Find me a better product!

Jenson Chang, Chukwunonso Ebele-Muolokwu, Catherine Meng, Jingyuan Wang

Table of contents

Executive Summary

FinlyWealth is an affiliate marketing platform offering cashback on financial products, now expanding into e-commerce with a broader product catalog. To support this transition, a team of Master of Data Science students from the University of British Columbia is developing a scalable, multimodal product search engine.

Our hybrid retrieval strategy combines CLIP-based embeddings (Radford et al. 2021) with FAISS indexing (Johnson, Douze, and Jégou 2017) for efficient, large-scale similarity search. The system comprises a Streamlit frontend (Streamlit Inc. 2019), a Flask API backend (Ronacher 2010), and a vector store for embeddings. We will evaluate performance by recall@K, query latency, and human relevance judgments. The final prototype will deliver intelligent product discovery beyond traditional keyword matching.

Introduction

As FinlyWealth expands from personal finance into e-commerce, they face the challenge of building a product search experience that can scale across a diverse and rapidly growing catalog. To support this transition, a team of Master of Data Science students from the University of British Columbia is developing a machine learning-powered search engine capable of handling both text and image queries.

Search in the e-commerce domain is uniquely difficult due to inconsistent metadata, varied product categories, and the need to interpret nuanced user intent—such as "pants under $100" or image-based queries. The current system relies on basic keyword matching, lacking semantic understanding, multimodal input support, and performance benchmarking at scale.

To address these gaps, this project will implement a semantic search engine that fuses text and image embeddings, providing accurate and flexible retrieval for a multimillion-item catalog.

Objective

Design and implement a fast, scalable search system with multimodal capabilities. Architecture components include:

- **Frontend**: Streamlit for interactive text and image queries (Streamlit Inc. 2019)
- **Backend API**: Flask for query handling and embedding generation (Ronacher 2010)
- **Similarity Engine**: FAISS for approximate nearest neighbor search (Johnson, Douze, and Jégou 2017)
- **Vector Store**: PostgreSQL or equivalent for embedding storage and metadata

Over eight weeks, we will iterate on the prototype in collaboration with FinlyWealth. Success metrics include:

- Recall@K for retrieval accuracy
- Latency for query responsiveness
- Manual relevance assessments for qualitative validation

Data Science Techniques

Data Source and Description

The dataset consists of multi-modal product data, including images, textual descriptions, and structured metadata. It includes:

- **14,684,588 JPEG images** (67 GB), each under 150 × 150 pixels. Filenames correspond to the `Pid` field in the metadata CSV. Although this is slightly fewer than the number of unique product entries (15,147,805), we assume a 1:1 mapping between images and product listings based on confirmation from our project partner.
- **A 12 GB CSV file** with 15,384,100 rows and 30 columns, each row representing a product listing. Metadata fields include `Pid`, `Name`, `Description`, `Category`, `Brand`, `Color`, `Price`, `Gender`, and product URLs.

This dataset was provided by FinlyWealth via its affiliate network and will underpin both exploratory analysis and model development.

Exploratory Data Analysis

Several metadata columns contained substantial missing values and were therefore dropped in favor of more complete fields (see Table 1). Attributes like `Color`, `Gender`, and `Brand` still have missing entries but were retained because they add modeling value (see Table 2). The missingness in these fields may be due to certain product types—such as books—lacking those attributes (see the category distribution in Figure 1).

Table 1: Table: Summary of Dropped Columns and Reasons

Attribute	Reason Dropped
`ShortDescription`	91% missing
`Keywords`	99% missing
`CategoryId`	Redundant with `Category`
`ImageURL`	Not used directly in modeling
`SalePrice`	Redundant with `FinalPrice` / incomplete usage
`PriceCurrency`	Not used in current modeling scope
`MPN`	High cardinality (~2.3 M values), sparse
`UPCorEAN`	High cardinality (~3.5 M values), sparse
`SKU`	Redundant with `Pid` (~10 M values)
`AlternateImageUrl*`	Not used; only primary image needed
`DeepLinkURL`	Not relevant for modeling
`LinkUrl`	Not relevant for modeling

Table 2: Table: Summary of Retained Columns and Their Characteristics

Group	Attribute	Description / Examples
Identifiers	`Pid`	Unique product ID; links to image filenames
Text Fields	`Name`	Product title (0.2% missing)
	`Description`	Product description (0.03% missing)
	`Category`	Product category (28% missing; ~15 K unique values)
Pricing & Availability	`Price`	Listed price
	`FinalPrice`	Final price after discounts
	`Discount`	Discount percentage or value
	`isOnSale`	Boolean flag
	`IsInStock`	Boolean flag
Branding	`Brand`	Brand name (53% missing; ~21 K unique values)
	`Manufacturer`	Manufacturer name (34% missing; ~26 K unique values)

Group	Attribute	Description / Examples
Product Features	`Color`	Product color (49% missing; ~170 K unique values)
	`Gender`	Target gender (54% missing; 3 values: e.g., male/female)
	`Size`	Product size (46% missing; ~55 K unique values)
	`Condition`	Product condition (e.g., new, used; 5 values)

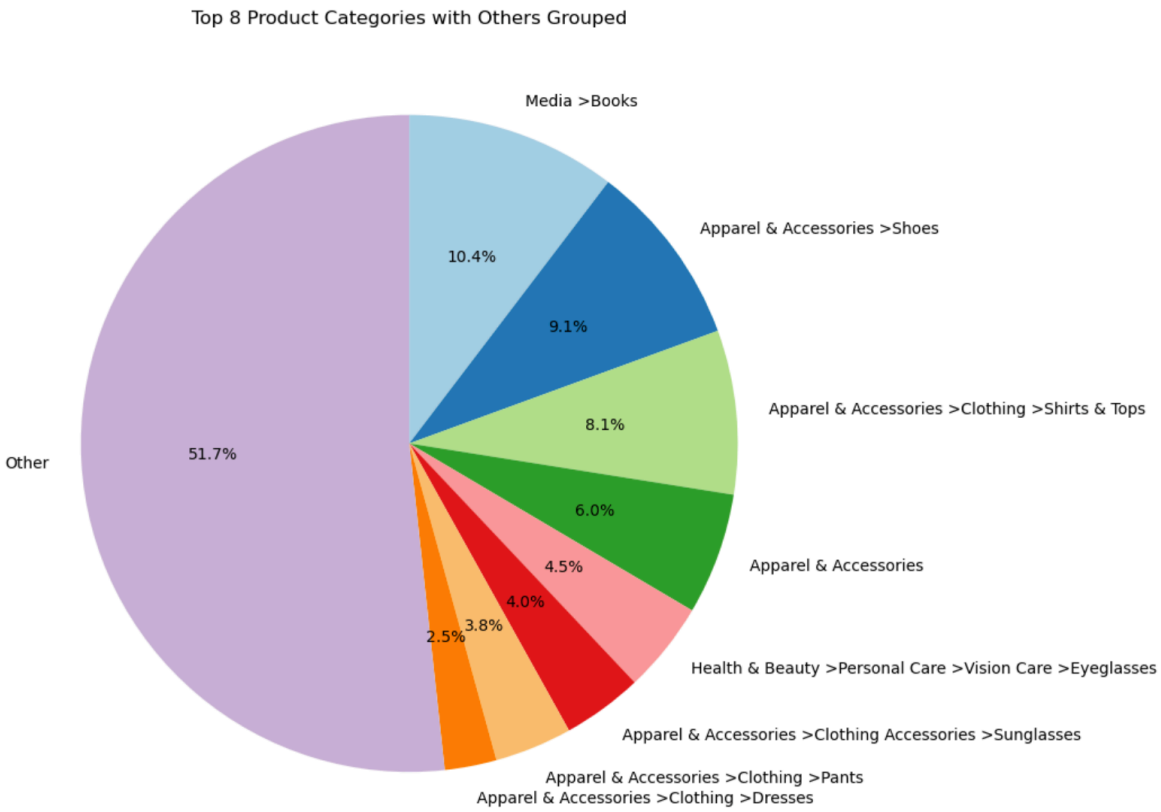

Figure 1: Top 8 Product Categories with Others Grouped

High-cardinality fields such as `Category`, `Size`, and `Brand` (see Figure 2) may require grouping rare values to avoid overfitting and reduce memory usage. We merged `Brand` and `Manufacturer` into a single `MergedBrand` column to reduce duplication while preserving distinct information. We also excluded non-English market entries—retaining approximately 70% of the dataset—as confirmed by our partner.

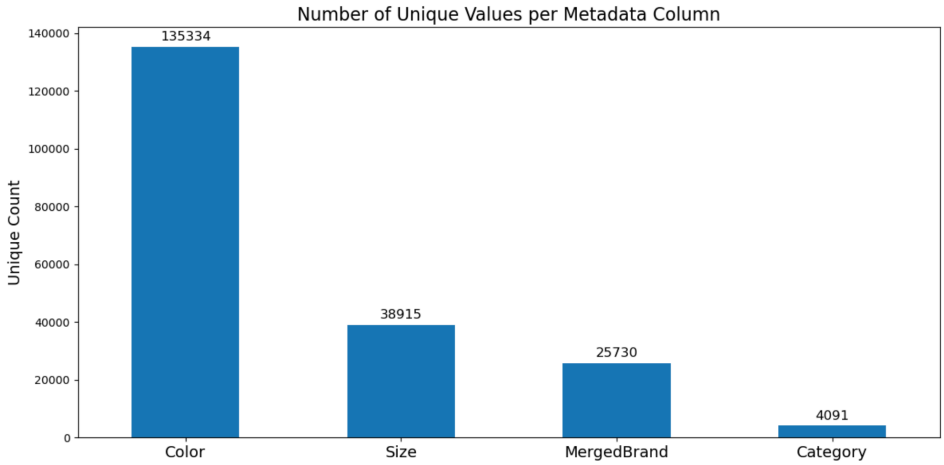

Figure 2: Unique Value Counts per Column After Data Cleaning

To understand likely user searches, we analyzed the 50 most frequent words in product names (see Figure 3). Common terms like "womens", "mens", "size", and "black" appear frequently and often overlap with metadata fields such as `Gender` and `Color`.

Figure 3: Top 50 Words in Product Names

System Workflow

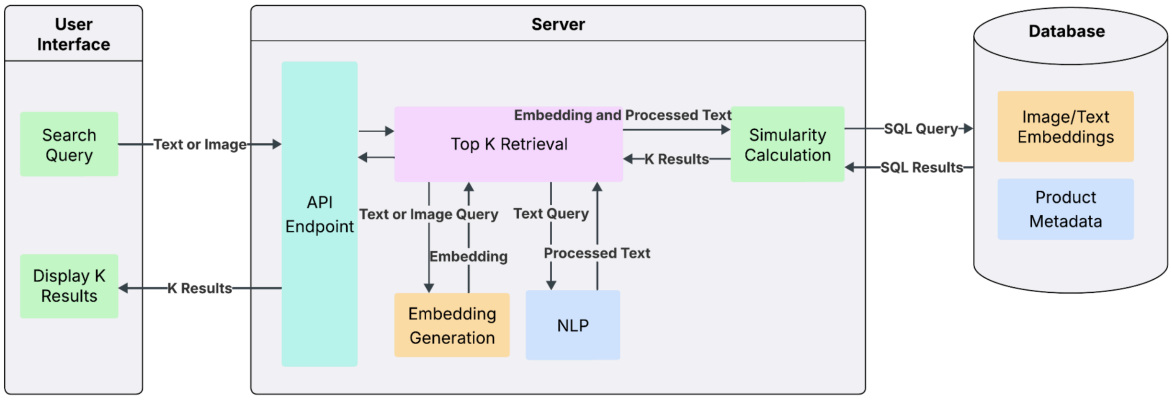

Figure 4: Workflow Diagram

Figure 4 illustrates the end-to-end architecture, including a user interface, an API server for query processing, and a database for storing embeddings and metadata.

- **User Interface:** A web interface accepts text or image queries, designed for intuitive interaction.
- **API Endpoint & Preprocessing:** Queries are sent to our Flask(Ronacher 2010) server, where they undergo preprocessing.
- **Embedding Generation:** After preprocessing, text is transformed into representations that can be compared with stored data. We generate embeddings using OpenAI CLIP(Radford et al. 2021) (Contrastive Language–Image Pre-training) or create text vectors via TF-IDF(Aizawa 2003).
- **Similarity Calculation & Retrieval:** We calculate similarity (e.g., cosine similarity(Xia, Zhang, and Li 2015), Euclidean distance, dot product(Ogita, Rump, and Oishi 2005)) between query and product representations. FAISS(Johnson, Douze, and Jégou 2017) supports fast, scalable retrieval.
- **Database & Indexing:** Preprocessed embeddings and metadata are stored in a centralized database. We evaluated vector-aware options such as Pinecone(Pinecone Systems 2021), ChromaDB(Team 2023), and PostgreSQL(PostgreSQL Global Development Group 1996) with pgvector for efficient similarity search.

Baseline Approach: We implement a TF-IDF-based retrieval model as an interpretable, lightweight baseline.

Tools and Libraries: See the appendix for a complete list.

Implementation Challenges

- Scalability (latency and memory) for efficient retrieval over a large catalog
- Data-quality issues (e.g., high-cardinality metadata) that can degrade TF-IDF and CLIP performance
- Defining and implementing metrics to evaluate multimodal query effectiveness
- Experimentally tuning and validating hybrid-retrieval weights (semantic vs. lexical)

Timeline

Due to the dataset's size, we plan to start early to ensure sufficient time for optimization, as shown in Figure 5.

- **Week 1:** Develop the initial pipeline, finalize the project proposal, and complete the necessary exploratory data analysis (EDA).
- **Week 2:** Focus on developing and integrating the core components locally, including the user interface, API, and data processing pipeline, to ensure smooth local operation before cloud deployment.
- **Week 3:** Process the full dataset on the cloud, connect the frontend with the backend, and explore the impact of advanced NLP techniques such as Named Entity Recognition (NER).
- **Week 4:** Begin performance testing and refinements, focusing on internal demos and retrieval benchmarks.
- **By Week 6 (before June 9):** Deliver a runnable draft to our mentor for review and final presentation preparation, ensuring it fully meets our capstone partner's requirements.
- **Final Presentation:** Scheduled for June 12 or 13. We will then have 10 days to finalize the product and report, aiming for final submission by **June 26 (Week 8)**.

We acknowledge the project's complexity and will adjust the timeline as needed, except for hard submission deadlines.

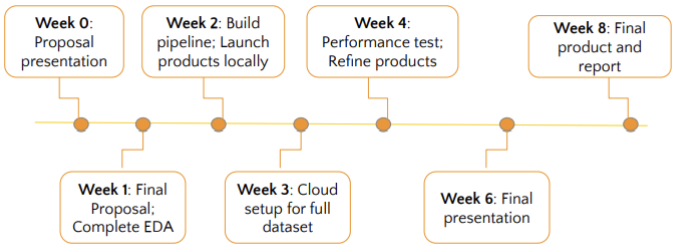

Figure 5: Timeline

Appendix

Tools and Libraries

Library	Purpose in Project
NumPy	Efficient numerical operations, especially for vector manipulation and math ops.
Flask	Lightweight web framework used for rapid prototyping of API endpoints.
FAISS	Approximate nearest neighbor search for CLIP embeddings; enables fast vector search.
Hugging Face	Access to pretrained models like CLIP; used for text and image embedding.
Pillow	Image processing library used for resizing, normalization, and format conversion.
spaCy	Natural language processing toolkit for tokenization, NER, and text normalization.
Pinecone	Scalable, cloud-based vector database for fast and persistent similarity search.
PostgreSQL	Relational database to store Embeddings. Allows for multiple columns to have ebeddings

Definitions

CLIP: Generates embeddings for both text and images, mapping them into a shared embedding space. We are not training any embedding model, instead we use off-the-shelf CLIP models to generate embeddings.

Embedding Generation: The preprocessed query is then transformed into a numerical representation (an embedding) that captures its semantic meaning.

FAISS (Facebook AI Similarity Search) is a library that allows developers to quickly search for embeddings of multimedia documents.Enables efficient approximate nearest neighbor search over embeddings.

TF-IDF: A numerical statistic used to evaluate the importance of a word in a document within a collection of documents

References

Aizawa, Akiko. 2003. "An Information-Theoretic Perspective of Tf–Idf Measures." *Information Processing & Management* 39 (1): 45–65.

Johnson, Jeff, Matthijs Douze, and Hervé Jégou. 2017. "FAISS: A Library for Efficient Similarity Search and Clustering of Dense Vectors." https://github.com/facebookresearch/faiss.

Ogita, Takeshi, Siegfried M Rump, and Shin'ichi Oishi. 2005. "Accurate Sum and Dot Product." *SIAM Journal on Scientific Computing* 26 (6): 1955–88.

Pinecone Systems, Inc. 2021. "Pinecone: Vector Database as a Service." https://www.pinecone.io/.

PostgreSQL Global Development Group. 1996. "PostgreSQL: The World's Most Advanced Open Source Relational Database." https://www.postgresql.org/.

Radford, Alec, Jong Wook Kim, Luke Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, et al. 2021. "Learning Transferable Visual Models from Natural Language Supervision." *Proceedings of the International Conference on Machine Learning (ICML)*. https://github.com/openai/CLIP.

Ronacher, Armin. 2010. "Flask: Web Development, One Drop at a Time." https://flask.palletsprojects.com/.

Streamlit Inc. 2019. "Streamlit." https://streamlit.io.

Team, Chroma. 2023. "Chroma: The Open-Source Embedding Database." https://www.trychroma.com.

Xia, Peipei, Li Zhang, and Fanzhang Li. 2015. "Learning Similarity with Cosine Similarity Ensemble." *Information Sciences* 307: 39–52.