Semantic Product Search & Recommendation

Capstone Project – submitted by Jayesh Reddy

End-to-End Machine Learning Lifecycle Capstone Project - ReadMe

Project Overview

This project demonstrates the end-to-end application of the machine learning lifecycle on a real-world, industry-relevant e-commerce dataset. The focus is on designing and evaluating a content-based product recommendation and retrieval system that allows users to describe what they are looking for in natural language, instead of relying on exact keyword search.

The project spans problem framing, data understanding, preprocessing and feature engineering, model implementation, evaluation, and critical reflection on ethical and fairness considerations.

Business Context

Traditional e-commerce platforms typically rely on:

- Keyword-based search, which assumes users know the correct terminology, or
- Generic recommender widgets which offer limited support when user intent is vague or descriptive.

This project explores how semantic retrieval can improve product discovery by matching free-text user queries to relevant products, even when keyword overlap is limited.

Dataset

- **Source:** Kaggle
- **Dataset:** Walmart Product Data (December 2019)
- **URL:** https://www.kaggle.com/datasets/promptcloud/walmart-product-data-2019

The dataset contains approximately 30,000 products with metadata including titles, descriptions, categories, brands, availability, and pricing information.

Raw data is not included in the submission as it is publicly available via Kaggle. All derived datasets and model artefacts can be regenerated using the submitted code, with key artefacts bundled to support reproducibility.

Project Structure

```
Project Structure

The GitHub repository for this project is organised as follows:

semantic-product-search/
├── README.md
├── requirements.txt
├── .gitignore
│
├── src/
│   ├── step3a_preprocessing_eda.py
│   ├── step3b_feature_engineering_dimensionality_reduction.py
│   └── step4_retrieval_evaluation.py
│
├── artifacts/
│   └── step3b_artifacts_bundle.zip
│
├── data/
│   ├── raw/
│   │   └── README.md
│   └── processed/
│       └── README.md
│
└── docs/
    └── formal_report.pdf
```

.

Workflow Summary

Each step in the workflow is implemented as a standalone Python script in the GitHub repository, with explicit artefacts persisted between steps to support reproducibility and independent execution.

Step 1: Problem Framing

- Framed as a content-based recommendation / information retrieval task.
- Focused on ranking products by relevance to natural-language queries.
- Success metrics defined using ranking-based measures (Precision@K, Recall@K, nDCG@K).

Step 2: Data Understanding

- Dataset suitability assessed with respect to the problem context.
- Feature types, missingness, and distributional characteristics examined.

- Data source and provenance clearly documented (Kaggle).

Step 3A: Data Preprocessing & Exploratory Data Analysis

- Data cleaning, de-duplication, and text normalisation.
- Handling of missing values and removal of duplicate product records.
- Exploratory analysis of:
    - category and brand distributions,
    - price distributions, and
    - title and description length characteristics.
- Outputs from this step were saved as a cleaned dataset to support reproducibility and downstream modelling.

Step 3B: Feature Engineering & Representation Learning

- Construction of text-based feature representations using:
    - TF-IDF vectors as a lexical baseline, and
    - Sentence embeddings as a semantic representation.
- Dimensionality reduction using PCA to support analysis and visualisation.
- Artefacts (vectorisers, matrices, embeddings, and indices) were persisted to disk and bundled to enable reuse in subsequent steps.

Step 4: Model Implementation & Evaluation

- Implemented two retrieval models:
    - TF-IDF cosine similarity (baseline)
    - Semantic embedding similarity using a pretrained transformer
- Evaluation performed using:
    - category-based relevance as a scalable proxy,
    - ranking metrics (Precision@K, Recall@K, nDCG@K), and
    - manual qualitative evaluation with realistic user queries.
- Model artefacts were reused directly from Step 3B to ensure a reproducible, artefact-driven workflow.

Step 5: Ethical AI & Bias Auditing

- Critical analysis of explainability, bias, and fairness considerations.
- Focused on structural biases (category dominance, brand frequency, price-tier effects).
- Proposed practical mitigation strategies such as diversity-aware and category-aware re-ranking.

Reproducibility & Environment Notes

Artefact-Based Workflow

This project follows an artefact-driven workflow rather than relying on implicit notebook state or execution order. Each major stage produces explicit outputs that are reused downstream.

- **Step 3A** produces a cleaned and deduplicated product dataset following preprocessing and exploratory analysis.
  This dataset is saved to disk and represents the canonical input for all subsequent modelling steps.
- **Step 3B** consumes this cleaned dataset to generate feature representations (TF-IDF vectors and semantic embeddings), which are then persisted as reusable artefacts.
- **Step 4** operates entirely on artefacts produced in Step 3B, ensuring that evaluation is decoupled from feature generation.

This approach mirrors common industry practice and supports clarity, traceability, and reproducibility.

Google Colab Compatibility

Several notebooks are designed to run in Google Colab, which uses an ephemeral execution environment. To accommodate this:

- The cleaned dataset generated in **Step 3A** is manually uploaded when starting **Step 3B** in Colab.
- Feature artefacts generated in **Step 3B** (vectorisers, matrices, embeddings, and indices) are bundled into a ZIP file for easy reuse.
- **Step 4** includes environment checks that prompt the user to upload this artefact bundle when running in Colab.

In a persistent local environment, these manual upload steps are not required; they are retained to ensure portability and to allow for easy accessibility in the assessment of this project.

Design Rationale

While manual file uploads would be unnecessary in a fully persistent environment, retaining this pattern:

- makes data and model dependencies explicit,
- avoids reliance on hidden notebook state, and
- allows each step to be executed independently for review.

This design choice prioritises transparency and reproducibility over execution convenience.

Code Availability

All code developed for this project is included as part of the submission and organised by project step. The primary implementation is provided as Python (.py) files exported from the original development notebooks, in line with submission requirements.

The code covers:

- **Step 3A – Data Preprocessing and Exploratory Analysis**
  Data cleaning, handling of missing values, exploratory analysis, and preparation of a cleaned dataset for downstream modelling.
- **Step 3B – Feature Engineering and Representation Learning**
  Implementation of the TF-IDF baseline, semantic embeddings using a lightweight sentence transformer, dimensionality reduction, and visualisation.
- **Step 4 – Retrieval, Ranking, and Evaluation**
  Construction of retrieval pipelines, quantitative evaluation using ranking metrics, and qualitative inspection through manual query examples.

Models, vectorisers, and intermediate artefacts are saved explicitly to support reproducibility and to accommodate execution in non-persistent environments (e.g. Google Colab). Execution order and environment considerations are described elsewhere in this README.

Submitted Artefacts

As part of the submission, selected intermediate artefacts generated during the modelling process are included to support transparency and reproducibility. These artefacts are included in both the formal submission and the accompanying GitHub repository.

In particular, the artefact bundle produced at the end of **Step 3B** is included in the submission. This bundle contains the trained vectorisers, feature representations, and supporting indices used during retrieval and evaluation in Step 4. Including these artefacts allows the evaluation stage to be reproduced without re-running the full feature engineering pipeline and reflects common practice in applied machine learning workflows.

Raw source data is not included in the submission, as it is publicly available via Kaggle and is referenced explicitly elsewhere in this README. All artefacts included in the submission can be regenerated using the provided code if required.

Model Choices & Scope

Model selection was driven by task appropriateness, and not for the sake of model variety.

Given the problem was framed as a content-based recommendation and retrieval task, experimentation focused on:

- A traditional lexical retrieval baseline (TF-IDF)
- A semantic embedding-based retrieval approach

Supervised classifiers and clustering models were intentionally excluded from Step 4, as they are less naturally aligned with ranking-based retrieval in the absence of labelled relevance data.

Key Dependencies

- Python 3.x
- pandas, numpy, scikit-learn
- sentence-transformers
- matplotlib / seaborn
- scipy
- joblib

See the requirements.txt file for full details.

Notes for Assessors

- All modelling and evaluation decisions are explicitly justified in the accompanying report.
- Step 5 is primarily analytical and builds on artefacts generated in Steps 3 and 4.
- No demographic data was available; bias analysis therefore focuses on structural and representational effects relevant to e-commerce systems.