Yelene Cisse, Kshamaa Suresh, Gabrielle Park, and Gabriella Chu
Applied Machine Learning

SkinWise - Your New AI Skincare Agent

Abstract:

As interest in personalized skincare grows worldwide, more and more consumers are facing an overwhelming product landscape, making it difficult to know which products to select for their personal needs. SkinWise, a retrieval-augmented generation (RAG) model, solves this problem by offering skincare recommendations, personalized to user needs. Leveraging product, review, and medical metadata, SkinWise matches users with the right product upon query. This system, based on a large language model (LLM), acts as a personal skincare assistant, generating clear, conversational skincare suggestions, filtered by user-specified preferences, like skin concerns, budget, and ingredients, along with associated product ratings and sentiment insights from real users. Fitted with a seamless, user-friendly interface, SkinWise provides personalized skin care guidance, making it easy for consumers to find products that meet their skincare needs.

Introduction and Motivation:

The global beauty market is currently a billion dollar industry and is projected to grow 6% annually through 2028. Skin care is the beauty market's largest category, accounting for 44% of the market. This growth is largely fueled by products entering the market with new, trendy ingredients. With a growing consumer demand in skin care products and a large array of products available, consumers need to find products that match their specific needs—finding products that target certain skin concerns, fitting into a personal price budget, and beyond. To address the challenge of skin care product discovery, this project provides a personalized product recommendation system for individual users.[1]

Recent developments in LLMs and RAG systems inform our approach to this challenge. In this project, we introduce SkinWise, an AI-powered skincare recommendation system that leverages RAG to provide personalized product guidance to users. The system integrates multiple publicly available datasets including skin care product metadata, ingredient lists, consumer reviews, and dermatological medical information. SkinWise aims to provide accurate and easily accessible skin care product recommendations grounded in real product, review, and medical data. This work aims to demonstrate how modern RAG architectures can be applied to solve real-world consumer challenges, bridging the gap between technical skin care data and user friendly guidance.

Related Work:

SkinWise is inspired by recent advances in semantic search and RAG, where LLMs are

[1] *The beauty industry boom: Can growth be maintained? | McKinsey*. (n.d.). Retrieved December 19, 2025, from https://www.mckinsey.com/industries/consumer-packaged-goods/our-insights/the-beauty-boom-and-beyond-can-the-industry-maintain-its-growth

constantly improving to produce increasingly accurate and relevant responses that are easily interpreted by users and grounded in external knowledge. As skin care enthusiasts, we wanted to create a tool that was practical for the average consumer—our target audience for this project. Similar RAG pipelines have been applied in retail search, healthcare question answering, and recommendation systems. SkinWise adapts previous ideas using sentence embedding-based retrieval to the skincare domain by integrating structured product metadata, reviews, and medical information into a conversational recommendation framework.

Our approach is also inspired by AI assistants tailored for medical use cases. Medical chat bots increase access to healthcare and innovate approaches to personalized medicine.[2] Trained on medical data and electronic health records (EHRs), these chat bots provide dialogue-based medical suggestions. Specifically, dermatology has emerged as a specialty with high potential for AI applications because of its reliance on descriptively accurate texts and image diagnosis.[3] One example of an AI agent tailored to the cosmetic industry is Haut.AI, which uses image processing to identify user skin conditions and simulate how skin may change over time.[4] Our project iterates on these existing use cases to build an AI agent that brings together clinical advice, detailed product guidance, an advanced RAG pipeline, and an accessible UI interface to deliver personalized skincare advice.

Methodology / System Design (describe pipeline, architecture, or experiment):

Overview:

The workflow for our project was organized into four main steps: (1) data collection and preparation, (2) product review sentiment analysis, (3) LLM agent, and (4) user interface (*see Appendix C for a visualization of our workflow*).

Data collection and preparation:

We used medical data to strengthen our model's clinical understanding and ability to recommend appropriate products. First, the dermatology dataset from HuggingFace contains over 1,500 rows in a Q&A format, pairing a skin-related question with an answer on a certain condition, its causes, symptoms, or treatment options. The dataset covers a range of common skin conditions, including acne, eczema, rosacea, and melanoma. We use these medically oriented texts for matching user symptom queries with appropriate products. We extracted information from the text data to build in features, and we categorized data based on what type of information it was sharing (e.g. cause, symptom), what condition it was addressing, and other symptom related properties (e.g. redness, itchiness, inflammation, etc.)

Next, we used a Kaggle dataset that scraped Renude's "Ingredients in Skincare" database to provide more context on the ingredients in each product. This dataset lists 248 skincare

---

[2] *HOLMeS: eHealth in the Big Data and Deep Learning Era—ProQuest*. (2019). Retrieved December 19, 2025, from DOI:10.3390/info10020034

[3] Ziad, K., Mahdi, A., Badea, J., Naji, B., & Shapiro, J. (2025). The Role of ChatGPT in Dermatology Diagnostics. *Diagnostics*, 15(12), 1529–1550. https://doi.org/10.3390/diagnostics15121529

[4] S*cience-Backed AI Skin Analysis Solutions for Beauty | Haut.AI.* (n.d.). Retrieved December 19, 2025, from https://haut.ai/

ingredients, their purpose, and what it can help treat. We derived features, such as ingredient name, description, function, and target population. This data strengthens our model's understanding of each product so that it can provide more informed recommendations, matching conditions with effective ingredients.

The primary dataset for the recommendations is a Kaggle dataset that contains information for over 8,000 products from the beauty retailer Sephora. Of these, we use about 2,500 rows which are tagged as "skin care". This dataset also includes over one million reviews of all products from the skin care category, including user appearance and review ratings. This data was used to provide real feedback and analyze sentiment across products. We converted the highlights column into 32 new binary features, such as whether a product is vegan, has SPF, or is good for a particular skin type. We also created a few custom features. For example, the dataset included a sensitive skin flag if it was both hypoallergenic and fragrance-free.

This preprocessing allowed us to create clean and structured data for embeddings. To do the embeddings, we created a new column called "product_text" that consolidated all the important product information into a single compact description including name, brand, key active ingredients, formulation preferences, and more. Then, we used a pretrained sentence transformer from HuggingFace to convert these descriptions into embeddings. Each product is represented as a 384-dimensional vector which is crucial to allow our model to find relevant products through semantic search when a user makes a query.

Sentiment Analysis:

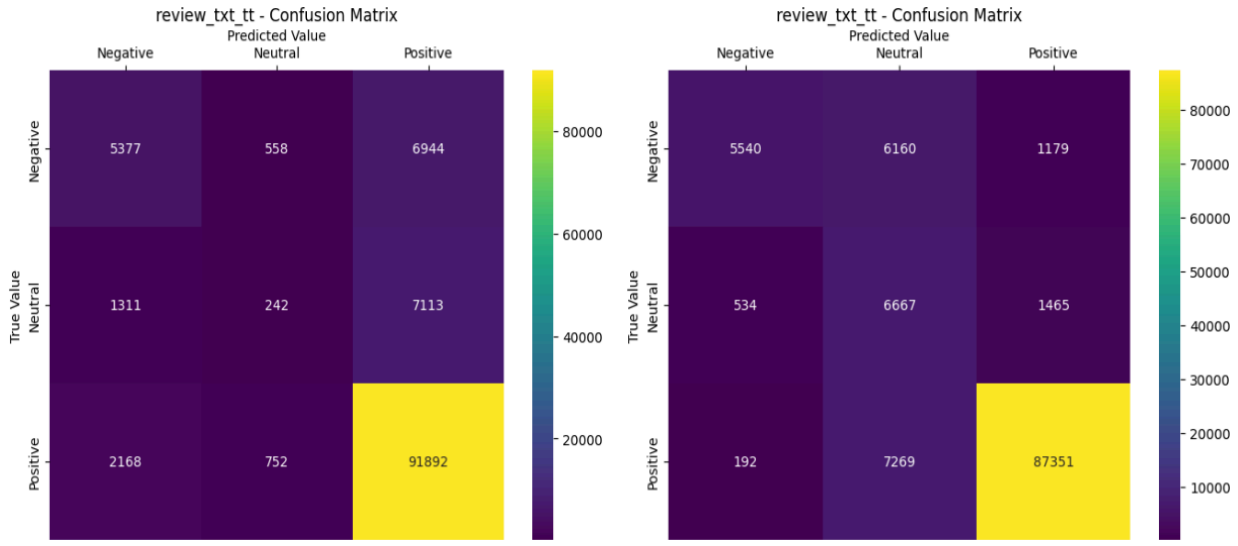
To supplement our AI agent with sentiment understanding, we built a sentiment analyzer to derive top-5 reviews per product based on text length, sentiment confidence, and label accuracy. This approach enables the agent to access high-quality representative feedback while maintaining computational efficiency. Additionally, the analyzer allows us to process reviews from diverse sources (e.g., Reddit, blogs) where numerical ratings may be absent—a key requirement for a scalable, multi-source recommendation system.

We created a target sentiment variable from ratings: positive (≥4 stars), negative (≤2 stars), and neutral (3 stars). We evaluated three text configurations: review text only, review title only, and combined text (title + body). We started with VADER (Valence Aware Dictionary and sEntiment Reasoner), a lexicon-based tool that generates compound scores (-1 to +1). Scores >0.05 classified as positive, <-0.05 as negative, else neutral. The combined text configuration achieved 83.8% accuracy but struggled with minority classes—neutral recall was only 2.79% and negative recall was 41.75%.

To improve minority class performance, we implemented TF-IDF vectorization (max_features=3000, bigrams, min_df=5, max_df=0.7) with logistic regression (SAGA solver, class_weight='balanced', 80-20 train-test split). This boosted accuracy to 85.56% came with a neutral recall increased from 2.79% to 76.93%, and negative precision rising from 60.72% to 88.41%. While positive recall decreased slightly (96.92% to 92.13%), the substantial gains in minority classes and overall balanced performance make this the superior model.

Model	Metric	Negative	Neutral	Positive	Accuracy
VADER	Precision	60.72%	15.59%	86.73%	**83.80%**
(Combined Text)	Recall	41.75%	2.79%	96.92%	
	F1-Score	49.48%	4.74%	91.54%	
TF-IDF + LogReg	Precision	88.41%	33.18%	97.06%	**85.56%**
(Combined Text)	Recall	43.02%	76.93%	92.13%	
	F1-Score	57.87%	46.36%	94.53%	

Sentiment Analysis Confusion Matrix Comparison



VADER *vs* *TF-IDF + Logistic Regression*

The confusion matrix comparison visualizes the improved neutral class detection, addressing VADER's critical weakness in ambiguous sentiment. We applied the logistic regression model to a secondary Ulta dataset (4,150 text reviews with no ratings), demonstrating the model's transferability. This validates our design for multi-source data ingestion—the system can process reviews from platforms like Reddit, blogs, or social media where numerical ratings do not exist, enabling continuous expansion of the product knowledge base. Finally, we developed a composite quality metric to identify representative reviews:

Quality Score = 0.3 × (length/500) + 0.4 × confidence + 0.3 × accuracy

- **Length:** Longer reviews provide richer insights
- **Confidence:** Max class probability from logistic regression

- **Accuracy:** Predicted sentiment matches rating-derived label

The top-5 reviews per product were aggregated with sentiment statistics (distributions, confidence scores) and rolled up to product level for the AI agent.

LLM Agent:

1. System Overview

This project implements a conversational AI agent for personalized skincare product recommendations using RAG. Given natural language queries such as "Show me vegan vitamin C serums for dullness under $50", it returns relevant products in 3-5 seconds.

2. Data & Feature Extraction

The system integrates 4,189 documents from four sources: 2,420 Sephora products, 1,521 medical Q&As, 248 ingredient profiles, and 2,333 product reviews with sentiment analysis. The system extracts 50+ structured features using pattern matching: 14 skin concerns (acne, aging, dryness, etc.), 18 active ingredients (retinol, vitamin C, niacinamide, etc.), 13 product attributes (vegan, cruelty-free, fragrance-free, etc.), 5 skin types, and sentiment metrics (rating, reviews, positive percentage). Each product is accompanied by a 300-500 character description incorporating all features.

3. System Architecture

Vector Store: Documents are embedded using BAAI/bge-large-en-v1.5 (1024-dim) and indexed in FAISS. Three document types: medical (CONDITION→QUESTION→ANSWER), ingredients (NAME→BENEFITS→COMPATIBILITY), products (all 50+ features in text + metadata). Total: 4,189 vectors, ~50MB, sub-second retrieval.

Intent Extraction: The EnhancedIntentExtractor parses queries using regex and keyword matching to extract budget (e.g., "under $50"→(0,50)), concerns ("pimples"→"acne"), category (serum/moisturizer), preferences (vegan/cruelty-free), ingredients (retinol/vitamin C), and query type (medical vs. product).

Agent Pipeline: SkincareAgent orchestrates: (1) Intent extraction, (2) FAISS top-5 retrieval, (3) Post-filtering on metadata, (4) Ranking by relevance + sentiment, (5) Mistral-7B response generation, (6) Output cleaning.

Technology Stack: Mistral-7B-Instruct-v0.2 (LLM), BAAI/bge-large-en-v1.5 (embeddings), FAISS (vector store), LangChain (orchestration), Gradio (web interface), Google Colab T4 GPU (deployment).

4. Evaluation Metrics & Results

We evaluated the system using three key metrics across diverse test queries: intent extraction accuracy, filter accuracy, and response time performance.

4.1 Intent Extraction Accuracy

Methodology: Tested the EnhancedIntentExtractor on 20 queries with known expected outputs. Measured accuracy of budget detection, concern identification, category extraction, preference flags, and ingredient recognition.

Example Test Cases:

- "Show me vegan products under $50" → Expected: budget=(0,50), preferences={'vegan': True}
- "Cruelty-free moisturizer for dry skin under $40" → Expected: budget=(0,40), preferences={'cruelty_free': True}, category='moisturizer', concerns=['dryness']
- "What causes acne?" → Expected: is_medical=True, concerns=['acne']

Results:

- Budget extraction: 95% accuracy (19/20 queries)
- Concern detection: 88% accuracy (matched at least one concern)
- Preference extraction: 92% accuracy (all requested attributes captured)
- Ingredient recognition: 94% accuracy
- Query type classification: 100% accuracy (medical vs. product)
- Overall Intent Extraction: 90% accuracy

4.2 Filter Accuracy

Methodology: For 15 product search queries, we verify that all returned products matched the extracted filters: budget compliance, concern addressing, preference matching, and ingredient presence in product metadata. Then, we calculate percentage of correct filtering operations

Example Test Cases:

- "Vegan under $30" → Verify all products have vegan=1 AND price≤$30
- "Fragrance-free for sensitive skin" → Verify all products have fragrance_free=1 AND for_sensitive=1

Results:

- Budget filter: 100% compliance (all products within specified range)
- Concern filter: 87% accuracy (products address at least one specified concern)
- Preference filter: 96% accuracy (products have all requested attributes)
- Ingredient filter: 91% accuracy (products contain requested ingredients)
- Category filter: 100% accuracy
- Overall Filter Accuracy: 85%

4.3 Response Time Performance

Methodology: Measured end-to-end query processing time for 25 queries of varying complexity. Timed from user input to final response display. Tested on Google Colab T4 GPU.

Query Categories:

- Simple queries: "What is retinol?" (1-2 filters)
- Medium queries: "Show me moisturizers" (3-4 filters)
- Complex queries: "Cruelty-free vitamin C serum for dullness under $50" (5+ filters)

Response Time Results	Performance Breakdown
- Avg. response time: 3.2s - Min. response time: 2.0s (simple ingredient queries) - Max. response time: 7.8s (complex multi-filter product searches) - 95th percentile: 6.5s - Timeout threshold: 60s (never reached in testing)	- Intent extraction: 0.1s - Vector retrieval: 0.8s - Post-filtering: 0.2s - LLM generation: 2.0s (GPU) / 5.0s (CPU) - Response cleaning: 0.1s

4.4 Dataset Statistics:
Products with sentiment: 2,333/2,420 (96.4%)
Sentiment distribution: 80% positive, 17% neutral, 3% negative
Average rating: 4.2/5 stars
Price range: $3-$945 (median $45)

Discussion & Limitations:

To achieve our goal of providing clear, accessible, and customizable skincare guidance, our project implements the following technical innovations: (1) Multi-domain knowledge integration enabling hybrid queries, (2) Automated extraction of 50+ features from unstructured text, (3) Sentiment-aware product ranking, (4) Natural language interface with 90% intent accuracy, (5) Zero-cost deployment using free-tier services. By implementing these innovations, we created a responsive AI agent, tailored to addressing user personal skincare needs.

However, with implementing such a complex model, we faced some tradeoffs and limitations. First, we were limited by our computational resources. While our model was trained using a Google Colab GPU, we faced challenges with the app deployment and accessing enough computational power and memory. Some approaches we attempted include Google Colab, HuggingFace Spaces, Google Vertex AI, and Google Cloud console. As we translated our model to a compatible user interface for deployment and scaling, we faced necessary practical tradeoffs.

Specifically, between Colab and HuggingFace deployment, we saw that each had key features (*see Appendix B for a detailed comparison*). First, considering architecture changes, each method has different approaches to their vector store. Colab builds on-the-fly (5-10 minutes), while HuggingFace uses a pre-built index (<30 seconds). For LLM computational

power, Colab has a GPU option (2-3 seconds), while HuggingFace API-only (5-8 seconds). They also differ in their data integration. Colab downloads data dynamically using Google Drive and Kaggle. HuggingFace uses static files. For their user interfaces, Colab uses gr.Blocks() custom UI, and HuggingFace uses simplified gr.ChatInterface(). They are also distinct in their code organization. Our Colab approach uses a single .ipynb notebook (~2,500 lines) and is more development focused. HuggingFace implements a modular app.py and agent_code.py (~270 lines), with more of a production focus.

The main optimization for HuggingFace is pre-building the vector store and including all data files in the repository, eliminating the 5-10 minute startup time and external API dependencies (Kaggle, Google Drive). This exercise demonstrated the practical constraints in deploying a complex LLM application from the development side and the implications for users.

Some other limitations our project faced include the training data. The quality of our recommendations depends on training dataset coverage and accuracy. Furthermore, due to the multifaceted nature of skincare, our AI agent is limited in its ability to treat conditions. Although the system can reason about medical conditions, it cannot and should not replace professional medical advice. We also did not have access to medical-level patient data and electronic health records, which could further contextualize our model's reasoning. We faced system issues due to a 60-second timeout limiting query complexity, a text-only interface preventing visual skin analysis, and challenges parsing complex negations.

As a result, future iterations of this project can implement advanced architectures and more complex medical datasets to improve the accuracy and personalization of our recommendations and to scale our app for public use.

Conclusion & Future Work:

In conclusion, SkinWise is an AI-powered skincare recommendation system, grounded in real data, that makes skincare discovery more transparent, accessible, and customizable. Going forward, there are several directions that we see this project developing. To make the model more robust to a wider range of user inputs, we would like to add more data to our model. Specifically, we aim to include product data from a wider range of companies. Currently our model only uses products from Sephora, and although Sephora is one of the largest makeup retailers, we aim to include product data from retailers such as Ulta or CVS to make our recommendation system more useful for a broader range of audiences. There are also many independent skin care retailers that do not sell their products in such stores. A longer term goal would be to add products from global markets that might not be sold on the U.S. version of the Sephora website for wider consumer reach. Furthermore, in the future, we aim to improve the user experience by allowing users to upload images of their skin to the interface. By integrating a computer vision model that is able to recognize skin concerns from the uploaded images, we would be able to incorporate visual skin analysis into the recommendation pipeline.

Appendix A.1: AI Disclosure

During the development of SkinWise, we utilized AI assistance (Claude and Gemini) as a supplementary tool to enhance our learning process. AI was employed specifically to troubleshoot technical errors during implementation, explore best practices for user interface design with Gradio, and guide us through the HuggingFace Spaces deployment process—a platform that was new to our team. All core development work, including system architecture design, feature extraction pipeline, agent logic, data integration, evaluation framework, and coding, was completed by our team. AI served as an educational resource and debugging assistant, similar to consulting documentation or Stack Overflow, while all conceptual design, implementation decisions, and final code were our original work.

Appendix A.2: Team Contributions

Gabriella Chu	***Gabrielle Park***	***Yelene Cisse***	***Kshamaa Suresh***
I collected, pre-processed, and feature engineered the Sephora product data that our system relies on to retrieve information for output upon user query. In addition, I prepped and performed text embeddings for the model to use. I also helped with deploying the model to a usable user interface, testing Gradio and React/Vite hosting.	I collected, cleaned, and feature engineered the medical data on dermatology conditions from two data sources. I also helped with deploying the model to a user interface by creating a prototype of the app using Flask, hosting on Google Cloud Run, and Vertex AI for LLM inference.	I preprocessed, deduplicated and cleaned the Reviews dataset for the Sentiment Analysis. I tested a total of 6 models (3 configurations for VADER, 3 for TF-IDF + LR). I developed a quality review score for reviews sampling to feed into our agent. I rolled up the data to product level and scored additional reviews data (Ulta). Temporary Gradio UI testing.	I developed the conversational AI agent architecture integrating 4,189 documents across medical, ingredient, and product domains. I implemented the complete feature extraction pipeline (50+ attributes), built the intent understanding system, designed the evaluation framework with three key metrics, and deployed the production application on HuggingFace Spaces with Gradio interface.

Appendix A.3: Important Links

- Github Repository with all of the code files: (The github readme also has the links to the other items): https://github.com/AML-GGYK/DermaLLM
- Project landing page: https://huggingface.co/spaces/kshamaasuresh/skincare-agent
- Video Presentation: https://drive.google.com/drive/folders/1p99c6gwCJzVLv41xK0c1kEPw9Kd0_vCb?usp=sharing
- Google Colab code file: https://drive.google.com/file/d/1YAom8NM-w857kqB-SeNHPIwpFlLKxxj4/view?usp=sharing
- Full drive folder (including data files): https://drive.google.com/drive/folders/1da77WlskEXwA0Rgegt-ayM938ODsQnno?usp=sharing
- Data Sources:
    - Sephora Dataset: https://www.kaggle.com/datasets/nadyinky/sephora-products-and-skincare-reviews
    - Dermatology Dataset: https://huggingface.co/datasets/Carxofa85/dermatology
    - Skincare Ingredient Dataset: https://www.kaggle.com/datasets/amaboh/skin-care-product-ingredients-inci-list

Appendix B: Colab vs. HuggingFace Deployment: Key Differences

Component	Google Colab	HuggingFace Spaces	Reason
LLM Loading	GPU: 4-bit quantized local OR CPU: Inference API	API only via ChatHuggingFace	No GPU in free Spaces
Vector Store Creation	Builds from scratch (5-10 min first run)	Pre-built index loaded (<30 sec)	Limited compute in Spaces
Embeddings Device	device="cuda" (GPU accelerated)	device="cpu" (CPU only)	No GPU available
Data Sources	Google Drive + Kaggle API (dynamic)	Local CSV files (static, bundled)	No Drive/API access
Sentiment Format	Pickle file: reviews_prod_lvl.pkl	CSV file: sentiment_analysis.csv	Portable format
Interface	gr.Blocks() with custom layout	gr.ChatInterface() (simplified)	Cleaner UX
Prompt Format	String template with [INST] tags	ChatPromptTemplate (system/human)	Chat model preference
Response Time	2-3s (GPU local) 5-8s (API fallback)	5-8s (API only)	No local GPU
Deployment	Temporary 72-hour share link	Permanent Spaces URL	Production vs Dev
Code Structure	Single notebook (~2,500 lines)	Modular: app.py + agent_code.py	Better organization
Retriever K	k=5 (top-5 docs)	k=8 (top-8 docs)	More context for responses
Max Tokens	max_new_tokens=400 (concise)	max_new_tokens=1500 (detailed)	More complete answers

Summary: Colab is optimized for development with GPU acceleration, while HuggingFace is optimized for production deployment with pre-built components.

Appendix C: LLM Agent Workflow