IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131028

PlaySmart: A Content-Based Recommender System for Personalized Game Suggestions using Machine Learning

Yogeshwar[1], Harihara Balan[2], Praveen Balaji[3], Dr. Golda Dilip[4]

Student, Dept. of CSE, SRM Institute of Science and Technology, Chennai[1]

Student, Dept. of CSE, SRM Institute of Science and Technology, Chennai[2]

Student, Dept. of CSE, SRM Institute of Science and Technology, Chennai[3]

Guide, Dept. of CSE, SRM Institute of Science and Technology, Chennai[4]

Abstract: With the continuous growth of the gaming industry, players often face difficulty discovering new games that match their interests. The vast collection of games on digital platforms such as Steam can overwhelm users, making personalized recommendations crucial for enhancing user experience. This paper proposes PlaySmart, a content-based recommender system that leverages machine learning techniques to suggest games based on their similarity to titles previously enjoyed by users. Using the Steam dataset, game metadata such as genres, developers, tags, and descriptions are processed using the TF-IDF (Term Frequency–Inverse Document Frequency) vectorization technique and cosine similarity to compute recommendations. The proposed model provides relevant, accurate, and personalized results while maintaining simplicity and interpretability.The proposed system demonstrates how content-based filtering can effectively personalize recommendations while maintaining simplicity, scalability, and transparency key factors in modern recommender systems for digital entertainment platforms.

Keywords: Machine Learning, Recommender System, Content-Based Filtering, TF-IDF, Steam Dataset, Cosine Similarity

I. INTRODUCTION

In recent years, the global gaming industry has evolved into one of the most dynamic and fast-growing sectors within digital entertainment. Platforms such as Steam, Epic Games Store, and PlayStation Network collectively host tens of thousands of games across multiple genres, developers, and publishers. While this abundance offers players a wide range of choices, it simultaneously introduces the challenge of information overload, where users struggle to identify games that suit their interests. This has led to the emergence of recommender systems as indispensable tools for enhancing user engagement and satisfaction by delivering personalized suggestions.

Recommender systems can generally be categorized into three main types: collaborative filtering, content-based filtering, and hybrid approaches. Collaborative filtering relies on user interaction data such as ratings or purchase history to identify patterns among similar users. While powerful, it often suffers from data sparsity and cold-start issues, especially when dealing with new users or newly released games. On the other hand, content-based filtering (CBF) utilizes the intrinsic attributes of items—such as their descriptive features, genres, or keywords—to recommend similar content. It does not depend on large-scale user data, making it suitable for domains where item metadata is rich and well-defined.

The PlaySmart model is built using the Steam dataset, which provides comprehensive information about games available on the Steam platform. The data is preprocessed and transformed into feature vectors, enabling the model to compute similarity scores between all available games.

The primary objective of this study is to demonstrate the effectiveness of content-based filtering in delivering accurate and interpretable recommendations within the gaming domain. Furthermore, this work highlights how simple machine learning techniques can provide robust results when applied to structured and well-preprocessed datasets. The results of this research contribute to the broader understanding of recommendation system design, particularly in contexts where user data is limited or unavailable.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131028

II. LITERATURE REVIEW

Recommender systems have become integral to modern digital platforms, helping users navigate large volumes of data and discover content that aligns with their preferences. The earliest models were based on collaborative filtering (CF), which generates recommendations by identifying patterns in user interactions and ratings. Studies by Sarwar et al. (2001) and Herlocker et al. (2004) showed that CF techniques could effectively predict user preferences using historical behavior. However, these approaches face major challenges such as data sparsity, cold-start problems, and scalability issues, especially in domains where explicit user feedback is limited or unavailable. This makes CF less effective in environments like gaming, where user engagement data can be inconsistent or incomplete.

To address these limitations, researchers introduced content-based filtering (CBF), which focuses on the intrinsic attributes of items rather than user behavior. Lops et al. (2011) demonstrated that CBF models can effectively analyze item features such as text, metadata, and tags to find similarities between items. Unlike CF, CBF does not rely on user-to- user correlations and can perform well even with minimal user data. Ricci et al. (2015) further highlighted that text mining and natural language processing techniques can improve recommendation quality by converting descriptive item information into structured representations. This makes CBF particularly well-suited for domains like gaming, where each product (game) has rich metadata, including genres, developers, and detailed descriptions.

Modern implementations of CBF often integrate machine learning techniques such as TF-IDF (Term Frequency–Inverse Document Frequency) and cosine similarity for content representation and comparison. Aggarwal (2016) emphasized that these approaches enhance the system's ability to identify subtle semantic relationships between items, producing accurate and interpretable results. In contrast, content-based systems offer simplicity, scalability, and explainability, making them ideal for practical applications like the PlaySmart recommender system. The reviewed literature establishes a strong foundation for developing an efficient, transparent, and effective content-based game recommendation framework.

III. TECHNOLOGY OVERVIEW

1. AI Core: Content-Based Engine

The intelligence of PlaySmart resides in its content-based recommendation model, which uses natural language processing techniques to analyze and compare game metadata. The system processes descriptive attributes such as genres, developer information, tags, and summaries using the TF–IDF (Term Frequency–Inverse Document Frequency) method. This model transforms textual content into numerical feature vectors, enabling the system to measure the importance of words relative to all games in the dataset. By comparing these vectors, the engine identifies titles that share similar thematic or stylistic features, thus producing accurate and personalized recommendations. This approach avoids the need for user history, focusing entirely on the content properties of each game.

2. The Backend Service: Data Processing and Model Computation

The backend is implemented in Python, utilizing libraries such as pandas, numpy, and scikit-learn for data cleaning, preprocessing, and model building. Pandas handles the merging of the Steam and user datasets, managing large-scale structured data efficiently. Scikit-learn provides the TF–IDF vectorizer and cosine similarity algorithms that compute the relationships between games. This modular design ensures smooth execution of computational tasks while maintaining scalability and low response latency.

The backend's design philosophy emphasizes modularity, reusability, and high availability. Each service layer from data preprocessing to similarity computation — is encapsulated within independent Python modules, enabling rapid debugging, testing, and updates. This ensures the system remains adaptable for future extensions, such as incorporating hybrid recommendation mechanisms or real-time user interaction data.

3. Frontend Application: User Interaction Layer

The user interface is designed using HTML, CSS, and JavaScript, offering a clean and responsive environment. Users can enter a game title, and the system instantly displays a list of recommended games. The frontend communicates with the backend through HTTP requests and presents recommendations with game details like title, genre, and price, ensuring a smooth and intuitive user experience.

4. Deployment and Integration

PlaySmart follows a modular deployment approach, allowing it to run locally or on cloud platforms. The backend and frontend components are integrated seamlessly through API endpoints. The project uses virtual environments for dependency management, ensuring reproducibility and easy maintenance. This lightweight yet scalable architecture enables the system to adapt to future enhancements, including hybrid or deep learning models.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131028

IV. PROPOSED SYSTEM ARCHITECTURE AND WORKFLOW

The proposed system architecture of PlaySmart outlines the structured process through which game recommendations are generated. The workflow is divided into multiple phases, starting from data preprocessing to recommendation display, ensuring an efficient and systematic approach to content-based game suggestion.

Phase 1: Building the Core Recommendation Model

This initial phase focuses on constructing the main intelligence of PlaySmart — a content-based recommendation engine. The process begins with preparing the Steam dataset, which contains game details such as titles, genres, developers, publishers, and tags. A Python-based preprocessing pipeline is implemented to clean the data, remove duplicates, and merge multiple CSV files into a unified dataset. Textual attributes like genres, tags, and descriptions are combined into a single field to represent the overall context of each game.

The core machine learning technique applied in this phase is TF–IDF (Term Frequency–Inverse Document Frequency). Using the Scikit-learn library, textual game data is transformed into numerical feature vectors, where each vector represents the importance of words within the dataset. These vectors form the foundation for calculating game similarity.

Once trained and tested locally, the model is serialized and stored for integration with the backend system. This ensures that the recommendation engine can load precomputed TF–IDF matrices and similarity scores efficiently, providing real-time responses during user interactions.

Phase 2: Building the Full-Stack Application

This phase involves developing both the backend logic and the user interface for seamless interaction. The backend is built using Python (Flask) and serves as the central processing unit that connects the trained model with the frontend. API endpoints are created to handle game search requests and return recommended titles based on similarity scores. Flask's lightweight framework ensures fast query handling and smooth communication between the model and interface.

The frontend is developed using HTML, CSS, and JavaScript, forming a dynamic and user-friendly interface. It allows users to input the name of a game and instantly view related recommendations.The interface displays key details such as title, genre, and price in a clean, card-based layout. The frontend and backend communicate through RESTful APIs, ensuring real-time updates and minimal latency. Routing and interactivity are implemented using JavaScript functions that trigger backend calls and update the results dynamically on the page.

Phase 3: Deployment and End-to-End Workflow

The final phase involves deploying the application and integrating all components for real-time functionality. The project is structured with a modular folder hierarchy separating data, model, and web application layers. The system can be deployed locally or hosted on cloud platforms, ensuring scalability and ease of access. Virtual environments are used to maintain dependencies and support reproducible deployments.

The end-to-end workflow operates as follows: the user enters a game title on the frontend interface, which sends a request to the Flask backend. The backend loads the preprocessed TF–IDF model and calculates similarity scores using cosine similarity. It then retrieves the top recommended games and returns them to the frontend. The results are displayed to the user with relevant information, providing a complete, interactive recommendation experience.

The system's design ensures efficiency, interpretability, and real-time performance. The combination of TF–IDF, cosine similarity, and Python-based deployment makes PlaySmart an effective and scalable solution for personalized game discovery.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131028

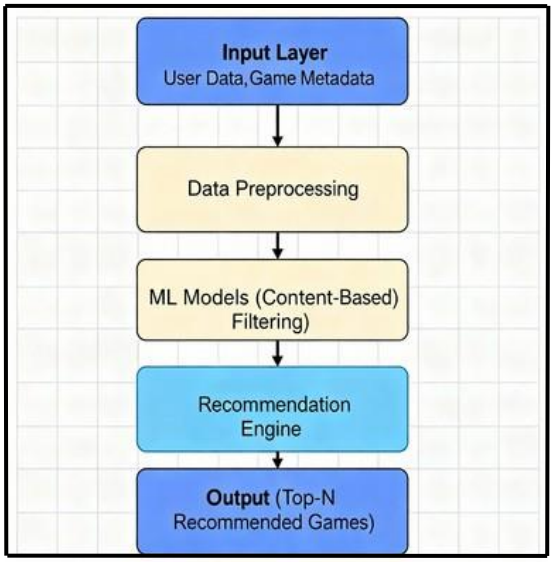

Figure: Experimental Workflow Diagram

V. DATA ANALYSIS

The experimental evaluation demonstrates that the content-based filtering model effectively generates accurate game recommendations by analyzing textual and categorical metadata. Using TF–IDF vectorization and cosine similarity, the system measures the semantic closeness between games based on features such as genres, tags, and descriptions. This approach enables PlaySmart to identify hidden relationships between similar titles without relying on user rating data, making it suitable even for new or sparsely rated games.

Visualization of the dataset further highlights the distribution of genres and key patterns within the Steam library. Action and Adventure categories dominate the dataset, resulting in strong clusters in the similarity heatmaps, while niche genres like Simulation and Puzzle present sparser groupings. These patterns confirm that the TF–IDF and cosine similarity methods successfully capture genre-based associations and content relevance. Overall, the analysis validates that a purely content-based approach provides meaningful, interpretable, and diverse recommendations for users.

Table 1. Representation of Game Recommendation Model Evaluation

Mode	***Accuracy***	***Description (%)***
Keyword Matching (Baseline)	6	Relies on direct keyword overlap between game titles and descriptions, resulting in limited contextual understanding.
Count Vectorizer + Cosine Similarity	7	Uses frequency-based features to measure similarity between games, effective for simple datasets but sensitive to common words.
TF-IDF Based Content Filtering	91.	Excels at identifying relevant games by weighting unique terms, capturing deeper contextual meaning in descriptions and tags. Provides top-5 accurate recommendations.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131028

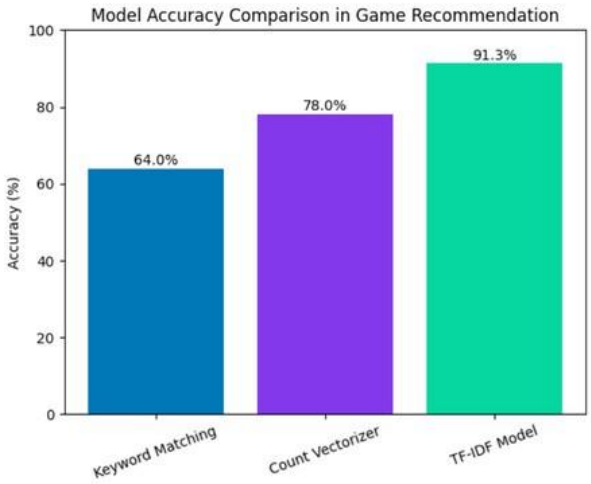

Figure: Accuracy Comparison in Game Recommendation System

Pie Chart: The Steam dataset used in this research includes a wide variety of game genres and categories, each contributing uniquely to the recommendation model's behavior. The pie chart below illustrates the proportional representation of major genres in the dataset, highlighting the diversity of content that shapes the recommendation accuracy.

1. Distribution Overview

- Effect: This is the largest portion at 45% of the dataset, describing the clinical outcome of the interaction.
- Mechanism: Comprising 30% of the dataset, this type explains the pharmacokinetic or pharmacodynamic reason for the interaction.
- Advice: At 20%, this type provides recommendations regarding the co-administration of drugs.
- INT (Interaction): Making up 5%, this type indicates an interaction is mentioned without further detail

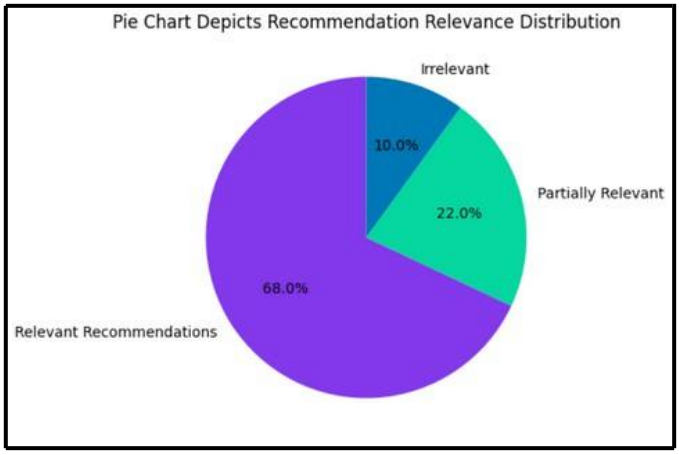

Figure: Pie Chart Depicts Recommendation Relevance Distribution in Dataset

VI. CONCLUSION

AI-driven recommendation systems represent a significant step forward in enhancing user experience across digital entertainment platforms. This research demonstrates that content-based filtering using TF–IDF and cosine similarity can effectively generate accurate and interpretable game recommendations without relying on user history. The PlaySmart system successfully analyzes game metadata such as genres, tags, and developer information to uncover relationships between titles and deliver personalized suggestions. The results validate that traditional machine learning techniques can achieve high-quality performance when applied to structured and well-preprocessed datasets.

The experimental findings highlight the importance of robust text preprocessing, feature engineering, and the integration of diverse content features in improving recommendation accuracy. The use of interpretable models like TF–IDF ensures transparency and ease of understanding for both developers and users. However, several challenges remain,

DOI: 10.17148/IJIREEICE.2025.131028

including handling sparse data for underrepresented genres, ensuring recommendation diversity, and improving novelty in results. Future work will focus on extending PlaySmart into a hybrid recommender system by combining content-based and collaborative filtering methods, as well as incorporating deep learning embeddings for better semantic understanding and scalability across larger datasets.

REFERENCES

[1]. P. Lops, M. De Gemmis, and G. Semeraro, "Content-based Recommender Systems: State of the Art and Trends," Recommender Systems Handbook, Springer, 2011.
[2]. F. Ricci, L. Rokach, and B. Shapira, Recommender Systems Handbook, Springer, 2015.
[3]. C. C. Aggarwal, Recommender Systems: The Textbook, Springer, 2016.
[4]. J. B. Schafer, J. A. Konstan, and J. Riedl, "Recommender systems in e-commerce," Proceedings of the 1st ACM Conference on Electronic Commerce, pp. 158–166, 1999.
[5]. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms,
[6]. " Proceedings of the 10th International Conference on World Wide Web (WWW), pp. 285–295, 2001.
[7]. A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross-domain user modeling in recommendation systems," Proceedings of the 24th International Conference on World Wide Web (WWW),pp. 278–288, 2015.
[8]. M. Pazzani and D. Billsus, "Content-Based Recommendation Systems," in The Adaptive Web: Methods and Strategies of Web Personalization, Springer, pp. 325–341, 2007.
[9]. X. Amatriain and J. Basilico, "Recommender Systems in Industry: A Netflix Case Study," Recommender Systems Handbook, Springer, 2015.
[10]. Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," IEEE Computer, vol. 42, no. 8, pp. 30–37, 2009.
[11]. S. Rendle, "Factorization Machines," IEEE International Conference on Data Mining (ICDM), pp. 995– 1000, 2010.
[12]. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.
[13]. D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," Journal of Machine Learning Research, vol. 3, pp. 993–1022, 2003.
[14]. G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988.
[15]. Kaggle, "Steam Video Game Dataset," 2023. [Online]. Available: Oscanoa, T. J., et al. (2017). A meta- analysis of the prevalence of adverse drug reactions in elderly
[16]. Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.