Optimized Decision-Making in Physical Retail Stores Through an Adaptive Hybrid System

A. De Paola, P. Ferraro, S. Imperiale, G. Lo Re

Proceedings of the 22nd International Conference on Distributed Computing and Artificial Intelligence (DCAI 2025)

DOI: https://doi.org/10.1007/978-3-032-04160-9_27

Publisher: Springer

Optimized Decision-Making in Physical Retail Stores through an Adaptive Hybrid System

Alessandra De Paola[1], Pierluca Ferraro[1], Sergio Imperiale[1], and Giuseppe Lo Re[1]

Department of Engineering, University of Palermo, Palermo, Italy
alessandra.depaola@unipa.it, pierluca.ferraro@unipa.it, sergio.imperiale@unipa.it, giuseppe.lore@unipa.it

Abstract. Physical retail stores have to deliver product recommendations to enhance customer experience, and accurately forecast sales to optimize inventory levels. While traditional recommender systems typically rely on explicit customer preferences, behavior similarities, and popularity metrics, such systems encounter limitations in offline retail contexts due to insufficient explicit customer data and the complexity of customer-store interactions. This paper introduces a novel hybrid solution that combines various recommendation strategies with advanced predictive analytics, including Recurrent Neural Networks and statistical modeling approaches. This approach is designed to improve both the precision of product recommendations and the accuracy of sales predictions. The system was extensively evaluated on a publicly available real-world dataset, using metrics such as Precision@K and NDCG@K to assess its performance. The results demonstrate that the proposed hybrid method successfully balances personalization and accuracy, offering a robust solution for optimized decision-making in physical retail settings.

Keywords: Recommender Systems · Predictive Analytics · Physical Retail Environments.

1 Motivation and Related Work

Recommender systems intelligently tailor suggestions by analyzing user preferences or patterns inferred from similar individuals. These systems are commonly used in digital environments like e-commerce platforms, streaming services, and social networks, where they increase engagement by adapting services to individual behavior. User preferences can be collected explicitly, for instance through ratings, or implicitly, by observing interactions with products [1]. In physical retail settings, they act as decision-support tools not only for customers, but also for store managers. Indeed, traditional decision-making in these contexts often relies on manual rules or manager intuition, which may lead to inefficient inventory planning and missed opportunities for personalization. In addition, the operational constraints of physical retailing (including limited shelf space, perishable goods, and unpredictable demand patterns) increase the difficulty of

matching supply to customer needs. Recommender systems (RS) in this setting can help address these issues by suggesting relevant alternatives when items are unavailable or by promoting soon-to-expire products. However, despite their promise, RS face several limitations. The effectiveness of their suggestions is directly tied to the quantity and density of interaction data, which is often sparse in non-digital environments [2]. For example, the so-called cold-start problem limits the system's ability to generate accurate suggestions when recommending new or rarely purchased items [3]. Standard approaches to mitigate this include collaborative filtering (based on user similarity), content-based methods (relying on item attributes), and hybrid models that combine both. These conventional methods, although widely studied, are often ill-suited to the dynamics of offline retail. Data scarcity, difficulties in tracking user behavior [4, 5], and the need for real-world testing complicate their deployment. Moreover, traditional data-driven decision strategies frequently overlook temporal trends and fail to adapt in real time [6], resulting in missed opportunities to optimize stock levels or customize offers. Forecasting models provide a complementary capability by anticipating future demand based on historical sales data. Recurrent Neural Networks (RNNs), well suited to sequence modeling, are commonly used to detect evolving patterns in time-series data [7]. These are often supported by statistical models that capture broader trends and relationships within the data, enabling more robust and interpretable insights for store planning [8].

This paper presents an integrated hybrid system that addresses the complex demands of physical retail by combining recommendation and forecasting functionalities. The recommendation module exploits content-based filtering, user-based collaborative filtering, and popularity-based ranking to generate personalized suggestions. These are delivered via a mobile app that interacts with IoT infrastructure in-store, allowing real-time personalization based on contextual data. In parallel, the forecasting component equips store managers with predictive tools that leverage both RNNs and statistical models. While RNNs analyze temporal sales dynamics to anticipate product demand, statistical models reveal overarching trends that support high-level inventory and pricing decisions. We evaluate this system using a publicly available dataset, measuring performance with metrics such as Precision@K and NDCG@K. Results show the system's ability to effectively address the specific challenges of offline retail and support both personalized experiences and strategic planning.

The remainder of this paper is structured as follows. Section 2 introduces the system architecture. Section 3 describes the experimental setup and presents the results. Section 4 concludes with directions for future work.

2 Proposed Hybrid System

The proposed system is built to achieve two goals: providing personalized recommendations to enhance customer satisfaction and optimizing store performance through reliable forecasts. Unlike online platforms, physical stores typically lack mechanisms to collect explicit user feedback, such as ratings, on individual items.

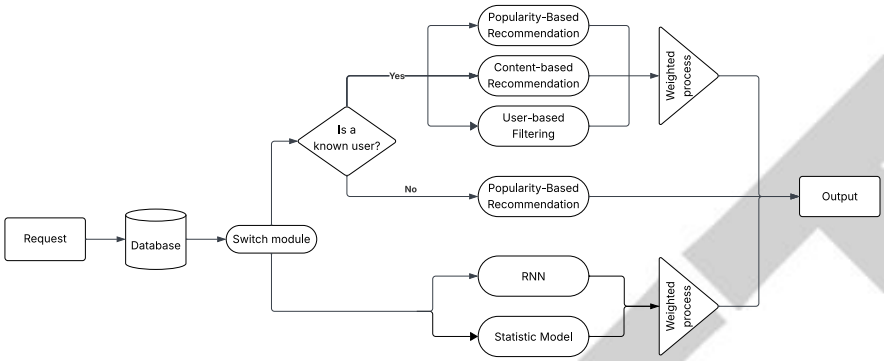

Fig. 1: The architecture of the proposed hybrid system.

As highlighted in [9], customer needs are often inferred through implicit feedback. For example, repeated purchases of the same product or consistent buying patterns may reveal preferences that would normally be obtained through direct input.

The system operates in two distinct phases: a recommendation phase, which produces tailored item suggestions for individual customers [10], and a forecasting phase, which anticipates product demand based on historical data. These two phases are described in detail in the following sections. The recommendation module leverages three established paradigms, i.e., content-based filtering, user-based collaborative filtering, and popularity-based ranking. Each model independently generates a ranked list of items, and their outputs are subsequently aggregated into a unified recommendation list delivered to the user's mobile device, as shown in Figure 1. The algorithm relies on five key inputs: the user identifier, the set of ratings or feedback associated with that user, the number of items to be recommended, historical sales data, and detailed metadata for each item in the store's inventory. Each module processes this information to generate a personalized recommendation list, and these outputs are then combined using a weighted strategy to produce the final set of suggested products.

2.1 Switch module

The *switch module* is responsible for routing each incoming request to the appropriate subsystem and determining whether it should be handled by the recommendation or prediction component. If the request is for the recommendation component, the switch module first checks for historical user feedback. If previous interactions exist, the module formats the input data according to the specific requirements of the three recommendation strategies used by the system: user-based, content-based, and popularity-based models. In the absence of prior feedback, the module automatically defaults to the popularity-based approach, which operates independently of user history. This conditional logic is especially important when dealing with new or infrequently active users, for whom insuffi-

Algorithm 1: User-Based module

```
Initialize a structure to store scores for each item
for each item in the list do
    Retrieve user ratings for the item
    Compute a weighted average of the ratings using user similarity as weights
    Store the resulting score for the item
end for
Convert the structure into a sorted list of items based on their scores
return the final ranked list of recommended items
```

cient data prevents reliable identification of individual preferences or behavioral patterns. By dynamically adjusting to the availability of user information, this module ensures that the system continues to deliver useful recommendations even when personalization is not possible.

2.2 Popularity-Based module

The *Popularity-based* module, as mentioned before, can be used as a robust fallback mechanism that can generate recommendations using only aggregated item sales in absence of user-specific data.

The algorithm begins by analyzing the store's sales records and dividing them into two subsets: one representing the complete sales data, and another capturing only recent sales from the past week. This dual perspective allows the model to account for both long-standing popularity and short-term demand fluctuations.

To merge these two sources of information, the system applies a weighted ranking formula inspired by the method introduced in [11]:

$$O_{pb}(i) = \lambda \cdot P_{History}(i) + (1 - \lambda) \cdot P_{LastWeek}(i), \tag{1}$$

where i represents an individual item, and $\lambda \in [0, 1]$ is a tunable parameter that controls the relative influence of long-term versus recent sales. $P_{History}$ and $P_{LastWeek}$ represent the ranking positions of item i based on full and recent sales data, respectively. By adjusting *lambda*, the module can prioritize items that have been consistently popular or, alternatively, highlight recent trends.

2.3 User-based module

The *User-based* module relies on collaborative filtering and begins by computing user-to-user similarity scores. The measure used to assess similarity is cosine similarity, defined as follows:

$$cos(\overline{X}, \overline{Y}) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2} \sqrt{\sum_{i=1}^{d} y_i^2}}, \tag{2}$$

where $\overline{X} = (x_1, \ldots, x_d)$ and $\overline{Y} = (y_1, \ldots, y_d)$ are d-dimensional vectors, and x_i and y_i denote the i-th components of each vector, respectively.

Algorithm 2: Content-Based module

```
Initialize a structure to store similar products with their scores
for each item in the list of top-ranked products do
    Retrieve the most similar products for the item, along with their similarity scores
    Update the structure, keeping the highest score for each similar product
end for
Convert the structure into a sorted list of similar products based on score
Remove any duplicates
return the final ranked list of recommended products
```

Applying this measure across all user pairs produces a square similarity matrix of size n x n. Each entry in the matrix reflects the similarity between two users, ranging from from -1 (opposing preferences) to 1 (identical preferences).

Once similarity scores are established, the module computes a weighted average in which ratings from users with higher similarity scores contribute more heavily to the final recommendation score for each item, allowing the model to suggest products likely to match the target user's tastes Algorithm 1 shows this process in detail.

2.4 Content-based module

Content-based recommendation systems are particularly effective in scenarios where items are described through structured attributes or detailed textual metadata. Unlike collaborative methods, these systems rely exclusively on a user's own interaction history to infer preferences, matching the user with items that share characteristics with those previously appreciated. A central component of this module is the the *Tf-Idf* (*Term Frequency - Inverse Document Frequency*) function [11], a numerical representation used to measure the importance of a term within a specific document relative to a larger corpus. It is composed of two factors; the *Term Frequency* (*Tf*) quantifies how often a term appears in a single document:

$$tf_{i,j} = \frac{n_{i,j}}{d_j}, \tag{3}$$

where $n_{i,j}$ is the number of occurrences of term i in document j, and d_j is the total number of terms in that document. The *Idf* is defined as:

$$idf_i = \log(\frac{n}{n_i}), \tag{4}$$

where n is the total number of documents, and n_i is the number of documents containing term i. The final *Tf-Idf* score is obtained by multiplying these two components:

$$(tf\text{-}idf)_{i,j} = tf_{i,j} \times idf_i \tag{5}$$

After preprocessing, the *Tf-Idf* function is applied to convert the textual information into weighted feature vectors. Each word receives a weight reflecting its relative significance within the product description and across the entire catalog. These vectors are then used to compute item-to-item similarity scores.

To generate recommendations, the system identifies items most similar to those ranked highly by the user. Algorithm 2 provides an overview of this process.

2.5 Recurrent Neural Network module

The Recurrent Neural Network (RNN) module performs sales forecasting by modeling temporal patterns in historical data. It is implemented using Long Short-Term Memory (LSTM) units, which address the vanishing gradient problem common in standard RNNs and are well suited to learning both short-term and long-term dependencies in sequential data [7].

The input to the model is a *tensor* of time-ordered sequences, where each step may represent features such as past sales, customer counts, or contextual indicators. Multiple stacked LSTM layers process this sequence hierarchically, extracting increasingly abstract temporal features. To prevent overfitting, *dropout regularization* [12] is applied between layers during training.

The final layer of the network is a fully connected layer that maps the hidden state from the last LSTM time step to a scalar output, representing the predicted sales value for the next time step:

$$O_{RNN} = W_{out} h_T + b_{out}, \tag{6}$$

where W_{out} and b_{out} are the weights and bias of the output layer, and h_T is the hidden state at the final time step.

2.6 Statistical module

The *Statistical module* focuses on analyzing past sales data to produce forecasts through machine learning models that exploit temporal patterns and long-term trends. Its design leverages structured features derived from historical observations to capture recurring behaviors and anticipate future demand. For this purpose, the module employs *Gradient Boosting*, an ensemble technique that incrementally builds a strong predictor by combining multiple weak learners such as decision trees into a unified model [13]. In the context of retail forecasting, Gradient Boosting Decision Trees have shown superior performance over traditional statistical methods, offering enhanced accuracy and better adaptability to complex patterns in the data. Their ability to handle heterogeneous feature sets, combining categorical variables, time-based signals, and numerical trends makes them particularly effective for modeling retail demand.

Although gradient boosting is not inherently sequential, it can be adapted effectively to time series forecasting by reformulating the problem as a regression task. Historical sales are transformed into input-output pairs, where the inputs consist of engineered features such as temporal indicators and lagged values, while the output represents the expected sales at a given future time point. This transformation enables the model to learn mappings between past behaviors and future outcomes. The process starts with a baseline model, and new trees are

then added in sequence, each one trained to minimize the residual errors produced by the ensemble so far. At each step, the algorithm fits a new decision tree to the gradient of the loss function, focusing the learning effort on the least accurate predictions. This iterative procedure continues until either a predetermined number of trees is reached or further additions yield minimal gains in accuracy, thus balancing predictive power with computational efficiency. The final output is a weighted sum of all the trees in the ensemble, where each tree's contribution is scaled by the learning rate:

$$O_{stat} = F_M(x_i) = \sum_{m=1}^{M} \eta \cdot h_m(x_i), \tag{7}$$

where O_{stat} is the predicted output, $F_M(x_i)$ denotes the gradient boosting model after M iterations, $h_m(x_i)$ is the prediction from the m-th base learner given input x_i and η is the learning rate that controls the influence of each learner in the final result [13].

2.7 Weight Process Module

The *Weight Process Module* serves as a key integration point for both the recommendation and prediction subsystems, enabling the combination of multiple model outputs. Its role is to assign specific weights to each contributing module based on optimization goals, contextual constraints, or application-specific requirements, producing a single, aggregated output for each subsystem. In the *recommendation subsystem*, this module merges the outputs from the popularity-based, user-based, and content-based recommenders. By doing so, it balances global popularity trends, user similarity scores, and item-level content features, generating more personalized suggestions. The final recommendation score for each item i is computed through a weighted linear combination:

$$O_{rs}(i) = \alpha \cdot O_{pb}(i) + \beta \cdot O_{ub}(i) + \gamma \cdot O_{cb}(i), \tag{8}$$

where $O_{pb}(i)$, $O_{ub}(i)$, and $O_{cb}(i)$ represent the scores from the popularity-based, user-based, and content-based modules, respectively. The weights α, β, and γ are non-negative and constrained such that $\alpha + \beta + \gamma = 1$. These parameters can be optimized empirically to adapt the recommendation strategy to different contexts (e.g., emphasizing diversity, relevance, or novelty). In the *prediction subsystem*, the module combines the outputs from two fundamentally different modeling paradigms: a neural model (RNN) that learns temporal dynamics, and a statistical model (gradient boosting) that captures structured patterns and interpretable features. The combined forecast is computed as:

$$O_{ps} = \delta \cdot O_{RNN} + \zeta \cdot O_{stat}, \tag{9}$$

where O_{ps} is the final prediction, O_{RNN} is the output from the LSTM-based module, and O_{stat} is the output from the gradient boosting model. The coefficients δ and ζ are also non-negative and sum to one. These weights are selected with the goal of minimizing validation error and improving generalization.

The ensemble formed by this combination benefits from the strengths of each approach: the RNN captures sequential dependencies and temporal patterns, while the statistical model identifies interpretable structures and reacts more robustly to seasonality or promotional events. Together, they reduce the likelihood of overfitting to a single modeling strategy and enhance the system's ability to generalize.

This architecture also creates a natural link between recommendation and forecasting logic. For example, predicted demand peaks from the forecasting module can be used to adjust the relevance weights in the recommendation subsystem, aligning product suggestions with expected inventory trends. Through this synergy, the system supports coordinated retail decision-making in areas such as stock replenishment, personalized marketing, and price optimization.

3 Experimental Evaluation

Evaluating the proposed system involves two main challenges: the scarcity of realistic offline retail datasets and the definition of suitable evaluation strategies. Due to the limited availability of transactional data from physical stores, our evaluation primarily targets the recommendation module. The forecasting component could not be rigorously assessed in the absence of time-stamped, item-level sales data from actual retail environments.

To simulate retail-like recommendation behavior, we employed the MovieLens 100K dataset provided by GroupLens [14]. While this dataset originates from the movie domain, its structure mirrors the typical components found in retail recommendation scenarios (such as product features, purchase logs, and user preferences). Prior literature often leverages MovieLens to benchmark collaborative, content-based, and hybrid recommenders, and its use here allows us to analyze user-specific and global patterns that are transferable to retail contexts, especially for cold-start or sparse interaction scenarios. Our experiments combine two MovieLens sub-datasets: one containing movie metadata and another with user ratings and timestamps. In the popularity-based model, we assume item popularity correlates with the number of received ratings, reflecting broad appeal, which is a reasonable assumption valid in many retail settings where frequently purchased products tend to attract consistent customer attention. Recommendations were evaluated using two standard metrics: *Precision@K* and *NDCG@K*, both well suited to top-K recommendation contexts typical of retail decision-making. These are demanding metrics that penalize irrelevant suggestions and reward both accuracy and ranking. *Precision@K* measures the proportion of relevant items in the list and is particularly strict, as a single poor recommendation can substantially lower the score. *NDCG@K* complements this by accounting for item relevance and position, assigning higher weights to correctly ranked items. An item was considered relevant if its rating exceeded the user's average. These metrics were computed for $K \in \{5, 7, 10\}$ for each recommender system, using an 80/20 train-test split. To improve statistical robustness, results were averaged over 50 runs, with randomly sampled users in each run and varying the

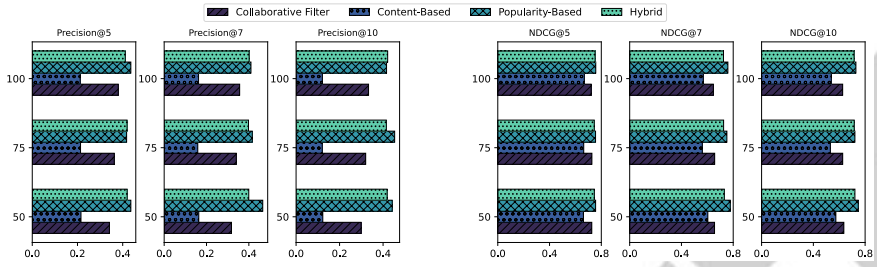

Fig. 2: Comparison using *Precision@K* and *NDCG@K* for different models.

number of users ($U \in \{50, 75, 100\}$). Figure 2 present the experimental results, averaged across all runs, illustrating the performance of the tested modules. Figure 2 summarizes the performance of all tested recommenders. The hybrid and popularity-based models consistently outperform the individual collaborative and content-based modules. Notably, the popularity-based model slightly surpasses the hybrid system in most scenarios. This result reflects a property of the MovieLens dataset itself: users often converge on a shared set of highly rated, popular items. In such a context, popularity alone becomes a strong predictor of user interest. However, this does not invalidate the hybrid approach. In real-world retail environments, user preferences are influenced by many variables, such as time of day, stock availability, promotions or seasonal effects, that are absent in MovieLens. While popularity may work well on this dataset, it lacks personalization, especially in contexts where users exhibit more diverse or localized behaviors. The hybrid model, by combining personalized, content-aware, and trend-based signals, is more robust to those variations, making it a better candidate for deployment in real-world retail systems.

4 Conclusions

This paper presented a hybrid intelligent system tailored for offline retail, combining recommendation and forecasting components to support both personalized customer interaction and operational planning. The recommendation module integrates various approaches, enabling flexible adaptation to different user profiles. Experimental results confirmed the system's robustness, with consistent performance in standard top-K recommendation metrics. Beyond accuracy, the system is designed with scalability and real-time adaptability in mind. This makes the system suitable for deployment in dynamic retail settings, where inventory, promotions, and customer behavior evolve rapidly. Integration with mobile applications, in-store IoT devices, and even motion sensors further supports real-time personalization and contextual recommendations [15]. Future developments will focus on refining the hybrid weighting strategies within the recommendation module, with the goal of improving the trade-off between accuracy and diversity across different retail scenarios.

Acknowledgments. This research is partially funded by the "Smart Venues" Project (POC Sicilia 2014/2020).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Agate, V., Concone, F., Gaglio, S., Giammanco, A.: A hybrid recommender system for cultural heritage promotion. In: 2021 IEEE International Conference on Smart Computing (SMARTCOMP) (2021)
2. Batmaz, Z., Yurekli, A., Bilge, A., Kaleli, C.: A review on deep learning for recommender systems: challenges and remedies. Artif. Intell. Rev. 52(1) (2019)
3. De Paola, A., Gaglio, S., Giammanco, A., Lo Re, G., Morana, M.: A multi-agent system for itinerary suggestion in smart environments. CAAI Transactions on Intelligence Technology 6(4), 377–393 (2021)
4. Agate, V., Ferraro, P., Gaglio, S.: A cognitive architecture for ambient intelligence systems. In: 6th International Workshop on Artificial Intelligence and Cognition, AIC 2018. CEUR Workshop Proceedings. vol. 2418, pp. 52—58 (2019)
5. De Paola, A., Lo Re, G., Gaglio, S., Ortolani, M.: An ambient intelligence architecture for extracting knowledge from distributed sensors. ACM International Conference Proceeding Series 403, 104–109 (2009)
6. Agate, V., Drago, S., Ferraro, P., Lo Re, G.: Anomaly detection for reoccurring concept drift in smart environments. In: 18th International Conference on Mobility, Sensing and Networking (MSN). pp. 113–120. IEEE (2022)
7. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5(2) (1994)
8. Varadharajan, V., Smith, N., Kalla, D., Samaah, F., Polimetla, K., Kumar, G.R.: Stock closing price and trend prediction with lstm-rnn. Journal of Artificial Intelligence and Big Data 4 (2024)
9. De Paola, A., Ferraro, P., Lo Re, G., Morana, M., Ortolani, M.: A fog-based hybrid intelligent system for energy saving in smart buildings. Journal of Ambient Intelligence and Humanized Computing 11, 2793–2807 (2020)
10. Ferraro, P., Lo Re, G.: Designing ontology-driven recommender systems for tourism. Advances onto the Internet of Things: How Ontologies Make the Internet of Things Meaningful pp. 339–352 (2014)
11. Aggarwal, C.C.: Data Mining: The Textbook. Springer Cham, 1 edn. (2015), springer International Publishing AG, part of Springer Nature 2015. eBook Packages: Computer Science, Computer Science (R0)
12. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(56) (2014)
13. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics 29(5) (2001)
14. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TIIS) 5(4) (2015)
15. Lo Re, G., Morana, M., Ortolani, M.: Improving user experience via motion sensors in an ambient intelligence scenario. PECCS 2013 - Proceedings of the 3rd International Conference on Pervasive Embedded Computing and Communication Systems pp. 29—34 (2013)