation/

15. Retrieval Metrics Tutorial: Recall@k and MRR Explained [Internet]. 31.10.2025 [viitattu 30.12.2025]. Saatavilla https://medium.com/@rajnish_khatri/retrieval-metrics-tutorial-recall-k-and-mrr-explained-d2f12afb9c89

16. PyPI Docs. trafilatura. [internet]. [Viitattu 6.1.2026]. Saatavilla: https://pypi.org/project/trafilatura/

17. PyPI Docs. pymupdf4llm. [internet]. [Viitattu 6.1.2026]. Saatavilla: https://pypi.org/project/pymupdf4llm

18. Faiss Documentation. [internet]. [Viitattu 6.1.2026]. Saatavilla: https://faiss.ai/

19. Rajapakse T, Yates A, Rijke M. Negative Sampling Techniques for Dense Passage Retrieval in a Multilingual Setting. [internet]. [Viitattu 6.1.2026]. Saatavilla: https://staff.fnwi.uva.nl/m.derijke/wp-content/papercite-data/pdf/rajapakse-2024-study.pdf

20. Wang L, Yang N, Huang X, Yang L, Majumder R, Wei F. Multilingual E5 Text Embeddings: A Technical Report. [internet]. 2024. Saatavilla: https://doi.org/10.48550/arXiv.2402.05672

21. sbert Docs. [internet]. [Viitattu 6.1.2026]. Saatavilla: https://sbert.net/docs/package_reference/sentence_transformer/evaluation.html#informationretrievalevaluator

22. Sentence Transformers Training Overview. [internet]. [Viitattu 6.1.2026]. Saatavilla: https://sbert.net/docs/sentence_transformer/training_overview.html

23. Mitigating False Negatives in Multiple Negatives Ranking Loss for Retriever Training. 25.5.2025. [internet]. [Viitattu 6.1.2026]. Saatavilla: https://huggingface.co/blog/dragonkue/mitigating-false-negatives-in-retriever-training

24. Sbert Docs. Losses. [internet]. [Viitattu 6.1.2026]. Saatavilla: https://www.sbert.net/docs/package_reference/sentence_transformer/losses.html#tripletloss

25. Sbert Docs. Losses. [internet]. [Viitattu 6.1.2026]. Saatavilla: https://sbert.net/docs/package_reference/sentence_transformer/losses.html#contrastcontras

26. Epoch in Machine Learning | Understanding the Core of Model Training. [internet]. [Viitattu 25.1.2026]. Saatavilla: https://medium.com/@saiwadotai/epoch-in-machine-learning-understanding-the-core-of-model-training-bfd64bbd5604

27. Belcic I, Stryker C. What is learning rate in machine learning? [internet]. [Viitattu 25.1.2026]. Saatavilla https://www.ibm.com/think/topics/learning-rate

28. Sbert Docs. Training Arguments. [internet]. [Viitattu 6.1.2026]. https://sbert.net/docs/package_reference/sentence_transformer/training_args.html#sentence_transformers.training_args.SentenceTransformerTrainingArguments

29. Lo A. Weight Decay and Its Peculiar Effects. 2021. [internet]. [Viitattu 6.2.2026]. Saatavilla: https://towardsdatascience.com/weight-decay-and-its-peculiar-effects-66e0aee3e7b8/

30. Huggingface Models. [internet]. [Viitattu 6.1.2026]. Saatavilla: https://hugging-face.co/microsoft/Phi-4-mini-instruct

RAG-järjestelmän lähdeaineistot

- Verkkosivut (url-haku):

https://www.keva.fi/tyonantajalle/elakemaksut/ (alasivuineen),

https://www.keva.fi/henkiloasiakkaalle/tietoa-elakkeista/elakevaihtoehdot/tyokyvytto-myyselake-ja-kuntoutustuki/,

https://www.keva.fi/henkiloasiakkaalle/tietoa-elakkeista/elakevaihtoehdot/osatyokyvytto-myyselake-ja-osakuntoutustuki/,

https://www.keva.fi/henkiloasiakkaalle/tietoa-elakkeista/elakevaihtoehdot/tyouraelake/,

https://www.keva.fi/henkiloasiakkaalle/elakkeensaajalle/tyonteko-elakkeella/,

https://www.telp.fi/ohjeet/etuudet/tyokyvyttomyyselake/tyokyvyttomyyselakkeen-maara/,

- verkkosivuilla olevat pdf-muotoiset ilmoittamisen ohjedokumentit

https://www.keva.fi/globalassets/2-tiedostot/ta-tiedostot/tyonantajien-ohjeet/kevan-ohjeet-2025---ilmoittaminen.pdf,

https://www.keva.fi/globalassets/2-tiedostot/ta-tiedostot/tyonantajien-ohjeet/mitka-ansiot-kuuluvat-juel-vakuutukseen.pdf,

https://www.keva.fi/globalassets/2-tiedostot/ta-tiedostot/tyonantajien-ohjeet/kevan-koodisto-2025.pdf

- JuEL:n (julkisten alojen eläkelaki) mukaisen vakuuttamisen sisäiset Case-esimerkit pdf-muodossa, painottuen erilaisiin eläkevakuuttamisen tulkintatilanteisiin

+ Laajempi aineisto

https://www.keva.fi/henkiloasiakkaalle/tietoa-elakkeista/elakevaihtoehdot/vanhuuselake/,

https://www.keva.fi/henkiloasiakkaalle/tietoa-elakkeista/elakevaihtoehdot/osittainen-varhennettu-vanhuuselake/,

https://www.keva.fi/henkiloasiakkaalle/tietoa-elakkeista/elakevaihtoehdot/perhe-elake/,

https://www.keva.fi/henkiloasiakkaalle/tietoa-elakkeista/elakevaihtoehdot/taloudellinen-tuki/,

https://www.keva.fi/henkiloasiakkaalle/tietoa-elakkeista/elaketta-palkattomilta-ajoilta/,

https://www.keva.fi/henkiloasiakkaalle/tietoa-elakkeista/elakkeen-maara/,

https://www.keva.fi/henkiloasiakkaalle/tietoa-elakkeista/ammattiryhmien-elake/omais--ja-perhehoitajat/,

https://www.keva.fi/henkiloasiakkaalle/elakkeensaajalle/elakkeen-maksupaivat/,

https://www.keva.fi/henkiloasiakkaalle/elakkeensaajalle/elakkeensaajan-kuoltua/,

https://www.telp.fi/ohjeet/vakuuttaminen/elakkkeen_-perusteena_-olevat_-tyoansiot/elakkeen-perusteena-olevat-tyoansiot/,

https://www.telp.fi/ohjeet/etuudet/tyokyvyttomyyselake/tyokyvyttomyyselakkeen-yleiskuvaus/,

https://www.telp.fi/ohjeet/kansainvaliset-asiat/,

https://www.telp.fi/ohjeet/kansainvaliset-asiat/yleisohjeita-kansainvalisiin-tyoskentelytilanteisiin/a1-ja-muut-todistukset-sovellettavasta-sosiaaliturvalainsaadannosta/,

https://www.telp.fi/ohjeet/kansainvaliset-asiat/suomesta-ulkomaille-vakuuttaminen/eu-maassa-tyoskentelyn-vakuuttaminen/