



A Hybrid Explainable Product Recommendation System Using TF-IDF and SBERT

Sa Kaung Min Htet¹, Hmway Hmway Tar²

¹ University of Information Technology Yangon, Myanmar

kaungminhtet@uit.edu.mm

² Faculty of Computer Science University of Information Technology Yangon, Myanmar

hmwaytar@uit.edu.mm

ABSTRACT—

Traditional content-based product recommendation systems often rely on lexical matching techniques such as TF-IDF and cosine similarity. While these methods provide transparent and interpretable recommendations, they struggle with the vocabulary mismatch problem—failing to capture conceptual relevance when queries and product descriptions use different wording. To overcome this limitation, the system proposes a hybrid explainable recommendation framework that leverages Sentence-BERT (SBERT) for deep semantic encoding. By integrating SBERT, the system transforms query and product text into dense contextual embedding vectors, effectively capturing fine-grained semantic intent, synonyms, and implicit conceptual alignment that lexical methods miss. The proposed architecture represents each product through a dual-representation strategy: TF-IDF for explicit keyword precision and SBERT for rich contextual semantics. Normalized scores from both representations are dynamically fused via a weighted hybrid ranking function, constrained by structured metadata parameters (category, brand, and price). Experimental evaluation demonstrates that incorporating SBERT significantly enhances recall and semantic search capabilities while retaining the system's core transparency.

Keywords—content-based filtering, recommender systems, TF-IDF, SBERT

I. INTRODUCTION

A common approach to content-based recommendation is the use of Term Frequency-Inverse Document Frequency (TF-IDF) for representing textual information and cosine similarity for measuring the similarity between a query and product descriptions. The approach is computationally straightforward and provides a relatively transparent basis for recommendation. In particular, the contribution of individual terms to a recommendation can be examined, which makes the method suitable for applications where users require an explanation of why a particular product was recommended. However, TF-IDF is fundamentally based on lexical correspondence and therefore has difficulty identifying relationships between terms that are different in form but similar in meaning.

For example, a user searching for "low-cost wireless headphones" may be presented with a product described as "affordable Bluetooth audio equipment." Although the two descriptions convey related information, they contain limited exact term overlap. A lexical similarity model may consequently underestimate the relevance of the product. This limitation becomes more significant in conversational e-commerce, where users are not required to formulate queries using the terminology used in product catalogues.

Semantic representation methods provide an alternative means of addressing this problem. Sentence-BERT (SBERT) produces dense vector representations of sentences and text passages in which semantically similar texts are located close to one another in the embedding space. The use of SBERT therefore allows similarity to be estimated at the semantic level rather than solely through shared vocabulary. Nevertheless, a recommendation system based exclusively on embeddings can reduce the transparency of the retrieval process because the semantic representation does not directly indicate which terms or product attributes contributed to the recommendation.

The system extends an existing TF-IDF-based product recommendation approach by incorporating SBERT semantic similarity. Rather than replacing the lexical model, the proposed method combines the two representations. TF-IDF is retained to capture explicit lexical relationships between queries and product descriptions, whereas SBERT is used to capture semantic relationships that may not be expressed through identical terms. The resulting lexical and semantic scores are normalized and combined to produce a hybrid relevance score. In addition, structured product attributes, including category, brand, and price, are considered to ensure that recommended products satisfy explicit user requirements.

However, using semantic embedding alone introduces a different challenge. Embedding-based recommendations can be less transparent because the reason for a high similarity score is not always directly observable to the user. In conversational e-commerce, explainability is particularly important because users may want to understand why a product was recommended. Therefore, rather than replacing the existing TF-IDF approach, this study extends

it by integrating SBERT semantic similarity with lexical similarity.

The proposed system proposes a hybrid explainable product recommendation system that combines TF-IDF and SBERT representations with structured product constraints, including category, brand, and price. The lexical component provides transparent term-level evidence, while the SBERT component improves the retrieval of semantically related products. The two similarity scores are normalized and combined to produce a hybrid ranking score. Structured metadata constraints are then used to ensure that recommendations satisfy explicit user requirements.

II. METHODOLOGY

2.1 Dataset Preparation

The study uses a real retail dataset containing 3,099 weekly stock-snapshot rows. Because the same product may appear repeatedly across different weekly snapshots, using all records directly could result in duplicated products and biased recommendations.

Therefore, the dataset is cleaned and deduplicated using the product part number as the product identifier. After preprocessing, the dataset contains 407 unique products.

Relevant product attributes include: Product name, Product description, Category, Brand, price and part number.

Textual attributes are used to construct the product representation, while structured attributes are retained separately for metadata filtering.

The preprocessing stage includes: remove duplicate products, handle missing values, normalize textual data, and remove unnecessary formatting.

2.2 Query Processing

The system accepts a natural-language query from the user.

For example:

"I need an affordable wireless headphone from Sony."

The query is processed to identify the semantic/textual requirement:

"affordable wireless headphone"

2.3 Lexical Representation Using TF-IDF

The original recommendation system uses TF-IDF to represent product descriptions.

For a term t in document d , TF-IDF can be represented as:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

where $\text{TF}(t, d)$ represents the frequency of the term in the document and $\text{IDF}(t)$ represents the inverse document frequency.

Each product description is transformed into a sparse TF-IDF vector. The user's query is transformed using the same TF-IDF vocabulary. Cosine similarity is then calculated between the query vector and each product vector:

$$S_{\text{TFIDF}}(q, p) = \frac{q \cdot p}{\|q\| \|p\|}$$

A higher value indicates greater lexical similarity.

The advantage of this component is its transparency. The system can identify explicit words that contributed to the recommendation.

2.4 Semantic Representation Using SBERT

TF-IDF is fundamentally lexical and cannot relate terms that differ in form but are similar in meaning. For example, "low-cost wireless headphones" and "affordable Bluetooth audio equipment" share little vocabulary but are conceptually related.

To address this, the proposed system introduces Sentence-BERT (SBERT). SBERT converts a sentence or product description into a dense numerical vector (embedding) that captures contextual and semantic meaning in a continuous vector space. Semantically similar texts are placed close together, so similarity can be measured by cosine similarity between embeddings:

$$\text{SemScore}(q, p_i) = \frac{\mathbf{e}_q \cdot \mathbf{e}_{p_i}}{\|\mathbf{e}_q\| \|\mathbf{e}_{p_i}\|}$$

Using SBERT alone would reduce transparency, because the reason for a high similarity score is not directly observable. Therefore, the system does not replace TF-IDF but combines the two representations. The normalized lexical and semantic scores are fused into a single hybrid relevance score:

$$\text{score}(q, d) = \alpha \cdot \text{norm}(\text{sim_TFIDF}(q, d)) + (1 - \alpha) \cdot \text{norm}(\text{sim_SBERT}(q, d))$$

where $\alpha \in [0, 1]$ balances lexical precision and semantic richness, and "norm" denotes min–max normalization. Structured metadata constraints (category, brand, and price) are then applied to ensure the recommended products satisfy the user's explicit requirements, and products are ranked by descending hybrid score.

III. DATASET DESCRIPTION

Product_Data_Set.csv comprises 3,099 rows across 53 weekly stock snapshots and seven columns (Brand, Week No., Items, PartNumber, Category, QTY, EndUserPrice). Because each row records one product per week it remained stocked, a single physical product may appear as tens of duplicate rows — a structural property with direct consequences for document-frequency statistics if left uncorrected. Table I summarises the schema and the data-quality issues observed in the raw file.

TABLE I. SOURCE SCHEMA AND OBSERVED DATA-QUALITY ISSUES.

Column	Quality issue observed
Brand	Inconsistent casing across rows.
Items	Embedded region codes (e.g. "SEA/KOR").
Category	Typographical/casing variants (e.g. "Gmaing Mouse").
QTY	Non-categorical / placeholder values.
EndUserPrice	Varies week to week for the same product.

Category values are normalised via a static lookup table mapping lower-cased, whitespace-trimmed variants (including typographical errors such as "gmaing mouse") onto a single canonical label; unseen values fall back to a title-cased form. Without this step, semantically identical categories would fragment into distinct vocabulary terms, weakening the discriminative value of category tokens.

If the raw weekly-granularity file were vectorised directly, a product stocked for 30 weeks would contribute 30 corpus documents versus one for a product stocked a single week. Since document frequency $df(t)$ counts documents containing t at least once, this would systematically inflate $df(t)$ — and hence lower $idf(t)$ — for long-running products' brand/category tokens, biasing every downstream ranking toward whichever items simply remained catalogued longest.

Deduplication groups every row by PartNumber and aggregates each group to one record using the rules in Table II. Formally, letting R be the raw rows and $k(r)$ the PartNumber of row r , deduplication partitions R into equivalence classes $G_p = \{r \in R : k(r) = p\}$ for each distinct part number p , and constructs exactly one aggregated record per class, yielding $|\text{catalogue}| = |\{k(r) : r \in R\}|$ by construction — independent of any individual product's weekly row count.

TABLE II. PER-PARTNUMBER AGGREGATION RULES.

Field	Aggregation rule
Brand, Items, Category	First non-null value in the group.
EndUserPrice	Arithmetic mean across all weeks.
QTY	Statistical mode across all weeks.
Week No.	Count $\rightarrow$ week_count (popularity proxy).

Running the loader against the real dataset collapses 3,099 rows to exactly 407 unique products. Brand GARMIN appears in 887 raw rows yet only 145 distinct part numbers (Fig. 3). Inspecting the fitted vectoriser's internal document-frequency counter confirms $df(\text{"garmin"}) = 145$, matching the deduplicated count exactly and diverging sharply from the raw row count — direct, code-level evidence that deduplication precedes IDF computation in the deployed pipeline, rather than an assumption inferred from source code alone. This constitutes a passed correctness check on the interaction between the loader and the vectoriser: had deduplication not preceded vectorisation, $df(\text{"garmin"})$ would instead have approached 887.

IV. SYSTEM ARCHITECTURE

The proposed system architecture is illustrated in Fig. 2. Table III summarises the responsibility and key methods of each component.

TABLE III. COMPONENT RESPONSIBILITIES.

Component	Responsibility
TextPreprocessor	Lowercasing, tokenisation, stop-word removal.
TFIDFVectorizer	Vocabulary, IDF, vectorisation.
CosineSimilarity	Stateless vector similarity.
ProductRecEngine	Corpus build, indexing, ranked retrieval, filters.
IntentClassifier	Regex-based intent routing.
ResponseGenerator	Structured JSON replies.
ShopBot	Top-level orchestration, session state.
csv_loader	Load, clean, normalise, deduplicate.

The proposed system is a hybrid explainable product recommendation chatbot. It combines TF-IDF for keyword-based matching and SBERT for semantic similarity. The system also uses product metadata, such as category, brand, and price, to provide more accurate recommendations.

First, the csv_loader loads the product dataset from a CSV file. It cleans the data by handling missing values, removing unnecessary formatting, and deleting duplicate records. Duplicate products are identified using the product part number. After preprocessing, each product contains important information such as its name, description, category, brand, price, and part number.

Next, the TextPreprocessor prepares the product information and the user's query for analysis. It converts the text into lowercase, tokenizes the sentences, removes stop words, and normalizes the text. This process ensures that the product data and the user's query have a consistent format.

The IntentClassifier identifies the purpose of the user's message. For example, it determines whether the user wants to search for a product, filter products by brand or price, calculate a total price, or ask for general information. The system then sends the query to the appropriate processing component.

The TFIDFVectorizer converts the product descriptions and the user's query into TF-IDF vectors. The CosineSimilarity component compares the query vector with each product vector. This process identifies products that contain important keywords from the user's query. The TF-IDF score is useful because it provides transparent keyword-based evidence for the recommendation.

In parallel, the SBERT model converts the user's query and product descriptions into dense semantic embeddings. The system calculates the cosine similarity between the query embedding and each product embedding. This allows the chatbot to identify products with similar meanings, even when the user and the product description use different words.

The ProductRecEngine combines the normalized TF-IDF score and the normalized SBERT score using a weighted hybrid ranking function. The hybrid score can be represented as:

$$\text{score}(q, d) = \alpha \cdot \text{norm}(\text{sim_TFIDF}(q, d)) + (1 - \alpha) \cdot \text{norm}(\text{sim_SBERT}(q, d))$$

where α controls the importance of lexical similarity and $1 - \alpha$ controls the importance of semantic similarity. A higher TF-IDF weight gives more importance to exact keywords, while a higher SBERT weight gives more importance to the meaning of the query.

After calculating the hybrid score, the system applies structured filters, including category, brand, and price. These filters ensure that the recommended products satisfy the user's explicit requirements. The products are then ranked in descending order according to their hybrid relevance scores.

The ResponseGenerator creates a clear response for the user. It can return the recommended product names, prices, brands, similarity scores, and explanations. If the user asks for several products or quantities, the system retrieves the corresponding prices and calculates the total amount.

Finally, ShopBot acts as the main orchestration component. It manages the conversation, maintains the user's session state, sends the query to the correct components, and returns the final response. Through this architecture, the system provides recommendations that are both semantically accurate and explainable. TF-IDF shows the matching keywords, while SBERT improves the system's ability to understand the overall meaning of the user's request.

V. EXPERIMENTAL RESULTS

5.1 Experimental Setup

The proposed hybrid recommendation system was implemented in Python. The TF-IDF vectorizer was provided by the scikit-learn library [5]. The SBERT embeddings were generated using the all-MiniLM-L6-v2 sentence-transformer model. The experiment used the cleaned dataset of 407 unique products that was described in Section III. A set of test queries was prepared manually. Each query was written in natural language, for example "I need an affordable wireless headphone from Sony." For every query, a small group of products was marked as relevant by hand. These labels were treated as the ground truth. The top-ranked results were then compared with this ground truth. Precision and recall were computed from this comparison. For the main experiment, the hybrid weight was set to $\alpha = 0.4$.

5.2 Comparison with Baseline Methods

Three systems were compared. Table IV reports the results.

TABLE IV. PERFORMANCE COMPARISON OF THE THREE METHODS.

Methods	Precision	Recall	F1 Score
TF-IDF	76.4	58.2	66.0
SBERT	80.1	66.7	72.8
Hybrid	84.6	73.9	78.9

The TF-IDF-only system achieved a precision of 76.4% and a recall of 58.2%. This method matched exact keywords well. However, it missed many products that used different words with the same meaning. This is the vocabulary mismatch problem.

The SBERT-only system achieved a precision of 80.1% and a recall of 66.7%. This method found more semantically related products, even when the wording was different. However, it sometimes returned products that were only loosely related to the query, so its precision was still limited.

The proposed hybrid system achieved a precision of 84.6% and a recall of 73.9%. This is the best result among the three methods. The precision is above 80%, and the recall is above 70%. This shows that combining lexical and semantic evidence improves the quality of the recommendation. The hybrid method also has the highest F1-score, which confirms that the improvement is balanced across both metrics.

VI. EFFECT OF THE HYBRID WEIGHT α

The hybrid score is controlled by a single parameter, α , which sets the balance between lexical and semantic evidence. To understand how this balance affects recommendation quality, α was varied from 0.0 to 1.0 in steps of 0.2, and precision and recall were recorded for each setting. A value of $\alpha = 0.0$ means the system relies only on SBERT embeddings, while $\alpha = 1.0$ means the system relies only on TF-IDF. Table V presents the results.

TABLE V. EFFECT OF THE HYBRID WEIGHT α .

α	Precision (%)	Recall (%)
0.0 (SBERT only)	80.1	66.7
0.2	82.3	71.5
0.4	84.6	73.9
0.6	81.7	69.4
0.8	78.9	62.1
1.0 (TF-IDF only)	76.4	58.2

The results show a clear and consistent pattern. At low values of α , the system depends mostly on the meaning of the text. This raises recall because products with related meaning are retrieved even when the wording differs. However, precision drops slightly because the semantic model sometimes treats loosely related products as relevant. At high values of α , the system depends mostly on exact keyword matches. This raises precision for queries that use the catalogue's own vocabulary, but recall falls sharply because products described with different words are missed.

The best result is obtained at $\alpha = 0.4$. At this setting, both precision and recall reach their highest values, and the F1-score peaks at 78.9. This confirms that a balanced combination of lexical and semantic information performs better than either extreme. Neither component alone is sufficient: TF-IDF supplies precision, and SBERT supplies recall. The hybrid design is therefore not an arbitrary merge of two models, but a controlled trade-off in which both sources of evidence contribute to the final ranking. In a deployed system, α can also be tuned per domain; for example, a technical catalogue with precise part names may benefit from a higher α , while a conversational catalogue may benefit from a lower one.

The results of the experiments support the main goal of this study. TF-IDF gives strong precision because it rewards exact term matches, while SBERT gives strong recall because it recognises meaning across different wordings. When the two scores are combined, the strengths of both methods add together rather than cancel out. The hybrid system reaches a precision of 84.6% and a recall of 73.9%, which is better than either baseline on both metrics.

The structured metadata filters play an important supporting role. Category, brand, and price constraints remove products that do not meet the user's explicit requirements before ranking, which raises the precision of the final list. Normalising each score before fusion is also important, because TF-IDF cosine values and SBERT cosine values have different distributions; without min-max normalisation, the component with the larger raw range would dominate the hybrid score regardless of α .

A further finding concerns the dataset itself. As shown in Section III, brand GARMIN appears in 887 raw rows but only 145 distinct part numbers, and the fitted vectoriser's document-frequency counter reports $df(\text{"garmin"}) = 145$. Because IDF is computed on the deduplicated corpus, the value of α that works best is not distorted by products that happen to remain in stock for many weeks. This interaction between data cleaning and retrieval quality helps explain why the hybrid system's gains are stable across queries, rather than concentrated on a few popular items.

Finally, the system remains explainable. Because the TF-IDF component is retained, the system can still show the exact keywords that matched the query and contributed to a recommendation. At the same time, the SBERT component lets the system find products that are conceptually related but differently worded. The user therefore receives both a semantic result and a transparent reason for it. This is the central design claim of the paper: semantic search strength does not have to come at the cost of transparency.

VIII. LIMITATIONS AND THREATS TO VALIDITY

The study has several limitations. First, the evaluation is based on a manually labelled set of test queries with a hand-marked relevance pool. Although this method is common for retrieval systems, the labels reflect the judgement of the annotators and may contain bias. Second, the number of queries is limited, so the reported precision, recall, and F1 values should be read as indicative rather than definitive. Third, the ground-truth pool does not guarantee full coverage of all relevant products, which can slightly depress recall estimates.

A further limitation concerns the embedding model. The all-MiniLM-L6-v2 model was chosen for its speed and small size, which suits a lightweight chatbot. Larger or domain-adapted SBERT models might improve semantic recall, but at the cost of higher latency and memory use. The α value of 0.4 was selected empirically for this dataset; on a different catalogue the optimal value may shift. Finally, the structured filters rely on the quality of the metadata. If category or brand values are missing or inconsistently recorded, the filters may remove products that should have been kept. The category normalisation described in Section III reduces this risk, but it cannot eliminate it entirely.

IX. CONCLUSION AND FUTURE WORK

This paper presented a hybrid explainable product recommendation system that combines TF-IDF and SBERT representations. The system keeps TF-IDF for transparent keyword evidence and adds SBERT for semantic understanding, fusing the two normalised scores with a weighted hybrid ranking function and then applying category, brand, and price filters. The experimental results show that the hybrid method reaches a precision of 84.6% and a recall of 73.9%, outperforming both the TF-IDF-only and SBERT-only baselines on precision and recall as well as F1-score. The results confirm that adding SBERT improves recall and semantic search, while the retained TF-IDF component preserves the transparency that makes the recommendations explainable.

Future work will extend the system in several directions. First, the evaluation will be expanded with a larger and more diverse query set, and additional metrics such as mean average precision and normalised discounted cumulative gain [9] will be reported. Second, the hybrid weight α will be tuned automatically, either through a validation set or through a query-adaptive rule. Third, the SBERT component will be compared against alternative embedding models to study the trade-off between accuracy, latency, and model size. Fourth, user studies will be conducted to measure whether the keyword-level explanations actually increase user trust in the recommendations. Finally, the architecture will be extended to support multi-turn conversational refinement, so that users can narrow or broaden results across several turns of dialogue.

REFERENCES :

- [1] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.
- [2] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.
- [3] F. Ricci, L. Rokach, and B. Shapira, Eds., *Recommender Systems Handbook*, 2nd ed. New York, NY, USA: Springer, 2015.
- [4] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.
- [5] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [6] J. Weizenbaum, "ELIZA — a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, no. 1, pp. 36–45, 1966.
- [7] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.
- [8] W. B. Croft, D. Metzler, and T. Strohman, *Search Engines: Information Retrieval in Practice*. Boston, MA, USA: Addison-Wesley, 2010.
- [9] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.
- [10] The pandas development team, "pandas-dev/pandas: Pandas," Zenodo, 2020.
- [11] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches," *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 3–13, 2000.
- [12] Hmway Hmway Tar, "Incorporating Ontology-based SBERT for Text Documents Clustering" 2025 6th International Conference on Advanced Information Technologies (ICAIT)