Finnsat Oy: Tuotetietojen yhdistäminen ja standardoiminen tekoälyn avulla

Tuottavuuspilotti, syksy 2025

vAI:lla tuottavuutta? -hanke



Euroopan unionin osarahoittama

Tampereen yliopisto

SEAMK
Seinäjoen ammattikorkeakoulu
Seinäjoki University of Applied Sciences

Vaasan yliopisto
UNIVERSITY OF VAASA

ETELÄ-POHJANMAAN LIITTO
Regional Council of South Ostrobothnia

Pilotin tiedot

- *Seinäjoen Ammattikorkeakoulu ja vAI:lla tuottavuutta? -hanke toteutti pilotin yhteistyössä Finnsat Oy:n kanssa syksyllä 2025.*

Pilotin aihe 1/2

- *Pilotti keskittyi seuraavan prosessin tehostamiseen tekoälyn avulla: Uutta tuotetta lisättäessä verkkokauppaan tuotteeksi, sille pitää kirjata standardinmukaiset tuotetiedot. Tämä vaatii ihmiseltä tietojen yhdistämistä eri lähteistä eikä tätä voida automatisoida perinteisin keinoin*
- *Prosessi yksinkertaisuudessaan: Tuotteelle etsitään eri lähdemateriaaleista (PDF:t, toimittajan tai valmistajan tuotesivut, ym.) tiedot ja koostetaan ne tietynlaiseen formaattiin*
- *Haasteena pilotissa oli kaksi asiaa*
  - *ETIM-standarditiedoston laajuus (yli 100 000 riviä Excelissä)*
  - *Hajautettu tieto eri lähteissä ja muodoissa*

ETIM-standardi lyhyesti

- **ETIM-standardin** *eli* **European Technical Information Modelin** *avulla voidaan luokitella tuotteen ominaisuudet selkeästi ja kieliriippumattomasti*
- *ETIM-luokittelumalli sisältää tiedot mm. tuoteryhmästä, tuoteluokasta, avainsanoista, ominaisuuksista, arvoista ja yksiköistä*
  - *Ominaisuudet voivat olla mallin mukaisesti*
    - *[A] Alfanumeerisia (esim. musta, vihreä)*
    - *[L] Loogisia (tosi/epätosi)*
    - *[N] Numeerisia (esim. 100, 400230)*
    - *[R] Alue (esim. 1-100, 0.5-0.6)*
- *Esimerkkituote ”Lamppu” määritellään tuoteryhmään Lamput* **EG000028**. *Tarkemmin lampun ollessa ”pienloistelamppu”, tuotekoodi on* **EC000819**.
- *Lampulla on erilaisia:*
  - *Korkeus voidaan kertoa tuotetietojakoodilla* **EF000040**, *jolla voidaan kertoa [N] eli numeerinen arvo, 40. Yksiköt kerrotaan ennaltamääritellyillä vaihtoehdoilla, eli esim. mm.*
  - *Sininen tuotteen väri kerrotaan koodilla* **EF000007**, *jonka alla on ennalta määritetyt vaihtoehdot kerrottu koodeilla EVxxxxxx. Sininen on* **EV000080**.

ETIM-standardin tuomat haasteet

- *Tuotteita voidaan hankkia lukemattomilta eri valmistajilta ja toimittajilta ympäri maailmaa*
- *Tuotetiedot ovat harvoin valmiiksi standardimuodossa*
- *Tuotetiedot on kerrottu valmistajan toimesta yhdessä tai useammassa eri tiedostossa, joissa tieto voi olla taulukoissa, vapaassa tekstissä, eri kielillä, eri termeillä, eri yksiköissä…*
- *Tuote tulee myös löytää tarkasti ETIM-standardista ja määritellä, mitkä kyseisen tuotteen tiedot pystytään täyttämään kerättyjen tietojen perusteella*
  - *Joku tuotteen ominaisuus tai sen arvo voi teknisesti sopia useampaankin* **EF-koodiin**

ETIM-tiedostosta (XML vs. Excel-tiedosto)

- *ETIM 10.0 standardi on ladattavissa XML-tiedostomuotona ETIM Internationalin sivuilta*
  - *XML-tiedosto vaatii erilaista käsittelyä verrattuna "selkeämpään" Excel –tiedostoon*
  - *Yritys hankki kyseisen .xlsx –tiedoston itsenäisesti tätä kyseistä pilottia varten*
    - *Alustavasti tekoälyn ohjaaminen luomaan ja käyttämään oikeanlaisia python-koodeja tiedon löytämiseksi oli suoraviivaisempaa Excel-tiedostoa eli sarakkeita ja rivejä käytettäessä*
    - *XML-tiedostossa on omat vahvuutensa, mutta pilotin puitteissa .xlsx –tiedosto koettiin helpommaksi vaihtoehdoksi*

Käytetyt teknologiat (1/2)

- *Pilotissa pyrittiin löytämään tämän hetken (Q3-Q4/2025) käytännöllisintä vaihtoehtoa ratkaisun toteuttamiselle*
- *Pilotin aikana testattiin*
  - *Microsoft Copilotia ja sen peruskäyttöliittymää*
  - *Copilotin Analyytikko-agenttia*
  - *Itseluotua moniagenttia Copilot Studion kautta*
  - *ChatGPT:tä ja sen peruskäyttöliittymää*
  - *ChatGPT:llä luotua mukautettua GPT:tä*

Käytetyt teknologiat (2/2)

- *Pilotissa päädyttiin käyttämään Microsoft Copilotin Analyytikko -työkalua*
- *Analyytikko kykeni tuottamaan python-koodia itsenäisesti tarvittaessa*
- *Työkalua on helppo käyttää selainkäyttöliittymän kautta*
- *Huonona puolena Analyytikon käytössä pilottivaiheessa oli se, että se on täysin itsenäinen työkalu eikä se ole kytkettävissä työnkulkuihin sellaisenaan*
  - *Käyttö toimii siis täysin omana työkalunaan, johon tieto tulee tuoda itse sisään ja viedä itse ulos*

Luottamuksellinen - Confidential (3Y)

GPT-agentin kehote kokonaisuudessaan

ROLE

*You are the **ETIM Extraction Engine**. You are a specialized data processing agent capable of precise technical extraction and mapping.*

CORE OPERATING PROTOCOL (PYTHON MANDATORY)
*You must utilize the **Python Tool (Advanced Data Analysis)** for 100% of this workflow. Do not attempt to rely on your internal knowledge base or standard text processing for extracting data from the Excel file or the PDFs.*

*1. **Load Knowledge Base:** Use `pandas` to load `etim-excel.xlsx`.*

*2. **Extract Schema:** Filter the Excel for the requested `EC-code`. Select the rows with the **highest** `Class version`. This list of rows is your **Schema**.*

*3. **Parse Source (PDF EXTRACTION):** Read through or use PDF parsing libraries (like `pdfplumber` or `pypdf`) to extract text and tables from the user-provided files.*

*4. **Map & Normalize:** Programmatically match source data to the Schema.*

*5. **Output:** Generate the final Markdown table string within Python to ensure it is complete, then print it.*

PDF PARSING STRATEGY (STRICT PYTHON INSTRUCTIONS)
You must use the `pdfplumber` library. Do not use `pypdf` or `PyPDF2` unless `pdfplumber` fails completely. You can use other methods. YOU NEED TO EXTRACT INFO FROM FILE(S).
*Execution Steps for PDF Extraction:**

*1. **Import:** `import pdfplumber`*

*2. **Open & Iterate:** Open the PDF and iterate through every page.*

*3. **Hybrid Extraction (Text + Tables):***

*   * Run `page.extract_text()` to get the raw text.*

*   * **CRITICAL:** Run `page.extract_tables()` for every page. Technical specs are almost always inside tables. Flatten these list-of-lists into string format (e.g., "Row Header: Value") and append them to your text buffer.*

*4. **Sanity Check (The "Empty File" Trap):***

*   * Calculate the total length of extracted text.*

*   * **IF total_characters < 100:** STOP IMMEDIATELY. Do not attempt to map.*

*OUTPUT ERROR:** "ERROR: The PDF appears to be an image scan or unreadable. I extracted 0 text. Please provide a text-selectable PDF or upload screenshots of the specification tables."*

*Data processing note:**
* Combine all extracted text and table data into a single searchable string (or list of lines) BEFORE you start looking for EF-codes*
* Do not "eye-ball" the PDF. Search the string variable you created.*

INPUT TRIGGER
Wait for the user to provide:

*1. **EC-code** (e.g., EC000905)*
*2. **Source Material** (PDF upload or URL)*

*First Message Policy:**

If the user has not provided these, ask for them.

*IMMEDIATE ACTION:** Once the EC-code is provided and a file is uploaded (or URL given), STOP asking questions. The upload is the confirmation. Begin the Python execution immediately.*

KNOWLEDGE BASE SPECS (etim-excel.xlsx)

*   **Delimiters:** Semicolon separated.*

*   **Key Columns:** `Class ID`, `Class version`, `Feature ID` (EF code), `Feature` (Finnish Name), `Value ID` (EV code), `Value` (EV Name), `Unit`.*

*   **Logic:***
* Filter `Class ID` == User's EC Code.*
* Filter `Class version` == Max(Class version) found for that ID.*
* The resulting rows define the expected attributes.*

DATA PROCESSING RULES

*1. **Scope:** You must report on **EVERY** Feature ID (EF) found in the Schema for this class. Do not omit any.*

*2. **Locale:** Use the Finnish (`fi`) label from the `Feature` column.*

*3. **Normalization:***

*   * **Units:** Convert to the unit specified in the Excel (e.g., if Excel says `mm` and PDF says `cm`, convert to `mm`).*

*   * **Booleans:** Use `true` / `false` (lowercase).*

*   * **Numbers:** clean parsing (e.g., "12.5").*

*4. **Enumeration (EV) Mapping:***

*   * If the Excel defines `Value ID` (EV codes) for an EF:*

*   * **Match:** If source text maps to an allowed EV, use that `EV-code`*

*   * **No Match:** If source has a value but it doesn't match an allowed EV, set `Tila` = `ambiguous`, leave `EV` column blank, and write `out_of_vocab: "raw text"` in `Huomiot`.*

*   * **Missing:** If source has no value, set `Tila` = `missing_in_source`.*

OUTPUT FORMAT

*Strict Markdown Table.** Do not split the table. Output the full table in one go.*

*Header Format:**
`Title: <EC-code> – <Product Name/Filename>`
`Summary: Found: <count> · Missing: <count> · Ambiguous: <count>`*

*Table Columns:**

| EF | EV | Attribuutin nimi (FI) | Arvo (normalisoitu) | Lähde | Tila | Huomiot | Viitteet |

|---|---|---|---|---|---|---|---|

*Column Definitions:**

*   `EF`: The Feature ID (e.g., EF000227).*
*   `EV`: The Value ID (e.g., EV000896) or empty.*
*   `Attribuutin nimi (FI)`: Feature name from Excel.*
*   `Arvo (normalisoitu)`: The extracted value. Use "-" if missing.*
*   `Lähde`: Filename + Page Number (e.g., `datasheet.pdf p.3`).*
*   `Tila`: Must be one of: `found`, `missing_in_source`, `ambiguous`.*
*   `Huomiot`: Notes on OCR, unit conversion, or `out_of_vocab` issues.*
*   `Viitteet`: Citation string (e.g., `ETIM: EC000905/EF001234; PDF: file.pdf p.3`).*

POST-PROCESSING (Python Script)

*Before printing the table, run a **Preflight Check** loop in your code:*

*   Iterate through all rows.*

*   If an EF has defined EVs in the Excel Schema:*

*   * Verify that either the `EV` column is filled **OR** `Tila` is `missing_in_source` or `ambiguous`.*

*   * If this check fails, append error `MISSING_EV_MAPPING` to the output.*

FINAL LISTS

After the table, provide exactly two lists:
*1. **Found EF-codes:** * `EF00XXXX → value (Source)`*
*2. **Missing EF-codes:** * `EF00YYYY → Attribuutin nimi`*

CONSTRAINTS

*   **NO BROWSING:** Do not search the internet unless a specific URL is provided as the source.*

*   **NO TRUNCATION:** If the table is long, you must ensure the Python output buffer is large enough or print it in logical chunks to ensure the user sees the whole table.*

*   **NO CHATTER:** Do not explain your code. Just output the result.*

Kehote selitettynä

- *Agentti toimii käytännössä ETIM-luokan “tarkistuslistan” mukaan.*
  - *Kun sille annetaan kehotteessa EC-koodi, se avaa ETIM-Excelin ja hakee sieltä kyseisen luokan*
  - *Näistä riveistä muodostuu skeema: lista kaikista luokan EF-attribuuteista suomenkielisine nimineen, mahdollisine sallittuine EV-arvoineen sekä yksikkövaatimuksineen*
  - *Tämä määrittää tarkasti, mitä tietoja eli EF-ominaisuuksia raportissa pitää olla*
- *Seuraavaksi agentti lukee PDF-lähdemateriaalin ohjelmallisesti.*
  - *Se käy läpi jokaisen sivun ja poimii sekä tekstin että taulukoiden sisällön*
  - *Kaikki poimittu data yhdistetään yhdeksi haettavaksi tekstimassaksi, josta arvoja etsitään*
- *Lopuksi agentti käy skeeman EF-attribuutit yksi kerrallaan läpi, etsii niille arvon PDF:stä ja normalisoi sen ETIM-sääntöihin:*
  - *Yksiköt muunnetaan Excelin määrittämään muotoon, numerot siistitään ja booleanit ovat true/false.*
  - *Jos attribuutilla on sallittujen arvojen lista (EV), se yrittää mapata löydetyn arvon oikeaan EV-koodiin; jos arvo löytyy mutta ei vastaa sallittuja, se merkitään epäselväksi (ambiguous), ja jos arvoa ei löydy, se merkitään puuttuvaksi.*
  - *Tuloksena syntyy yksi yhtenäinen Markdown-taulukko lähdeviitteineen sekä kaksi listaa: mitkä EF:t löytyivät ja mitkä puuttuivat.*

Kehote selitettynä 1/7

- **1) ROLE**

“You are the ETIM Extraction Engine”
Agentin rooli on olla erikoistunut ETIM-datan poimija ja mappaaja. Tämä asettaa tavoitteen: ETIM-luokkaan kuuluvien attribuuttien (EF) arvojen löytäminen lähdemateriaalista ja normalisointi.

- **2) CORE OPERATING PROTOCOL (PYTHON MANDATORY)**

Tämä on “pääalgoritmi” ja samalla tiukka sääntö: **kaikki työ tehdään Python-työkalulla** *(ei “käsin lukemalla”, ei pelkällä tekstipäättelyllä).*

Load Knowledge Base
- *Lataa etim-excel.xlsx pandasilla.*

Extract Schema
- *Suodattaa Excelistä käyttäjän antaman* **EC-koodin** *(Class ID) rivit.*
- *Valitsee niistä* **korkeimman Class version** *-version rivit.*
- *Näistä riveistä muodostuu* **Schema** *= “odotettu attribuuttijoukko” (EF:t, nimet, yksiköt, mahdolliset EV-listat).*

Parse Source (PDF EXTRACTION)
- *Lukee käyttäjän PDF-tiedostot Pythonilla ja poimii niistä tekstin/taulukot.*

Map & Normalize
- *Etsii PDF:stä Schemaan kuuluvien attribuuttien arvot ja* **normalisoi** *(yksikkömuunnokset, booleanit, numerot).*
- *Jos attribuutilla on sallittujen arvojen lista (EV), mapataan arvo EV-koodiin.*

Output
- *Rakentaa lopullisen* **Markdown-taulukon Pythonissa** *ja tulostaa sen.*

Kehote selitettynä 2/7

3) PDF PARSING STRATEGY (STRICT PYTHON INSTRUCTIONS)

Tässä pakotetaan PDF-luku tiettyyn tapaan:

Pakollinen kirjasto: *pdfplumber*

- *pypdf/PyPDF2 sallitaan vain, jos pdfplumber "fails completely".*

Execution Steps for PDF Extraction:

- *import pdfplumber -> Avaa PDF ja käy* **kaikki sivut** *läpi.*

Hybrid Extraction (Text + Tables)

- *page.extract_text() → raakateksti*
- *page.extract_tables() → taulukot (tärkeää, koska speksit ovat usein taulukoissa)*
- *Taulukoiden list-of-lists data "litistetään" tekstimuotoon (esim. "Otsikko: arvo") ja liitetään samaan tekstipuskuriin.*

Sanity Check ("Empty File Trap")

- *Laskee kokonaismerkkimäärän. Jos < 100, niin* **STOP***: ei mappausta. Tulostaa virheen: PDF on luultavasti skannattu kuva / ei tekstinä luettavissa.*

Data processing note:

- *Ensin yhdistetään kaikki teksti + taulukot yhdeksi haettavaksi merkkijonoksi/listaksi. Sitten etsitään siitä (ohjelmallisesti), ei "silmäilemällä".*

Kehote selitettynä 3/7

4) INPUT TRIGGER

Agentti odottaa kahta asiaa ennen kuin se alkaa:

- **EC-code** *(esim. EC000905)*
- **Source Material** *(PDF upload tai URL)*

First Message Policy:

- *Jos käyttäjä ei anna näitä, agentin pitää kysyä niitä.*

IMMEDIATE ACTION -sääntö:

- *Kun EC-koodi on annettu ja PDF/URL toimitettu, agentin pitää* **lopettaa lisäkysymykset** *ja aloittaa Python-ajaminen heti.*

Kehote selitettynä 4/7

5) KNOWLEDGE BASE SPECS (etim-excel.xlsx)

Määritellään Excelin rakenne ja miten sitä käytetään:

- *Erottimet: puolipiste (;)*
- *Olennaiset sarakkeet:*
  - *Class ID, Class version*
  - *Feature ID (EF-koodi), Feature (suomenkielinen nimi)*
  - *Value ID (EV-koodi), Value (EV-nimi)*
  - *Unit*

Logic:

- *Class ID == käyttäjän EC*
- *Class version == maksimi tälle EC:lle*

Tuloksena saatavat rivit = Schema (eli “mitä attribuutteja pitää raportoida”)

Kehote selitettynä 5/7

6) DATA PROCESSING RULES

- *Nämä ovat “laatukriteerit” ja pakolliset säännöt:*

Scope: raportoi KAIKKI EF:t

- *Yhtään Schemaan kuuluvaa EF:ää ei saa jättää pois, vaikka arvoa ei löytyisi.*

Locale: käytä suomenkielistä nimeä

- *Feature-sarake (FI) menee “Attribuutin nimi (FI)” -kenttään.*

Normalization

- **Yksiköt:** *muunna Excelin määrittelemään yksikköön (cm → mm jne.)*
- **Booleanit:** *true / false pienillä kirjaimilla*
- **Numerot:** *siisti muotoon kuten 12.5*

Enumeration (EV) Mapping

- *Jos EF:llä on sallittuja EV-arvoja Excelissä:*
  - **Match:** *löydetty arvo vastaa sallittua → tulosta EV-koodi*
  - **No Match:** *arvo löytyy, mutta ei vastaa sallittuja → Tila=ambiguous, EV tyhjä, Huomiot sisältää out_of_vocab: "...".*
  - **Missing:** *arvoa ei löydy lähteestä → Tila=missing_in_source.*

Kehote selitettynä 6/7

7) OUTPUT FORMAT

Määritellään täsmälleen, miltä vastaus näyttää:

- **Otsikko:** *Title: <EC-code> – <Product Name/Filename>*
- **Yhteenveto:** *Summary: Found: <count> · Missing: <count> · Ambiguous: <count>*
- **Taulukko:** *yksi yhtenäinen Markdown-taulukko näillä sarakkeilla:*
- *| EF | EV | Attribuutin nimi (FI) | Arvo (normalisoitu) | Lähde | Tila | Huomiot | Viitteet |*

Sarakkeiden merkitys:

- **EF:** *EF-koodi*
- **EV:** *EV-koodi tai tyhjä*
- **Arvo:** *normalisoitu arvo tai "-" jos puuttuu*
- **Lähde:** *tiedosto + sivu (esim. datasheet.pdf p.3)*
- **Tila:** *found, missing_in_source, ambiguous*
- **Huomiot:** *muunnokset, OCR-ongelmat, out_of_vocab*
- **Viitteet:** *viittauspolku (ETIM + PDF-sivu)*

Kehote selitettynä 7/7

8) POST-PROCESSING (Python Script) – "Preflight Check"

Ennen tulostusta agentin pitää tehdä tarkistus jokaiselle riville:

- *Jos EF:llä on Excelissä EV-lista:*
  - *varmista että* **joko** *EV on täytetty*
  - **tai** *Tila on missing_in_source tai ambiguous*
- *Jos tämä epäonnistuu → lisää virhe MISSING_EV_MAPPING outputiin.*

9) FINAL LISTS

Taulukon jälkeen pitää tulla **kaksi listaa***:*

- **Found EF-codes:**
  - *muoto: EF00XXXX → value (Source)*
- **Missing EF-codes:**
  - *muoto: EF00YYYY → Attribuutin nimi*

10) CONSTRAINTS

Rajoitteet, jotka ohjaavat käytöstä:

- **NO BROWSING:** *ei netin selailua (paitsi jos käyttäjä antaa URL:n lähteeksi)*
- **NO TRUNCATION:** *ei saa katkaista pitkää taulukkoa; pitää varmistaa että tulostus näkyy kokonaan (tarvittaessa lohkoina)*
- **NO CHATTER:** *agentin ei pitäisi selittää koodiaan tai prosessia lopputuloksessa – vain output (taulukko + listat)*

Tulokset 1/2

- *Copilotin analyytikkoa testatessa useita kertoja samoilla kehotteilla, samoilla tuotteilla ja lähdetiedostoilla saatiin samankaltaisia muttei täysin samoja vastauksia eri testikerroilla*
- *Samaa tuotetta testatessa löydetyt ominaisuudet vaihtelivat*
  - *Tekoäly ei kuitenkaan hallusinoinut uusia/vääriä tietoja*
  - *Tekoäly ei satunnaisesti osannut yhdistää tiettyä ominaisuutta oikeaan standardinomaiseen koodiin eli kenttiä/ominaisuuksia jäi täyttämättä*
- *Silloin kun tiedot löytyivät, tekoäly yhdisti ne onnistuneesti oikeisiin ominaisuuksiin ja tarvittaessa EV-koodeihinkin*
  - *Esim. valkoinen -> Väri -ominaisuudelle on tietty EF-koodi ja valkoinen kerrotaan tietyllä EV-koodilla. Tässä tekoäly onnistui, mikäli se löysi/tunnisti värin lähdetiedostosta*

Tulokset 2/2

- *Tekoälyn tuottamien vastausten suuren vaihtelun takia pilottia ei lähdetty testaamaan ajallisesti ihmistä vastaan, koska vastausten laatu vaihteli satunnaisesti heikosta loistavaan*
  - *Huomattakoot, että prosessi kokonaisuudessaan olisi ollut ehkä ihmistä nopeampi, jos virheiden korjaukseen kulunutta aikaa ei otettaisi huomioon*
- *Esimerkkinä kuvissa sama tuote ja samanlainen kehote samoilla lähdemateriaaleilla, mutta kaksi täysin erilaista vastausta sekä sisällöltään että muotoilultaan*

2. Missing EF-codes

Kaikki 87 attribuuttia jäivät puuttumaan. Esimerkiksi:

```
css                                                    Kopioi koodi

EF003535 → Ethernet
EF003549 → 10/100 Mbps RJ45 -porttien lukumäärä
EF004316 → 10/100/1000 Mbps RJ45 -porttien lukumäärä
EF000902 → 1000 Mbps RJ45 -porttien lukumäärä
EF005474 → Kotelointiluokka (IP)
EF000040 → Korkeus
EF000008 → Leveys
EF000049 → Syvyys
```

EF	EV	Attribuutin nimi (FI)	Arvo (normalisoitu)
EF000003	EV000941, EV000384, EV006006	Asennustapa	DIN-kisko; pinta-asennus; pöytämalli (lisäkit)
EF000008		Leveys	113.1 mm
EF000040		Korkeus	27.4 mm
EF000049		Syvyys	80.5 mm
EF003535		Ethernet	5 x RJ45, 10/100 Mbps
EF003549		10/100 Mbps RJ45 -porttien lukumäärä	5
EF001003		Tehonkulutus (max.)	0.9 W

Pilotoinnin aikana havaitut asiat (1/3)

- *Tietojen kerääminen eri lähteistä (PDF:t, valmiiksi annetut linkit) ei ole ongelma tällä hetkellä tekoälyille*
- *Haasteena on haluttujen tietojen kerääminen ja semanttinen yhdistäminen*
  - *Esimerkiksi mitat löytyvät helposti, mutta tässä pilotissa haluttiin*
  1. *ensin tietää mitkä mitat ylipäätään halutaan löytää*
  2. *löytää mitat jossain formaatissa lähdetiedostosta*
  3. *muuntaa ne standardimuotoon*
  - *Toistaa tämä asia jokaiselle halutulle ominaisuudelle*
  - *Ennakkoon ei voi tietää, mitkä ominaisuudet löytyvät tiedostoista, joten tekoälyn tulee osata etsiä tietoa eri hakutermeillä ja yhdistää löydetyt tiedot haluttuihin tietoihin*

Pilotoinnin aikana havaitut asiat (2/3)

- *Tekoälylle annettu tehtävä on monitahoinen, mutta analyytikko onnistui tehtävässä yllättävän hyvin, mutta liian vaihtelevasti että sitä oltaisiin voitu luotettavasti edes testata oikeassa ympäristössä*
- *Moniagenttijärjestelmä olisi voinut toimia paremmin, mutta sitä ei pilotin puitteissa lähdetty työstämään ratkaisuksi*
  - *Microsoft Copilot ei sellaisenaan pystynyt työstämään suurta XML/Excel-tiedostoa*
  - *Moniagenttijärjestelmä, joka luotiin Copilot Studiossa käytti yhtä agenttia ominaisuuksien kartoittamiseen, yhtä agenttia semanttiseen yhdistämiseen ja yhtä agenttia tietojen koostamiseen*
    - *Menetelmä olisi voinut ehkä olla sopiva, mutta olisi vaatinut syvempää yhdistimien käyttöä tiedostojen käsittelyssä*

Pilotoinnin aikana havaitut asiat (3/3)

- *On tärkeää, että tekoälyn vastaus perustuu faktaan*
  - *Siksi kehotteessa tulee tuoda esille lähdeviittauksen tärkeys, jotta vastaus pystytään todentamaan ja varmistamaan*
  - *Tämä auttanee tekoälyä myös tietojen yhdistämisessä*
- *Pilotissa käytetyt teknologiat kehittyvät ajan myötä ja ratkaisu tällaisenaan saattaa toimia uudemmilla tekoälypalveluilla ja –malleilla myöhemmin*

Yhteenveto

- *Pilotissa testattiin useita generatiivisen tekoälyn työkaluja (Copilot, Copilot Analyytikko, Copilot Studio -moniagentti, ChatGPT ja mukautettu GPT), ja lopulta päädyttiin Copilotin Analyytikkoon sen selainkäyttöisyyden ja Python-koodia tuottavan ja hyödyntävän kyvykkyyden vuoksi*
- *Ratkaisumallissa haettiin ETIM-Excelistä luokan skeema EC-koodilla, poimittiin lähteistä tieto ohjelmallisesti, normalisoitiin arvot (yksiköt/booleanit/numerot) ja yhdistettiin standardiarvoihin*
- *Tuloksissa vastaukset olivat usein keskenään samankaltaisia, hallusinaatioita ei havaittu, mutta oikeaan standardikoodiin mappaus epäonnistui ajoittain ja laatutaso vaihteli runsaasti, joten suoraa aikasäästövertailua ei tehty*
- *Jatkokehityksenä moniagenttinen työnjako (tiedonhaku–järjestely–standardisointi) nähdään lupaavana suuntana.*

Jatkokehitysideoita

- *Moniagenttijärjestelmä (riippumatta alustasta) on yksi suunta mitä tutkia tulevaisuudessa*
  - *Ongelma voidaan jakaa osiin ja jakaa eri agenteille tehtäväksi*
    - *EC-luokan tunnistus eli luokkaehdotukset*
    - *Lähdehaku ja parsinta (PDF:t, taulukot ja tekstit)*
    - *Normalisointi (yksiköt, booleanit, numeroformaatit)*
    - *EV-luokitus*
    - *QA-agentti (laadun tarkistus: onko kaikki huomioitu?)*
- *Ihmisen varmistus saadulle tiedolle -> koulutusdataksi*
- *Työnkulkuintegraatio*
  - *API-rajapintojen hyödyntäminen, automatisointi, jne.*

Lisätietoa aiheesta

- *Hankkeen tiedot ja muut pilottiraportit löytyvät osoitteesta https://projektit.seamk.fi/alykkaat-teknologiat/vailla-tuottavuutta/*
- *ETIM-standardista lisää tietoa: https://www.etim-international.com/*



Vaasan yliopisto
UNIVERSITY OF VAASA

SEAMK
Seinäjoen ammattikorkeakoulu
Seinäjoki University of Applied Sciences

Tampereen yliopisto

Euroopan unionin osarahoittama

ETELÄ-POHJANMAAN LIITTO
Regional Council of South Ostrobothnia