CS 4641 ML Project final Report

Cayman is a clean, responsive theme for GitHub Pages.

View on GitHub Download .zip Download .tar.gz

Introduction/Background

Literature Review

The rating systems have evolved significantly. People focus on improving prediction accuracy and propose different machine learning methods to fulfill their requirements. Priya et al. (2022) propose a hybrid collaborative and content-based filtering approach utilizing a CNN-based ReLU network which demonstrated improved accuracy on the Movie Tweetings and Open Movie Database datasets. Liu and Singh (2018) explored recurrent neural networks (RNNs), particularly attention-based RNNs, to process user reviews, demonstrating improved accuracy in capturing user preferences through textual analysis. Lastly, Khan et al. (2020) conducted a comprehensive deep learning-based rating model. Deep learning plays a crucial role in addressing challenges like scalability, cold-start, and data sparsity in movie rating predictions.

Dataset Description

Our project will mainly use the dataset from Netflix Prize Data on Kaggle (Netflix Prize Data) for training and testing. Netflix held the Netflix Prize open competition for the best algorithm to predict user ratings for films years ago. This dataset was used in that Netflix prize competition. There are two parts to this dataset. The first part is a tar of a directory containing 17770 files, one per movie. The first line of each file contains the movie id followed by a colon. Each subsequent line in the file corresponds to a rating from a customer and its date in the following format: `CustomerID, Rating, Date`.

The second part is a file containing specific information about the movie with the following format: `MovieID, YearOfRelease, Title`.

With the above information, we want to make a prediction on a movie's rating solely based on the title of that movie to analyze how the naming of the movie would impact its rating. We also find several similar movie datasets that contain useful information such as IMDB dataset and TMDB dataset. In the future we can also potentially analyze how some specific factors such as genre or cast might influence the rating of a movie using these datasets.

Dataset Links

- TMDB 5000 Movie Data
- Netflix Prize Data
- IMDB 5000 Movie Data
- MovieLens 20M Dataset

Problem Definition

Problem

When selecting a movie to watch, the title is often the first and sometimes only information available to potential viewers. Despite this, the process of creating a compelling and effective movie title remains largely subjective. Filmmakers and marketers struggle to predict how a title might influence audience perception and ultimately a movie's success. The challenge lies in understanding how a simple string of words can convey critical information about the movie's genre, tone, or plot while appealing to a wide audience. Without clear data-driven insights, movie production companies are left with uncertainty about how a title might affect user ratings and viewer engagement, leading to potentially missed opportunities in marketing and audience retention.

Motivation

Given the impact that movie titles have on attracting and engaging audiences, there is a need for a systematic approach to evaluate and optimize them. A well-crafted title can boost viewership by effectively communicating the essence of the movie and aligning with audience expectations. By leveraging machine learning techniques, we can predict how a movie title may influence user ratings, providing valuable feedback for film studios during the marketing phase. This system offers the potential to refine the process of title selection, transforming it from a creative guesswork into a data-driven strategy that maximizes a movie's reach and appeal. This solution could significantly benefit filmmakers and production companies by helping them make informed decisions that lead to better audience perception and increased success at the box office or on streaming platforms.

Methods

Data Preprocessing

Ratings Value Accumulation

From the 4 `combined_data.txt` files, each movie's average rating and most frequent rating is calculated. This data is then stored in a pandas DataFrame where:

- The first column is the movie ID,
- The second column is the mean of the user ratings,
- The last column is the mode.

Inclusion of Movie Titles

The `movie_titles.csv` contains the following data format: `MovieID, YearOfRelease, Title`. So, a script was created to add a new column to the data frame that contains the movie title for each movie based on the movieID. Since the movie titles aren't encapsulated with "" and could contain "," in their names each line in the csv was split with the first two ","s. These titles are then added to a list which is then merged with the initial data frame. Finally, the data frame is sorted in ascending order with movieID as the key.

Vector Representation of Movie Titles

1. **Tokenize the Titles**: Convert titles to integer sequences, and pad the sequences to a fixed length. Since the titles can contain special characters, abbreviations, and phrases, treat each unique token (word, abbreviation, or symbol) as a standalone unit. Each token is split by

whitespace and treated as unique, so phrases like "IV" and "TT" are handled just like any other token.

2. **Padding**: After tokenizing, we pad sequences to a uniform length as the token can be of different lengths. This helps with matrix operations on the tokens.

3. **Embedding Model**: The PyTorch nn.Embedding layer is used to convert integer tokens into dense vector embeddings.

4. **Batch Size and Embedding Dimension**:

    - The batch size defines the number of samples (in this case, movie titles) that the model processes before updating its weights.
    - The `embedding_dim` parameter in an embedding layer defines the size (dimensionality) of the dense vector that each unique token (in this case, each unique word or token in the movie title) will be mapped to.

Why Embeddings Were Used

- **Dimensionality Reduction and Memory Efficiency**: Embeddings map each word to a dense, lower-dimensional vector space (e.g., 50, 100, or 300 dimensions) instead of using the full vocabulary size. This significantly reduces the memory footprint and makes computations much faster.
- **Capturing Semantic Relationships**: Word embeddings are designed to capture semantic relationships between words by placing similar words close to each other in the embedding space

- **Suitability for Deep Learning Models**: Embeddings, especially when trained as part of a neural network (such as in your case), integrate seamlessly with deep learning architectures. They allow neural networks to learn and refine the embeddings based on the task at hand (e.g., predicting movie ratings), making the representation more task-specific and useful for the model.

Principal Component Analysis (PCA)

To perform PCA on the embeddings produced by the model for the movie titles, the embeddings for each title were flattened or averaged to create a single embedding vector per title. This is because PCA operates on fixed-length vectors, so a consistent shape is required across all titles. The steps used were:

1. **Tokenization and Padding**: Titles are tokenized, and padding is added to make sequences uniform in length. Each token is mapped to an integer using `token_to_index`.

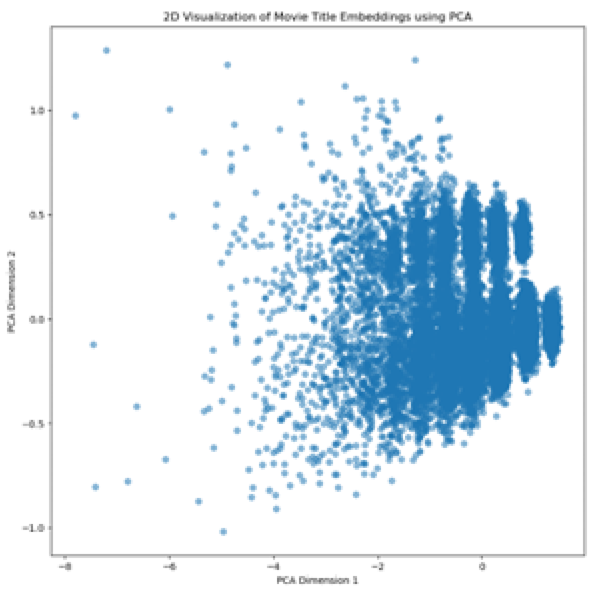
2. **Embedding Layer**: The `EmbeddingModel` class includes an embedding layer, which converts token IDs to dense vectors.

3. **Average Pooling**: For each batch, the model generates a 3D tensor of shape `[batch_size, sequence_length, embedding_dim]`. By averaging across the `sequence_length` dimension, we convert each title's sequence of embeddings into a single vector of size `[embedding_dim]`, resulting in a 2D tensor of shape `[batch_size, embedding_dim]`.

4. **Concatenate Batches**: After processing all batches, concatenate them into a single array of all title embeddings.

5. **PCA**: Perform PCA on the averaged embeddings to reduce their dimensionality to 2D.

6. **Plotting**: Use Matplotlib to plot the 2D embeddings.



Interpretation

Each point in the plot represents a movie title in a 2D space derived from the high-dimensional embedding space.

- A large number of points are densely clustered on the right side of the plot. This indicates that, after PCA, many of the embeddings share similar characteristics in the high-dimensional space, which are preserved to some degree in the reduced 2D space. If these embeddings were trained or influenced by some specific movie attributes (like genre, theme, or common words in titles), this clustering could suggest that these titles share some common semantic or linguistic features
- Some points are spread out toward the left, especially along the negative side of the first PCA dimension. This could mean these titles have uncommon words, unique phrases, or distinct semantic characteristics compared to the main body of titles.

Convolutional Neural Network (CNN)

To predict the average user rating of each movie based on its title embeddings, a Convolutional Neural Network (CNN) was designed and implemented. CNNs, which are typically used for image processing tasks, were chosen here for their ability to capture spatial patterns in data, in this case, the sequential relationships within the title embeddings. Each embedding vector for a title serves as a 1-dimensional “image” input to the model, enabling the CNN to identify patterns across adjacent words and phrases within the embedding space.

Model Architecture

- **Input Layer**: The model starts with an input layer that accepts a reshaped version of the title embeddings, structured to match the CNN’s expected input format. Each title embedding was reshaped to have dimensions (100,1)(100, 1)(100,1), where 100 represents the embedding length, and 1 signifies a single-channel input.
- **Convolutional and Pooling Layers**:
    - The first convolutional layer employs 64 filters with a kernel size of 3, utilizing the ReLU activation function to introduce non-linearity. This layer detects local patterns in the embeddings, such as common sequences of words or semantic relationships between adjacent words.

    - A max-pooling layer with a pool size of 2 follows, reducing the spatial dimensions by half to extract the most relevant features while minimizing the computational load.
    - The second convolutional layer is similar to the first but with 32 filters. This further refines the feature extraction, enabling the model to capture more specific and complex patterns within the embeddings.
    - Another max-pooling layer is applied, reducing the dimensions once again to retain only the essential features.
- **Flattening and Dense Layers**: The output from the convolutional layers is flattened into a 1-dimensional vector, effectively consolidating all detected patterns into a single representation. This flattened vector is then fed into a dense layer with 64 neurons, which applies further transformations to capture higher-level patterns across the entire title embedding.

- **Output Layer**: The final dense layer, comprising a single neuron, is designed to output a scalar value representing the predicted average rating. The model utilizes Mean Absolute Error (MAE) as a loss metric, which is well-suited for regression tasks involving continuous-valued outputs.

Training and Evaluation

The model was trained on a split of the data, with 20% reserved for validation. Over the course of 20 epochs, the model demonstrated a steady reduction in both training and validation loss, indicating effective learning and generalization. The training process yielded a Mean Squared Error (MSE) loss curve that continually declined, showing that the model was minimizing errors across epochs. Validation performance stabilized at an MAE of around 0.42, suggesting the model effectively captured the underlying patterns in the title embeddings to predict user ratings with reasonable accuracy.

Recurrent Neural Network (RNN)

This is a Recurrent Neural Network (RNN), specifically using Long Short-Term Memory (LSTM) layers. RNNs are well-suited for sequential data like your movie title embeddings. LSTMs are a type of RNN designed to handle long-term dependencies in data.

Model Architecture

Input Layer (Implicit)

- This layer is defined by the `input_shape` argument in the first LSTM layer.
- It expects input data with the shape `(100, 1)`, which corresponds to your reshaped title embeddings (100 features per embedding, treated as a sequence of length 100 with 1 feature at each time step).

LSTM Layer 1

- **Units**: 64 (This is the dimensionality of the output space, meaning this layer will output 64 features.)
- **Activation**: `relu` (Rectified Linear Unit - introduces non-linearity)
- **`return_sequences=True`**: This indicates that the output of this LSTM layer should be a sequence, which is then fed into the next LSTM layer.

LSTM Layer 2

- **Units**: 32
- **Activation**: `relu`
- `return_sequences=False` : This LSTM layer returns the final output of the sequence, not a sequence itself.

Dense Layer 1

- **Units**: 16
- **Activation**: `relu`
- This layer is a fully connected layer, meaning each neuron in this layer is connected to every neuron in the previous layer. It acts as a further feature extractor.

Dense Layer 2 (Output Layer)

- **Units**: 1
- This is the final output layer of your model. It has a single neuron because it’s predicting a single continuous value (the movie rating).

Training and Evaluation

The RNN was trained using the **Adam optimizer** on 80% of the dataset, with 20% reserved for validation. Over **10 epochs**, the training loss declined, reflecting the model’s ability to minimize prediction errors. The validation metrics, including **MSE**, **MAE**, and **RMSE**, stabilized, indicating the model successfully generalized to the validation data.

Key Metrics from the Training Process:

- **Validation MSE**: 0.0242
- **Validation MAE**: 0.1252
- **Validation RMSE**: 0.1555

Summary

The model takes **movie title embeddings** as input, processes them through **two LSTM layers** to capture sequential information, then passes the information through **two dense layers** for further processing, and finally produces a **single output** representing the predicted movie rating.

Deep Neural Network (DNN)

To predict the average user rating of movies based on their title embeddings, a Deep Neural Network (DNN) was implemented. DNNs, known for their versatility and ability to learn high-level abstractions, were chosen for this task to identify patterns and relationships within the embedding space. Unlike CNNs, which focus on local patterns, the DNN utilizes fully connected layers to capture global dependencies within the embeddings. This makes it suitable for analyzing the overall structure of the title embeddings and predicting user ratings.

Model Architecture

- **Input Layer**:
  - The input to the DNN is the title embedding vector, represented as a one-dimensional array with **100 features**. These embeddings are fed directly into the first dense layer,

preserving their numerical structure without requiring additional reshaping.

- **Dense Layers**:
  1. **First Dense Layer**:
     - Consists of **256 neurons** with **ReLU activation**.
     - This layer begins extracting meaningful patterns and relationships from the embedding vectors, introducing non-linearity to help model complex patterns.
  2. **Second Dense Layer**:
     - A hidden layer with **128 neurons** further refines these patterns, focusing on higher-level relationships within the embeddings.
  3. **Third Dense Layer**:
     - Contains **64 neurons** and consolidates the features extracted by the previous layers, capturing the most significant aspects of the embeddings.
- **Dropout Layers**:
  - To mitigate overfitting, a dropout rate of 30% is applied after each dense layer. This ensures that the model does not overly rely on specific neurons, improving its ability to generalize to unseen data.
- **Output Layer**:
  - The final dense layer comprises a single neuron that outputs a scalar value, representing the predicted average user rating for the movie. This regression task leverages Mean Squared Error (MSE) as the primary loss function and Mean Absolute Error (MAE) as an evaluation metric to measure performance.

Training and Evaluation

The DNN was trained using the Adam optimizer on 80% of the dataset, with 20% reserved for validation. Over 20 epochs, the training loss consistently declined, reflecting the model's ability to minimize prediction errors. The validation metrics, including MSE, MAE, and RMSE, stabilized, indicating the model successfully generalized to the validation data. Key metrics from the training process include:

- Validation MSE: 0.275
- Validation MAE: 0.420
- Validation RMSE: 0.524

These values suggest that the DNN effectively captured meaningful patterns in the title embeddings to predict user ratings with reasonable accuracy.

Explanation of the Quantitative Metrics

- **MSE (Mean Squared Error)**:
  - The **MSE** measures the average squared difference between predicted and actual ratings, penalizing larger errors more heavily. A lower MSE indicates better predictive accuracy.
  - The final validation **MSE of 0.275** demonstrates that the DNN minimized significant deviations during training.
- **MAE (Mean Absolute Error)**:
  - The **MAE** of **0.420** reflects the average absolute deviation between predicted and actual ratings, meaning the predictions were typically within **0.42 points** of the true ratings.

  - This low MAE suggests the DNN captured the essential relationships within the embeddings.
- **RMSE (Root Mean Squared Error)**:
  - The **RMSE**, calculated as the square root of MSE, provides an interpretable measure of prediction accuracy in terms of rating points.
  - The validation **RMSE of 0.524** indicates that the DNN made accurate predictions, though some deviations remain due to the inherent variability in the data.

Analysis and Next Steps

- The **DNN demonstrates strong predictive performance** with relatively low error metrics and smooth convergence. However, there is room for improvement, as the validation loss stabilizes slightly above the training loss, suggesting mild overfitting.

For Future Enhancements:

1. **Hyperparameter Tuning**:
   - Experiment with learning rates, batch sizes, and layer configurations to optimize performance.
2. **Regularization**:
   - Incorporate **L2 regularization** to penalize large weights and reduce overfitting further.
3. **Early Stopping**:
   - Use **early stopping** during training to halt the process if validation performance stops improving, avoiding unnecessary overfitting.
4. **Feature Engineering**:
   - Enhance the quality of the title embeddings or augment them with additional metadata, such as **movie genres** or **release years**, to provide richer input features.

Conclusion

The **DNN proved effective** in predicting user ratings based on title embeddings, achieving a validation **MAE of 0.420** and **RMSE of 0.524**. While its performance is promising, **fine-tuning** and additional enhancements could further boost its predictive power and generalization. The simplicity of the DNN architecture also makes it **computationally efficient** and easier to deploy for large-scale tasks.

Analysis and Comparison of the Three Models

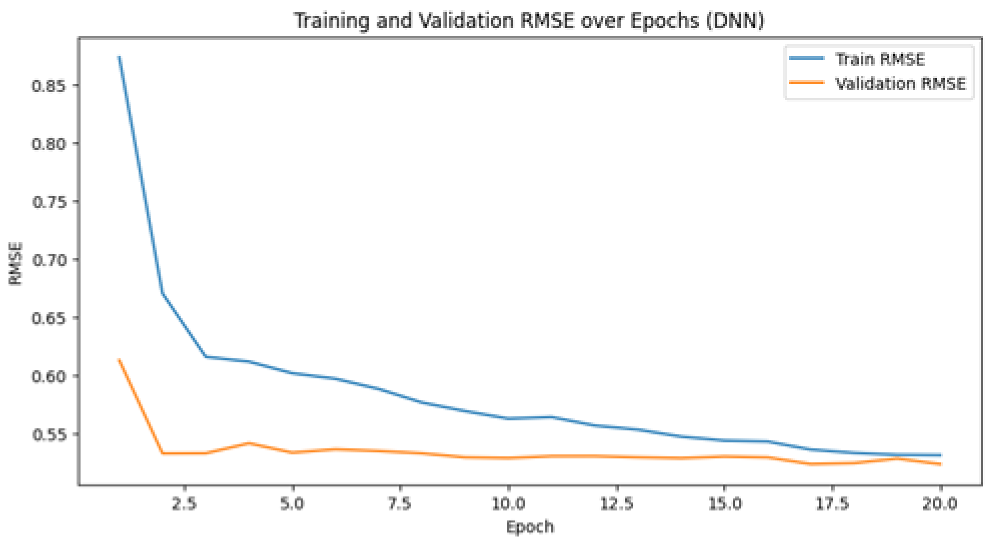
Deep Neural Network (DNN)

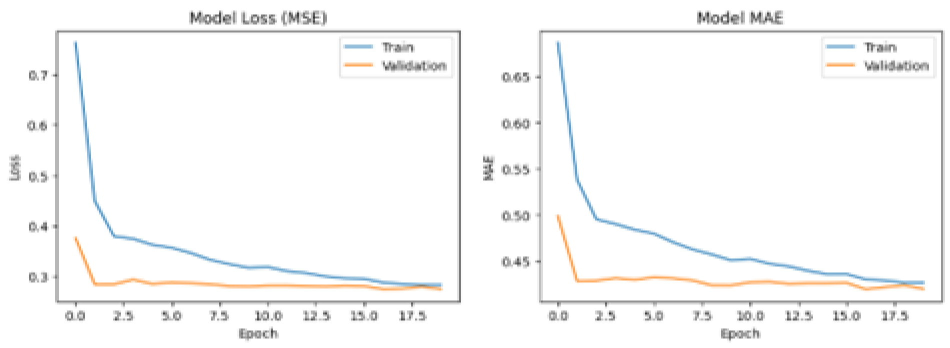
- **Strengths**:
  - Fully connected **dense layers** extract high-level patterns from the title embeddings.
  - **Straightforward architecture** makes it computationally efficient.
  - The model exhibits **smooth convergence** with minimal overfitting, as training and validation losses are closely aligned.
- **Weaknesses**:

- DNNs treat embeddings as **flat arrays**, potentially overlooking the **order of words** and relationships within the sequence.

Performance Metrics:

Metric	Value
Validation MSE	0.275
Validation MAE	0.420
Validation RMSE	0.524



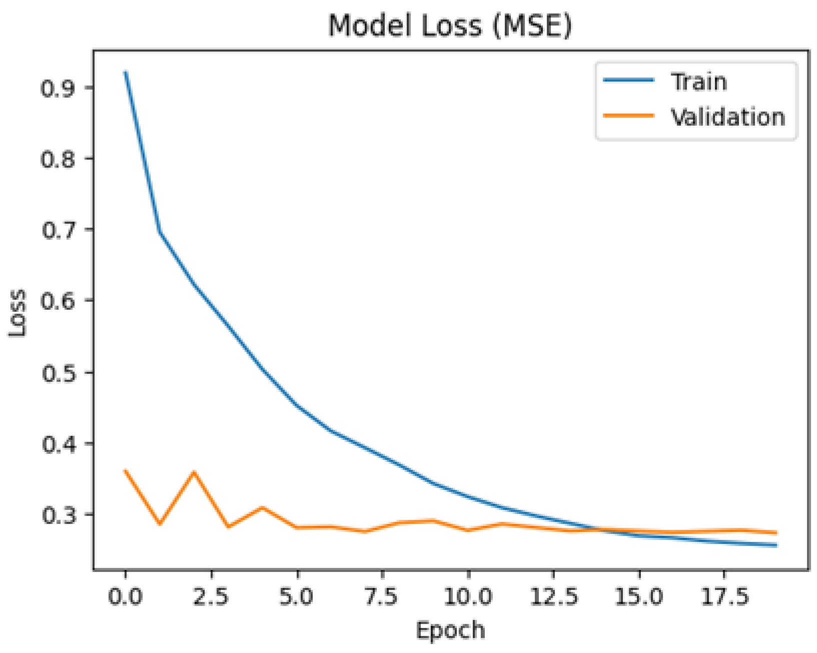


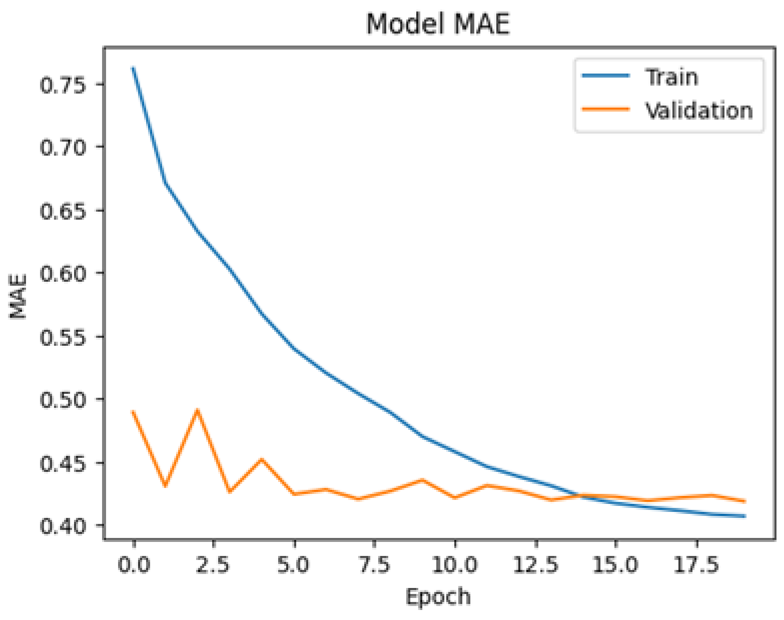
Convolutional Neural Network (CNN)

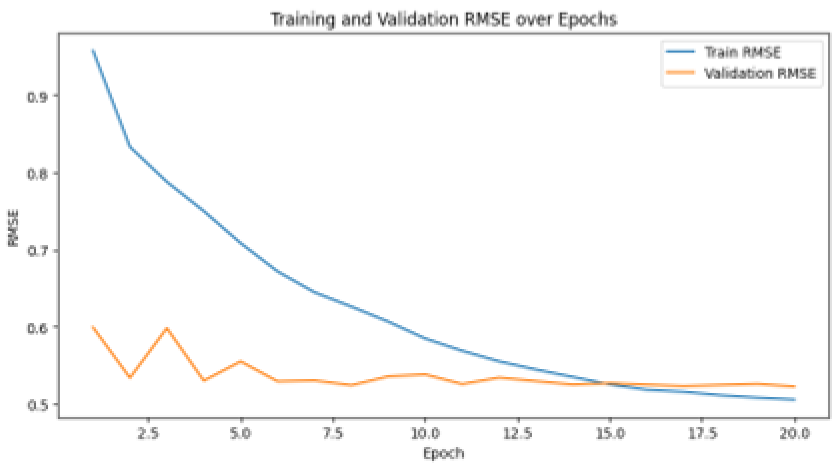
- **Strengths**:
  - **CNNs are adept** at capturing local dependencies in the title embeddings, such as semantic relationships between adjacent words or tokens.
  - The **multi-layered architecture** extracts increasingly complex features as data progresses through layers, refining raw embeddings into higher-level representations.
  - **MaxPooling layers** effectively reduce the spatial dimensions, retaining essential features while minimizing computational overhead.
- **Weaknesses**:
  - While CNNs excel at **local pattern recognition**, they are less effective at capturing **long-range dependencies** or global patterns across the entire embedding sequence.
  - Model performance is highly sensitive to the choice of **kernel size**, **number of filters**, and **pooling strategies**.

Performance Metrics:

Metric	Value
Validation MSE	0.272
Validation MAE	0.416
Validation RMSE	0.521







Recurrent Neural Network (RNN)

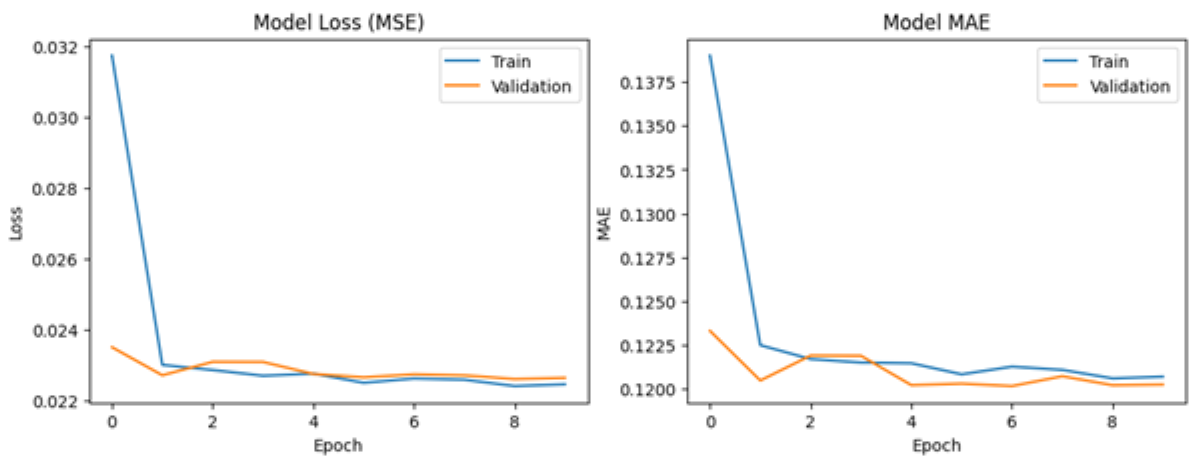
- **Strengths**:
    - **LSTM layers** are designed to process sequential data, effectively capturing both short-term and long-term dependencies in the movie title embeddings.

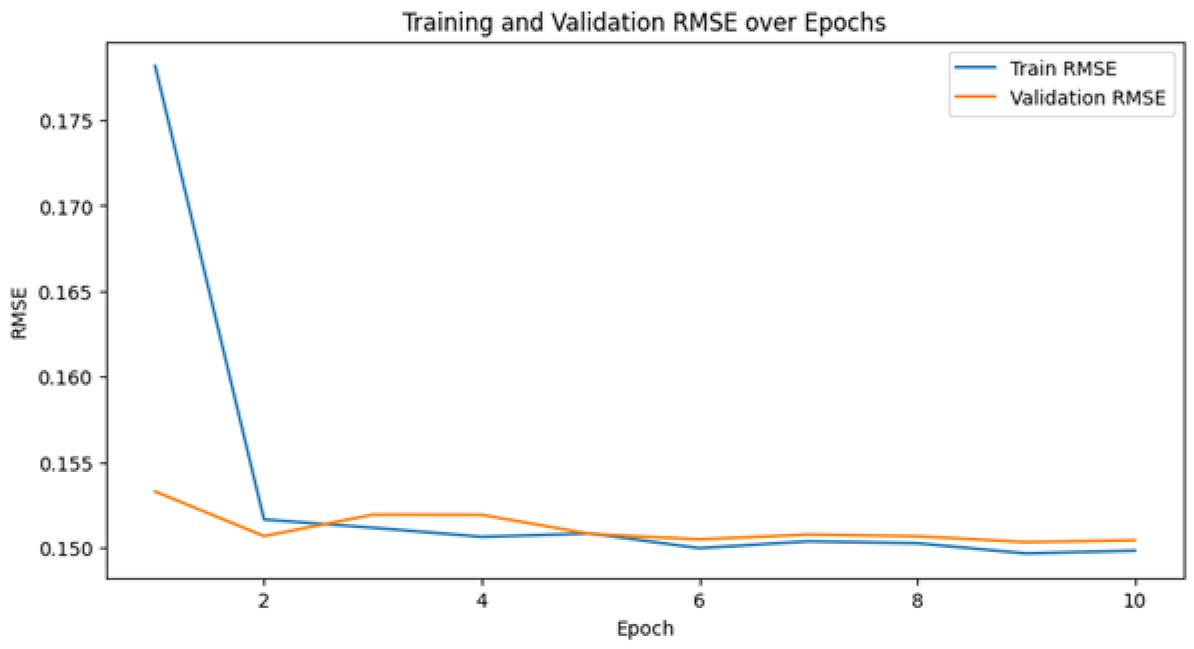
- The **gated mechanisms** in LSTMs (forget, input, and output gates) selectively retain or discard information, making them ideal for tasks requiring memory of past inputs.
- **High Predictive Accuracy**:
  - Achieves outstanding metrics, including:
    - **Validation MSE**: 0.024
    - **Validation MAE**: 0.125
    - **Validation RMSE**: 0.156
  - These metrics are significantly better than those of DNNs and CNNs, showcasing the model's ability to understand and utilize sequential patterns.

- **Weaknesses**:
  - **Training LSTMs** is computationally intensive due to their sequential processing nature, which prevents parallelization across sequence steps.
  - The performance of the RNN heavily depends on the choice of **hyperparameters**, such as the number of LSTM units, activation functions, and learning rate.

Performance Metrics:

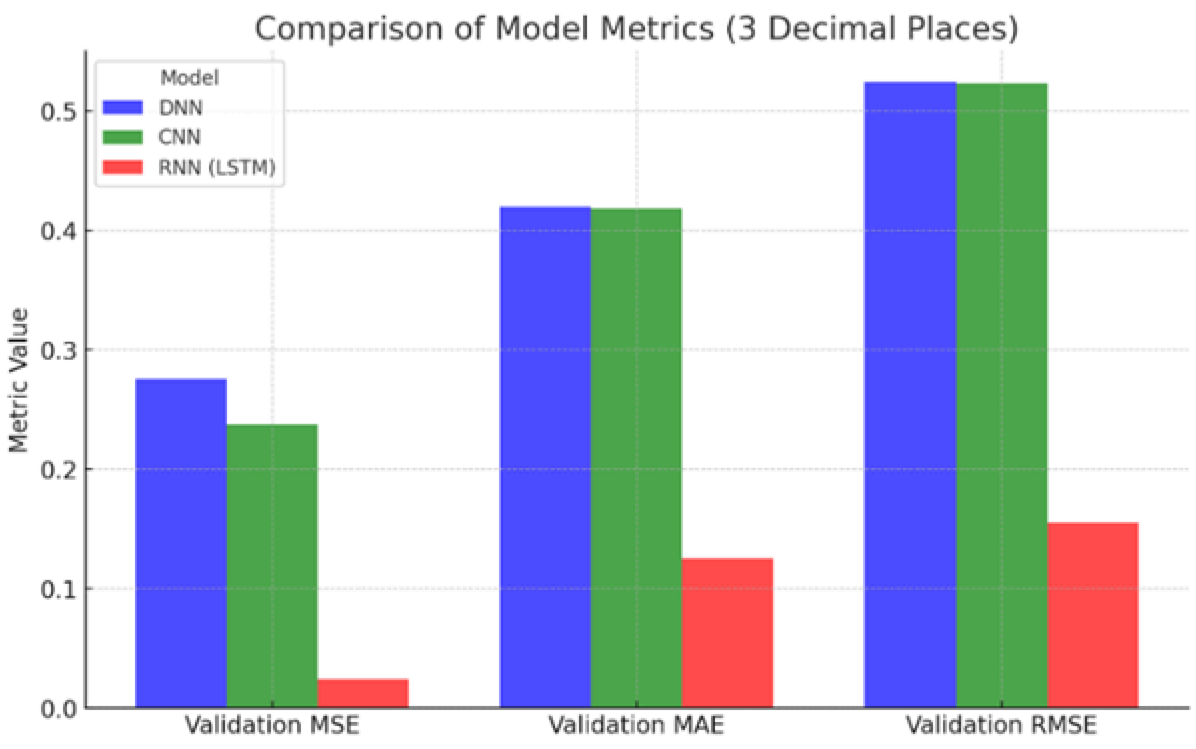
Metric	Value
Validation MSE	0.024
Validation MAE	0.125
Validation RMSE	0.156





Model Comparison

Model	Validation MSE	Validation MAE	Validation RMSE
DNN	0.275	0.420	0.524
CNN	0.272	0.416	0.521
RNN	0.024	0.125	0.156



- The comparison of the three models—DNN, CNN, and RNN (LSTM)—reveals distinct strengths and performance characteristics based on their validation metrics. The RNN emerges as the most accurate model, achieving the lowest validation MSE (0.024), MAE (0.125), and RMSE (0.156). This exceptional performance likely stems from the LSTM's inherent ability to process movie title embeddings as sequences, effectively capturing both short-term and long-term dependencies. The CNN performs moderately well, with metrics slightly better than the DNN: Validation MSE (0.272), MAE (0.416), and RMSE (0.521). This exceptional performance likely stems from the LSTM's inherent ability to process movie title embeddings as sequences, effectively capturing contextual relationships at various time scales. The DNN, while computationally efficient and simpler in architecture, exhibits the highest error rates among the three models, with Validation MSE (0.275), MAE (0.420), and RMSE (0.524). This relatively weaker performance reflects the model's fundamental limitation in processing sequential data, as it treats each element of the embedding independently. Overall, in this task of movie title embedding, the RNN yields the most promising results, then followed by CNN and DNN.

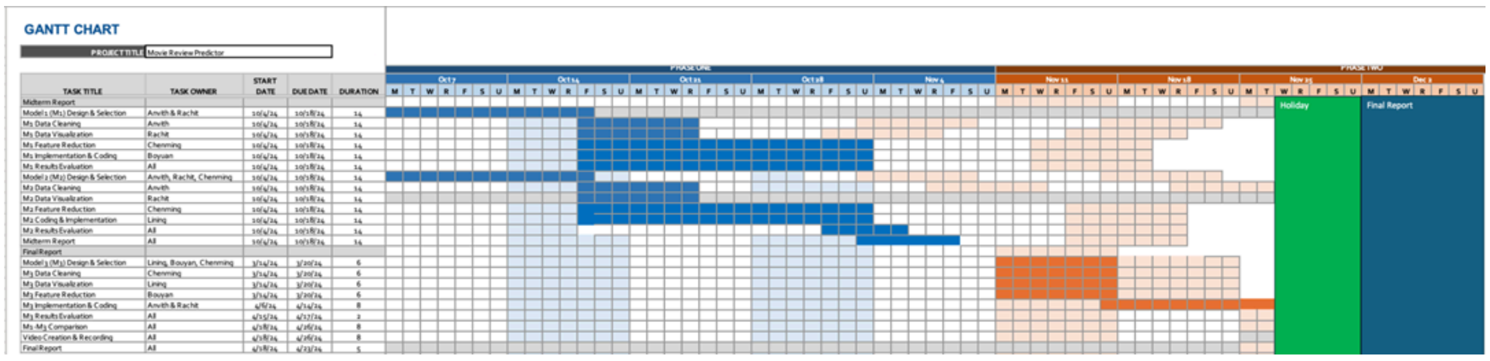
References

- Priya, S. K., Manonmani, T., Dharshana, N., & Ragaanasuya, K. (2022). Movie recommendation system with hybrid collaborative and content-based filtering using convolutional neural network. *International Journal of Health Sciences*, 6(S8), 5357-5372.
- Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., & Li, Y. (2023). A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems*, 1(1), 3. https://doi.org/10.1145/3568022
- Khan, Z. Y., Niu, Z., Sandiwarno, S., & Prince, R. (2021). Deep learning techniques for rating prediction: A survey of the state-of-the-art. *Artificial Intelligence Review*, 54(1), 95–135.

https://doi.org/10.1007/s10462-020-09892-9

- Lund, J., & Ng, Y. K. (2018). Movie recommendations using the deep learning approach. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)* (pp. 47-54). IEEE. https://doi.org/10.1109/IRI.2018.00015
- Liu, D. Z., & Singh, G. (2016). A recurrent neural network based recommendation system. In *International Conference on Recent Trends in Engineering Science & Technology*.

Gantt Chart



GANTT CHART

PROJECT TITLE Movie Review Predictor

TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION
Midterm Report				
Model 1 (M1) Design & Selection	Anvith & Rachit	10/4/24	10/18/24	14
M1 Data Cleaning	Anvith	10/4/24	10/18/24	14
M1 Data Visualization	Rachit	10/4/24	10/18/24	14
M1 Feature Reduction	Chenming	10/4/24	10/18/24	14
M1 Implementation & Coding	Boyuan	10/4/24	10/18/24	14
M1 Results Evaluation	All	10/4/24	10/18/24	14
Model 2 (M2) Design & Selection	Anvith, Rachit, Chenming	10/4/24	10/18/24	14
M2 Data Cleaning	Anvith	10/4/24	10/18/24	14
M2 Data Visualization	Rachit	10/4/24	10/18/24	14
M2 Feature Reduction	Chenming	10/4/24	10/18/24	14
M2 Coding & Implementation	Lining	10/4/24	10/18/24	14
M2 Results Evaluation	All	10/4/24	10/18/24	14
Midterm Report	All	10/4/24	10/18/24	14
Final Report				
Model 3 (M3) Design & Selection	Lining, Bouyan, Chenming	3/14/24	3/20/24	6
M3 Data Cleaning	Chenming	3/14/24	3/20/24	6
M3 Data Visualization	Lining	3/14/24	3/20/24	6
M3 Feature Reduction	Bouyan	3/14/24	3/20/24	6
M3 Implementation & Coding	Anvith & Rachit	4/6/24	4/14/24	8
M3 Results Evaluation	All	4/15/24	4/17/24	2
M1-M3 Comparison	All	4/18/24	4/26/24	8
Video Creation & Recording	All	4/18/24	4/26/24	8
Final Report	All	4/18/24	4/23/24	5

Contribution Table

Team Member	Contributions
Lining Song	Analysis and Comparison of models
Boyuan Chen	Video report and putting contents to webpage
Chenming Fan	Create the powerpoint for presentation
Rachit Gupta	Implementations and analysis of DNN model
Anvith Anand	Implementations and analysis of RNN model