International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

AI-BASED PERSONALIZED MOBILE APP RECOMMENDATIONS USING CROWDSOURCED EDUCATIONAL DATA

BANTU MAHESH[1], NAGISETTI KEERTHANA[2], GONGA AKSHAY[3], KOLLI SAI ASHRITHA[4], KOMMA BALAJI[5], KETHAPALLY SHIVA CHARAN[6]

ASSISTANT PROFESSOR[1], UG SCHOLAR[2,3,4,5&6]

DEPARTMENT OF CSE, NARSIMHA REDDY ENGINEERING COLLEGE (UGC- AUTONOMOUS) MAISAMMAGUDA (V), KOMPALLY, SECUNDERABAD, TELANGANA-500100

ABSTRACT

The rapid growth of mobile applications has created a need for intelligent recommendation systems that help users discover relevant and useful apps. Traditional mobile app recommendation methods often rely on user ratings, download counts, or simple collaborative filtering techniques, which may not fully capture users' educational needs and preferences. To address this limitation, this study proposes an AI-based personalized mobile app recommendation system that leverages crowdsourced educational data. The proposed framework collects and analyzes crowdsourced information such as user feedback, learning outcomes, usage patterns, and educational relevance of mobile applications. Machine Learning and Deep Learning algorithms are applied to process this large-scale data and identify patterns that reflect user interests and learning objectives. By integrating collaborative filtering, content-based filtering, and deep learning models, the system generates personalized recommendations tailored to individual users. The proposed approach improves recommendation accuracy by incorporating real-world educational insights gathered from the crowd, enabling the system to better understand the effectiveness and relevance of mobile applications in educational contexts. Experimental results demonstrate that the AI-driven recommendation model provides more relevant, adaptive, and personalized suggestions compared to traditional recommendation systems. This system can assist students, educators, and lifelong learners in discovering suitable educational applications, thereby enhancing the overall mobile learning experience and promoting effective digital education.

INTRODUCITON

With the rapid growth of mobile technologies, thousands of educational applications are available on various app platforms. These apps provide learning opportunities in areas such as language learning, programming, science education, and skill development. However, due to the large number of available applications, users often face difficulties in identifying the most relevant and effective apps that match their learning needs and preferences. Traditional mobile app recommendation systems typically rely on user ratings, download counts, or basic collaborative filtering techniques. While these methods can suggest popular applications, they often fail to provide personalized recommendations that consider individual learning goals, interests, and performance levels. As a result, users may receive recommendations that are not well suited to their educational requirements. Artificial Intelligence (AI), particularly Machine Learning (ML) and Deep Learning (DL) techniques, has significantly improved the capability of recommendation systems. These technologies can analyze large volumes of user interaction data, identify hidden patterns, and generate more accurate and personalized suggestions. By leveraging AI models, recommendation systems can better understand user behavior, preferences, and learning patterns. In addition to AI, crowdsourced educational data plays an important role in improving recommendation quality. Crowdsourcing involves collecting feedback, usage data, ratings, reviews, and learning experiences from a large number of users. This collective information provides valuable insights into the effectiveness, usability, and educational value of different mobile applications. The proposed system focuses on developing an AI-based personalized mobile app recommendation framework that utilizes crowdsourced educational data to improve recommendation accuracy and relevance. By analyzing user preferences, learning activities, and community feedback, the system can recommend educational apps that best match the user's learning objectives. This approach not only enhances the user experience but also helps learners discover high-quality educational resources more efficiently. Ultimately, AI-driven recommendation systems combined with crowdsourced insights can play a significant role in supporting personalized digital learning environments and improving educational outcomes.

LITERATURE REVIW

1. Recommender Systems for Mobile Applications

Recommender systems play an important role in helping users discover relevant applications from a large number of available mobile apps. These systems analyze user behavior, preferences, ratings, and contextual information to generate personalized recommendations. Traditional recommendation approaches include **collaborative filtering, content-based filtering, and hybrid models**, which predict user preferences by analyzing past interactions and similarity among users or items. Mobile recommender systems must also consider contextual factors such as location, device characteristics, and temporal behavior, which increases system complexity. Early recommender system research relied heavily on collaborative filtering datasets such as **MovieLens**, where user ratings are used to generate predictions. These datasets enabled researchers to test algorithms for personalized recommendations and develop more advanced models for predicting user preferences.

2. Machine Learning Techniques for Recommendation Systems

Machine learning techniques have significantly improved the performance of recommender systems. Algorithms such as **decision trees, Bayesian models, support vector machines, and matrix factorization** are widely used to analyze user interaction data and predict personalized recommendations. These models automatically learn patterns from large datasets and improve recommendation accuracy compared with rule-based systems.

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

Recent studies propose ML-based recommendation frameworks that analyze learners' online behavior and interaction patterns with digital resources. These systems capture user activities such as page navigation, resource access time, and content preferences to identify learning interests and recommend relevant educational resources or applications.

3. Deep Learning in Mobile App Recommendation

Deep learning models have further advanced personalized recommendation systems. Techniques such as **Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and neural collaborative filtering** are capable of capturing complex relationships between users and applications. Deep learning models can analyze large datasets and extract high-level features automatically, improving recommendation accuracy and relevance. Research on ML/DL-based mobile app recommendation systems demonstrates that deep learning models can better capture contextual and behavioral patterns from crowdsourced data, significantly improving recommendation quality compared to traditional methods.

4. Crowdsourced Educational Data in Recommendation Systems

Crowdsourcing has become an important source of data for building intelligent recommendation systems. Crowdsourced educational data includes **user ratings, reviews, feedback, learning activities, and usage patterns** collected from a large community of learners. Mobile crowdsensing enables smartphones and connected devices to collect data from many users simultaneously, providing large datasets for training AI models. Crowdsourced data allows recommender systems to learn from collective user experiences and generate recommendations that better match the needs of diverse learners. Platforms such as Google's Crowdsource demonstrate how user-generated data can be used to improve machine learning models and enhance digital services.

5. Mobile App Review and Feedback Analysis

User reviews and feedback collected from app stores are valuable sources of crowdsourced information for improving recommendation systems. Researchers have developed automated tools that analyze thousands of app reviews using **natural language processing (NLP), sentiment analysis, and machine learning**. These techniques extract user preferences, identify app features, and detect user satisfaction levels to support better recommendations. However, challenges remain in analyzing large-scale review data due to language complexity, data noise, and scalability issues. Automated analysis techniques must handle multilingual data, informal text, and contextual meanings to accurately interpret user feedback.

6. Research Challenges and Future Directions

Despite significant progress, several challenges still exist in AI-based mobile app recommendation systems:

- Handling **large-scale crowdsourced datasets** efficiently
- Improving recommendation accuracy for **new users and new apps (cold-start problem)**
- Ensuring **data privacy and security** in crowdsourced learning environments
- Integrating **context-aware and real-time recommendation techniques**

Future research focuses on combining **machine learning, deep learning, and crowdsourced educational data** to build more intelligent, scalable, and personalized mobile app recommendation systems.

SYSTEM ANALYSIS

EXISTING SYSTEM

Traditional mobile app recommendation systems mainly rely on **basic recommendation techniques such as popularity-based filtering, collaborative filtering, and content-based filtering**. These systems suggest apps to users based on download statistics, ratings, user preferences, and historical usage data.

In many existing systems, recommendation engines analyze **user interaction data** such as app installations, reviews, and browsing behavior. Some platforms also use machine learning algorithms to improve recommendations by identifying patterns in user activity.

However, these systems often have several limitations. They usually depend on **limited user data and static recommendation models**, which may not fully capture the diverse learning needs of users. Many recommendation systems also suffer from the **cold start problem**, where new users or newly released apps lack sufficient data for accurate recommendations. Additionally, most traditional systems do not effectively utilize **crowdsourced educational insights**, such as feedback from teachers, students, and learning communities, which could improve recommendation relevance. As a result, users may receive **generic or less personalized app suggestions**.

PROPOSED SYSTEM

The proposed system introduces an **AI-Based Personalized Mobile App Recommendation System using Crowdsourced Educational Data** to enhance the accuracy and relevance of app suggestions.

In this system, data is collected from multiple sources including **user behavior, app ratings, reviews, educational feedback, and crowdsourced recommendations from students, educators, and learning communities**. This diverse dataset provides richer information about app quality, usability, and educational value.

Machine learning and deep learning models are then applied to analyze the collected data and identify meaningful patterns. Algorithms such as **collaborative filtering, neural networks, and recommendation models** are used to generate personalized app suggestions tailored to each user's learning interests, preferences, and skill levels.

The system also continuously updates recommendations by learning from **real-time user interactions and crowdsourced feedback**. This dynamic learning approach allows the recommendation engine to adapt to changing user needs and emerging educational apps.

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

By integrating **crowdsourced knowledge with AI-driven recommendation techniques**, the proposed system provides more accurate, personalized, and context-aware app recommendations for learners.

ADVANTAGES OF THE PROPOSED SYSTEM

- Highly personalized mobile app recommendations
- Utilization of crowdsourced educational insights
- Improved recommendation accuracy
- Reduced cold start problem
- Continuous learning from user feedback
- Better support for educational app discovery

IMPLEMENTAION

DECISION TREE CLASSIFIERS

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C1, C2, …, Ck is as follows:

Step 1. If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf labeled with this class

Step 2. Otherwise, let T be some test with possible outcomes O1, O2,…, On. Each object in S has one outcome for T so the test partitions S into subsets S1, S2,… Sn where each object in Si has outcome Oi for T. T becomes the root of the decision tree and for each outcome Oi we build a subsidiary decision tree by invoking the same procedure recursively on the set Si.

GRADIENT BOOSTING Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.[1][2] When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest.A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

K-NEAREST NEIGHBORS (KNN)

- Simple, but a very powerful classification algorithm
- Classifies based on a similarity measure
- Non-parametric
- Lazy learning
- Does not “learn” until the test example is given
- Whenever we have a new data to classify, we find its K-nearest neighbors from the training data

Example

- Training dataset consists of k-closest examples in feature space
- Feature space means, space with categorization variables (non-metric variables)
- Learning based on instances, and thus also works lazily because instance close to the input vector for test or prediction may take time to occur in the training dataset

LOGISTIC REGRESSION CLASSIFIERS

Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name *logistic regression* is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name *multinomial logistic regression* is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar. Logistic regression competes with discriminant analysis as a method for analyzing categorical-response variables. Many statisticians feel that logistic regression is more versatile and better suited for modeling most situations than is discriminant analysis. This is because logistic regression does not assume that the independent variables are normally distributed, as discriminant analysis does. This program computes binary logistic regression and multinomial logistic regression on both numeric and categorical independent variables. It reports on the regression equation as well as the goodness of fit, odds ratios, confidence limits, likelihood, and deviance. It performs a comprehensive residual analysis including diagnostic residual reports and plots. It can perform an independent variable subset selection search, looking for the best regression model with the fewest independent variables. It provides confidence intervals on predicted values and provides ROC curves to help determine the best cutoff point for classification. It allows you to validate your results by automatically classifying rows that are not used during the analysis.

NAÏVE BAYES

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature .Yet, despite this, it appears robust and efficient. Its performance is comparable to

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model easy to interpret and deploy, they does not understand the interest of such a technique. Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This largely explains the good performance of the method in comparison to others. In the second part, we use various tools on the same dataset (Weka 3.6.0, R 2.9.2, Knime 2.1.1, Orange 2.0b and RapidMiner 4.6.0). We try above all to understand the obtained results.

RANDOM FOREST

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance. The first algorithm for random decision forests was created in 1995 by Tin Kam Ho[1] using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg. An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.).The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman[13] in order to construct a collection of decision trees with controlled variance. Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

SVM

In classification tasks a discriminant machine learning technique aims at finding, based on an *independent and identically distributed* (*iid*) training dataset, a discriminant function that can correctly predict labels for newly acquired instances. Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point x and assigns it to one of the different classes that are a part of the classification task. Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed. From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space. SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter—in contrast to *genetic algorithms* (*GAs*) or *perceptrons*, both of which are widely used for classification in machine learning. For perceptrons, solutions are highly dependent on the initialization and termination criteria. For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set, whereas the perceptron and GA classifier models are different each time training is initialized. The aim of GAs and perceptrons is only to minimize error during training, which will translate into several hyperplanes' meeting this requirement.

1. Data Collection Module

The first step is collecting **crowdsourced educational data** from multiple users. This data may include **user ratings, reviews, download statistics, app usage behavior, learning preferences, course interests, and demographic information**. Data can be gathered from educational platforms, mobile app stores, surveys, or user feedback systems. The collected dataset forms the foundation for training the recommendation model.

2. Data Preprocessing Module

Raw crowdsourced data often contains **missing values, duplicate entries, and inconsistent formats**. In this stage, data cleaning and preprocessing are performed. Tasks include **removing noise, handling missing values, normalizing numerical data, and**

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

converting textual reviews into structured features using techniques such as tokenization or TF-IDF. This process ensures the dataset is suitable for machine learning models.

3. Feature Extraction Module

Relevant features are extracted from the dataset to represent user preferences and app characteristics. Features may include:

- User interests and learning goals
- App category (education, programming, language learning, etc.)
- User ratings and reviews
- App popularity and usage statistics

Text-based features can be processed using **Natural Language Processing (NLP)** techniques to understand user opinions and feedback about apps.

4. Model Training Module

Machine Learning or Deep Learning models are trained using the prepared dataset to learn patterns in user preferences. Common models used for recommendation systems include:

- **Collaborative Filtering**
- **Content-Based Filtering**
- **Matrix Factorization**
- **Neural Network–based recommendation models**

The dataset is divided into **training and testing sets**, and the model learns to recommend apps based on similarities between users, apps, and educational needs.

5. Personalization Engine Module

The personalization engine analyzes individual user behavior and preferences to generate **customized mobile app recommendations**. It combines outputs from trained models with real-time user interactions to provide relevant suggestions for educational applications.

6. Recommendation Generation Module

Based on the trained model and personalization engine, the system generates a **ranked list of recommended mobile applications** for each user. The recommendations may consider factors such as learning level, subject interest, previous downloads, and user feedback.

7. Model Evaluation Module

The recommendation system is evaluated using performance metrics such as:

- **Precision**
- **Recall**
- **F1 Score**
- **Mean Average Precision (MAP)**
- **Recommendation accuracy**

These metrics measure how effectively the system recommends relevant educational apps to users.

8. Deployment and User Interface Module

Finally, the system is deployed on a **web or mobile platform** where users can interact with the recommendation system. A user-friendly interface displays personalized educational app suggestions, ratings, and reviews, helping learners choose suitable applications for their needs.

CONCLUSION

When the collaborative filtering (CF) and deep learning (DL) algorithms are compared, clear performance patterns are seen, and GRU (Gated Recurrent Unit) is the best method for student-focused app suggestions. Although they show considerable accuracy, traditional CF approaches like SVD++ (MAE=0.560, RMSE=0.773) and KNN Basic (MAE=0.548, RMSE=0.753) have intrinsic limitations when it comes to addressing sparse and temporally dynamic student behavior. In contrast to other CF models such as Slope One (MAE=0.638, RMSE=0.907) or Co Clustering (MAE=0.677, RMSE=0.960), SVD++ and KNN Basic perform better, but their errors are still much larger than those of GRU (MAE=0.2246, RMSE=0.2516). This discrepancy highlights CF's difficulties with data sparsity, which is a typical problem in student datasets because of the limited diversity of app usage, as well as its incapacity to adjust to temporal changes in preferences, such as students switching from productivity tools during exams to entertainment apps after the semester.On the other hand, GRU's recurrent design achieves stability over longer prediction horizons (k=16: MAE=0.2648) while excelling at capturing short-term sequential patterns (such as spikes in app usage during study sessions). Its low error rates demonstrate its capacity to generalize from sparse encounters, which is crucial

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

considering the students' erratic app usage. Comparable temporal modeling is provided by LSTM (MAE=0.3453, RMSE=0.3522), but decreasing results are produced by its increased complexity. It is important to connect the architecture with recommendation tasks because DL models such as Graph Auto-Encoder (MAE=0.8992) and Stacked Auto-Encoder (MAE=0.0001, probably over fit) perform poorly.In conclusion, GRU is the best option for student app suggestions because it strikes a balance between precision, computational effectiveness, and flexibility to accommodate changing tastes. Although CF techniques like KNN Basic and SVD++ offer baseline benefit, student populations are less well-suited to them due to their rigidity when handling sparse, time-sensitive data. The current results solidify GRU as the state-of-the-art for tailored, real-time student recommendations, but future research could investigate hybrid models that combine CF's neighborhood-based benefits with GRU's temporal insight.

REFERENCES

[1] M. Zhai, X. Wang, and X. Zhao, "The importance of online customer reviews characteristics on remanufactured product sales: Evidence from the mobile phone market on Amazon.com," J. Retailing Consum. Services, vol. 77, Mar. 2024, Art. no. 103677.

[2] C. C. Aggarwal, "An introduction to recommender systems," in Recommender Systems: The Textbook. Cham, Switzerland: Springer, 2015, pp. 1–28.

[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item to- item collaborative filtering," IEEE Internet Comput., vol. 7, no. 1, pp. 76–80, Jan. 2003.

[4] N. Tintarev and J. Masthoff, "Beyond explaining single item recommendations," in Recommender Systems Handbook. Cham, Switzerland: Springer, 2022, pp. 711–756.

[5] C. Musto, M. de Gemmis, P. Lops, F. Narducci, and G. Semeraro, "Semantics and content-based recommendations," in Recommender Systems Handbook. Cham, Switzerland: Springer, 2022, pp. 251–298.

[6] P. Winoto and T. Y. Tang, "The role of user mood in movie recommendations," Expert Syst. Appl., vol. 37, no. 8, pp. 6086–6092, Aug. 2010.

[7] P. Li and A. Tuzhilin, "A dynamical system framework for exploring consumer trajectories in recommender system," SSRN, May 2024,
pp. 1–55.

[8] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Adv. Artif. Intell., vol. 2009, no. 1, Oct. 2009, Art. no. 421425.

[9] M. Asad, S. Shaukat, E. Javanmardi, J. Nakazato, and M. Tsukada, "A comprehensive survey on privacy-preserving techniques in federated
recommendation systems," Appl. Sci., vol. 13, no. 10, p. 6201, May 2023.

[10] M. Singh, "Scalability and sparsity issues in recommender datasets: A survey," Knowl. Inf. Syst., vol. 62, no. 1, pp. 1–43, Jan. 2020.

[11] M. Moser, "College Tennis University recommendation system (CTURS)," M.S. thesis, Dept. Psychol., Univ. Salzburg, Salzburg,
Austria, 2022.

[12] I. Hossain, M. A. H. Palash, A. T. Sejuty, N. A. Tanjim, M. A. A. L. Nasim, S. Saif, A. B. Suraj, M. M. A. Haque, and N. Karim, "A survey of recommender system techniques and the ecommerce domain," 2022, arXiv:2208.07399.

[13] G. B. Martins, J. P. Papa, and H. Adeli, "Deep learning techniques for recommender systems based on collaborative filtering," Expert Syst., vol. 37, no. 6, p. 12647, Dec. 2020.

[14] F. Camilli and M. Mézard, "The decimation scheme for symmetric matrix factorization," J. Phys. A, Math. Theor., vol. 57, no. 8, Feb. 2024, Art. no. 085002.

[15] H. Nabli, R. B. Djemaa, and I. A. B. Amor, "Improved clustering-based hybrid recommendation system to offer personalized cloud services," Cluster Comput., vol. 27, no. 3, pp. 2845–2874, Jun. 2024.