Video Recommendation System

A Comparative Analysis of Three Algorithms for Short-Form Content

Authors: Jedrzej Irla, Wienczyslaw Wlodyga

Abstract

This project presents a comprehensive evaluation of three distinct recommendation algorithms for short-form video content: a Two-Tower neural network model, a Hybrid content-collaborative filtering approach, and a Multi-Armed Bandit strategy. We implemented a full-stack application using FastAPI and React, developed an AI-powered video content generation system, and conducted user testing with four participants over 10-18 minute sessions.

Our experimental results showed minimal performance differences between algorithms, with average watch percentages ranging from 53.7% to 57.0%, suggesting that user preferences and content quality may be more influential than algorithmic sophistication in limited-data scenarios.

Introduction

Short-form video platforms have revolutionized digital content consumption, with recommendation systems serving as their core technology for matching content with user preferences. These platforms face significant technical challenges including the cold start problem, sparse user interaction data, and the need for real-time adaptation to rapidly changing user preferences.

This study compares three fundamentally different recommendation strategies commonly used in the industry. The Two-Tower neural network model represents the sophisticated approach employed by major platforms like YouTube and TikTok. The Hybrid recommender combines content-based filtering with collaborative filtering techniques. The Multi-Armed Bandit approach provides a simple yet effective method for real-time learning from user feedback.

Our research aims to understand how these algorithms perform under realistic conditions with limited data, addressing a common scenario for emerging platforms or new user acquisition phases. We implemented a complete system including content generation, user interface, and comprehensive analytics to provide meaningful insights into recommendation system effectiveness.

System Architecture and Implementation

Technical Stack

Our system employs a modern full-stack architecture designed for scalability and ease of development:

- **Backend:** FastAPI with Python providing RESTful API endpoints for each recommendation algorithm
- **Frontend:** React with Vite for responsive user interface and seamless video consumption experience
- **Database:** SQLite for lightweight data persistence with three core tables
- **Machine Learning:** Scikit-learn for TF-IDF vectorization and similarity computations
- **Content Generation:** Ollama's Gemma3:4b model for synthetic story creation

Database Design

The system utilizes a simple but effective database schema consisting of three main tables:

Videos Table: Stores unique video identifiers and textual content descriptions that serve as the primary content for recommendation algorithms.

Labels Table: Contains categorical tags associated with each video, enabling content-based filtering. Labels include categories such as "nighttime," "suspense," "urban legend," "paranormal," and others that help characterize content themes.

Interactions Table: Records comprehensive user engagement data including user ID, video ID, watch percentage, explicit feedback (likes/dislikes), reaction timing, and timestamps. This table serves as the foundation for all recommendation algorithms.

API Design

The FastAPI backend exposes distinct endpoints for each recommendation algorithm, allowing direct performance comparison under identical conditions. Key endpoints include:

- `/recommend/twoTower/{user_id}` : Returns neural embedding-based recommendations
- `/recommend/hybrid/{user_id}` : Provides content-collaborative hybrid recommendations
- `/recommend/bandit/{user_id}` : Delivers Multi-Armed Bandit strategy recommendations
- `/interaction` : Records user feedback for continuous model training

- `/stats` : Provides system-wide analytics and performance metrics

AI-Powered Content Generation System

One of the most critical components of our implementation was the development of a sophisticated content generation pipeline. Since creating actual video content would have been resource-intensive and time-consuming, we developed an AI-powered system to generate realistic short-form content.

Story Generation Pipeline: We utilized Ollama's Gemma3:4b language model to create 100 unique, engaging stories designed to mimic authentic short-form video content. The generation process involved crafting specific prompts that encouraged the AI to create first-person narratives with the authentic, casual voice typical of social media content.

Content Labeling System: Each generated story was automatically tagged with 10-15 descriptive labels relevant to the narrative content. This labeling system enabled effective content-based filtering and provided the semantic understanding necessary for recommendation algorithms to function properly.

Quality Control: The generation system included JSON validation and content filtering to ensure consistent output format and appropriate content quality. Stories were designed to be 200-400 words in length, matching typical short-form video content duration when converted to speech.

Text-to-Speech Integration: We implemented a text-to-speech system using Google's gTTS (Google Text-to-Speech) to convert written stories into audio content, creating a more realistic video-like experience for users.

Recommendation Algorithms

Two-Tower Model

The Two-Tower architecture represents the most sophisticated approach in our comparison, mirroring design principles used by major platforms. This model creates separate neural embeddings for users and videos, enabling scalable similarity computation.

User Embedding Generation: The user tower creates embeddings based on interaction history, incorporating weighted content from previously watched videos. User interactions are weighted by engagement quality, where high watch percentages and likes increase content influence in user embeddings, while dislikes reduce it.

Video Embedding Creation: The video tower generates embeddings from textual descriptions and categorical labels using TF-IDF vectorization, allowing the system to understand content semantics and match them with user preferences.

Similarity Computation: Recommendations are generated using cosine similarity between user and video embeddings, providing personalized content suggestions based on learned pref-

erences.

Algorithm 1 Two-Tower Recommendation

```
function GetTwoTowerRecommendations(user_id, k=5)
    interactions ← GetUserInteractions(user_id)
    if interactions is empty then
        return RandomRecommendations(k)
    end if
    user_embedding ← CreateZeroVector(embedding_dimension)
    total_weight ← 0
    for each interaction in interactions do
        engagement_weight ← interaction.watch_percentage / 100
        if interaction.liked == 1 then
            engagement_weight ← engagement_weight × 1.5
        else if interaction.liked == -1 then
            engagement_weight ← engagement_weight × 0.3
        end if
        video_features ← GetTFIDFFeatures(interaction.video_id)
        user_embedding ← user_embedding + engagement_weight × video_features
        total_weight ← total_weight + engagement_weight
    end for
    user_embedding ← user_embedding / total_weight
    candidate_videos ← GetUnwatchedVideos(user_id)
    scores ← []
    for each video in candidate_videos do
        video_embedding ← GetTFIDFFeatures(video.id)
        similarity ← CosineSimilarity(user_embedding, video_embedding)
        scores.append((video.id, similarity))
    end for
    return TopKVideos(scores, k)
end function
```

Hybrid Recommender

The Hybrid approach combines content-based filtering with item-to-item collaborative filtering, providing a balanced solution that leverages both content similarity and user behavior patterns.

Content-Based Component: Uses TF-IDF to identify videos similar to the user's interaction history, focusing on textual and categorical content features.

Collaborative Filtering Component: Builds item-similarity matrices based on video features and user interaction patterns, enabling discovery of content liked by users with similar preferences.

Exploration Mechanism: Includes random recommendations to prevent filter bubbles and encourage content diversity, ensuring users discover new types of content.

Algorithm 2 Hybrid Recommendation

```
function GetHybridRecommendations(user_id, k=5)
    user_history ← GetUserInteractions(user_id)
    if user_history is empty then
        return RandomRecommendations(k)
    end if
    content_scores ← CreateEmptyDictionary()
    collaborative_scores ← CreateEmptyDictionary()
                                                    ▷ Content-based scoring
    for each interaction in user_history do
        engagement_weight ← CalculateEngagementWeight(interaction)
        candidate_videos ← GetUnwatchedVideos(user_id)
        for each candidate in candidate_videos do
            similarity ← TFIDFSimilarity(interaction.video_id, candidate.id)
            content_scores[candidate.id] += engagement_weight × similarity
        end for
    end for
                                                    ▷ Collaborative filtering
    for each candidate in candidate_videos do
        for each similar_video in GetSimilarVideos(candidate.id) do
            if similar_video in user_history then
                collab_scores[candidate.id] += item_similarity[candidate.id][similar_video]
            end if
        end for
    end for
                                                    ▷ Combine scores
    final_scores ← []
    for each candidate in candidate_videos do
        combined_score ← 0.7 × content_scores[candidate.id] + 0.3 × collab_scores[candidate.id]
        final_scores.append((candidate.id, combined_score))
    end for
                                                    ▷ Add exploration component
    if random() ¡ 0.2 then
        final_scores.append((RandomVideo(), 0.1))
    end if
    return TopKVideos(final_scores, k)
end function
```

Multi-Armed Bandit

The Multi-Armed Bandit approach treats each video as a "bandit arm" and employs an epsilon-greedy strategy to balance exploration of new content with exploitation of known high-performing videos.

Epsilon-Greedy Strategy: With 10% probability, the algorithm selects random content for exploration. With 90% probability, it chooses the best-performing content based on historical rewards.

Reward System: Immediate feedback incorporation where likes generate positive rewards, dislikes create negative penalties, and watch percentages provide normalized engagement rewards.

Real-Time Learning: The algorithm maintains running averages of performance metrics for each video across all users, enabling immediate adaptation to new user feedback.

Algorithm 3 Multi-Armed Bandit Recommendation

```
function GetBanditRecommendations(user_id, k=5, epsilon=0.1)
    available_videos ← GetUnwatchedVideos(user_id)
    recommendations ← []
    for i = 1 to k do
        if random() ¡ epsilon then                                  ▷ Exploration
            selected_video ← RandomChoice(available_videos)
            reason ← "Exploration (random)"
        else                                                        ▷ Exploitation
            if bandit_arms is not empty then
                best_videos ← [(vid, avg_reward[vid]) for vid in available_videos if vid in bandit_arms]
                if best_videos is not empty then
                    selected_video ← max(best_videos, key=lambda x: x[1])[0]
                    reason ← "Exploitation (best performing)"
                else
                    selected_video ← RandomChoice(available_videos)
                    reason ← "Cold start (no data)"
                end if
            else
                selected_video ← RandomChoice(available_videos)
                reason ← "Cold start (no data)"
            end if
        end if
        score ← avg_reward[selected_video] if selected_video in bandit_arms else 0.0
        recommendations.append((selected_video, score, reason))
        available_videos.remove(selected_video)
    end for
    return recommendations
end function

function UpdateBanditReward(video_id, user_feedback)
    reward ← 0.0
    if user_feedback.liked == 1 then
        reward += 1.0
    else if user_feedback.liked == -1 then
        reward -= 0.5
    end if
    if user_feedback.watch_percentage is not None then
        reward += user_feedback.watch_percentage / 100.0
    end if
    bandit_arms[video_id].count += 1
    bandit_arms[video_id].total_reward += reward
    bandit_arms[video_id].avg_reward ← bandit_arms[video_id].total_reward / bandit_arms[video_id].count
end function
```

Experimental Methodology

User Testing Protocol

We conducted controlled user studies designed to simulate realistic short-form video consumption patterns. The testing protocol involved four participants, each engaging in individual viewing sessions lasting 10-18 minutes until natural interest decline occurred.

Session Structure: Each session began with a queue of five videos, with automatic replenishment when only two videos remained in the queue. This approach ensured continuous content availability while maintaining realistic user engagement patterns.

Data Collection: We tracked comprehensive interaction data including watch percentages, explicit feedback through likes and dislikes, reaction timing within videos, and session timestamps. This granular data collection enabled detailed analysis of user engagement patterns across different recommendation strategies.

Algorithm Rotation: Users experienced all three recommendation algorithms across different sessions, allowing for direct comparison of algorithm performance with the same user base.

Evaluation Metrics

We employed multiple metrics to assess recommendation algorithm effectiveness:

- **Average Watch Percentage:** Measured user engagement depth by calculating the percentage of each video watched before the user moved to the next content or ended the session.
- **Like Ratio:** Tracked user satisfaction through explicit positive feedback, calculated as the ratio of liked videos to total videos with explicit feedback.
- **Total Interactions:** Counted all user actions including video views, likes, dislikes, and session activities to measure overall system engagement.
- **Interactions Per User:** Normalized engagement metrics by dividing total interactions by the number of active users to enable fair comparison across algorithms.
- **Content Category Distribution:** Analyzed which types of content received the most engagement to understand user preferences and algorithm effectiveness in content discovery.

Results and Analysis

Overall Performance Comparison

Our experimental results revealed surprisingly minimal differences between the three algorithms across key performance metrics:

Metric	Hybrid Model	Two-Tower Model	Bandit Model
Average Watch Percentage	57.0%	54.3%	53.7%
Average Like Ratio	0.67	0.75	0.78
Total Interactions	140	141	152
Interactions per User	35.0	35.2	38.0

Table 1: Performance Comparison Across Algorithms

The small performance variations suggest that with limited data and short interaction sessions, sophisticated algorithmic approaches may not provide substantial advantages over simpler methods.

User Engagement Analysis

Individual user analysis revealed significant variation within each algorithm. In the Hybrid Model, user engagement patterns showed considerable diversity, with some users achieving watch percentages above 60% while others remained around 54%. The scatter plot analysis demonstrates total interactions ranging from approximately 31 to 37 per user.

The Two-Tower Model exhibited more consistent but generally lower engagement patterns. Most users clustered around 52-53% average watch percentages with total interactions between 31 and 40. One user achieved the highest engagement in this model with approximately 52% watch percentage but generated fewer total interactions compared to users in other models.

The Bandit Model demonstrated superior performance in user satisfaction metrics, with some users achieving watch percentages approaching 55% and the highest total interaction count of approximately 40. The model showed better consistency in maintaining user engagement across the user base.

User Satisfaction Metrics

Like ratio analysis provided insights into user satisfaction across different algorithms. The Hybrid Model displayed like ratios ranging from 0.55 to 0.79, indicating variable user satisfaction levels. Some users showed relatively lower satisfaction (0.55 and 0.58) while others achieved higher satisfaction (0.75 and 0.79).

The Two-Tower Model demonstrated more consistent user satisfaction with like ratios between 0.71 and 0.78 across users. The Bandit Model achieved the highest overall satisfaction with like ratios ranging from 0.71 to 0.81, particularly excelling with certain user groups.

Watch Completion Distribution

Analysis of watch completion patterns revealed distinct behavioral characteristics for each algorithm:

Hybrid Model: Showed a mean completion rate of 57.0% with a relatively normal distribution centered around this value, indicating consistent moderate engagement across different content types.

Two-Tower Model: Achieved a mean of 54.3% with a more dispersed distribution, suggesting variable user responses to neural network recommendations. Some videos achieved very high completion rates while others were quickly abandoned, indicating potential issues with recommendation accuracy for certain content types.

Bandit Model: Despite having the lowest mean at 53.7%, displayed interesting bimodal characteristics with peaks at both low and high completion rates. This pattern suggests that the algorithm successfully identified both highly engaging content and less appealing videos through its exploration-exploitation strategy.

Content Preference Analysis

One of the most significant findings was the identical content preference distribution across all three recommendation algorithms. This uniformity suggests that user intrinsic preferences dominated algorithmic recommendation strategies in determining content consumption patterns.

The most popular content categories across all models were:

- Nighttime content: 11 interactions
- Suspense stories: 10 interactions
- Urban legend content: 8 interactions
- Deer-related content: 8 interactions
- Vehicle stories: 8 interactions
- Paranormal stories: 7 interactions
- Rain-themed content: 6 interactions
- Unexplained phenomena: 6 interactions
- Wildlife stories: 5 interactions
- Reddit-style narratives: 5 interactions

This consistent pattern indicates that content intrinsic appeal may be more critical than recommendation delivery mechanisms in driving user engagement.

Technical Insights and Implementation Challenges

Model Update Strategies

Each algorithm employed different update mechanisms reflecting their architectural requirements. The Two-Tower and Hybrid models utilized background thread updates every 30 seconds, balancing computational efficiency with responsiveness to new user data. This batch processing approach proved suitable for algorithms requiring extensive computation.

The Bandit model implemented immediate updates upon receiving user feedback, enabling real-time adaptation crucial for its learning strategy. This immediate feedback incorporation contributed to its superior performance in user satisfaction metrics.

Cold Start Problem

All algorithms faced significant challenges with new users who had no interaction history. The Two-Tower model struggled most with this scenario, often falling back to random recommendations with low confidence scores. The Hybrid model handled cold start situations through its exploration component and content-based recommendations. The Bandit model proved most resilient, treating cold start as pure exploration and quickly adapting based on initial user feedback.

Computational Efficiency

Performance characteristics varied significantly across algorithms:

- **Two-Tower Model:** Required the most computational resources due to TF-IDF vectorization and embedding computations, making it suitable primarily for batch processing scenarios.
- **Hybrid Approach:** Offered moderate complexity with reasonable accuracy-speed balance.
- **Bandit Model:** Remained lightweight and ideal for real-time applications with minimal resource overhead.

Content Generation System Performance

Our AI-powered content generation system proved highly effective in creating diverse, engaging content for algorithm testing. The Gemma3:4b model successfully generated 100 unique stories with realistic themes and appropriate length for short-form content. The automated labeling system provided consistent categorical tags that enabled effective content-based filtering across all recommendation algorithms.

Discussion and Implications

Algorithm Performance in Limited Data Scenarios

The minimal performance differences between sophisticated and simple algorithms highlight an important consideration for recommendation system development. In scenarios with limited user data and short interaction sessions, complex neural architectures may not provide meaningful advantages over simpler approaches. This finding has significant implications for startup platforms and new product launches where algorithmic investment should be carefully considered against available data resources.

Real-Time Adaptation Benefits

The Bandit model's superior performance in user satisfaction and total interactions demonstrates the value of immediate feedback incorporation. While other algorithms required batch processing cycles, the Bandit's real-time adaptation enabled rapid identification of user preferences within short experimental sessions. This responsiveness proves particularly valuable in short-session environments where users may not have time for gradual algorithm learning.

Content Quality as a Fundamental Factor

The identical content preference patterns across all models strongly suggest that content intrinsic quality and appeal significantly outweigh recommendation algorithmic sophistication. Users consistently gravitated toward engaging story categories regardless of how content was recommended, emphasizing that content creation and curation should be prioritized alongside algorithmic development.

Scalability Considerations

Our findings suggest that algorithm selection should consider deployment infrastructure constraints and real-time performance requirements. Simple algorithms like the Multi-Armed Bandit may be preferable for new platforms with limited computational resources, while sophisticated approaches like the Two-Tower model should be reserved for scenarios with substantial user data and computational capacity.

Conclusions and Future Directions

This comparative analysis demonstrates that algorithmic sophistication does not automatically translate to superior performance in limited-data scenarios. The minimal differences between complex neural network approaches and simple bandit strategies suggest that factors such as content quality, user experience design, and data collection may be more critical than algorithmic choice for emerging platforms.

Our results indicate that practitioners should consider a staged approach to recommendation system development: beginning with adaptive algorithms like Multi-Armed Bandits for immediate user feedback incorporation, transitioning to Hybrid approaches as user bases expand, and implementing sophisticated neural network models only after achieving sufficient scale and data density.

The study also highlights the critical importance of content generation and curation systems. Our AI-powered content creation pipeline proved essential for conducting meaningful algorithm comparisons and could serve as a model for platforms seeking to bootstrap content libraries efficiently.

Possible Future Research Improvements

Future research should focus on:

- **Scale Enhancement:** Scaling user studies to achieve statistical significance with larger participant pools
- **Temporal Analysis:** Implementing longer observation periods to capture preference evolution over time
- **Multimodal Content:** Integrating real video content with visual and audio features for more realistic testing scenarios
- **Advanced Algorithms:** Testing more sophisticated algorithms such as transformer-based models
- **A/B Testing Framework:** Implementing comprehensive A/B testing frameworks for deeper insights into recommendation system effectiveness at scale
- **Enhanced Content Generation:** Developing more sophisticated content generation systems, potentially incorporating visual and audio elements to enable more realistic testing scenarios and better understanding of multimodal recommendation challenges

The development of these enhanced systems would provide valuable insights into the challenges and opportunities presented by next-generation recommendation systems in the rapidly evolving landscape of short-form video platforms.