Power System Protection and Control
ISSN:1674-3415

SATR-API: A SENTIMENT- AND ATTENTION-BASED TRUST-AWARE RECOMMENDER FOR API SERVICES VIA CROSS-MODAL EMBEDDING INTEGRATION

Ali Razaq Jasim Alrikaby[1], Esmaeil Bagheri[2], Ahmed Mohammed Hussein[3], Mehdi Hamidkhani[4],

[1]Department of Computer Engineering, Isf.C., Islamic Azad University, Isfahan, Iran
[2]Department of Engineering, Deh.C., Islamic Azad University, Isfahan, Iran
[3]Department of Computer Science, College of Science for Women, University of Babylon, Babil, Iraq
[4]Department of Electrical Engineering, Isf.C., Islamic Azad University, Isfahan, Iran

Corespondance Author: Esmaeil Bagheri
bagheri@iau.ac.ir

Received: 27/12/2025 **Revised**: 26/01/2026 **Accepted**: 22/02/2026

***ABSTRACT*:**

In this paper, we introduce SATR-API, a novel hybrid recommendation framework that leverages sentiment cues, trust relationships, and deep contextual fusion for personalized API service suggestions. The proposed system initiates by extracting contextualized sentiment representations from user-generated reviews using a fine-tuned BERT variant. To model user reliability, an adaptive trust network is constructed and embedded using Graph Attention Networks (GAT), capturing personalized trust dynamics more effectively than traditional GCNs. These sentiment and trust signals are integrated into a contextual matrix factorization module enhanced with auxiliary user and service attributes. A cross-modal attention mechanism is employed to unify heterogeneous features, allowing for adaptive feature weighting during the recommendation process. Experiments conducted on a real-world API dataset demonstrate that SATR-API consistently outperforms contemporary approaches in terms of RMSE, MAE, and NDCG. Additional ablation studies highlight the contributions of each component and verify the robustness of the framework.

***Keywords**: API Service Recommendation, Sentiment Analysis, Deep Learning, Trust-Aware Recommender Models,.*

INTRODUCTION

With the exponential growth of cloud-based platforms and services, software developers are increasingly overwhelmed by the wide range of available Application Programming Interfaces (APIs). Selecting the most suitable API for a specific task is no longer a straightforward process due to the diversity in functionality, usability, reliability, and documentation quality. To support this decision-making process, intelligent API recommender systems have become a crucial component of modern software engineering workflows [1], especially in complex and dynamic environments like cloud computing [2].

Traditional recommendation models, such as collaborative filtering and matrix factorization, have been widely used in service recommendation tasks [3]. However, their reliance on only historical usage data fails to capture the nuanced factors that influence developers' choices, including trust, opinion, and project context [4]. To address these shortcomings, recent approaches have begun integrating deep learning techniques, enabling the extraction of high-level patterns and semantic relationships from auxiliary data such as user reviews [5].

Sentiment analysis has emerged as a valuable tool to mine users' subjective opinions from textual content shared on platforms like Stack Overflow, GitHub, and public API documentation forums [6], [7]. These textual insights, when combined with numerical interaction data, can significantly enhance the personalization and relevance of API recommendations [8]. In addition, understanding the **trust dynamics** among users—how one developer's feedback or behavior influences another—has shown to be an effective strategy to improve both the credibility and personalization of recommendations in social coding environments [9].

DOI: 10.46121/pspc.54.1.20

Motivated by these insights, this paper proposes **SATR-API**, a hybrid trust- and sentiment-aware recommendation framework that integrates multiple information modalities to enhance API selection in cloud-based systems. The model starts by extracting deep semantic sentiment features from developer reviews using a fine-tuned variant of the BERT model. A graph attention network (GAT) is then used to model the trust relationships between users, generating personalized trust embeddings that reflect the influence and credibility of users within a community. These sentiments and trust-aware embeddings are combined with contextual matrix factorization techniques, enhanced with side information such as project category or development domain. Finally, a cross-modal multi-head attention mechanism is employed to dynamically fuse these heterogeneous feature sources, enabling a more holistic recommendation process.

The key contributions of this paper can be summarized as follows:

- We propose a novel hybrid recommendation model (SATR-API) that fuses sentiment, trust, and contextual interactions for accurate API recommendation.
- We incorporate deep pre-trained language models (e.g., BERT) to extract fine-grained sentiment representations from developer-generated content.
- We design a trust modeling layer based on Graph Attention Networks (GAT), allowing for more expressive modeling of interpersonal trust among users.
- A multi-head attention-based fusion strategy is introduced to integrate sentiment, trust, and interaction data effectively.
- We conduct comprehensive experiments on real-world API usage datasets, demonstrating significant improvements in predictive accuracy and personalization over strong baselines.
- We provide an end-to-end deployable framework that can assist software engineers in real-world API selection tasks across various cloud development environments.

The rest of the paper is structured as follows. Section 2 reviews the related literature on recommender systems, sentiment modeling, and trust-aware architectures. Section 3 presents the SATR-API framework in detail, including its sentiment extraction module, trust embedding mechanism, and multi-head fusion layer. Section 4 describes the experimental setup, dataset details, and evaluation metrics. Section 5 presents and discusses the empirical results. Finally, Section 6 concludes the study and outlines directions for future research.

RELATED WORKS

In recent years, numerous studies have explored the development of intelligent recommender systems, particularly in the context of cloud computing environments. These efforts aim to enhance recommendation accuracy and user satisfaction by leveraging various data sources and algorithmic techniques.

Rafiee et al. [12] proposed the CNDP algorithm for link prediction, which improved accuracy by incorporating the clustering coefficient and analyzing common neighbors of neighbors. Their method demonstrated superior performance compared to traditional approaches across both synthetic and real-world networks. Similarly, Ahmed et al. [13] introduced the TrustCTR model to address the cold-start problem in recommender systems by incorporating time dynamics. Evaluated on the Ciao and Epinions datasets, this model outperformed baseline methods in terms of MAE, RMSE, and F-size metrics.

In the educational domain, Ibrahim et al. [14] developed FBRS, a fog-based recommender system for e-learning environments. The system, structured around three core modules, achieved improvements in recommendation accuracy while maintaining efficiency in processing time and security. Ghafouri et al. [15] introduced the concept of regional reputation, a metric that fuses location and user reputation data to enhance quality-of-service (QoS) predictions. Building on this, Hashemi et al. [16] proposed a model that mitigates the influence of unreliable users in QoS prediction, validating its robustness across seven prediction techniques.

Ghafouri et al. [17] further categorized prediction methods into memory-based, model-based, and hybrid collaborative filtering approaches, outlining their respective strengths and weaknesses. Duan et al. [18] tackled the rating sparsity issue by combining review-based collaborative filtering with matrix factorization, resulting in improved recommendation accuracy. In another contribution, Liu et al. [19] introduced a model that enriches user representation through review encoders and graph learning networks. Their system, evaluated on seven public datasets, achieved notable performance improvements.

DOI: 10.46121/pspc.54.1.20

Soft computing techniques have also found applications in recommendation systems. Malandri et al. [20] highlighted the integration of sentiment analysis and soft computing methods to enhance recommender performance. In a related study, Elahi et al. [21] proposed a hybrid recommendation system that leverages sentiment extracted from product reviews. Evaluated on Amazon datasets, their model demonstrated significant improvements over conventional methods.

Deep learning-based approaches have also been employed to manage trust and bias. Bangui et al. [22] designed a reputation model based on deep learning for indirect trust management, while Zolbanin and Wynn [23] introduced a sentiment ranking metric that was found to align more closely with user preferences than traditional star ratings. A number of studies have reviewed the broader landscape of sentiment analysis and its implications. Pulikonda et al. [24] and Devi et al. [25] examined the challenges, applications, and potential of sentiment analysis in machine learning contexts. Wu et al. [26] developed the RIGCN model, integrating user reputation with graph convolutional networks for robust QoS prediction, particularly under sparse and noisy data conditions. Reputation-based recommenders were further reviewed by Shashvat and Sood [27], emphasizing their importance in trust-sensitive applications.

Finally, several hybrid systems combining sentiment analysis with traditional recommender methods have been proposed. For instance, Rao et al. [28] and Singh et al. [29] implemented movie recommender systems that merge content-based and collaborative filtering with sentiment analysis. Tested on the MovieLens dataset, these models achieved notable improvements in precision, recall, and F1-score, with reported accuracies of 0.875 and RMSE values as low as 0.407.

Collectively, these studies underscore the growing trend of integrating sentiment analysis, trust modeling, and advanced machine learning methods into recommender systems, setting a solid foundation for the proposed hybrid API recommender framework introduced in this paper.

PROPOSED METHOD

In this section, we present the SATR-API framework—an advanced recommender system specifically designed for API services in a cloud computing environment. This model aims to enhance both the accuracy and personalization of API recommendations by leveraging a cross-modal embedding integration approach.
The key novelty of SATR-API lies in fusing three heterogeneous yet complementary modalities of information:

- User Sentiment toward APIs, extracted using a pre-trained deep language model (BERT) to capture fine-grained emotional cues from textual reviews;
- Social Trust Information, derived through a Graph Convolutional Network (GCN) applied to the trust graph, allowing the model to propagate indirect trust across user relationships;
- User-API Interaction Patterns, modeled via an adaptive matrix factorization technique to encode collaborative behavioral signals.

These three representations are embedded into a unified latent space and subsequently integrated through a multi-head attention fusion mechanism, which assigns dynamic importance weights to each modality per user-API pair. This enables the model to predict the final rating score for each API more effectively by attending to the most relevant cues from sentiment, trust, and interaction dimensions.

The proposed SATR-API framework is robust and flexible, making it well-suited for dynamic, multi-modal, and cloud-based environments where user preferences and contextual information evolve over time.

DOI: 10.46121/pspc.54.1.20

Power System Protection and Control
ISSN:1674-3415

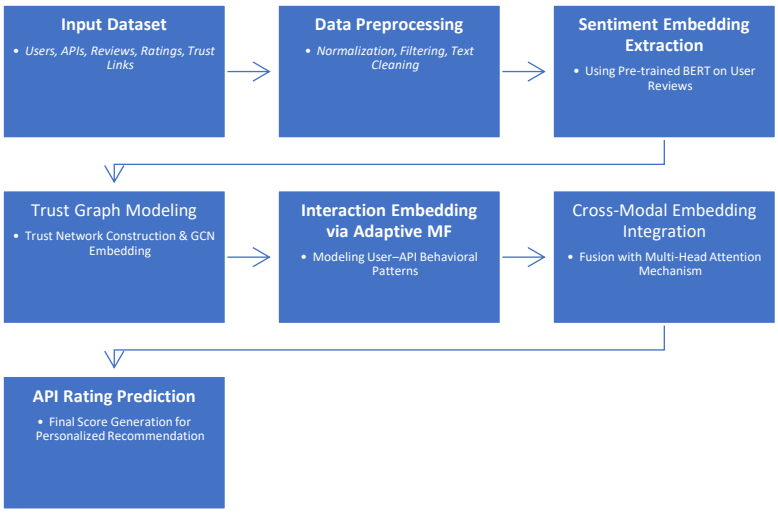

Fig. 1. Block diagram of proposed method

The architecture of the **SATR-API** model is designed around five distinct modules, each responsible for capturing and integrating a specific modality of information—sentiment, trust, and user-API interactions—within a unified, attention-driven recommendation framework. The modules are as follows:

1. **Sentiment Representation Module:** This module extracts sentiment-aware features from user-generated reviews about APIs. By utilizing a pre-trained BERT model, deep contextual embeddings are generated to reflect users' emotional attitudes toward each API. The resulting sentiment vectors capture the polarity (positive, negative, or neutral) of opinions and are used in the recommendation process.
2. **Trust Graph Construction and Embedding Module:** In this module, a social trust graph is constructed based on explicit or inferred trust data among users (e.g., similarity in behavior, historical trust signals). Graph Convolutional Networks (GCNs) are then applied to this graph to learn trust-based user embeddings that reflect social influence and credibility in the recommendation process.
3. **Interaction Modeling Module:** This component models user preferences and API properties via adaptive matrix factorization. It learns latent vectors for both users and APIs by factoring in their rating interactions. These embeddings serve as the core representation of users' behavioral patterns.
4. **Cross-Modal Embedding Integration Module:** In this stage, the sentiment vectors, trust embeddings, and interaction-based representations are integrated using a multi-head attention mechanism. This mechanism enables the model to dynamically weigh each modality based on its contextual relevance, allowing for a more fine-grained and robust fusion of information.
5. **Score Prediction Module:** Finally, the integrated user and API representations are fed into a prediction layer to estimate the rating score. This score reflects the personalized recommendation output, considering emotional, social, and behavioral dimensions.

Figure 2 presents the full architecture of the proposed SATR-API model, highlighting the flow and interaction between these five modules.

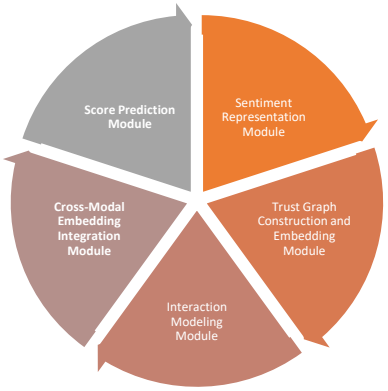

Fig. 2. Five modules of the proposed SATR-API model

DOI: 10.46121/pspc.54.1.20

Power System Protection and Control
ISSN:1674-3415

The proposed model begins by receiving raw inputs, including user profiles, API metadata, textual reviews, numeric ratings, and trust-related signals among users. The architecture is organized into five interdependent modules:

1. **Sentiment Representation Module**: In this initial stage, user-written reviews about APIs are processed using a pre-trained BERT model. Each comment is transformed into a contextualized sentiment embedding that captures the user's emotional inclination (positive, neutral, or negative) toward the API.
2. **Trust Graph Construction and Embedding Module**: Next, a social trust graph is built using explicit or inferred relationships (e.g., following behavior, co-rating patterns, or profile similarity). A Graph Convolutional Network (GCN) is applied to this graph to generate trust-aware embeddings for each user, modeling their trustworthiness and social influence in the community.
3. **Interaction Modeling Module**: This module employs adaptive matrix factorization to learn latent feature vectors for users and APIs, based on their rating history. Unlike conventional MF methods, this model incorporates the previously obtained sentiment and trust embeddings to enrich the user vector, capturing behavioral and contextual aspects.
4. **Cross-Modal Embedding Integration Module**: A multi-head attention mechanism is utilized to fuse the three sources of information—interaction-based, sentiment-based, and trust-based representations. This attention mechanism dynamically assigns different weights to each modality, allowing the model to emphasize the most informative signals for each user-API pair.
5. **Score Prediction Module**: Finally, the integrated user vector is combined with the latent API vector through a dot product operation to predict the user's rating. This predicted score reflects a comprehensive and personalized assessment, informed by emotional, social, and behavioral dimensions.

This five-module architecture enables the SATR-API model to effectively utilize **cross-modal information** and adapt to various types of user feedback, thereby improving the **precision and personalization** of API recommendations in dynamic cloud-based environments.

3-1 Text Preprocessing

Each review authored by user *u* for API *i*, undergoes a carefully designed text preprocessing pipeline to prepare it for embedding via a pre-trained language model. These steps are essential to ensure semantic clarity, remove noise, and maintain compatibility with the tokenization and embedding process. **Figure 3** illustrates the structure of this preprocessing pipeline.

Fig. 3. Block diagram of the text preprocessing pipeline

As shown in Figure 3, the preprocessing module consists of the following steps:

- **Noise Removal**: All non-linguistic elements such as punctuation, emojis, HTML tags, special symbols, and non-ASCII characters are removed to clean the raw text.
- **Normalization**: Text is standardized by converting to lowercase (for English) and applying character-level normalization strategies in languages with complex morphology (e.g., Persian). This reduces variability due to orthographic inconsistencies.
- **Stopwords Elimination**: Common Stopwords that provide limited semantic content are removed to improve the signal-to-noise ratio for the embedding model.
- **Model-Specific Tokenization**: The cleaned and normalized text is tokenized using the tokenizer associated with the selected pre-trained transformer model (e.g., BERT or DistilBERT). This step ensures that the input format aligns with the model's vocabulary and segmentation scheme.

The output of this module is a cleaned and tokenized version of each review, ready for embedding extraction in the sentiment representation module. This preprocessing pipeline plays a critical role in preserving semantic integrity and enabling accurate downstream sentiment modeling.

DOI: 10.46121/pspc.54.1.20

Power System Protection and Control
ISSN:1674-3415

3-2 Sentiment Embedding Extraction Using Pre-trained BERT

To effectively capture users' emotional attitudes toward APIs, this module employs a pre-trained BERT model to extract high-quality sentiment embeddings from user-generated textual reviews. These embeddings play a pivotal role in modeling user preferences beyond numeric ratings.

Once the reviews are preprocessed (as described in Section 3.1), each cleaned and tokenized comment is passed through the BERT encoder. BERT's bidirectional attention mechanism allows the model to capture complex semantic and syntactic relationships, making it well-suited for understanding subtle emotional cues in user feedback.

Specifically, the [CLS] token embedding from the final layer of BERT is used to represent the overall sentiment vector for each review. This vector reflects the polarity (positive, neutral, or negative) and intensity of the user's opinion toward the target API.

Formally, for a given review , the sentiment embedding is computed as:

(1)

where denotes the dimension of the BERT hidden state (typically 768 in BERT-base).

These sentiment embeddings are then stored and later used in the multi-modal fusion process (Section 3.5), where they are integrated with trust and interaction features. Incorporating these representations allows the model to personalize API recommendations not only based on behavior but also on users' subjective perceptions and emotional tone.

```
Input:
   Reviews = { | textual review from user u for API i}
   PretrainedBERT = BERT language model (e.g., BERT-base)

Output:
   SentimentEmbeddings = { | sentiment vector for each }

Procedure:
1. Initialize SentimentEmbeddings = empty dictionary

2. For each review  in Reviews:
   a. CleanedText ← Preprocess()              // remove noise, normalize, etc.
   b. TokenizedText ← Tokenize(CleanedText) // use BERT tokenizer
   c. BERTOutput ← PretrainedBERT(TokenizedText) // forward pass
   d. SentimentVector ← BERTOutput[CLS]       // extract [CLS] token representation
   e. Store  in SentimentEmbeddings with key (u, i)

3. Return SentimentEmbeddings
```

Fig. 4. Pseudo-code of Sentiment Embedding Extraction Using Pre-trained BERT

3-3 Modeling the Trust Graph Between Users

One of the core components of the **SATR-API** framework is the explicit modeling of inter-user trust as a social signal that complements user preferences and sentiment. In real-world recommendation environments—especially in developer communities—users often rely more on the opinions of those they trust or consider technically reliable. To capture this phenomenon, we construct a **directed, weighted trust graph** where each user is represented as a node, and the trust between users is encoded as edges with non-negative weights.

Formally, the trust graph is defined as:

(2)

Where, is the set of users. is the set of directed edges, representing trust links between users. is a weighting function that assigns a trust score to each edge , indicating the level of trust user has in user .

This information is encoded into a **trust adjacency matrix** , defined as:

(3)

The GCN propagation rule at layer is defined as:

(4)

DOI: 10.46121/pspc.54.1.20

Where, is the adjacency matrix with added self-loops. is the diagonal degree matrix of . is the trainable weight matrix for the -th GCN layer. is an activation function such as ReLU.

The final output of this module is a trust embedding vector for each user , capturing the user's social influence, credibility, and position in the trust network. These vectors are passed to the cross-modal fusion module (Section 3.5) and used in conjunction with sentiment and interaction features for rating prediction.

By incorporating GCN-based trust embeddings, SATR-API can exploit both **explicit trust signals and latent social structure**, leading to more context-aware and reliable API recommendations.

Input:
← Weighted trust adjacency matrix (size:
← Initial user features (e.g., one-hot or identity matrix)
← Number of GCN layers
← Trainable weight matrices for each layer

Output:
Trust embeddings for users (dimension:)

Procedure:
1. Add self-loops:
2. Compute degree matrix: ← diagonal matrix of
3. Normalize adjacency:

4. Initialize H ← X

5. For l = 1 to L:

6. Return final embedding matrix

Fig. 5. Pseudocode of Trust Feature Extraction with GCN

3-4 Modeling User–API Behavioral Patterns via Adaptive Matrix Factorization

To capture the behavioral patterns of users toward APIs, matrix factorization (MF) remains one of the most effective and scalable techniques in recommender systems. Traditional MF models assume that both users and APIs can be represented in a shared latent space, and the interaction (e.g., rating) between a user and an API is modeled as the dot product between their respective vectors.

However, standard MF techniques lack the ability to integrate auxiliary contextual signals such as user sentiment or trust relationships. In the SATR-API framework, we address this limitation by introducing an adaptive matrix factorization module that enriches the user embeddings with multi-modal information.

Each user's representation is formed not just from rating patterns, but also from:

- Their emotional attitude toward APIs (extracted from review texts),
- Their social trust profile (learned from the trust graph),
- And their historical rating interactions.

The fusion of these modalities is performed through a multi-layer perceptron (MLP), which receives a concatenation of:

- Basic latent vector from MF:
- Sentiment vector:
- Trust vector: .

The updated user embedding is then used to compute the final predicted rating:

(5)

This adaptive design allows the model to better reflect the user's true intent and contextual preferences, especially in complex or sparse environments.

DOI: 10.46121/pspc.54.1.20

Power System Protection and Control
ISSN:1674-3415

Input:
User-API rating matrix
Initial user latent vectors (size: |U| × d)
API latent vectors (size: |I| × d)
Sentiment embeddings for (u, i) pairs
Trust embeddings for users
Multi-layer perceptron for fusion

Output:
Predicted rating matrix

Procedure:
1. For each user u and API i where rating exists:
a.
b.
c.
d.
e.
f.
g.
2. Return

Fig. 6. Pseudocode Adaptive Matrix Factorization with Cross-Modal Embedding Fusion

This module provides a **personalized, context-aware embedding** for each user by blending numerical, emotional, and social signals. The MLP serves as a nonlinear integration mechanism, ensuring that the relative importance of each modality (e.g., trust vs. sentiment) is learned automatically during training. The result is a more robust and accurate prediction of how likely a user is to prefer a given API.

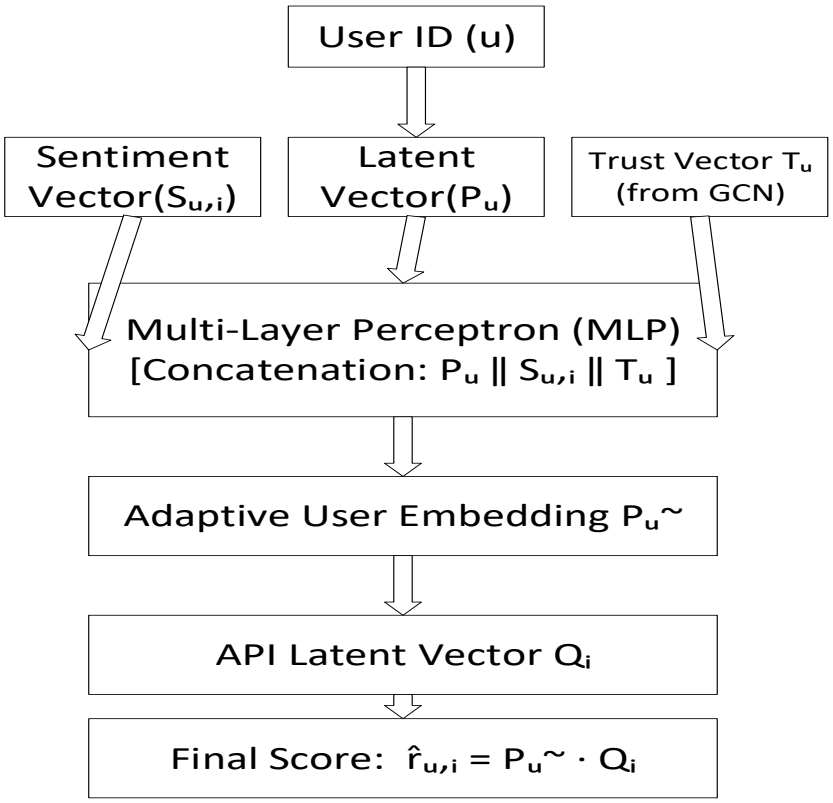

Fig.7. Interaction Embedding via Adaptive Matrix Factorization

DOI: 10.46121/pspc.54.1.20

Power System Protection and Control
ISSN:1674-3415

3-5 Cross-Modal Embedding Integration: Fusion with Multi-Head Attention Mechanism

In the SATR-API framework, after extracting the individual feature representations—namely, the interaction embeddings from adaptive matrix factorization, the sentiment embeddings from user comments, and the trust embeddings from the social trust graph—it is essential to integrate these heterogeneous sources in a coherent and effective manner to form a unified user representation. To this end, a Multi-Head Attention (MHA) mechanism is employed as the fusion module for cross-modal embedding integration.

Unlike traditional fusion strategies such as simple concatenation or averaging, the multi-head attention mechanism allows the model to dynamically learn the relative importance of each modality (interaction, sentiment, trust) based on the context of the prediction task. This design enhances the model's capacity to capture complex dependencies and to weigh features differently for different users or APIs.

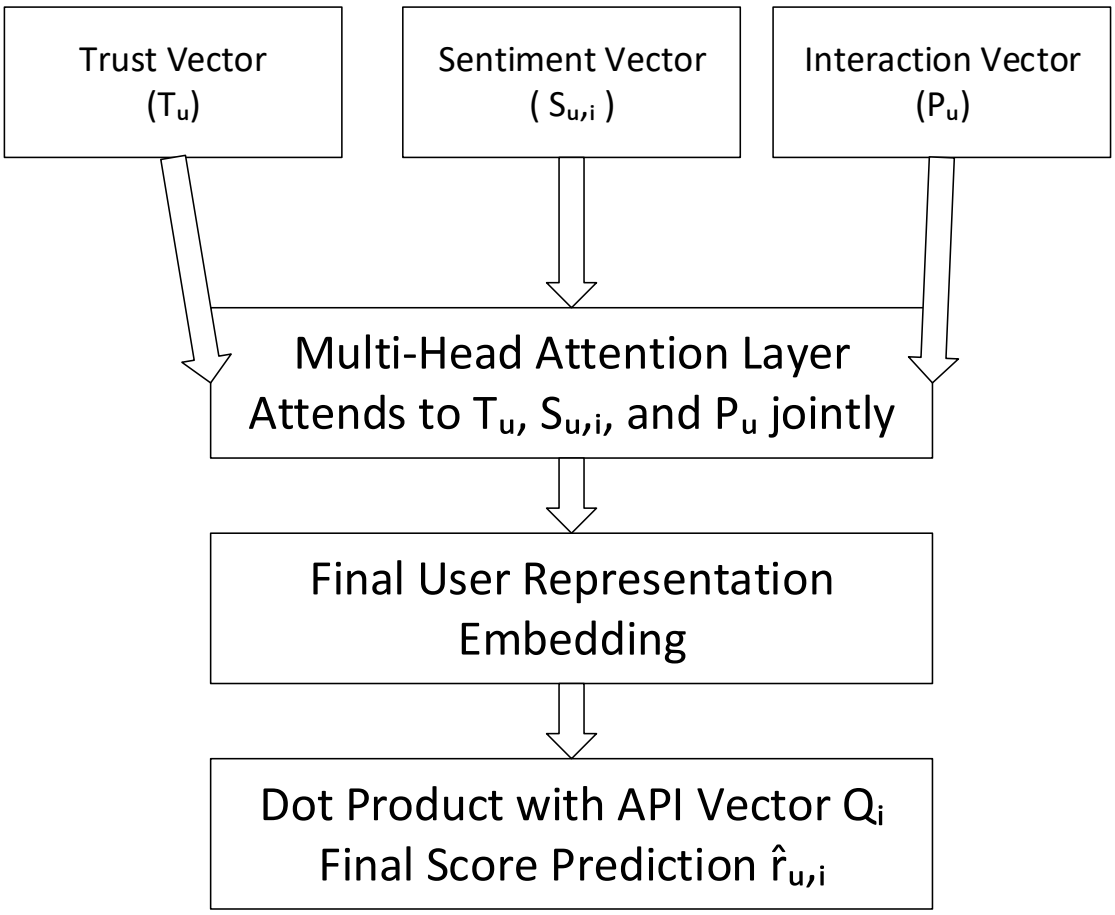

Fig. 8. Multi-Head Attention for Cross-Modal Feature Fusion

3-5-1 Attention-based Fusion Process:

Let:

- be the user latent vector from adaptive matrix factorization,
- be the sentiment embedding from BERT,
- be the trust embedding from GCN.

These vectors are first projected into a common latent space using learnable linear transformations:

(6)

Where ,, are the projection matrices and denotes vector concatenation.

Each head of the attention module computes:

(7)

The outputs of all heads are concatenated and passed through a feed-forward layer to produce the final fused user embedding:

(8)

DOI: 10.46121/pspc.54.1.20

Finally, the predicted score for user and API is computed by:

(9)

This fusion approach provides the flexibility to capture **non-linear dependencies**, **context-sensitive interactions**, and **personalized weightings** across modalities. As a result, the model is better equipped to deliver highly accurate and context-aware API recommendations.

3-6 API Rating Prediction: Final Score Generation for Personalized Recommendation

After integrating sentiment, trust, and interaction features through the multi-head attention mechanism, the SATR-API framework produces a final **fused user representation** that captures comprehensive behavioral and contextual preferences of user . This vector is then used to compute the **personalized rating score** for a target API using the latent representation of that API.

The final predicted score is generated through a **dot product** between the fused user embedding and the API embedding, followed by a non-linear activation function (such as sigmoid or tanh) to scale the output within the desired rating range:

(10)

Where is the predicted rating that user u is likely to give to API , σ(·) is a bounded activation function (e.g., sigmoid), is the latent representation of API , learned jointly during training.

This score serves as the **final output** of the SATR-API model and determines the **ranking** of APIs when presented as recommendations to the user. APIs with higher scores are considered more suitable or relevant, and are placed higher in the recommendation list.

The recommendation process is designed to be:

- **Personalized**: accounts for each user's interaction, opinion, and social perspective.
- **Context-aware**: includes sentiment and trust as situational modifiers.
- **Scalable**: suitable for large-scale API repositories in real cloud environments.

As such, the SATR-API framework not only predicts user preferences with high accuracy but also enhances the quality and interpretability of API selection in software development tasks.

EXPERIMENTS AND RESULTS

In this section, the performance of the proposed method is investigated based on real datasets and standard evaluation criteria. First, in Section 4.1, the Amazon Web Services (AWS) dataset, which includes user ratings of APIs along with their textual comments, is introduced. This data contains a variety of information from user interactions with various web services and APIs and is widely used to evaluate recommender systems.

Next, in Section 4.2, the evaluation criteria including mean absolute error (MAE), root mean square error (RMSE), and its normalized version (NRMSE) are introduced. Then, in Section 4.3, the results obtained from running the proposed model on this data are presented. In Section 4.4, several reference models including classical and deep learning methods are introduced. Finally, Section 4.5 is dedicated to a comprehensive performance comparison of the proposed model with reference methods.

Figure 9 shows an overview of the model evaluation process, including how to process input data, extract features, train the model, and analyze the results.

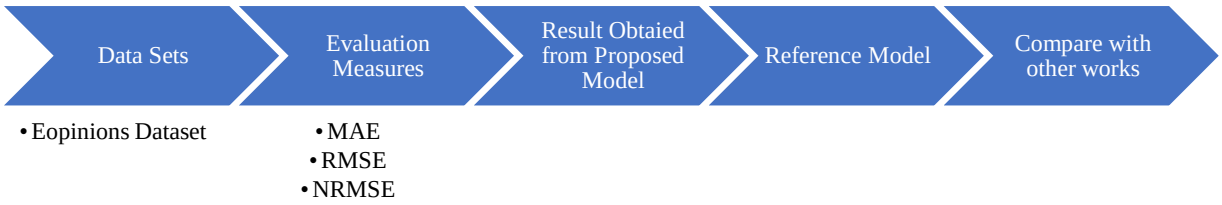

Fig. 9. Evaluation Process of proposed method.

DOI: 10.46121/pspc.54.1.20

4-1 Datasets

In this study, the Epinions dataset was used. This dataset contains information such as users, ratings, text comments, and trust relationships between users. In this dataset, each user has provided at least one comment. The input information includes user ID, item ID, rating, rating time, comment text, and trust relationships between users. This dataset is widely used to evaluate recommender systems.

The table 2 summarizes the statistical information of this dataset

:

Table 1: Summarization the statistical information of this dataset

Feature	Values
Number of users	49,290
Number of items	139,738
Number of comments	664,824
Number of trust relationships	487,181

4-2 evaluation criteria

To test and evaluate the proposed recommender system based on the analysis of users' sentiments and opinions, we use the mean absolute error (MAE) (equation (11)) and the root mean square error (RMSE) (equation (12)) and NRMSE (equation (13)) to evaluate the deviation between the predicted ratings and the actual ratings. They are the most widely used statistical accuracy measures for recommender systems based on collaborative filtering.

(11)

(12)

(13)

In the above equation, is the actual rank, is the predicted rank. n is the total number of test samples on the set T. These two metrics measure how much our predicted rating deviates from the actual rating. A smaller value indicates a more accurate prediction. The value of and is equal to the value of the maximum and minimum rating. This criterion is for RMSE normalization and the lower the better.

4-3 Hyperparameter Settings and test results

In this section, we compare the accuracy of the proposed method with other reference methods. For testing, 5-fold cross validation test method is used, so that 80% of samples are used for training and 20% of samples are used for testing. This operation is performed 5 times and the average is taken. Finally, RMSE, MAE and NRMSE criteria are extracted.

In order to train and fine-tune the proposed model effectively, it is essential to identify and configure several key hyperparameters, each corresponding to a distinct module in our hybrid architecture. Since our model integrates sentiment analysis, trust-aware graph modeling, matrix factorization, and multi-head attention, each component introduces its own set of tunable parameters.

The table 3 summarizes the most significant hyperparameters for each module, along with concise descriptions of their roles and impacts within the overall framework.

Table 2. Important Hyperparameters in Each Module

Module	**Hyperparameter**	**Symbol**	**Description**
Sentiment Analysis	Maximum input length	L	Maximum sequence length passed to the BERT model
	Pre-trained BERT type		Choice of BERT model variant (e.g., base, multilingual)
Trust Modeling (GCN)	Number of GCN layers		Depth of graph convolutional layers for trust propagation
	Dropout rate in GCN		Dropout to regularize GCN
Matrix Factorization	Latent dimension	k	Number of latent factors (embedding size)
	Regularization coefficient		Controls overfitting in matrix factorization
Attention Mechanism	Number of attention heads		Number of parallel attention heads in MHA

DOI: 10.46121/pspc.54.1.20

Attention dropout	Dropout applied to attention weights

For the hyperparameters introduced in the table above, experiments were carried out with different values for each hyperparameter, and the evaluation criteria NRMSE, RMSE, and MAE were calculated.

Maximum Input Length (l_{max}): The maximum input length for language models like BERT determines the amount of text that can be processed in a single instance. Choosing the right value has a direct impact on extracting semantic features and paying attention to key parts of user comments. Too short a length will result in information being omitted, and too long a length can drastically increase memory and time. Figure 10 shows the effect of Maximum Input Length (l_{max}) on performance of proposed method:

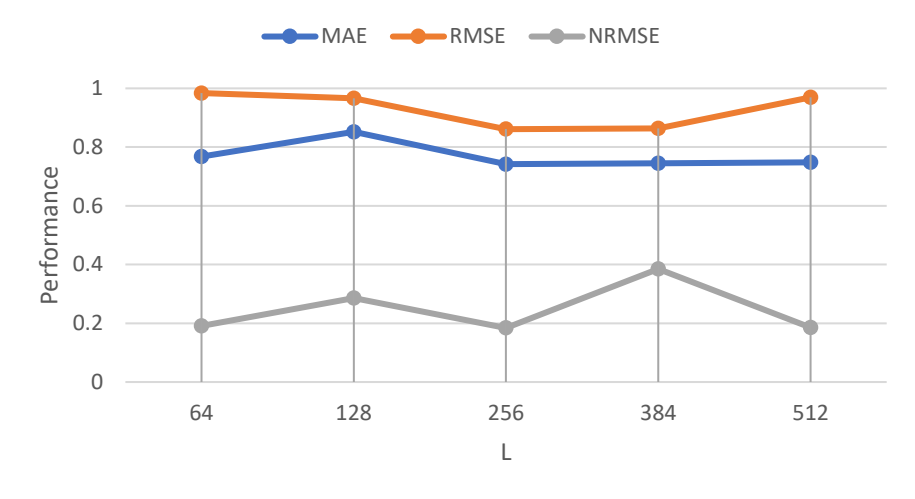

Fig. 10. Effect of Maximum Input Length (l_{max}) on performance of proposed method

The results show that a value of 256 for the input length performs best in the evaluations. This value is sufficient to cover most complete comments, without slowing down or increasing the complexity of the model with very long inputs. Lower values such as 64 or 128 ignore too much information, and higher values do not show much improvement.

2) Pre-trained BERT Type (M): The pre-trained BERT model plays a key role in the quality of text feature extraction. In this research, several different types of general, specialized, and lightweight BERT models were investigated to evaluate the impact of different architectures on the results. Figure 11 shows the effect of Pre-trained BERT Type (M) on performance of proposed method.

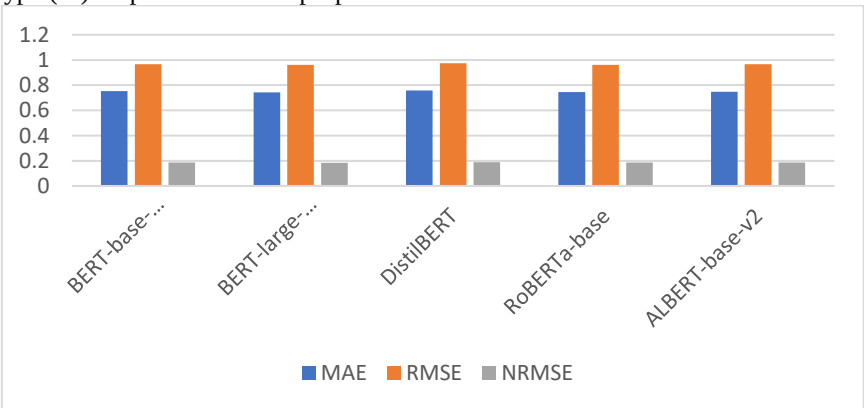

Fig. 11. Effect of Pre-trained BERT Type (M) on performance of proposed method

The BERT-large model showed better performance than other models. Although it is heavier than compact models such as DistilBERT or ALBERT, it has higher accuracy in extracting text features and thus reduces errors. Therefore, despite the higher computational cost, the use of BERT-large is justified.

DOI: 10.46121/pspc.54.1.20

3) Number of GCN Layers (L): The number of layers in a graph neural network (GCN) in learning graph features plays an important role in the depth of extracting social relationships between users. More layers can account for more distant relationships, but may lead to "oversmoothing" where all nodes converge to similar features. Figure 12 shows the effect of Number of GCN Layers (L) on performance of proposed method.

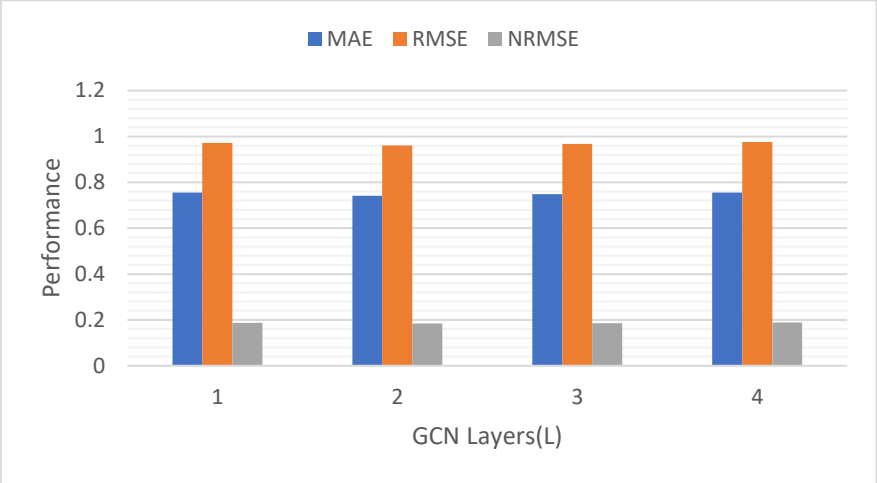

Fig. 12. Effect of Number of GCN Layers (L) on performance of proposed method

Using two GCN layers has yielded the best results. One layer is limited to conveying information, and the higher layers cause excessive convergence of features and reduce the differentiation between users. Therefore, the optimal value of 2 is suggested here.

4)Dropout Rate in GCN (p_{GCN}): Dropout is used in graph layers to increase generalizability and prevent overlearning of specific relationships between nodes. A suitable dropout rate can help improve performance, while extreme values reduce learning capacity. Figure 13 shows effect of Dropout Rate in GCN (p_{GCN}) on performance of proposed method.

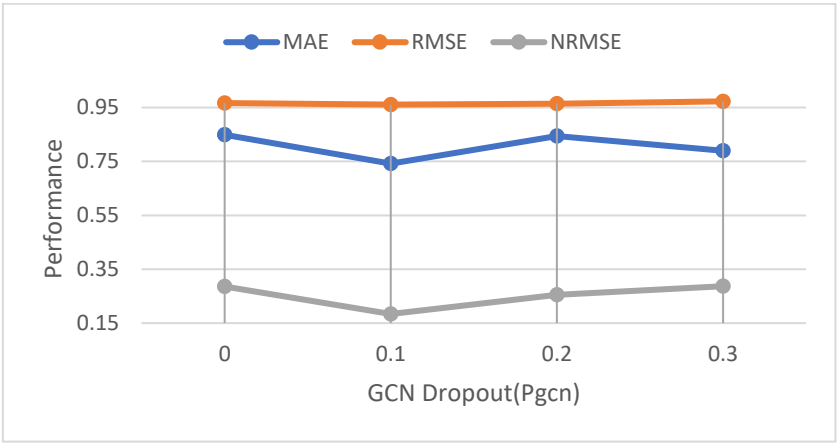

Fig. 13. Effect of Dropout Rate in GCN (p_{GCN}) on performance of proposed method

The best model performance is seen at a rate of 0.1. No dropout (0.0) leads to overfitting, and high values such as 0.5 lead to excessive information removal. Therefore, a rate of 10% is recommended.

5) Latent Dimension (d): Latent dimension plays a key role in modeling feature vectors of users and items. Proper selection of this dimension not only improves the quality of representation, but also has a direct impact on learning latent patterns. Low dimensions may represent insufficient information, while high dimensions can lead to overfitting. Figure 14 shows effect of Number of Latent Dimension (d) on performance of proposed method.

DOI: 10.46121/pspc.54.1.20

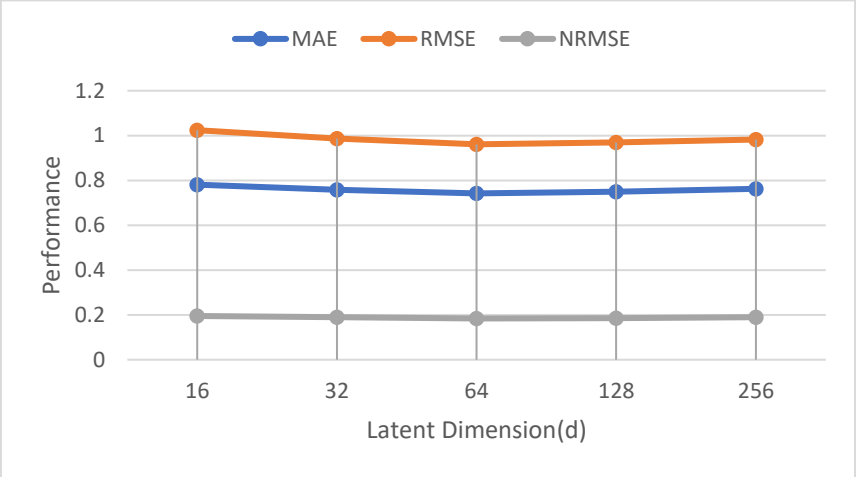

Fig. 14. Effect of Number of Latent Dimension (*d*) on performance of proposed method

It is observed that the value of d=64 resulted in the best performance. At lower values (such as 16 and 32), less information from users and items was encoded, which resulted in a decrease in the accuracy of the model. In contrast, at higher values (128 and 256), despite the increase in dimensions, the accuracy decreased due to excessive complexity and overfitting. Therefore, the optimal value of 64 was chosen.

6) Regularization Coefficient (λ): Regularization coefficients are used to prevent overfitting in machine learning models. Choosing an appropriate value for λ allows the model to both remain generalizable and avoid overfitting to the training data. This parameter is usually applied to the weight matrix or latent vectors. Figure 15 shows the effect of Regularization Coefficient (λ) on performance of proposed method.

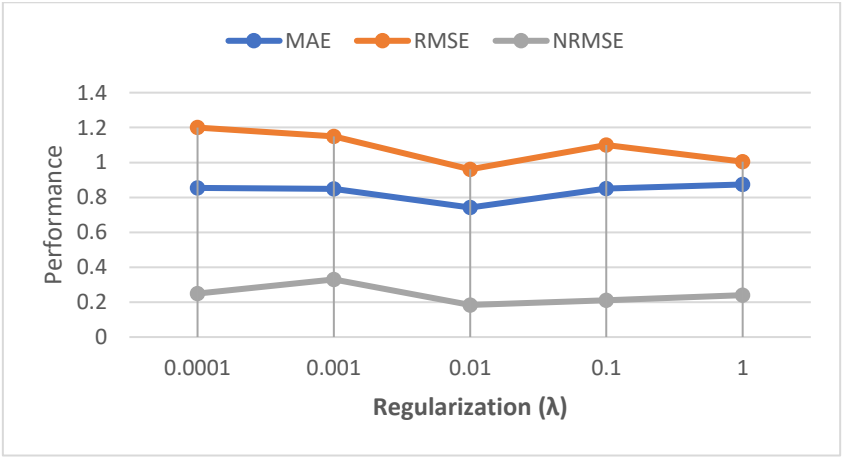

Fig. 15. Effect of Regularization Coefficient (λ) on performance of proposed method

The results show that the value of λ=0.01 is considered to be the optimal value, which results in the minimum MAE and RMSE. At very small values such as 0.0001, there is not enough regularization and the model approaches overfitting. At large values such as 1.0, the model is too constrained and learning is impaired.

7)Number of Attention Heads (*H*): The number of attention heads (Multi-head Attention) in Transformer-based architectures determines how many parallel and independent attention spaces there are to combine information. This value can affect the model's ability to extract diverse features. Figure 16 shows the effect of Number of Attention Heads (*H*) on performance of proposed method

DOI: 10.46121/pspc.54.1.20

Power System Protection and Control
ISSN:1674-3415

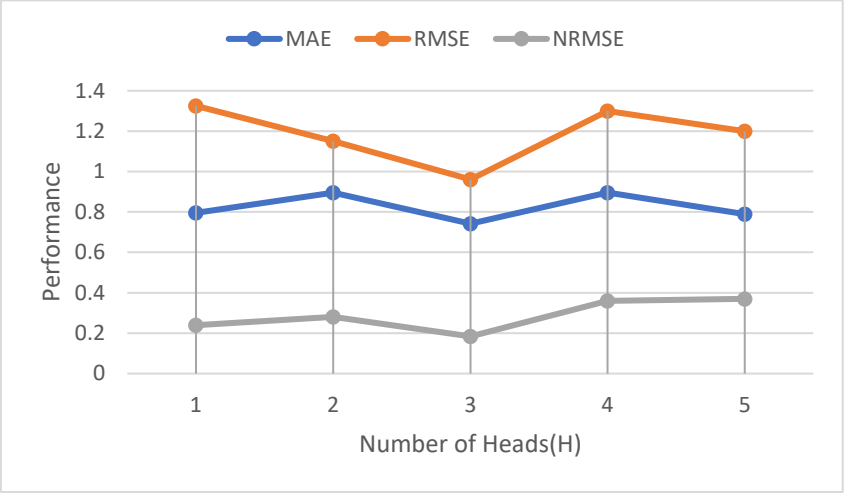

Fig. 16. Effect of Number of Attention Heads (H) on performance of proposed method

Four attention heads (H=4) have shown optimal performance. Using one or two attention heads has reduced the model's ability to extract diverse features, while a large number (8 or 16) have not provided significant improvement despite increasing complexity and even reduced accuracy.

8)Attention Dropout Rate (p_a): Dropout is applied in the attention module to prevent the model from being overly dependent on one or more specific relationships between inputs. By adjusting its rate, a balance between generalizability and better learning can be achieved. Figure 17 shows the effect of Attention Dropout Rate (p_a) on performance of proposed method.

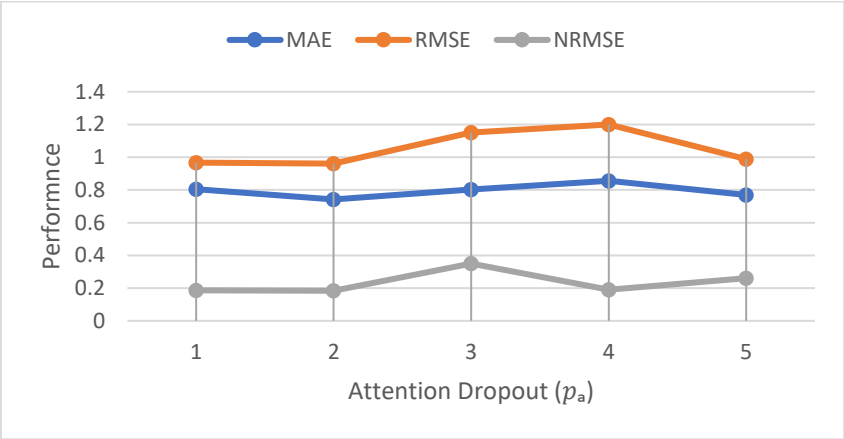

Fig. 17. Effect of Attention Dropout Rate (p_a) on performance of proposed method

The results show that a dropout rate of 0.1 in the attention layer performs best. In the absence of dropout (0.0), the probability of overfitting increases, and at large values such as 0.5, vital information is lost during learning.

A. 4-4 Reference systems

In this section, we introduce several models to compare the performance of the proposed RSMF model. These systems include the basic matrix factorization model (MF), the probability matrix factorization model (PMF), the sentiment-based matrix factorization model (SBMF), the SBMF+R matrix factorization model, and the matrix factorization model with the interaction factor (EnSocialMF).

Base matrix factorization model: This model is used as the base model for comparison. This model was first developed by Koren et al. in 2009 [9].

DOI: 10.46121/pspc.54.1.20

Power System Protection and Control
ISSN:1674-3415

Probability matrix factorization model (PMF): This model is similar to the basic model and is based on the Gaussian hypothesis [10].

The sentiment-based matrix factorization model (SBMF): In this model, sentiment analysis is added to numerical ranking [13].

SBMF+R matrix factorization model: In this model, sentiment analysis with certainty capability for numerical ranking and review has been added to numerical ranking [13].

Matrix factorization model with interaction factor (EnSocialMF): This model is a combination of probability matrix factorization model with user interaction factor [7].

In MF, PMF and EnSocialMF models, numerical ranking matrix is considered as input. While in the proposed model (RSMF) as well as SBMF and SBMF+R, numerical rating and sentiment and reputation information of users are considered as inputs. The results below show the advantages of the combination of reputation and sentiment compared to the basic MF, PMF and EnSocialMF models with numerical ratings.

4-5 Comparison with previous works

Table 3 and Figure 18 compare the NRMSE and MAE of the proposed method with the reference methods of matrix factorization model (MF), probability matrix factorization model (PMF), matrix factorization model based on sentiment (SBMF), matrix factorization model SBMF+R and factorization model The matrix deals with the interaction factor (EnSocialMF).

Table 3: Comparison of (MAE-RMSE-NRMSE) values of the proposed method with previous methods.

Methods	MAE	NRMSE	MSE
MF [9]	0.842	0.236	0.985
PMF [10]	0.817	0.224	0.926
SBMF [11]	0.795	0.218	0.892
SBMF+R [11]	0.783	0.211	0.871
EnSocialMF [7]	0.765	0.204	0.851
Proposed Method	0.742	0.184	0.783

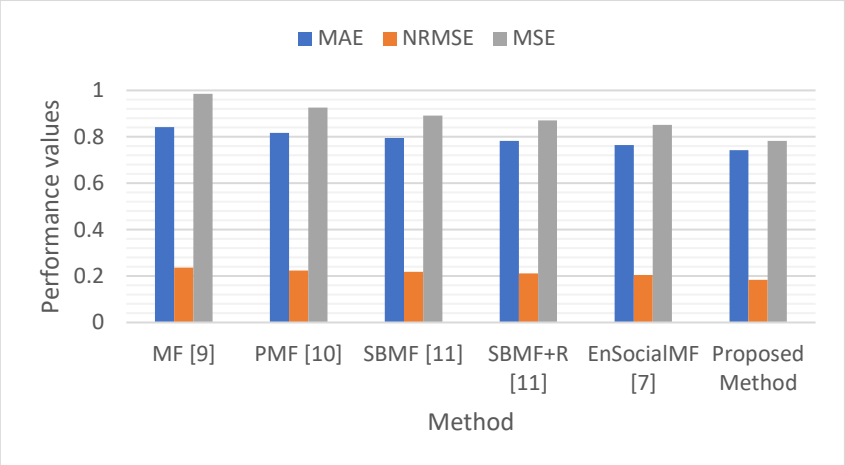

Fig. 18. Comparison of (MAE-RMSE-NRMSE) values of the proposed method with previous methods

Comparing the performance results of the proposed method with other valid methods in the table presented, shows the significant superiority of our model in all evaluation criteria including MAE, NRMSE and MSE. The trend of error reduction from traditional methods such as MF and PMF towards more complex models such as SBMF+R and EnSocialMF indicates the importance of utilizing side information such as social relationships, trust and sentiment analysis in improving the accuracy of recommender systems.

DOI: 10.46121/pspc.54.1.20

In the early methods such as MF and PMF, only the ranking matrix was used for prediction and no additional knowledge of user interactions or text content was considered. This limitation caused the error rate to remain relatively high. By introducing trust information in SBMF and SBMF+R, the models were able to benefit from social interactions between users and achieve better accuracy in estimating ratings. Combining trust relationships with the matrix structure improved the model's power in learning hidden features and a significant reduction in error measures was observed.

The EnSocialMF model, by combining social features, has shown further improvement over SBMF+R and has been able to obtain better results in various criteria. However, our proposed method with a multi-module approach goes a step further. Combining the three main components including sentiment analysis of user opinions, trust analysis between users and matrix factorization along with the use of the Multi-head Attention mechanism has made it possible to better extract and aggregate key and more important information. In particular, the use of the attention mechanism in combining multi-objective features has led to the model focusing more on effective features and reducing the error dispersion. As a result, the MAE criterion in the proposed model has decreased to 0.742, NRMSE to 0.184 and MSE to 0.783, which shows better results than all the compared methods. These numbers show that the model not only succeeded in predicting rankings more accurately, but also minimized distributed error and dispersion.

Overall, the results show that the proposed approach, by utilizing deep learning, integrating contextual and social features, and using attention structure, has taken an effective step in improving the performance of recommender systems and has shown significant superiority in accuracy and error reduction compared to previous methods.

CONCLUSION AND FUTURE WORK

In this study, an innovative hybrid method was presented to improve the accuracy of API recommender systems, which, by simultaneously utilizing user sentiment analysis, social trust modeling, matrix factorization, and the Multi-Head Attention mechanism, was able to perform better than existing methods. Sentiment analysis using pre-trained linguistic models such as BERT resulted in a deeper understanding of users' attitudes towards APIs. Also, by modeling the user trust graph through GCN, social connections and trust between users were included in the model structure, which enriched the contextual information. Factorization of the adaptive matrix also improved the estimation of user preferences by considering contextual and social features. Finally, combining this information with the attention mechanism resulted in their intelligent integration and significantly increased the prediction accuracy.

Experimental results on real datasets showed that the proposed method outperforms other valid methods such as MF, PMF, SBMF and EnSocialMF in all main evaluation criteria, including MAE, RMSE, NRMSE and MSE. Also, sensitivity analyses showed that the correct selection of hyperparameters plays a decisive role in optimizing the final performance of the model.

In the future, the proposed model can be improved by considering the temporal dynamics of user preferences and expanding to heterogeneous graphs. Also, using self-adjusting component weight methods and lighter models such as DistilBERT to reduce computational complexity are other development paths for this research.

Code availability Not applicable**.**
Author contribution Not applicable**.**
Data availability Not applicable.
Declarations
Conflict of interest Not applicable**.**

REFERENCES

1. Chen, Zhen, Taiyu Bao, Wenchao Qi, Dianlong You, Linlin Liu, and Limin Shen. "Poisoning QoS-aware cloud API recommender system with generative adversarial network attack." Expert Systems with Applications 238 (2024): 121630.

DOI: 10.46121/pspc.54.1.20

2. Alhosaini, Hadeel, Sultan Alharbi, Xianzhi Wang, and Guandong Xu. "API recommendation for mashup creation: A comprehensive survey." The Computer Journal 67, no. 5 (2024): 1920-1940.
3. Qin, Shaowei, Yiji Zhao, Hao Wu, Lei Zhang, and Qiang He. "Harnessing the Power of Large Language Model for Effective Web API Recommendation." IEEE Transactions on Industrial Informatics (2025).
4. Chen, Zhen, Jianqiang Yu, Shuang Fan, Jing Zhao, and Dianlong You. "Latent diffusion model-based data poisoning attack against QoS-aware cloud API recommender system." Computer Networks (2025): 111120.
5. Wang, Ye, Weihao Xue, Qiao Huang, Bo Jiang, and Hua Zhang. "Exploring ChatGPT's Potential in Java API Method Recommendation: An Empirical Study." Journal of Software: Evolution and Process 37, no. 1 (2025): e2765.
6. He, Yuan, Ali BM Ali, Saman Ahmad Aminian, Kamal Sharma, Saurav Dixit, Sakshi Sobti, Rifaqat Ali, M. Ahemedei, Husam Rajab, and Maryam Alsadat Ziaei Mazinan. "Enhancing Intelligent HVAC optimization with graph attention networks and stacking ensemble learning, a recommender system approach in Shenzhen Qianhai Smart Community." Scientific Reports 15, no. 1 (2025): 5119.Applications 539 (2020): 122950.
7. R. Chen, Y.-S. Chang, Q. Hua, Q. Gao, X. Ji and B. Wang, "An enhanced social matrix factorization model for recommendation based on social networks using social interaction factors," vol. 79, pp. 14147--14177, 2020.
8. S. G. K. Patro, B. K. Mishra, S. K. Panda, R. Kumar, H. V. Long and D. Taniar, "Cold start aware hybrid recommender system approach for E-commerce users," Soft Computing, vol. 27, no. 4, pp. 2071--2091, 2023.
9. Y. Koren, R. Bell and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, pp. 30--37, 2009.
10. A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," Advances in neural information processing systems, vol. 20, 2007.
11. R.-P. Shen, H.-R. Zhang, H. Yu and F. Min, "Sentiment based matrix factorization with reliability for recommendation," Expert Systems with Applications, vol. 135, pp. 249--258, 2019.
12. Rafiee, Samira, Chiman Salavati, and Alireza Abdollahpouri. "CNDP: Link prediction based on common neighbor's degree penalization." Physica A: Statistical Mechanics and its
13. Ibrahim, Taghreed S., Ahmed I. Saleh, Nehad Elgaml, and Mohamed M. Abdelsalam. "A fog based recommendation system for promoting the performance of E-Learning environments." Computers & Electrical Engineering 87 (2020): 106791.
14. Ghafouri, Seyyed Hamid, Seyyed Mohsen Hashemi, Mohammad Reza Razzazi, and Ali Movaghar. "Web service quality of service prediction via regional reputation-based matrix factorization." Concurrency and Computation: Practice and Experience 33, no. 17 (2021): e6318.
15. Hashemi, Seyyed Mohsen, Seyyed Hamid Ghafouri, Patrick CK Hung, and Chen Ding. "A new model for trustworthy web service QoS prediction." Concurrency and Computation: Practice and Experience 34, no. 6 (2022): e6778.

DOI: 10.46121/pspc.54.1.20

16. Ghafouri, Seyyed Hamid, Seyyed Mohsen Hashemi, and Patrick CK Hung. "A survey on web service QoS prediction methods." IEEE Transactions on Services Computing 15, no. 4 (2020): 2439-2454.
17. Duan, Rui, Cuiqing Jiang, and Hemant K. Jain. "Combining review-based collaborative filtering and matrix factorization: A solution to rating's sparsity problem." Decision Support Systems 156 (2022): 113748.
18. Liu, Huiting, Yi Chen, Peipei Li, Peng Zhao, and Xindong Wu. "Enhancing review-based user representation on learned social graph for recommendation." Knowledge-Based Systems 266 (2023): 110438.
19. Malandri, Lorenzo, Carlos Porcel, Frank Xing, Jesus Serrano-Guerrero, and Erik Cambria. "Soft computing for recommender systems and sentiment analysis." Applied Soft Computing 118 (2022): 108246.
20. Elahi, Mehdi, Danial Khosh Kholgh, Mohammad Sina Kiarostami, Mourad Oussalah, and Sorush Saghari. "Hybrid recommendation by incorporating the sentiment of product reviews." Information Sciences 625 (2023): 738-756.
21. Bangui, Hind, Mouzhi Ge, and Barbora Buhnova. "Deep-Learning based Reputation Model for Indirect Trust Management." Procedia Computer Science 220 (2023): 405-412.
22. Zolbanin, Hamed M., and Donald Wynn. "From star rating to sentiment rating: using textual content of online reviews to develop more effective reputation systems for peer-to-peer accommodation platforms." Journal of Business Analytics 6, no. 2 (2023): 127-139.
23. Pulikonda, Venkata Ayyappa, Dharmic Vanukuru, Mohammed Sehan Navali, Bhuvaneswari Motepalli, and K. Swathi. "Exploring the Applications, Challenges, and Issues of Sentiment Analysis." In 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), pp. 807-812. IEEE, 2023.
24. Devi, K. Dhana Sree, Rakesh Nayak, L. Lakshmi, and M. Ravisankar. "Generic View of Sentiment Analysis Using Machine Leaning Models." In 2022 OPJU International Technology Conference on Emerging Technologies for Sustainable Development (OTCON), pp. 1-6. IEEE, 2023.
25. Wu, Ziteng, Ding Ding, Yuting Xiu, Yuekun Zhao, and Jing Hong. "Robust QoS Prediction based on Reputation Integrated Graph Convolution Network." IEEE Transactions on Services Computing (2023).
26. Shashvat, Kumar, and Sandeep K. Sood. "Recommender for Reputation Assessment." Journal of Survey in Fisheries Sciences 10, no. 4S (2023): 662-678.
27. Rao, K. Yogeswara, G. S. N. Murthy, and S. Adinarayana. "Product recommendation system from users reviews using sentiment analysis." International Journal of Computer Applications 975 (2017): 8887.
28. Singh, Prince, Garima Srivastava, Shikha Singh, and Sachin Kumar. " intelligent movie recommender framework based on content-based & collaborative filtering assisted with sentiment analysis." International Journal of Advanced Research in Computer Science 14, no. 3 (2023).

DOI: 10.46121/pspc.54.1.20