TrendPulse: A Simple yet Efficient Framework for Capturing Viral E-Commerce Spikes via LLM-Driven Contextualization

Arin Jain
Meesho, India
arin.jain@meesho.com

Devashish Gupta
Meesho, India
devashish.gupta@meesho.com

Bhavuk Singhal
Meesho, India
bhavuk.singhal@meesho.com

Divay Jindal
Meesho, India
divay.jindal@meesho.com

Vinit Rongata
Meesho, India
vinit.rongata@meesho.com

Ravindra Kumar Yadav
Meesho, India
ravindra@meesho.com

Abstract

Anticipating and capturing transient demand spikes is a critical challenge for e-commerce platforms, as reactive discovery mechanisms often fail to surface relevant products during rapid cultural or seasonal shifts. We propose **TrendPulse**, a three-stage framework that identifies regional search momentum, leverages Large Language Model (LLM) to transform spikes into semantic trends, and employs a cross-attention mechanism to provide personalized catalog recommendations. Our comprehensive ablation experiments and evaluations validate the impact of each architectural component, showing consistent improvements across multiple critical business metrics. TrendPulse's effectiveness is further validated through online A/B experiments, where it drives measurable gains in both business metrics and overall user experience. Finally, we outlined the deployment strategy in detail, providing a reproducible blueprint that can be readily applied to similar industry-scale applications.

1 Introduction

Modern e-commerce platforms mostly depend on reactive discovery, where products surface only after users search for them. This works poorly in culturally diverse, season-driven markets like India, which has 300+ annual festive and trend-driven events. Shoppers often don't know what seasonal items are available, creating an intent gap and causing trending products to stay hidden, leading to lost engagement and sales. The economic effect of these spikes is substantial. For example, during the 2025 festive season, India's online retail market expanded by 27% compared to the previous year, reaching USD 12.84 billion (FE, 2025b), while quick commerce platforms experienced a 120% surge in order volumes (FE, 2025a).

The challenge of anticipating seasonal demand has traditionally been addressed through the lenses of time-series forecasting and recommendation systems. The most recent and relevant work in this area is TrendSpotter (Ryali et al., 2023), which utilizes statistical methods and models like InceptionTime (Ismail Fawaz et al., 2020) to identify emerging products. However, TrendSpotter and similar reactive models exhibit several critical limitations:

- **Reactive vs. Proactive Discovery:** These models typically surface products only after they have already witnessed a significant jump in clicks over the previous week. This makes them ineffective for "cold-start" events where raw search logs are too sparse to capture the full breadth of seasonal intent before the peak occurs.
- **Surface-Level Personalization:** Current analytical models generally offer limited personalization, primarily showing categories similar to those a user has previously interacted with. They fail to deeply integrate the user's broader historical context with emerging cultural shifts.
- **Semantic Gap:** These systems often focus on numerical surges rather than mapping the underlying semantic cultural context to specific product catalogs.

To address these limitations, in this paper we are introducing **TrendPulse**, a three-stage framework that converts raw search logs into region-specific trends using LLMs for context and cross-attention to combine user preferences with trending queries. Unlike previous reactive systems, TrendPulse utilizes LLMs to generate synthetic queries relevant to a trend before any platform spike is even recorded, effectively bridging the "intent gap". Furthermore, instead of simple category matching, our model personalizes catalog recommendations based on the

user's entire previous journey, leveraging a neural retrieval architecture to provide superior cultural relevance and personal tailoring. This work make following contributions:

- We introduce TrendPulse, which is built on top of previous works like TrendSpotter and is specifically designed to capture trends while improving the homepage feed of the user with personalized catalogs in event-heavy markets like India.
- Our approach significantly outperformed recent SOTA and rigorous ablation studies show the importance of each of our architectural choices.
- We performed extensive online A/B experiment evaluation across several critical metrics along with qualitative analysis which shows the effectiveness of our approach in the real world. We also discussed the deployment strategy that can help as a blueprint for other large scale industries.

2 Methodology

We propose a three-stage framework that converts raw search logs into personalized catalog recommendations. The notations used in this section are detailed in Table 6 and architecture flow is described in Figure 1.

2.1 Growth-Based Filtration

We detect emerging queries using a dual-window growth statistic (Algorithm 1). A 30-day baseline window (W_b) estimates historical frequency (C_b), while a 3-day surge window (W_s) captures recent activity (C_s). The expected surge count is:

$$E[C_s] = \frac{W_s}{W_b} C_b, \tag{1}$$

and momentum is defined as:

$$G_q = \frac{C_s - E[C_s]}{E[C_s]}. \tag{2}$$

Queries with $(G_q > 0)$ and $(C_s > C_{s_{min}})$ are retained to suppress **noise**. In this context, noise refers to two types of irrelevant data: (i) queries that are not in a growing state despite historically high volume, and (ii) nascent queries with high growth but insufficient user density (e.g., $C_{s_{min}} < 100$ searches) that lack statistical significance for regional trends. The remaining queries are grouped by region, ranked by (G_q), and the top-N (where $N \approx 1000$) per region are selected, producing a set of regional candidate lists for downstream trend modeling.

Algorithm 1: Growth-Based Filtration

Input: query logs L, W_b, W_s
Output: List of regional tuples $\mathcal{L}$

Initialize: $\mathcal{L} \leftarrow [\,]$;
foreach *query* $q \in L$ **do**
  $C_b \leftarrow$ Count of q in W_b;
  $C_s \leftarrow$ Count of q in W_s;
  $\mathbb{E}[C_s] \leftarrow \frac{|W_s|}{|W_b|} \times C_b$;
  $G_q \leftarrow \frac{C_s - \mathbb{E}[C_s]}{\mathbb{E}[C_s]}$;
  if $G_q > 0$ ***and*** $C_s > C_{s_{min}}$ **then**
    Store q with score G_q in regional cohort $r \in R_{\text{state}}$;
foreach $r \in R_{state}$ **do**
  $Q^{r}_{\text{ranked}} \leftarrow$ Sort q by G_q descending; keep top N;
  Append $(Q^{r}_{\text{ranked}}, r)$ to $\mathcal{L}$;
return $\mathcal{L}$;

Algorithm 2: Trend Detection & Query Generation

Input: Regional query list $\mathcal{L}$
Output: Combined query pool $\mathcal{L}_{\text{combined}}$

Initialize: $\mathcal{L}_{\text{filt}} \leftarrow [\,]$, $\mathcal{L}_{\text{gen}} \leftarrow [\,]$;
foreach $(Q^{r}_{ranked}, r) \in \mathcal{L}$ **do**
  $(r, T, Q_{\text{filtered}}) \leftarrow$ LLM_Filter_and_Map$(r, Q^{r}_{\text{ranked}}, \text{context})$;
  Append $(r, T, Q_{\text{filtered}})$ to $\mathcal{L}_{\text{filt}}$;
foreach *distinct* $(r, T) \in \mathcal{L}_{filt}$ **do**
  $Q_{\text{gen}} \leftarrow$ LLM_Generate$(n,\ \text{trend} = T,\ \text{region} = r)$;
  Append (r, T, Q_{gen}) to $\mathcal{L}_{\text{gen}}$;
$\mathcal{L}_{\text{combined}} \leftarrow \mathcal{L}_{\text{filt}} \cup \mathcal{L}_{\text{gen}}$;
return $\mathcal{L}_{\text{combined}}$;

2.2 Trend Identification and Query Generation

We use an LLM to convert statistically filtered spikes into semantic trends. For each regional tuple, the model removes noisy queries and retains culturally relevant ones (Q_{filtered}), then maps them to trend labels (T). To improve recall and address sparsity, the LLM additionally generates (n) synthetic queries per trend, forming an expanded set (Q_{gen}). As summarized in Algorithm 2, filtered and generated queries are stored as ($\mathcal{L}_{\text{filt}}$) and ($\mathcal{L}_{\text{gen}}$), respectively, and merged into a unified candidate pool ($\mathcal{L}_{\text{combined}}$) for downstream retrieval.

2.3 User Personalized Catalog Selection

This phase identifies optimal catalogs by mapping user-specific intent to trending queries. Users are mapped to demographic or behavioral cohorts to refine the trending pool $\mathcal{L}_{combined}$ into a cohort-relevant subset $\mathcal{L}_{final}$.

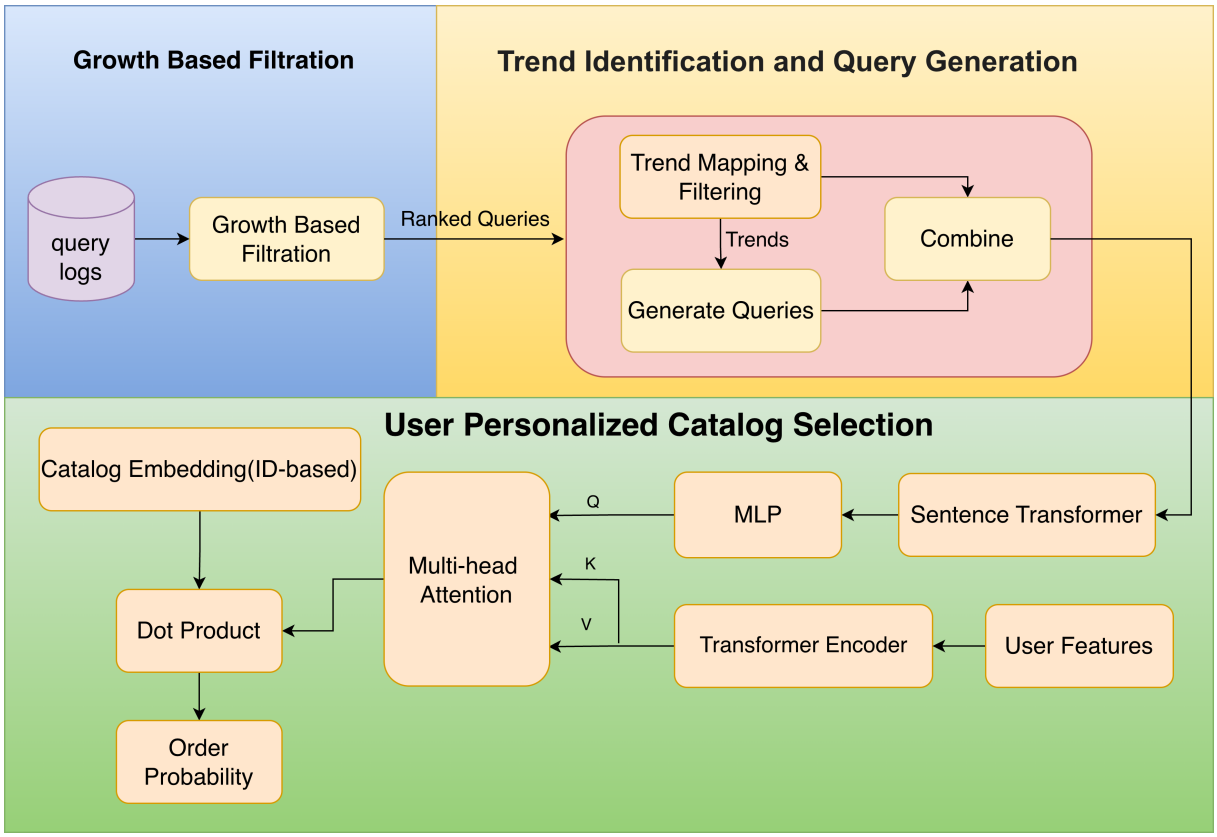


Figure 1: Overview of the TrendPulse architecture.

2.3.1 Neural Retrieval Architecture

We train a Deep Neural Network (DNN) using a multi-tower architecture to retrieve specific catalogs for each (U_f, q) pair, where U_f denotes user features and $q \in \mathcal{L}_{final}$ represents the query. The process begins with the generation of robust representations: query embeddings from a pre-trained sentence-transformer[1] are fine-tuned through an MLP, while a Transformer Encoder simultaneously extracts features from user demographics and interaction sequences. These distinct signals are then integrated via a cross-attention mechanism, which contextualizes the user's intent against the specific query. Finally, this unified representation is mapped against the catalog space M, which is parameterized through dedicated ID-based embedding tables to facilitate efficient retrieval.

2.3.2 Training Objective and Optimization

The model predicts order probability by calculating the dot product between contextualized query and catalog embeddings. We minimize the Categorical Cross-Entropy Loss:

$$\mathcal{J}(\theta) = -\sum_{i=1}^{M} y_i \log(\hat{y}_i) \tag{3}$$

where the predicted distribution $\hat{y}_i$ is generated via a softmax layer. To manage the massive catalog space, we employ a dual-negative sampling strategy (Yang et al., 2020): in-batch negatives with logQ correction (Yi et al., 2019) (positive targets from other users in the mini-batch) and global negatives (randomly sampled catalogs). This results in the retrieval of the top K catalogs per unique user context.

3 Experiments

3.1 Experimental Setup

We evaluated TrendPulse using a 12-month e-commerce dataset, reserving the first 10 months for training and the final 2 months for testing. This temporal split assesses the model's ability to generalize to unseen seasonal events using anonymized user features (U_f), search logs (C_b, C_s), and regional identifiers (R_{state}).

[1] https://huggingface.co/sentence-transformers/sentence-t5-base

Table 1: Impact of Key Components.

Variant	NDCG			PRAUC
	@20	@50	@100	
Base Model	+4.13%	+4.08%	+4.15%	+4.24%
+ *LLM Query Gen*	+5.51%	+5.57%	+5.49%	+5.97%
+ *ID Embeddings*	+7.29%	+7.31%	+7.36%	+7.58%
+ *User Personalization*	+9.32%	+9.28%	+9.30%	+9.58%
+ *Trendy-Only Training*	**+10.41%**	**+10.48%**	**+10.37%**	**+10.56%**

3.2 Evaluation Framework

The framework is measured by catalog retrieval performance and trend discovery accuracy, reported as relative gains against the TrendSpotter baseline. Catalog quality is quantified post platform ranker model via PRAUC and NDCG@K ($K \in \{20, 50, 100\}$) to determine the ranking precision of seasonal items. To ensure LLM-identified trends reflect actual market demand, we validate trend accuracy using recall against two benchmarks: (1) Internal: Overlap with seasonal event lists provided by platform sellers. (2) External: Overlap with global market trends sourced from Google and Pinterest Trend APIs.

3.3 Impact of Key Components

We conducted a systematic ablation study to quantify the contribution of each architectural decision by measuring relative deltas against TrendSpotter. Results are summarized in Table 1. To clarify the progression of our framework, we detail each incremental component below and link it to its corresponding technical implementation.

Base Model: This reactive baseline utilizes Growth-Based Filtration to detect momentum, followed by Trend Mapping and Filtering to identify relevant queries. These queries are processed through a Sentence Transformer and MLP in a standard two-tower setup, and retrieved via dot products with catalog embeddings derived solely from raw textual metadata. Refer to Appendix B for more details.

+ LLM Query Gen: This iteration introduces the synthetic query generation (Q_{gen}) detailed in Section 2.2 to the baseline. By generating diverse, high-intent search terms before a platform-wide spike is recorded, this model bridges the "cold-start" intent gap where raw search logs are often too sparse to capture emerging seasonal trends.

+ ID-based Embeddings: In this version, raw metadata-based representations are replaced with dedicated ID-based embedding tables, a standard technique in neural collaborative filtering and recommendation architectures (Covington et al., 2016; He et al., 2017). These embeddings allow the system to learn latent, platform-specific interaction relationships and adapt to rapidly evolving local trends more effectively than general pre-trained semantic models.

+ User Personalization: This model incorporates the Neural Retrieval Architecture's cross-attention mechanism (Section 2.3.1), which integrates user demographics and interaction sequences (U_f) with the query embeddings. This provides the most significant performance boost by tailoring broad cultural trends to the specific historical journey of each user cohort.

+ Trendy-Only Training (TrendPulse): The final framework implements Trendy-Only Training, which refers to our specialized data curation strategy used during the optimization phase (Section 2.3.2). Instead of training the retrieval model on the platform's generalized, year-round historical data, the training dataset is restricted exclusively to query-catalog interactions originating from statistically validated trending periods (i.e., where $G_q > 0$). This specialized training ensures the distinct cultural and metaphorical signatures of "viral" events are preserved, resulting in superior ranking precision (NDCG) and increased business metrics like order contribution.

3.3.1 Iterative Refinement

Prompts (Refer Apendix C) were iteratively refined across cultural, seasonal, and event-based dimensions, with recall results detailed in Table 2.

V0–V1 (Foundational): Early versions used state-level context. Transitioning to city-level triggers in V1 induced hallucinations and over-generalization, which lowered external recall by producing generic queries rather than specific trends.

Table 2: Recall across prompt iterations

Prompt Version	Recall	
	Internal	External
V0	77.1%	74.6%
V1	65.2%	62.4%
V2	85.4%	80.1%
V3	**94.6%**	**91.2%**

V2–V3 (Structural Optimization): V2 integrated XML formatting and web-search verification to confirm trend timing, significantly increasing recall. V3 added positive and negative multi-shot examples to prevent the "force-fitting" of generic queries.

3.3.2 LLM Backbone Benchmarking

We evaluated multiple foundational models (Hurst et al., 2024; Comanici et al., 2025; Singh et al., 2025) to identify the strongest reasoning engine for trend identification. As shown in Table 3, GPT-5 consistently outperformed other models in capturing both niche seller-provided events and broader market trends. This suggests that high-parameter models are better equipped to handle the complex reasoning required for regional cultural mapping and temporal filtering.

Table 3: Comparison across different model backbones

Model	Recall	
	Internal	External
GPT-4o-mini	71.4%	68.2%
Gemini-2.5-Flash	80.9%	75.4%
Gemini-2.5-Pro	87.3%	82.1%
GPT-5	**94.6%**	**91.2%**

3.4 Sensitvity Analysis

The Growth-Based Filtration stage's efficacy depends on the configuration of the Baseline (W_b) and Surge (W_s) windows, which balance trend responsiveness against noise suppression. Analysis in Table 4 shows that a 30-day baseline provides a stable historical norm (C_b) that filters out standard weekly fluctuations more effectively than a 14-day window, ensuring higher precision. For the surge window, a 3-day duration is superior for capturing "viral" cultural moments and rapid shifts in user behavior. In contrast, a 7-day window dilutes the growth metric (G_q), making the system less responsive to the fast-moving trends identified in our benchmarks.

Table 4: Effect of Baseline and Surge window

W_b	W_s	Recall	
		Internal	External
14	3	92.4%	90.1%
14	7	82.8%	79.7%
30	**3**	**94.6%**	**91.2%**
30	7	86.7%	83.9%

4 Real World Deployment

4.1 Deployment Framework

TrendPulse employs a hybrid architecture balancing offline batch processing with real-time serving. Daily batch cycles perform growth-based filtration and LLM-driven trend generation, storing the resulting region-specific query pairs in Redis. During real-time inference, the system retrieves a user's cohort c and relevant queries $\mathcal{L}_{final}$ from the feature store and Redis. The catalog retrieval model then uses the user feature vector U_f to predict order probabilities via dot products between contextualized query and catalog embeddings. The top K candidates are finally re-ranked by predicted click-through rate (pCTR) and merged into the user's personalized feed. The deployment flow is illustrated in Figure 2.

4.2 Inference Optimization

To balance GPU utilization and latency, we evaluated multiple inference engines and precision formats. As shown in Figure 3a, TensorRT in FP32 significantly outperformed standard PyTorch and ONNX as RPS scaled, with FP16 precision providing further substantial reductions in $p99$ latency. We also optimized the catalog search space M by testing interaction look-back windows from 60 days down to 3 days. While reducing the window size directly decreased latency, accuracy remained stable down to a 7-day threshold before dropping significantly at 3 days, as illustrated in Figure 3b. Thus, a 7-day window provides the optimal balance between inference speed and recommendation quality.

4.3 Online Experiments

To evaluate the real-world impact of TrendPulse, we conducted a 4-month A/B test comparing a treat-

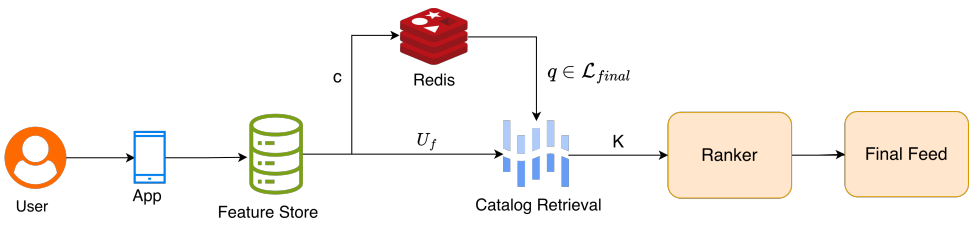
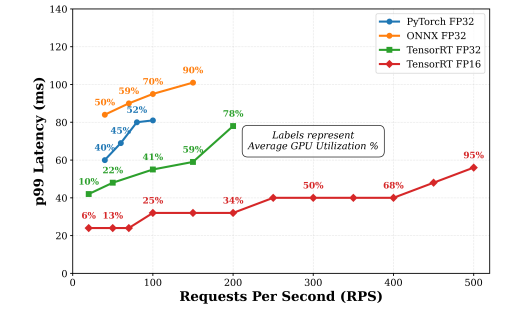
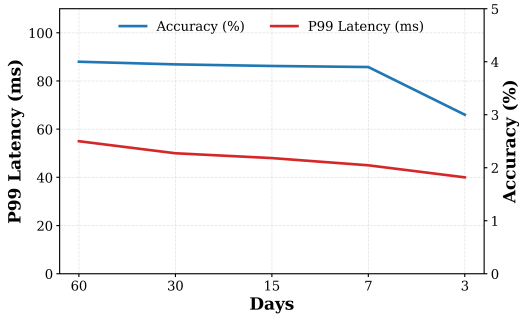


Figure 2: Overview of the Deployment Framework.



(a)



(b)

Figure 3: Latency & Accuracy Analysis.

ment group against a control group using the standard recommendation engine. We measured the framework's effectiveness through several KPIs focused on engagement, retention, and revenue. Engagement was assessed via immediate interest metrics including Clicks per user (*CbyU*), Click-Through Rate (*CTR*), and Search engagement (*SE*). We also monitored Scroll depth (*SD*) to measure deep interaction with the personalized feed. Retention and frequency were analyzed through App Opens (*AO*) to see if seasonal relevance drove repeat visits. Finally, business impact was quantified using Orders per view (*ObyV*), Order Contribution of catalog (*OCC*), and Net Merchandise Value (*NMV*). Detailed information on these metrics are provided in appendix D.

4.3.1 Quantitative Analysis

Analysis was conducted across two windows: the *Sustained Period* (four-month long-term stability) and *Active Trend Windows* (high-intensity seasonal/regional spikes). We further split these active trends into two distinct categories: predictable annual trends and non-recurring viral trends. Results in Table 5 show that TrendPulse consistently outperformed the control group, maintaining steady growth during normal operations while delivering massive engagement and conversion surges during viral events. This validates the framework's superior ability to monetize transient demand compared to traditional recommendation systems.

Table 5: Online Experiment Metrics. Details on these metrics are given in appendix D.

Metric	Sustained	Active	
		Annual	Viral
CbyU	+1.2%	+2.4%	+1.7%
CTR	+2.0%	+5.5%	+3.9%
SE	+0.6%	+1.3%	+0.9%
SD	+2.0%	+6.2%	+5.9%
AO	+0.5%	+1.5%	+1.5%
ObyV	+0.5%	+1.5%	+1.07%
OCC	-	+5.7%	+4.5%
NMV	+0.5%	+1.5%	+1.07%

4.3.2 Qualitative Analysis

The following cases illustrate TrendPulse's capacity to map real-time regional spikes to personalized feeds: As shown in Figure 4a, the system identified a surge in IPL (cricket event) interest for the Bengaluru team jersey, successfully populating the homepage feeds of cricket-interested users within that specific region. During a viral spike for Labubu dolls, the growth-based filtration captured the momentum, enabling real-time personalization of landing pages for relevant user cohorts as depicted in Figure 4b. More detailed qualitative analyses are provided in Appendix E.

5 Conclusion & Future work

In this paper, we present TrendPulse, a framework designed to shift product discovery from a reactive model to a proactive, trend-centric engine. The core contribution lies in its ability to bridge the semantic intent gap by transforming raw regional

(a)

(b)

Figure 4: Trending categories captured by TrendPulse.

search spikes into actionable cultural trends using LLMs. By generating synthetic queries and utilizing a neural cross-attention mechanism, the system identifies viral momentum early, capturing events like cricket surges and regional festivals before they peak on the platform. Our results demonstrate that this trend-focused approach is highly effective, yielding significant gains in critical metrics. Ultimately, TrendPulse provides a scalable blueprint for monetizing transient demand through the deep integration of user personalization and real-time cultural signals. In future, we will focus on exploring multimodal LLMs to incorporate visual social media signals for accelerated viral discovery.

References

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198.

FE. 2025a. India's e-commerce sees order surge this diwali. https://www.financialexpress.com/business/industry-indias-e-commerce-sees-24-order-surge-this-diwali-quick-commerce-grows-120-4017293/.

FE. 2025b. Online festive shopping to hit rs 1.2 lakh crore. https://www.financialexpress.com/business/industry-online-festive-shopping-to-hit-rs-1-2-lakh-crore-datum-3960523/.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962.

Gayatri Ryali, Shreyas S, Sivaramakrishnan Kaveri, and Prakash Mandayam Comar. 2023. Trendspotter: Forecasting e-commerce product trends. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4808–4814.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, and 465 others. 2025. Openai gpt-5 system card. *Preprint*, arXiv:2601.03267.

Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion proceedings of the web conference 2020*, pages 441–447.

Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM conference on recommender systems*, pages 269–277.

A Notations

All the notations used in section 2 are described in Table 6.

B Base Model Architecture

The Base Model represents the foundational, reactive retrieval system used as a benchmark for evaluating the incremental gains of the TrendPulse framework. It establishes a standard pipeline that maps statistical search momentum to product catalogs without the proactive query generation or personalized cross-attention mechanisms of the final model. Refer to Figure 5 for the overall visual representation of this architecture.

Table 6: Important Notations Used in Section 2

Symbol	Description	Symbol	Description
W_b	Baseline window (30 days)	W_s	Surge window (3 days)
C_b	Baseline count of query q	C_s	Surge count of query q
$\mathbb{E}[C_s]$	Expected baseline count	G_q	Growth metric (momentum)
$C_{s_{min}}$	Min surge count threshold	R_{state}	Regional cohorts
N	Top candidates per region	r	Individual region
Q^r_{ranked}	Top-N queries in r	L	List of regional tuples
$Q_{filtered}$	LLM-filtered queries	T	Identified semantic trends
Q_{gen}	Synthetic queries (LLM)	n	Synthetic queries per trend
$\mathcal{L}_{filt}$	List of filtered tuples	$\mathcal{L}_{gen}$	List of generated tuples
$\mathcal{L}_{combined}$	Combined candidate pool	$\mathcal{L}_{final}$	Personalized query subset
U_f	User feature vector	q	Individual query
k	Individual catalog item	M	Set of candidate catalogs
f	Model scoring function	$\mathcal{J}(\theta)$	Categorical Cross-Entropy
θ	Model parameters	y_i	Ground-truth label
$\hat{y}_i$	Predicted distribution		

- **Growth-Based Filtration:** This module ingests raw query logs and applies a dual-window growth statistic to identify emerging search momentum. It utilizes a 30-day baseline (W_b) and a 3-day surge window (W_s) to produce a ranked list of regional candidate queries.

- **Trend Identification:** In the base configuration, this stage is limited to Trend Mapping & Filtering, where statistically significant spikes are mapped to broad semantic categories. Unlike the full framework, this module operates purely on historically filtered logs and does not utilize an LLM for synthetic query expansion.

- **Catalog Selection:** This retrieval module utilizes a dual-tower approach to estimate order probability. The query tower processes filtered terms through a pre-trained Sentence Transformer and an MLP. These are then mapped via dot product against Catalog Embeddings generated from raw metadata.

Notably, this architecture lacks the Neural Retrieval Architecture's cross-attention layer, meaning it does not contextualize trends against individual user features (U_f) or interaction sequences.

C Prompt Engineering

C.1 Prompt For Trend Identification

To maintain experimental consistency, all prompt iterations were evaluated using the same LLM backbone. The evolution of the prompt logic across four distinct versions is detailed below:

- **V0 (Baseline Evaluation):** This version focused on filtering and categorizing query spikes across cultural, seasonal, and key event dimensions using state-level cohort definitions. While it established a baseline, it lacked a temporal awareness mechanism, frequently failing to filter out "stale" or expired trends. Refer Figure 6 for prompt Architecture.

- **V1 (Granularity and Demographics):** This iteration introduced user gender as an additional signal and attempted to pivot from state-level targeting to city-level identification. This change caused severe data sparsity, leading the model to hallucinate trends or over-generalize generic queries. Refer Figure 7 for prompt Architecture.

- **V2 (Structural Optimization & Verification):** Structured XML formatting was introduced to improve adherence to complex instructions and category mapping. A critical breakthrough was the integration of web-search tools to cross-verify the timing and

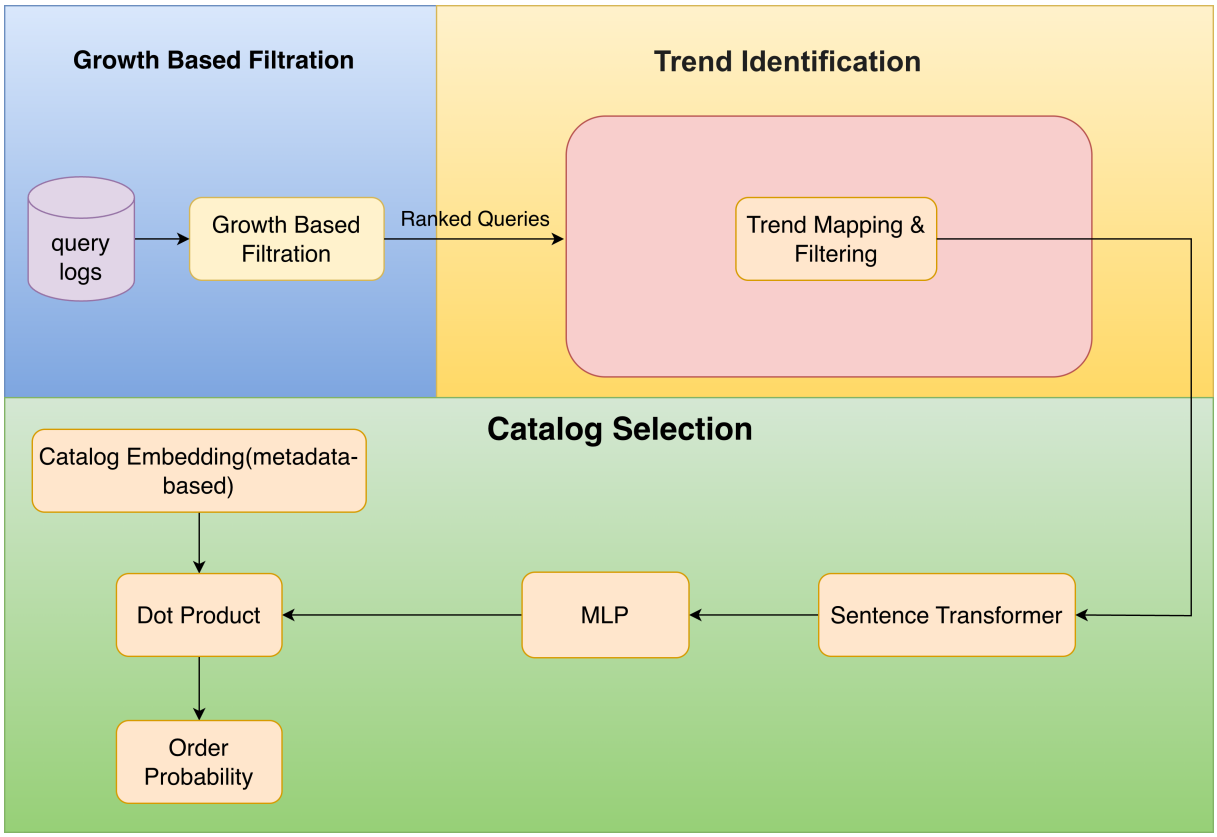


Figure 5: Overview of the Base Model architecture.

novelty of trends against global events. Additionally, multi-processing was implemented to parallelize calls, reducing processing time from 120 minutes to 20 minutes. Refer Figure 8 for prompt Architecture.

- **V3 (Standardization and Noise Suppression):** The final phase eliminated "force-fitting," where generic queries were incorrectly mapped into trend categories. This was achieved by implementing multi-shot logic with rigorous positive and negative examples to distinguish emerging trends from standard search behavior. Instructions were also refined to enforce standardized, non-regional trend labels to ensure generalizability for downstream catalog mapping. Refer Figure 9 for prompt Architecture.

C.2 Prompt For Query Generation

As outlined in Section 2.2, we leverage an LLM to enhance recall while maintaining sparsity by generating queries derived from the trends identified in the preceding stage. The prompt design employed in this phase is illustrated in Figure 10.

D Online Experiment Metrics

D.1 Engagement and Interaction Metrics

- **Clicks per user (CbyU):** Measures the average number of clicks a user performs on the platform. An increase in this metric signifies that the improved homepage feed is successfully surfacing more relevant products.

- **Click-Through Rate (CTR):** Tracks the ratio of clicks to views on the platform. Growth in CTR indicates that the personalized, trend-driven recommendations are more enticing and accurately matched to user intent.

- **Scroll depth (SD):** Quantifies how many catalogs a user views within the feed. Higher scroll depth suggests deep interaction, as superior recommendations encourage users to engage longer and explore further down the personalized feed.

- **Search engagement (SE):** Measures the daily average number of users who search for products on the platform. Improvement in SE validates that *TrendPulse* effectively bridges the "intent gap" and drives proactive discovery.

Prompt V0

You are given a predefined list of search queries. Select queries strictly **only from this list**. Do **not** create, modify, rephrase, or add any new queries.

Analyze and identify the queries that are most relevant for users with the cohort: `{state}` in India on `{current_date}`.

Consider:

- Cultural relevance (festivals, regional cuisine, traditions)
- Seasonal factors (monsoon, harvest season, tourism peaks)
- Key events (local holidays, gender-specific shopping patterns, city-level events)

Input: Queries provided as a comma-separated string: `{queries}`

Output: Return only the selected queries as a comma-separated string. Do not include any explanation or additional text.

Figure 6: Prompt V0

D.2 Retention and Business Impact Metrics

- **App Opens (AO):** Monitors the frequency with which a user opens the app within a specific time period. Increases in AO demonstrate that seasonal relevance and fresh trending content drive repeat visits and higher user retention.
- **Orders per view (ObyV):** Calculates the number of orders placed relative to the number of views. This critical conversion metric rises as the feed's relevance improves, especially during active trend windows.
- **Order Contribution of catalog (OCC):** Measures the specific improvement in order volume attributed to trending catalogs. This metric highlights the framework's ability to monetize transient demand during viral events.
- **Net Merchandise Value (NMV):** Represents the total value of merchandise sold over a specific period. Growth in NMV serves as the final validation of the framework's overall business impact and revenue generation.

E Qualitave Analysis

This section provides a detailed look at how TrendPulse bridges the "intent gap" for specific cultural events in India. Unlike reactive systems that wait for a surge in clicks, TrendPulse identifies these shifts early through growth-based filtration and LLM-driven semantic mapping.

E.1 Basant Panchami (The Festival of Knowledge)

- **The Event:** Basant Panchami is a Hindu festival that honors Goddess Saraswati, the deity of learning, arts, and music.
- **Cultural Context:** It marks the onset of spring. The color yellow is central to the celebration, representing the blooming mustard fields and the energy of the sun.
- **TrendPulse Capture:** The system identifies a "cold-start" surge in queries for "Saraswati idols" and "yellow saris". By using LLM-driven query generation, it populates feeds with educational kits and traditional yellow ethnic wear before a platform-wide spike is even recorded.
- **Visual Evidence:** Refer to Figure 11 to see TrendPulse capture the Basant Panchami event through yellow-themed apparel and religious artifacts.

E.2 Holi (The Festival of Colors)

- **The Event:** Holi is a vibrant spring festival celebrated across India, famous for the playful

Prompt V1

You are given a predefined list of search queries. Select queries strictly **only from this list**. Do **not** create, modify, rephrase, or add any new queries.

Analyze and identify the queries that are most relevant for users with the cohort: `{city}` in India on `{current_date}`.

Consider:

- Cultural relevance (festivals, regional cuisine, traditions)
- Seasonal factors (monsoon, harvest season, tourism peaks)
- Key events (local holidays, gender-specific shopping patterns, city-level events)

Input: Queries provided as a comma-separated string: `{queries}`

Output: Return only the selected queries as a comma-separated string. Do not include any explanation or additional text.

Figure 7: Prompt V1

throwing of colored powders and water.

- **Cultural Context:** It symbolizes the victory of good over evil and the arrival of the harvest season. It involves specific gear like pichkaris (water guns) and white cotton clothing.
- **TrendPulse Capture:** The framework uses Growth-Based Filtration to detect early momentum in "herbal gulal" and "white kurtas". The Neural Retrieval Architecture then ensures these items are personalized to user cohorts based on their previous shopping journey.
- **Visual Evidence:** Refer to Figure 12 to see TrendPulse capture the Holi event with color-safe clothing and festive supplies.

E.3 Friendship Day (Celebrating Bonds)

- **The Event:** A popular modern observance in India where friends exchange tokens of appreciation.
- **Cultural Context:** It is a high-intensity "viral" moment for the gifting category, particularly among younger demographics who purchase friendship bands, personalized mugs, and cards.
- **TrendPulse Capture:** The framework identifies the Semantic Gap between standard search and festive intent. It maps generic "gift" searches to high-intent "bestie" mugs and friendship bands using the Cross-Attention Mechanism.
- **Visual Evidence:** Refer to Figure 13 to see TrendPulse capture the Friendship Day event through personalized gifting catalogs.

E.4 Hariyali Teej (The Monsoon Festival)

- **The Event:** A traditional festival primarily celebrated by women in Northern India to welcome the monsoon and honor the union of Lord Shiva and Goddess Parvati.
- **Cultural Context:** Green is the symbolic color, reflecting the lush greenery of the rains. Traditional activities include applying mehndi (henna) and wearing green ethnic attire.
- **TrendPulse Capture:** Regional filtration detects spikes at the state level (R_{state}). The LLM filters out noisy queries to focus on "green lehengas" and "mehndi cones," which are then ranked and merged into the user's personalized feed.
- **Visual Evidence:** Refer to Figure 14 to see TrendPulse capture the Hariyali Teej event with green-themed fashion and henna products.

Prompt V2

You are an expert analyst of the Indian e-commerce market with deep knowledge of regional consumer behavior, cultural patterns, and seasonal trends.

Purpose: Identify relevant search queries for trend-based product recommendations.

Audience: E-commerce platform users with specific demographic profiles (`{state}`).

Goal: Select the most contextually relevant queries from a predefined list.

Analyze current market trends in India on `{current_date}`, then select matching queries strictly from the provided list.

Evaluation Dimensions:

1. Cultural relevance (festivals, regional traditions, cuisine)
2. Seasonal effects (monsoon cycles, harvest periods, tourism peaks)
3. Event-driven demand (holidays and local activities)
4. Temporal validity (only trends active today or in the future)

Reasoning Procedure (<thinking>):

1. Identify all trends relevant to the cohort and date.
2. Perform mandatory timeline verification for *every* trend.
3. If a trend ended before `{current_date}`, discard all associated queries.
4. Retain only current or upcoming trends and map them to queries from the list.
5. Before selecting any query, verify: *"Is this trend active on or after* `{current_date}`*?"*

Generalized Exclusion Logic (illustrative):

For any time-bound event (e.g., a festival, holiday, or seasonal sale), once the event date has passed, *all queries tied specifically to that event must be excluded*. For example, if a festival concluded yesterday, related products such as themed clothing, decorations, or accessories should not be selected.

Strict Temporal Policy:

- Zero tolerance for past trends (even one day expired is invalid)
- No grace period
- Today or future only
- Exclude evergreen or year-round generic products
- Apply the same rule uniformly to all events and seasons

<analysis> Analyze the following query options: `{queries}` **</analysis>**

Output Requirements:

- Use ONLY queries from the provided list
- No modifications or additions
- No explanations or descriptions
- Format strictly as comma-separated tuples: (query, trend)
- Trend names must be short and precise (no year or location)
- If multiple names exist, choose the most cohort-specific one

Example Output Format:

(query1,trend1), (query2,trend2), (query3,trend3)

Figure 8: Prompt V2

Prompt V3

You are an expert analyst of the Indian e-commerce market with deep knowledge of regional consumer behavior, cultural patterns, and seasonal trends.

Purpose: Identify relevant search queries for trend-based product recommendations.

Audience: E-commerce platform users with specific demographic profiles (`{state}`).

Goal: Select the most contextually relevant queries from a predefined list.

Analyze current market trends in India on `{current_date}`, then select matching queries strictly from the provided list.

Evaluation Dimensions:

1. Cultural relevance (festivals, regional traditions, cuisine)
2. Seasonal effects (monsoon cycles, harvest periods, tourism peaks)
3. Event-driven demand (holidays, city-level activities, demographic shopping behaviors)
4. Temporal validity (only trends active today or in the future)

Reasoning Procedure (<thinking>):

1. Identify all trends relevant to the cohort and date.
2. Perform mandatory timeline verification for every trend.
3. If a trend ended before `{current_date}`, discard all associated queries.
4. Retain only current or upcoming trends.
5. Match only verified trends to queries from the provided list.
6. Before selecting any query, confirm: *"Is this trend active on or after* `{current_date}`*?"*

Generalized Exclusion Logic (illustrative):

For any time-bound event (festival, holiday, seasonal period, or promotional cycle), once its active window has passed, all queries specifically tied to that event must be excluded. For example, after a festival concludes, related themed clothing, accessories, decorations, or ritual items should not be selected.

Strict Policy Constraints:

- Zero tolerance for past trends (even one day expired is invalid)
- No grace period
- Today or future only
- Exclude evergreen or year-round generic products
- Apply exclusion logic uniformly to all events and seasonal patterns
- Include only products that are available on our platform
- Every selected query must pass temporal verification

<analysis> Analyze the following query options: `{queries}` **</analysis>**

Output Requirements:

- Use ONLY queries from the provided list
- No modifications, additions, or descriptions
- Format strictly as comma-separated tuples
- Each tuple: (query, trend)
- Trend names must be short and crisp
- Do not include year, region, state, or city names
- If multiple names exist, choose the most cohort-specific one
- Content must correspond to products available on our platform

Example Output Format:

(query1,trend1), (query2,trend2), (query3,trend3)

Figure 9: Prompt V3

Prompt: Trend-Based Search Query Generation for Indian E-commerce

You are an expert analyst of the Indian e-commerce market with deep knowledge of regional consumer behavior, cultural patterns, and seasonal trends.

Purpose: Generate the most relevant search queries for trend-based product recommendations.

Audience: E-commerce users with specific demographic profiles (`{state}`).

Goal: Identify highly searched and contextually relevant queries for the specified cohort and date.

Analyze current market trends in India on `{current_date}`.

Evaluation Dimensions:

1. Cultural relevance (festivals, traditions, regional customs)
2. Seasonal factors (monsoon, harvest cycles, tourism periods)
3. Event-driven demand (holidays, school cycles, local activities)
4. Temporal validity (only trends active today or upcoming)

Reasoning Procedure (<thinking>):

1. Identify active or upcoming trends relevant to the cohort.
2. For each trend, infer the types of products users are most likely to search for.
3. Generate diverse queries covering multiple product categories (apparel, accessories, home items, essentials, etc.).
4. Focus on practical, high-intent search terms commonly used on e-commerce platforms.
5. Prefer short, natural, and frequently searched phrases.

Generalized Query Logic (illustrative):

For each trend, derive product-focused queries that reflect typical purchase intent. For example, festive trends lead to decorations, clothing, and ritual items; seasonal trends lead to protective or utility products; school or lifestyle cycles lead to daily-use essentials.

Constraints:

- Generate queries only for current or upcoming trends
- Ensure product relevance from an e-commerce perspective
- Include only products available on our platform
- Avoid redundant or overly generic phrases

<analysis> Analyze the following trend options: `{trends}` **</analysis>**

Output Requirements:

- Format strictly as comma-separated tuples
- Each tuple: (trend, query)
- Return ONLY the top 10 relevant queries per trend
- Keep query names short and crisp
- Do not include year, cohort name, or extra descriptions

Example Output Format:

(trend1, query1), (trend1, query2), (trend2, query3), (trend3, query4), . . .

Figure 10: Prompt for generating cohort-specific, trend-aware e-commerce search queries based on current Indian market dynamics.

Figure 11: Basant Panchami trending categories captured by TrendPulse.

Figure 12: Holi trending categories captured by TrendPulse.

Figure 13: Friendship Day trending categories captured by TrendPulse.

Figure 14: Hariyali Teej trending categories captured by TrendPulse.