Zpracování nutričních dat – dokumentace

ZADÁNÍ

Zpracovat nutriční data ze dvou Excel souborů (ReFresh a FIT) do jednoho CSV souboru s jednotnou strukturou.

Při řešení jsem používal AI asistenta (Claude) jako pomocníka. Pomohl mi hlavně s pochopením pandas syntaxe a kontrolou, jestli má logika správný postup.

JAK JSEM TO ŘEŠIL

1) Analýza dat

Nejdřív jsem si otevřel oba Excel soubory a podíval se, jak vypadají. ReFresh má hromadu listů (Tousty, Jogurty, Tortilly atd) a každý list má produkty a k nim suroviny. U každé suroviny jsou hodnoty na 100 g a pak přepočítané hodnoty podle toho kolik je jí v produktu. Je tam taky list Suroviny, který má databázi všech surovin, ale ten jsem nepotřeboval, protože ty hodnoty jsou už v těch produktových listech.

FIT soubor má úplně jinou strukturu – data jsou po 3 řádcích: název, hodnoty na porci, hodnoty na 100 g. V názvu je většinou napsaná velikost porce.

2) Co jsem použil

Python, protože jsem s ním už pracoval a má dobré knihovny na Excel. Použil jsem pandas (na čtení Excelu a práci s tabulkama), re (na hledání čísel v textu pro velikost porce), openpyxl (na zjištění viditelnosti listů), urllib (na komunikaci s OpenAI API) a concurrent.futures (na paralelní zpracování AI požadavků). Tyhle knihovny jsou standardní, každý je používá.

S AI (Claude) jsem konzultoval hlavně průchod řádky v pandas, validaci výstupu, ladění (např. konverzi čísel s čárkou), návrh promptů pro AI kategorizaci a implementaci batch processingu. U ReFresh jsem původně zvažoval sčítání surovin, ale finální hodnoty beru z řádku „Celkové výživové hodnoty produktu na 100g".

3) Jak to funguje

Udělal jsem funkce pro zpracování dat a přidal jsem AI kategorizaci místo ručních pravidel.

oprav_cislo() - Převádí čísla s čárkou na normální čísla s tečkou. Taky ošetřuje prázdné hodnoty a když je tam nějaký nesmysl. Dělám to takhle protože v Excelu jsou čísla s čárkama a Python chce tečky.

nacti_kategorie() - Načte číselník kategorií ze souboru categories.xlsx (ID, název, popis).

viditelne_listy() - Vrací seznam listů nastavených jako visible. Používá openpyxl kvůli zjištění viditelnosti listů. Excel umožňuje listy skrýt, chceme zpracovat jen to, co je aktivně viditelné, a tedy považované za aktuální zdroj.

vyber_kategorii() - Místo ručních pravidel s klíčovými slovy teď volá OpenAI API. AI dostane název produktu a kompletní číselník kategorií. Model sám vybere nejlepší kategorii podle názvu a popisů. Pokud AI není jistá nebo selže, vrátí prázdnou hodnotu místo špatné kategorie.

_precompute_categories() - Batch zpracování – vezme všechny názvy produktů najednou a pošle je do AI v dávkách po 10. Výsledky uloží do cache. To je mnohem rychlejší než posílat produkty jeden po druhém.

Pak mám dvě části pro zpracování:

ČÁST 1 - ReFresh data: projdu všechny (viditelné) listy kromě Suroviny, TESTOVÁNÍ a PV – přehled. U každého produktu neberu součet jednotlivých surovin jako finální produkt, ale načtu řádek „Celkové výživové hodnoty produktu na 100 g", kde jsou už přepočtené celkové hodnoty.

ČÁST 2 - FIT data: čtu po 3 řádcích. Z prvního řádku vezmu název a zkusím najít velikost porce (hledám "50 g", "100 g" a tak). Z druhého řádku vezmu nutriční hodnoty. Když nemám velikost porce, vypočítám ji z poměru kJ.

Nakonec všechno spojím do jednoho CSV se středníkem jako oddělovačem.

PROČ JSEM TO UDĚLAL TAKHLE

Proč Python? Mohl jsem to dělat v Excelu s makrama, ale to by bylo hrozně zdlouhavé a těžko se to spouští automaticky. Python se dá spustit kdykoli a udělá to vždycky stejně.

Proč nepoužívám list Suroviny? Protože ty hodnoty z něj jsou už zkopírované v produktových listech. Každý produkt má u surovin hodnoty na 100 g (stejné jako v databázi) a navíc přepočtené podle váhy.

Proč nesčítám hodnoty? Řádky se surovinami jsou jen jednotlivé komponenty (např. 220 g → 847 kJ). Kdybych je sečetl a vzal jako hotový produkt, vzniknou nadhodnocené hodnoty. Finální hodnoty hotového produktu proto beru z posledního řádku.

AI kategorizace místo ručních pravidel: Původně jsem měl v kódu funkci najdi_kategorii() s ručními pravidly – hledal jsem klíčová slova jako "jogurt", "tousty" atd. To fungovalo, ale mělo to problémy: musel jsem ručně psát pravidla pro každou kategorii, nefungovalo to dobře pro nestandardní názvy, a při přidání nové kategorie jsem musel upravovat kód. Nové řešení s AI je flexibilnější – stačí upravit categories.xlsx a AI sama rozhodne podle popisů. AI taky rozumí překlepům a cizím jazykům.

Batch processing a paralelizace: Místo jednoho produktu po druhém teď posílám 10 produktů najednou (AI_BATCH_SIZE=10) v 8 paralelních vláknech (AI_CONCURRENCY=8). To je mnohem rychlejší a levnější. První request platí plnou cenu, další jsou levnější díky prompt cachingu.

KONKRÉTNÍ KROKY

1. Příprava: Stáhl jsem zadání, prohlédl Excel soubory, udělal si poznámky, co kde je.

2. Nastavení: Otevřel jsem VS Code, nainstaloval pandas a openpyxl. Měl jsem problém, že VS Code používal jiný Python – musel jsem změnit interpreter na Anaconda.

3. Kódování: Napsal jsem funkce oprav_cislo() a najdi_kategorii() + viditelne_listy(). ReFresh zpracování – pro každý produkt čtu hodnoty z posledního řádku (sloupce 13–22). FIT zpracování – čtu po 3 řádcích a beru hodnoty na 1 porci, porci získám z názvu.

4. Testování: První pokus: chyba s pandas. Druhý pokus: divné hodnoty – opravil jsem indexy. Třetí pokus: fungovalo. Přidal jsem provozovatele ReFresh Bistro.

5. Upgrade na AI: Vytvořil jsem modul ai_categorizer.py s OpenAI API. Přidal jsem načítání .env souboru pro konfiguraci. Implementoval batch processing (10 produktů najednou) a paralelní zpracování (8 vláken). Přidal retry logiku pro 429 rate limits, error handling a cost tracking. Odstranil všechna ruční pravidla – kategorizace teď jde 100% přes AI.

6. Optimalizace: Přidal jsem prompt caching (první request platí plnou cenu, další jsou levnější), cenový limit (AI_MAX_COST_USD=20), pre-compute fázi (všechny produkty se zpracují najednou v batch requestech). Výsledek: 10× rychlejší, 5× levnější než jednotlivé requesty.

Hlavní problémy: VS Code špatný interpreter (20 minut googlení), nevěděl jsem, které sloupce jsou co, čísla s čárkou, regex jsem nikdy moc nepoužíval. A celkově byl problém pro mě ten formát tabulek, všude kde se něco učím (kurzy atd..), tak tabulky jsou jasné, tady to bylo složitější a také ty skryté listy. Při AI části: OpenAI API retry logika pro 429 rate limits, thread-safe operace pro paralelní zpracování, validace AI výstupu proti číselníku.

AI KATEGORIZACE - TECHNICKÉ DETAILY

Jak to funguje

OpenAI API (Responses endpoint):

• Používám model gpt-4o-mini (levný, rychlý, dostatečně přesný)

• API klíč je v .env souboru

• Structured outputs s JSON schématem – model MUSÍ vrátit validní JSON

Prompt strategie (soubor ai_categorizer.py):

AI dostane název produktu a kompletní číselník kategorií. Prompt říká AI, aby:

1. Normalizovala význam názvu (jazyk, překlepy, zkratky)

2. Odhadla typ produktu a jeho stav (čerstvé/mražené/sušené/konzervované/uzené)

3. Soustředila se na hlavní surovinu – dochucení, omáčky, koření jsou doplňky

4. Pokud je v názvu způsob úpravy (uzené, pečené), držela se toho

5. Sušené používala JEN když je výslovně uvedeno "sušené"

6. Uzené/zaúzené znamená UZENÉ

7. Porovnala s popisy kategorií a vybrala tu nejpřesnější

Validace:

• AI může vybrat POUZE z poskytnutých kategorií

• Model vrací JSON: {"id_kategorie": 42, "confidence": 0.95}

• Pokud AI vrátí ID mimo číselník, nastaví se prázdná hodnota

• Confidence (0–1) ukazuje, jak si je AI jistá

Ochrana proti chybám:

• Retry logika pro 429/5xx chyby (max 6 pokusů s exponenciálním backoffem)

• Maximální 2 pokusy na produkt (AI_MAX_ATTEMPTS)

• Tvrdý cenový limit 20 USD (AI_MAX_COST_USD)

• Thread-safe operace (locks pro statistiky a cost tracking)

Optimalizace rychlosti a ceny

Batch processing:

• Místo jednoho produktu posílám 10 najednou (AI_BATCH_SIZE=10)

• Rychlejší a levnější – prompt se počítá jednou místo 10×

Paralelní zpracování:

• 8 paralelních vláken (AI_CONCURRENCY=8)

• Produkty se zpracovávají současně

Prompt caching:

• Číselník kategorií se cachuje v paměti

• První request platí plnou cenu, další jsou o 50% levnější

• AI_PROMPT_CACHE_KEY a AI_PROMPT_CACHE_RETENTION v .env

Pre-compute fáze:

• Nejdřív se načtou všechny názvy produktů ze všech listů

• Pak se pošlou najednou do AI v batch requestech

• Výsledky se uloží do cache (slovník v paměti)

• Při sestavování výstupu už jen čteme z cache

Nastavení (.env)

Všechna nastavení pro AI jsou v souboru .env:

ZÁVĚR

Vytvořil jsem Python skript, který načte data z ReFresh a FIT souborů, získá finální hodnoty pro produkty (ReFresh z řádku „Celkové výživové hodnoty produktu na 100g", FIT z řádku „na 1 porci"), spojí to dohromady do jednoho CSV a automaticky přiřadí kategorie. Kód je jednoduchý, komentovaný a dá se snadno upravit.

AI mi při tom pomohlo hlavně jako konzultant – když jsem nevěděl, jak přesně něco v pandas udělat nebo jsem chtěl zkontrolovat, jestli mi logika dává smysl. Pomohlo mi taky s návrhem promptů pro AI kategorizaci a implementací batch processingu.

V rámci zpracování souboru nutrition_data_ReFresh.xlsx byly použity pouze listy, které jsou v Excelu nastavené jako viditelné. Skryté listy nebyly do zpracování zahrnuty.