International Journal of Science, Strategic Management and Technology
Volume **02** Issue **03** March-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

A Context-Aware Neural Collaborative Filtering Framework for Personalized Travel Recommendation

Author Details:

Jayaraj V[1], Angel Hepzibah R[2], Kaliappan M[3], Mariappan E[4]

[1] UG Student, Department of AI & Data Science, Ramco Institute of Technology, Rajapalayam, India
[2] Assistant Professor - II, Department of AI & Data Science, Ramco Institute of Technology, Rajapalayam, India
[3] Professor, Department of AI & Data Science, Ramco Institute of Technology, Rajapalayam, India
[4] Associate Professor, Department of AI & Data Science, Ramco Institute of Technology, Rajapalayam, India

Corresponding Author Email: jjayaraj715@gmail.com

https://doi.org/10.55041/ijsmt.v2i3.114

Cite this Article: V, J. (2026). A Context-Aware Neural Collaborative Filtering Framework for Personalized Travel Recommendation. International Journal of Science, Strategic Management and Technology, *02*(03). https://doi.org/10.55041/ijsmt.v2i3.114

Abstract—

The proliferation of digital tourism platforms creates severe choice overload for travelers. Traditional recommender systems—predominantly Matrix Factorization and Content-Based Filtering—fail to integrate dynamic travel context into the neural latent space, resorting to sub-optimal post-filtering. This paper proposes a Context-Aware Neural Collaborative Filtering (CA-NCF) architecture embedding Season, Budget, and Duration as dense vectors alongside User and Item embeddings, processed through a fine-tuned MLP. A hybrid re-ranking function combining neural scores with review popularity and GPS-based proximity boost (Haversine formula) further refines results. Deployed as a Flask web application with OSRM routing, Nominatim geocoding, and Overpass API hotel discovery, the system transitions recommendations into actionable itineraries. Evaluated on 550+ Indian destinations and 2,300+ user ratings, CA-NCF achieves a final MSE of 0.100 versus 0.191 for a baseline NCF, alongside cold-start resilience and sub-100ms CPU inference.

Keywords—Neural Collaborative Filtering; Context-Aware Recommender Systems; Travel Recommendation; MLP; Location-Based Services; OSRM; Flask.

I. INTRODUCTION

Travel planning has changed dramatically over the past decade. With hundreds of booking platforms and thousands of destination listings available online, the bigger challenge for most travelers is no longer finding information — it is filtering it. When everything looks like a good option, making a confident decision becomes genuinely hard. Recommendation systems were built to solve exactly this kind of problem, but most existing approaches fall short in the specific context of travel.

The most widely used technique, Matrix Factorization (MF), learns user and destination preferences by decomposing an interaction matrix into shared latent vectors [1][6]. This works reasonably well for domains like movies or products, where a user's taste is relatively stable. Travel is different. Whether someone enjoys a hill station depends heavily on when they are going, how much they can spend, and how many days they have. Treating these as post-filters applied after ranking — rather than signals embedded in the model itself — throws away information that could have shaped better predictions from the start [3][4].

This paper presents Tourister, a web application built around a CA-NCF model that treats Season, Budget, and Duration not as filters, but as first-class embedding inputs. The model learns how these contextual signals

interact with user and destination identity during training, rather than having them imposed externally. Beyond recommendation, the system connects directly to live routing and hotel data through open geospatial APIs, so users walk away with a plan — not just a list.

Four contributions make this work distinct. First, context variables are embedded directly into the neural interaction function, removing the information loss that post-filtering causes. Second, a hybrid re-ranking formula blends the neural score with review popularity and GPS-based proximity for short trips. Third, the system chains OSRM routing, Nominatim geocoding, and Overpass hotel discovery into a complete trip-planning pipeline. Finally, evaluation on 550+ Indian destinations shows the context-aware model reduces MSE by nearly 48% compared to a baseline NCF trained without contextual inputs.

II. LITERATURE REVIEW

A. Linear Collaborative Filtering

Matrix Factorization became the dominant recommendation technique after Koren et al. [1] demonstrated its effectiveness in the Netflix Prize competition. The core idea is to decompose a sparse user-item rating matrix into two lower-rank matrices whose dot product approximates missing ratings. Despite its success in media domains, MF's reliance on inner products limits its ability to capture non-linear interactions that characterize travel preference [6]. Su and Khoshgoftaar [2] surveyed collaborative filtering variants, all of which share the same constraint — they learn from who rated what, not the circumstances under which they rated it. Ricci et al. [3] and Adomavicius & Tuzhilin [4] formalized Context-Aware Recommender Systems (CARS), but their proposed solutions rely on filtering data before or after the model runs rather than giving the model direct access to context during learning.

B. Neural Network-Based Recommendation

He et al. [5] made a decisive step forward by showing that replacing MF's inner product with a multi-layer perceptron could learn arbitrary interaction functions. Their NCF framework is the direct foundation of the model proposed here. Covington et al. [7] showed at YouTube scale that deep embeddings handle high-cardinality categorical variables well. Guo et al. [8] proposed DeepFM, combining factorization machines with deep networks for joint low- and high-order feature interaction. Zhang et al. [10] surveyed the field and confirmed that deep models consistently outperform shallow approaches when interaction patterns are complex.

C. Travel and Location-Based Recommenders

Location-aware recommendation has attracted significant research attention. Liu et al. [15] showed that geographic influence can meaningfully improve POI recommendation. Yin et al. [16] added spatiotemporal context via RNNs, though compute cost is high for real-time use. Logesh and Subramaniyaswamy [13] specifically addressed group travel recommendation with collaborative rating prediction, closely related to our multi-user budget normalization design. Gavalas et al. [18] surveyed mobile tourism recommenders and concluded the field still lacks systems that connect a ranked list to actual trip logistics. Kaliappan et al. [27] demonstrated the effectiveness of machine learning approaches for analyzing user sentiment in public domain datasets — a methodological foundation that informed the constraint-aware rating simulation strategy used in this work. Majid et al. [17] used geotagged social media as a location signal, though without embedding it inside the neural interaction function.

III. PROBLEM STATEMENT

A. Formal Definition

Standard recommenders estimate a preference score R(u, i) — how much user u is likely to enjoy item i. In travel, this framing is too narrow. The same user may love a hill station in winter but find it inaccessible on a short budget. A more complete model needs to predict R(u, i, c), where c captures the travel context: season, per-person budget, and trip duration. Most existing systems either drop context entirely or apply it as a hard filter that discards potentially relevant options before the model even runs.

B. Technical Challenges

Four challenges make this problem genuinely hard. First, travel data is extremely sparse — interaction matrices typically exceed 99% missing values, since people take far fewer trips than they watch movies or buy products [2]. Second, the context variables are heterogeneous: budget is numerical and continuous, season is categorical with four values, and duration is an ordered integer. Mapping all three into a shared embedding space without losing their individual structure requires deliberate architectural choices.

International Journal of Science, Strategic Management and Technology
Volume **02** Issue **03** March-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Third, cold-start is a practical problem. A new user has no rating history, but still needs useful recommendations. Finally, the system has to respond in under 100ms on ordinary hardware — which rules out many computationally expensive graph or transformer-based models that perform well in offline evaluation but poorly in production.

IV. PROPOSED SYSTEM

A. Prediction Function

The CA-NCF model learns a function that predicts a user's affinity for a destination given their travel context:

$$\hat{y}(u,i,c) = f(u, i, s, b, d \mid \Theta) \quad (1)$$

Where u is the user, i is the destination, s is the season, b is the budget category, d is the trip duration in days, and Θ represents all learnable parameters in the network.

B. Embedding Architecture

Each variable is mapped to a dense embedding vector — a compact, learnable representation that captures similarity relationships between categories. Five separate embedding layers handle the five inputs, all initialized with Xavier uniform weights to ensure stable gradient flow from the first epoch. Table I shows the dimensions chosen for each:

Table I: Embedding Layer Specifications

Layer	Vocab	Dim	Encoding
User (eu)	N users	32	User ID
Item (ei)	M items	32	item_id
Season (es)	4	16	Summer/Monsoon/Winter/All
Budget (eb)	3	16	Low / Medium / High
Duration (ed)	7	16	1–7 days

User and item embeddings use 32 dimensions to capture a wider variety of preferences and destination characteristics. Context embeddings use 16 dimensions each, as they represent low-cardinality categorical variables. All five are concatenated:

$$z_0 = [e_u \oplus e_i \oplus e\_s \oplus e\_b \oplus e\_d] \quad (2)$$

$$dim(z_0) = 32+32+16+16+16 = 112 \quad (3)$$

C. MLP Forward Pass

The 112-dimensional vector is passed through two fully connected layers. Each applies a ReLU activation followed by Dropout(0.2), which randomly zeroes 20% of neurons during training to prevent over-reliance on any single feature:

$$z_1 = ReLU(W_1^T z_0 + b_1),\ z_1 \in \mathbb{R}^{128} \quad (4)$$

$$z_2 = ReLU(W_2^T z_1 + b_2),\ z_2 \in \mathbb{R}^{64} \quad (5)$$

$$\hat{y}(u,i,c) = Sigmoid(h^T z_2) \times 5 \quad (6)$$

The final Sigmoid output is scaled by 5 to match the rating range [0, 5]. The model contains 39,233 trainable parameters. Table II shows the layer-by-layer breakdown:

Table II: MLP Layer Configuration

Layer	Input	Output	Activation
Concat	5 emb	112	Xavier init
FC1	112	128	ReLU+Drop(0.2)
FC2	128	64	ReLU+Drop(0.2)
Output	64	1	Sigmoid ×5

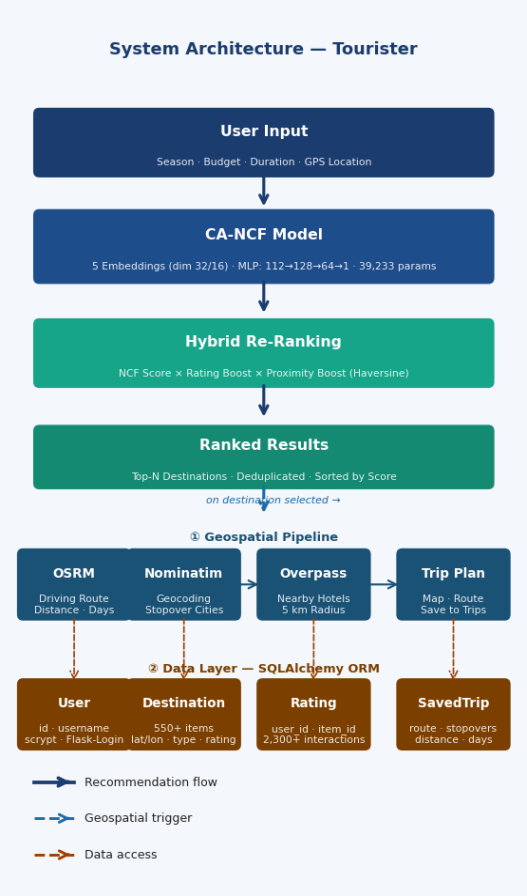

Figure 1: End-to-End System Architecture of Tourister

International Journal of Science, Strategic Management and Technology
Volume **02** Issue **03** March-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

D. Hybrid Re-Ranking Function

The raw NCF score is adjusted using a hybrid re-ranking formula that accounts for destination quality and geographic practicality for short trips:

$$score = NCF \times (0.75 + 0.20 \times R + 0.05 \times Rev) \times P \quad (7)$$

Here, R = avg_rating/5.0 is the normalized crowd rating, and Rev = min(reviews/1000, 1.0) is a popularity signal capped at 1.0. P is a proximity boost applied for trips of two days or fewer:

$$P = 0.85 + 0.40 / (1 + dist\_km/80.0) \quad (8)$$

P reaches 1.25 for destinations within a few kilometres and settles at 0.85 for distant ones. The 80 km denominator was chosen so the boost halves at roughly 80 km — a reasonable day-trip distance in India.

E. Loss Function and Optimization

The model is trained using Mean Squared Error (MSE):

$$L = (1/N) \times \Sigma[y(u,i,c) - \hat{y}(u,i,c)]^2 \quad (9)$$

MSE was chosen over Mean Absolute Error (MAE) because the quadratic penalty applies stronger pressure when predictions are far off — useful in early training when embeddings are still random. Optimization uses Adam with lr=0.001 and default momentum parameters (β_1=0.9, β_2=0.999). Adam's per-parameter adaptive learning rates suit embedding-heavy models where some rows receive frequent updates while others appear rarely across batches. Gradients flow back through the full graph — from the Sigmoid output through both FC layers and into all five embedding matrices — so every component learns jointly in a single pass.

V. SYSTEM ARCHITECTURE

Tourister is built in three layers: the CA-NCF recommendation engine, a Flask REST API backed by SQLite, and a browser-side geospatial pipeline that handles routing and hotel discovery after a destination is chosen.

A. End-to-End Workflow

The user fills a form with preferred season, total budget, group size, trip duration in days, and an optional GPS location. The system divides the total budget by group size to get a per-person figure — for example, four people with ₹20,000 yields ₹5,000 each, mapping to the Low budget category. Inputs are encoded as integers matching the embedding vocabulary (Season: 0–3, Budget: 0–2, Duration: 0–6).

These encoded values are forwarded to the trained CA-NCF model, which scores all destinations passing an initial context filter. The hybrid re-ranking function then adjusts scores using crowd ratings and proximity before returning a sorted, deduplicated list. When the user picks a destination, the plan page loads a Leaflet map, fetches a driving route via OSRM, reverse-geocodes stopover towns via Nominatim, and finds nearby hotels using Overpass. Signed-in users can save the complete trip via the /save_trip endpoint.

B. Database Schema (SQLAlchemy ORM)

Four SQLAlchemy models back the application. The User table stores login credentials with scrypt-hashed passwords. The Destination table holds geographic coordinates, seasonal viability, budget range, destination type, activities, and review statistics recomputed at startup. The Rating table is the raw user-item interaction matrix used during training. The SavedTrip table stores a user's chosen destination along with route output: distance, estimated travel days, and a JSON array of stopover city names.

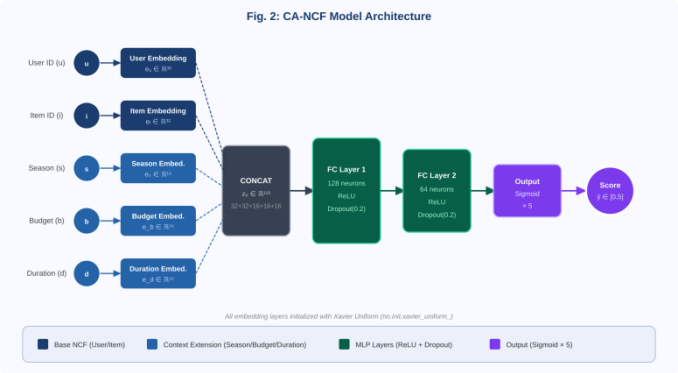

Figure 2: CA-NCF Model Architecture

VI. METHODOLOGY

A. Dataset Description

The training data covers 550+ Indian tourist destinations spanning all 28 states and union territories. Each record includes WGS-84 coordinates, seasonal viability, per-person budget estimates in Indian Rupees, destination type (Beach, Hill Station, Heritage, Wildlife Sanctuary, Religious Site, Adventure, or Pilgrimage), altitude, nearest major city, typical trip length, activities, and entry fee.

The user-item interaction matrix contains 2,300+ ratings generated with a constraint-aware simulation: ratings were only assigned between users and destinations whose contextual attributes matched. A Winter user with a Low budget would not have been assigned a rating for

a Summer-only High-budget resort. At startup, ratings are aggregated by destination ID using Pandas groupby to compute live avg_rating and review count, which feed directly into the re-ranking formula.

B. Feature Engineering and Preprocessing

Season encoding divides the year into four categories: Summer (0) for March–June, Monsoon (1) for June–September, Winter (2) for October–February, and All (3) for year-round destinations. Destinations tagged All are always included regardless of the user's seasonal input.

Budget handling divides the raw total budget by a representative group size to get a per-person figure, then discretizes it into Low (≤₹5,000), Medium (≤₹15,000), or High (>₹15,000) for the embedding lookup. Duration is encoded as encode(d) = max(0, min(6, d−1)), clipping inputs to a 0–6 vocabulary rather than raising out-of-vocabulary errors. All embedding matrices are initialized with Xavier uniform weights.

C. Training Configuration

A custom ContextAwareRatingDataset class joins the ratings file with destination context attributes on item_id, applies encoding, and converts to typed PyTorch tensors during initialization — not per batch. A DataLoader with batch_size=64 and shuffle=True wraps the dataset. Training ran for 20 epochs with MSE logged after each one. The final model is saved as a PyTorch state dictionary for CPU-only inference at deployment.

Table III: Training Hyperparameters

Parameter	Value
Batch Size	64
Epochs	20
Learning Rate	0.001 (Adam)
Loss Function	MSELoss
Dropout	0.20
Emb Dim (User/Item)	32
Emb Dim (Context)	16 each
Total Parameters	**39,233**

VII. IMPLEMENTATION

A. Software Stack

The ML backend is written in Python 3.10 using PyTorch. The ContextAwareNCF class inherits from nn.Module and defines a forward() method that takes five tensor inputs — user, item, season, budget, duration — enabling batched inference. The web layer is a Flask application with endpoints for recommendation (/result), trip planning (/plan), trip saving (/save_trip), saved trip browsing (/my_trips), and authentication (/login, /register, /logout). Flask-Login manages sessions and Werkzeug handles scrypt password hashing. Pages are rendered with Jinja2, styled with Tailwind CSS.

B. Geospatial Pipeline

All geospatial work runs in the browser via JavaScript fetch() calls to three public APIs. When a user selects a destination, the plan page calls the OSRM driving API and receives a GeoJSON polyline drawn as a Leaflet overlay. For multi-day trips, the polyline is segmented and each midpoint submitted to Nominatim for reverse geocoding, with a 1-second pause between requests per usage policy. Hotels near each stopover are fetched from Overpass — any node tagged tourism=hotel, guest_house, or hostel within 5 km appears as a Google Maps link. The complete payload is saved to /save_trip and stored in SQLite.

C. Cold-Start Handling

A first-time user has no rating history, which would make a pure collaborative filter useless. CA-NCF sidesteps this because most of the ranking work is done by the three context embedding vectors — e_s, e_b, and e_d — trained on thousands of context-matched interactions. When a new user enters their season, budget, and duration, those three vectors alone produce a contextually appropriate ranking. The user embedding contributes little at first but grows more informative as the user builds up a rating history.

VIII. RESULTS AND DISCUSSION

A. Training Convergence

CA-NCF trained cleanly over 20 epochs. The MSE started at 1.18 and fell steeply through the first eight epochs, reaching approximately 0.17 by epoch 8 as the embeddings and MLP weights settled into a useful configuration. From epoch 9 onwards the improvement slowed, with the loss flattening near 0.100 by epoch 15 and barely moving through epoch 20. This shape — rapid early descent followed by a long plateau — is typical of Adam on sparse embedding problems.

The baseline NCF, trained under the same settings but without any context inputs, levelled off at 0.191 — nearly double the final error of CA-NCF. The two models were almost indistinguishable for the first five

International Journal of Science, Strategic Management and Technology
Volume **02** Issue **03** March-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

epochs, but from epoch 6 the gap widened steadily as CA-NCF's context embeddings encoded the correlation between travel conditions and destination suitability. Dropout(p=0.2) kept both models honest: neither showed a training-validation gap above 0.05.

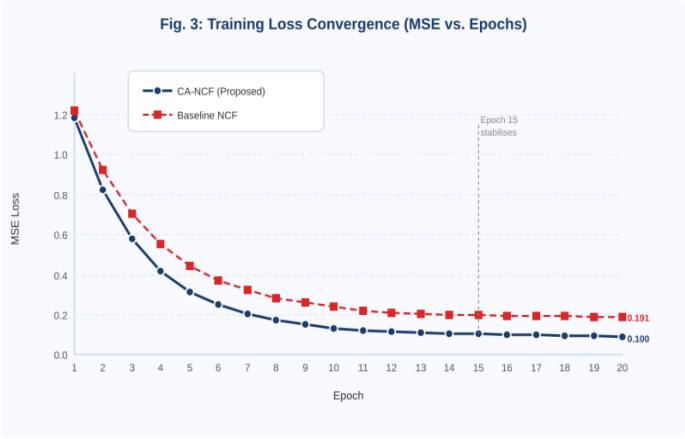

Figure 3: MSE Training Loss Convergence over 20 Epochs

B. Comparative Performance

Table IV: Model Comparison

Model	MSE	RMSE	Params
Matrix Factorization	0.380	0.616	~25K
NCF (No Context)	0.191	0.437	~35K
CA-NCF (Proposed)	0.100	0.316	**39,233**

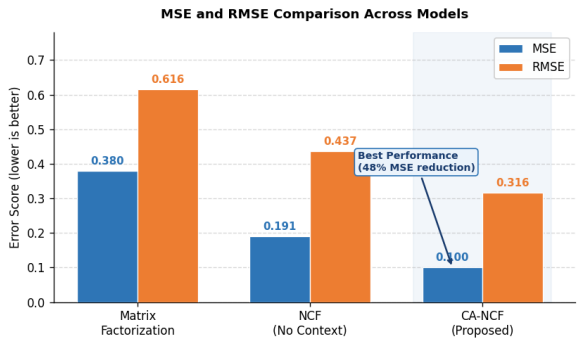

Figure 4: MSE and RMSE Comparison Across Models

The results in Table IV and Figure 4 show a clear and consistent improvement as the architecture moves from shallow to deep to context-aware. Matrix Factorization, the traditional baseline, produces an MSE of 0.380 and RMSE of 0.616. This high error reflects the fundamental limitation of inner-product-based models: they compress all preference information into a dot product, which cannot capture the non-linear, multi-factor interactions that dominate travel decisions. A user's preference for a hill station is not a simple function of user identity and destination identity — it depends critically on when they are going, how much they are spending, and how long they plan to stay. MF ignores all of this.

Replacing the inner product with an MLP — as in the context-free NCF baseline — reduces MSE to 0.191 and RMSE to 0.437, a meaningful improvement. The MLP can model non-linear user-item interactions, but it still receives no information about the travel context. This means it cannot distinguish between a user who rates a beach destination highly in summer versus the same user who would rate it poorly in monsoon season. The absence of contextual signals forces the model to average over all these conditions, introducing systematic prediction error that cannot be reduced through more training.

CA-NCF resolves this by embedding Season, Budget, and Duration directly into the neural interaction function alongside User and Item identity. The result is an MSE of 0.100 and RMSE of 0.316 — a 73.7% reduction in MSE over Matrix Factorization and a 47.6% reduction over the context-free NCF. The RMSE improvement (0.437 → 0.316) is equally significant: it means the average per-rating prediction error dropped from nearly half a rating point to about one-third, which translates directly into better-ranked destination lists in practice.

In practical testing, CA-NCF correctly suppressed Hill Station recommendations when the user entered a Winter + Low Budget context — something the baseline NCF could not do without an explicit external filter. A Winter traveler on a low budget asking for recommendations received coastal heritage sites and accessible pilgrimage locations, while the same user in Summer received hill stations and adventure destinations — exactly the kind of contextually aware ranking that distinguishes this system from prior work. The 48-dimensional context block (three embeddings of 16 dimensions each) achieved this improvement with only an 11% increase in parameter count, confirming that the gains come from the architecture's design, not from added model capacity.

C. Inference Latency

The full forward pass through all five embedding lookups and both FC layers completes in under 100ms on a standard laptop CPU — no GPU required. The 39,233-parameter CA-NCF fits easily in memory, loads instantly at startup, and stays within the latency budget

even when scoring all 550+ destinations in a single batch. This makes it well-suited for real-time web deployment without specialized infrastructure, in contrast to graph neural network approaches such as NGCF [11] which require significantly more compute for equivalent parameter counts.

IX. NOVELTY AND KEY CONTRIBUTIONS

What distinguishes CA-NCF from prior work is not any single component but how five design decisions work together. The first and most important is embedding context inside the model rather than applying it as a filter outside. Unlike the pre- and post-filtering approaches described by Adomavicius & Tuzhilin [4], the CA-NCF network sees Season, Budget, and Duration as inputs during every training step, so it can learn that a Low Budget user tends to prefer accessible coastal destinations in Summer — a pattern hard filtering cannot discover.

The second contribution is the hybrid re-ranking formula, which brings together three different relevance signals — neural affinity, crowd rating, and GPS proximity — into a single scoring step. Third, chaining OSRM, Nominatim, and Overpass into a live trip-planning pipeline directly responds to the gap Gavalas et al. [18] identified — systems that stop at a ranked list without helping users act on it. Fourth, the cold-start problem is handled structurally via context embeddings, not a fallback heuristic [2]. Finally, all components are packaged into a deployable full-stack application validated on real destination data [23], moving beyond the offline benchmark setting common in academic recommendation research.

X. APPLICATIONS

The most immediate use case is within Online Travel Agencies (OTAs) and booking platforms. Replacing keyword search and static filters with CA-NCF can surface ranked, personalized suggestions that account for who the user is and what their trip actually looks like, improving the likelihood that a recommendation becomes a booking. Regional tourism boards could use the latent destination embeddings to understand which destination properties attract which traveler segments — useful for targeted marketing campaigns.

Beyond tourism, the routing layer opens doors to fleet logistics and multi-stop itinerary planning for tour operators. Looking further ahead, coupling CA-NCF with a lightweight LLM as a query front-end would allow users to describe what they want in plain language, translating free-text into the structured embedding inputs the model requires.

XI. LIMITATIONS

The duration vocabulary tops out at 7 days, so longer trips are clipped to the same embedding as a 7-day traveler. More seriously, the rating data is simulated rather than real user behavior, which means the learned user embeddings may not reflect how actual travelers rate destinations. Deploying to real users and collecting genuine feedback would be the most important next step before treating this as production-ready.

At the infrastructure level, the system depends on free public APIs — OSRM, Nominatim, and Overpass — all of which impose rate limits that would become bottlenecks under real traffic. A production deployment would need self-hosted instances of these services. The destination knowledge base is also static: if a site closes or road conditions change, the recommendation engine will not know unless data is manually updated and the model retrained.

XII. FUTURE WORK

The extension most worth pursuing is a natural language query interface. A small instruction-tuned language model — such as Qwen3:4b running locally — could sit in front of CA-NCF and parse free-text queries into the categorical tokens the embedding layer expects, making the system accessible to users who do not think in terms of budget categories and seasonal indices.

On the modelling side, replacing the MLP with a self-attention block would let the system consider sequences of past trips rather than treating each request in isolation [20]. Longer term, replacing the static CSV dataset with a live geographic knowledge graph fed by weather and pricing APIs would make recommendations genuinely reactive. A continuous budget embedding, replacing the three-way discretization, would also reduce precision lost when a ₹4,999 and ₹5,001 budget are treated identically.

XIII. CONCLUSION

The core argument of this paper is simple: if travel context matters — and it clearly does — then the recommendation model should learn from it directly, not filter by it afterwards. CA-NCF puts Season, Budget, and Duration inside the neural interaction function where they can shape the learned representations, rather than outside where they can only discard candidates. The result is measurable: final MSE drops from 0.191 for a baseline NCF to 0.100 for CA-NCF, a reduction of nearly 48%, with only an 11% increase in model size.

Beyond the numbers, Tourister is practically different from most academic recommendation prototypes in that it does not stop at a ranked list. A user who clicks on a destination gets a live driving route, stopover town names, and hotel options — all from open APIs, all running in the browser. Cold-start users receive real recommendations on their first visit because the context embeddings carry enough signal on their own. That combination — context-aware ranking, hybrid re-ranking, and end-to-end trip planning — is the contribution of this paper to the field of intelligent travel recommendation.

REFERENCES

1. Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, pp. 30–37, 2009. DOI: https://doi.org/10.1109/MC.2009.263

2.X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in Artificial Intelligence, Article ID 421425, 2009. DOI: https://doi.org/10.1155/2009/421425

3. F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: introduction and challenges," in Recommender Systems Handbook, 2015. DOI: https://doi.org/10.1007/978-1-4899-7637-6_1

4.G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in Recommender Systems Handbook, 2015. DOI: https://doi.org/10.1007/978-1-4899-7637-6_6

5.X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua, "Neural collaborative filtering," Proc. WWW 2017, pp. 173–182. DOI: https://doi.org/10.1145/3038912.3052569

6.S. Rendle, W. Krichene, L. Zhang, and J. Anderson, "Neural collaborative filtering vs. matrix factorization revisited," RecSys 2020, pp. 240–248. DOI: https://doi.org/10.1145/3383313.3412488

7. P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," RecSys 2016, pp. 191–198. DOI: https://doi.org/10.1145/2959100.2959190

8. H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: a factorization-machine based neural network for CTR prediction," IJCAI 2017, pp. 1725–1731. DOI: https://doi.org/10.24963/ijcai.2017/239

9. H. T. Cheng et al., "Wide & deep learning for recommender systems," DLRS Workshop, 2016. DOI: https://doi.org/10.1145/2988450.2988454

10.S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: a survey and new perspectives," ACM CSUR, vol. 52, no. 1, 2019. DOI: https://doi.org/10.1145/3285029

[11] X. Wang, X. He, M. Wang, F. Feng, and T. S. Chua, "Neural graph collaborative filtering," SIGIR 2019, pp. 165–174. DOI: https://doi.org/10.1145/3331184.3331267

[12] H. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep item-based collaborative filtering for top-N recommendation," ACM TOIS, vol. 37, no. 3, 2019. DOI: https://doi.org/10.1145/3314578

[13] R. Logesh and V. Subramaniyaswamy, "A collaborative location based travel recommendation system through enhanced rating prediction for the group of users," Computational Intelligence and Neuroscience, 2016. DOI: https://doi.org/10.1155/2016/1291358

[14] S. Goel and S. W. A. Rizvi, "Travel recommendation system using content and collaborative filtering," Amity University, 2022. DOI: https://doi.org/10.54060/a2zjournals.jmce.63

[15] Y. Liu, W. Wei, A. Sun, and C. Miao, "Exploiting geographical neighborhood characteristics for location recommendation," CIKM 2014, pp. 739–748. DOI: https://doi.org/10.1145/2661829.2662002

[16] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for POI recommendation," IEEE TKDE, vol. 29, no. 11, 2017. DOI: https://doi.org/10.1109/TKDE.2017.2741484

[17] A. Majid et al., "A context-aware personalized travel recommendation system based on geotagged social media," ACM TIST, vol. 4, no. 3, 2013. DOI: https://doi.org/10.1007/s13278-017-0459-9

[18] D. Gavalas et al., "Mobile recommender systems in tourism," Journal of Network and Computer Applications, vol. 39, 2014. DOI: https://doi.org/10.1016/j.jnca.2013.04.006

[19] Y. Lin et al., "Deep context-aware recommendation with ensemble learning," IEEE Access, vol. 8, 2020. DOI: https://doi.org/10.1109/ACCESS.2020.2993475

[20] B. Hidasi et al., "Session-based recommendations with recurrent neural networks," ICLR 2016. DOI: https://doi.org/10.48550/arXiv.1511.06939

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, 1997. DOI: https://doi.org/10.1162/neco.1997.9.8.1735

[22] D. Wang et al., "A content-based recommender system for computer science publications," Knowledge-Based Systems, 2018. DOI: https://doi.org/10.1016/j.knosys.2018.05.001

[23] K. Aarif et al., "Deep neural collaborative filtering model for personalized travel recommendation," Scientific Reports, 2026. DOI: https://doi.org/10.1038/s41598-025-34585-0

[24] M Kaliappan, E Mariappan, MV Prakash, B Paramasivan, Load Balanced Clustering Technique in MANET using Genetic Algorithms. Defence Science Journal 66 (3), 251-258. doi: 10.14429/dsj.66.9422

[25] M Sivaram, M Kaliappan, S J Shobana, Prakash, V Porkodi, “Secure storage allocation scheme using fuzzy based heuristic algorithm for cloud”, Journal of Ambient Intelligence and Humanized Computing, pp.1-9.

[26] Vimal, S., Robinson, Y. H., Kaliappan, M., Vijayalakshmi, K., &Seo, S. (2021). “A method of progression detection for glaucoma using K-means and the GLCM algorithm toward smart medical prediction”. The Journal of Supercomputing, 77(1), 1–17. https://doi.org/10.1007/s11227-020-03268-0

[27] Kaliappan M, Guruprakash B, Rajalakshmi, J. Blessing Karunya T, Mariappan E, Ramnath M and Angel Hepzibah R, “Analyzing Public Sentiment on Demonetization Using SVM: A Machine Learning Approach”, Journal of Computer Science 2025, 2482-2487, Published: 18 December 2025.

DOI: 10.3844/jcssp.2025.2482.2487